Technologies for the Social Semantic Desktop

Michael Sintek[1], Siegfried Handschuh[2], Simon Scerri[2], and Ludger van Elst[1]

[1] Knowledge Management Department
German Research Center for Artificial Intelligence (DFKI) GmbH,
Kaiserslautern, Germany
`{firstname.surname}@dfki.de`
[2] DERI, National University of Ireland, Galway
`{firstname.surname}@deri.org`

Abstract. The vision of the *Social Semantic Desktop* defines a user's personal information environment as a source and end-point of the *Semantic Web*: Knowledge workers comprehensively express their information and data with respect to their own conceptualizations. *Semantic Web* languages and protocols are used to formalize these conceptualizations and for coordinating local and global information access.

A core challenge is to integrate existing legacy Desktop data into the *Social Semantic Desktop*. Semantic lifting is the process of capturing the semantics of various types of (semi-)structured data and/or non-semantic metadata and translating such data into *Semantic Web* conceptualizations.

From the way the vision of the *Social Semantic Desktop* is being pursued in the NEPOMUK project, we identified several requirements and research questions with respect to knowledge representation. In addition to the general question of the expressivity needed in such a scenario, two main challenges come into focus: i) How can we cope with the heterogeneity of knowledge models and ontologies, esp. multiple knowledge modules with potentially different interpretations? ii) How can we support the tailoring of ontologies towards different needs in various exploiting applications?

In this paper, we present semantic lifting as a means to create semantic metadata and the Nepomuk Representation Language (NRL) as a means to represent these metadata. NRL is an approach to these two aforementioned questions that is based on named graphs for the modularization aspect and a view concept for the tailoring of ontologies. This view concept turned out to be of additional value, as it also provides a mechanism to impose different semantics on the same syntactical structure.

We furthermore present some of the ontologies that have been developed with the help of NRL in the NEPOMUK project to build the semantic foundations for the *Social Semantic Desktop*.

1 Overview

This paper constitutes the material for the lecture on the (Social) Semantic Desktop given at the Reasoning Web Summer School 2009 (`http://reasoningweb.org/2009/`).

In Sect. 2, we present the basic ideas of the *Social Semantic Desktop*. The remaining sections describe technologies developed and used in NEPOMUK and other projects to build the *Social Semantic Desktop*. Lifting (Sect. 3) is the process of capturing the semantics of various types of (semi-)structured data and/or non-semantic metadata and translating such data into relations, attributes and concepts within an ontology. NRL (Sect. 4) is the NEPOMUK Representational (ontology) Language, developed as an extension to RDF/S with additional support for named graphs and views, in order to fulfill some of the requirements of a representational language for the *Social Semantic Desktop*. Finally, in Sect. 5, we present some of the resulting ontologies that have been developed in NEPOMUK.

2 The Social Semantic Desktop

2.1 Motivation

The very core idea of the *Social Semantic Desktop* is to enable data interoperability on the personal desktop based on *Semantic Web* standards and technologies, *e. g.*, Ontologies and semantic metadata. The vision [13] aims at integrated personal information management as well as at information distribution and collaboration, envisioning two expansion states: i) the *Personal Semantic Desktop* for personal information management and later ii) the *Social Semantic Desktop* for distributed information management and social community aspects.

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. Similarly, 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 means for a seamless communication between an application used by one person on their desktop and an application used by another person on another desktop. The knowledge exchange and integration problem on the desktop is thus similar to that which exists on the Web.

The *Social Semantic Desktop* paradigm adopts ideas from the *Semantic Web* (SW) paradigm [4], which offers a solution for the web. Formal Ontologies capture both a shared conceptualization of desktop data and personal mental models. RDF (Resource Description Format) serves as a common data representation format. 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 aligning the

Social Semantic Desktop paradigm with the *Semantic Web* paradigm, a *Semantic Desktop* can be seen as both the source and the end-point of the *Semantic Web*.

2.2 State of the Art

In the following we present a brief review of relevant research and development approaches for the *Social Semantic Desktop*.

Gnowsis [23] was among the first research projects targeting a *Semantic Desktop* system. Its goal was to complement, rather than replace, established desktop applications and the desktop operating system with *Semantic Web* features. The primary focus of Gnowsis was on Personal Information Management (PIM). It also addressed the issues of identification and representation of desktop resources in a unified RDF graph.

The Haystack [20] project presents a good example for an integrated approach to the *Social Semantic Desktop* 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 IRIS Semantic Desktop [8] (Integrate. Relate. Infer. Share) provided an application framework that enables users to create a personal map across their office-related information objects.

DeepaMehta [22] is an open source *Semantic Desktop* application based on the Topic Maps standard. 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.

Although the systems we have looked at focused on isolated and complementary aspects, they clearly influenced the vision of the *Social Semantic Desktop* presented in this paper. However our vision is more general and comprehensive.

2.3 Networked Collaborative Knowledge

We all face the problem of having increasingly more information on our desktops. The average workspace covers hundreds of thousands of different files (including emails), some of which we vaguely remember the place in which they were stored. To make matters worse for the desktop user, the web has not only enabled further information creation and dissemination, but has also opened wide the information floodgates. Furthermore, this information is highly confined. The computer desktop is our universal workspace, where we have all kinds of information in different formats, and use it for various purposes in different applications. Some of this data has little explicit representation, is not always suitably structured and is trapped and imprisoned in applications, *i. e.*, Data Silos. We have multiple isolated information spaces on the desktop, *e. g.*, email clients, file systems, music managers, web browsers. The same is true for the collaborative web information system we use, *e. g.*, wikis, sharepoint, BSCW. These data silos prevent us from joint problem solving and collaboration, as well as answering questions whose result is spread across multiple workspaces. In short, they hinder us from exchanging personal content from one workspace to another.

The central idea of the *Social Semantic Desktop* focuses on how social and collaborative activities and their coordination can be improved through semantic technologies. Semantics hold the promise of automatic understanding and better information organization and selective access, and providing standard means for formulating and distributing metadata and Ontologies. Hence, semantic collaborative information management facilitates the integration of information between desktop applications and the Web, *i. e.*, focused and integrated personal information management along with information distribution and collaboration.

Classical collaborative information management takes place in controlled, closed and comparatively small environments. In parallel, the WWW emerged as a phenomenon that is unstructured, social, open, and which distributes information on a large scale. Thus information is often disconnected on the Web. To solve this we require computers to make sense of this information, hence meaning, and thus semantics; to achieve computer-understandable data by exploiting existing resources. These existing resources can be lifted by using formal languages, such as RDF/S or NRL (*cf.* Sect. 4). This enables us to network the data and thus to achieve a higher level of new information.

Although knowledge is inherently strongly interconnected and related to people, this interconnectedness is not reflected or supported by current information infrastructures. The lack of interconnectedness hampers basic information management and problem-solving and collaboration capabilities, like finding, creating and deploying the right knowledge at the right time.

Besides the creation of knowledge through observation, networking of knowledge is the basic process to generate further knowledge. Networking knowledge, can produce a piece of knowledge whose information value is far beyond the mere sum of the individual pieces, *i. e.*, it creates new knowledge. With the Web we now have a foundational infrastructure in place that enables the linking of information on a global scale. Furthermore, with the desktop we have an infrastructure that stores all our personal information models. Adding meaning moves the interlinked information to the knowledge level: Web + Semantics + Desktop = *Social Semantic Desktop*.

Now is the time to tackle the next step: exploiting semantics to create an overall knowledge network that bridges the information islands in order to enable people, organizations and systems to collaborate and interoperate on a global scale.

In the following we will show how a *Social Semantic Desktop* can provide answers to the following questions:

Q1: How do you structure your personal information on your desktop? How do you structure your file system, your email and your bookmarks? Do you use other means to manage your information?

Q2: How do you share and exchange the data with your colleagues? With email—like most people, or with a Wiki, a share point system, *etc.*?

Q3: How do you find an expert in your organization, given it employs many people as to make it hard for you to keep an overview?

2.4 User Mental Models

Representation of the users mental models take the form of a personal information model (*cf.* Sect. 5.3). Lets envision an average desktop user called Claudia (*cf.* Fig. 1), who is organizing her information in folders and emails. A close look at it will reveal common topics in both structures, such as projects, organization, people, topics, *etc.*These mental models are currently isolated in her applications, and the goal of the *Semantic Desktop* is to free this information and represent it explicitly.

Therefore we propose to apply *Semantic Web* technologies to represent these mental models, by utilizing existing and/or extended standards and RDF/S vocabularies such as VCard for an optimal information representation and then to lift (*cf.* Sect.3) the existing structured data up to a NRL representation (*cf.* Sect. 4); thereby allowing the structuring of the mental model only once and not several times.

2.5 Interconnected Desktops

The explicit classification scheme, encapsulated within the Personal Information Model PIMO (*cf.* Sect. 5.3), helps individuals manage their desktop information. The semantics of NRL allows for the automatic processing of this model and the deduction of new knowledge. It creates a kind of a personal semantic web: a semantically-extended supplement to the user's view of their personal information.

This research contributes to the so-called vocabulary "onion" by providing PIMO, NIE (*cf.* Sect. 5.2) and other required vocabularies. An instance of the PIMO represents a single user's concepts, such as projects, tasks, organizations, *etc.*The NIE set of ontologies provides vocabularies for describing information elements (such as files, contact, calendar data, emails-s) which are commonly present on the desktop and in collaborative systems.

By interconnecting such personal semantic desktops (*cf.* Fig. 1), *e. g.*, via a client server model or the use of P2P technology; we can easily exchange information. Not only can we exchange data in the form of documents, but also structured information about project, people, events, *etc.*; as well as the personal models themselves. For example, Claudia might have developed a very good structure for a project in which Dirk is also working, so she shares this structure with Dirk, hence allowing him to re-use this information. Note that this is not possible with current desktop systems—one cannot easily transfer their project file folder structure to a colleague.

On top of the P2P networking (which allows content-based routing) we have social protocols and algorithms that enable an explicit representation of relationships which is similar to social systems (*e. g.*, LinkedIn[1] and others), yet is

[1] `http://www.linkedin.com/`

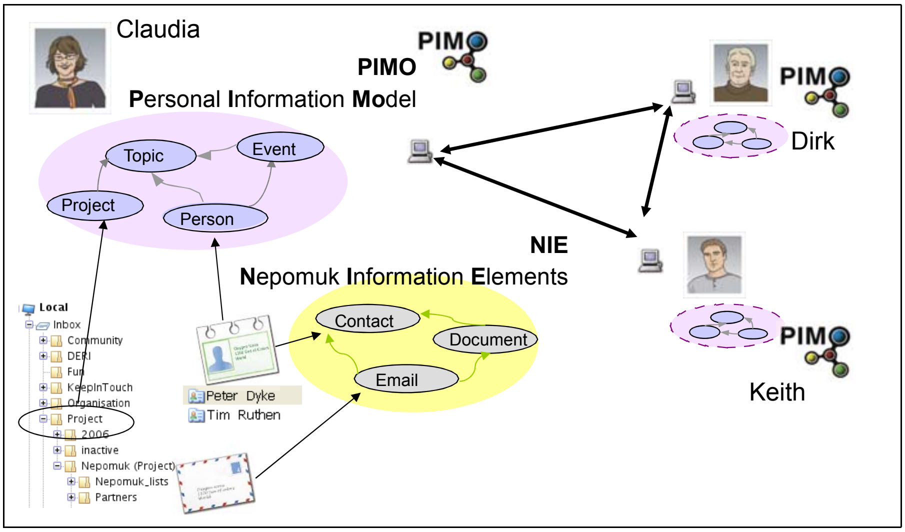

Fig. 1. Interconnected *Social Semantic Desktops*

open in the sense that it allows for the creation of new connections and the establishment of new relationships. This accelerates collaboration and allows for the maintenance of shared views.

2.6 Achievements

We can conclude that via the *Social Semantic Desktop* we achieve a universal platform for:

– Personal Information Management
– Distributed Information Management
– Social Expansion and Community Creation

The impact results in dramatic time savings, by i) filtering out marginal information, ii) discovering vital information and building, as well as participating, in communities of practice.

We manage personal information by mapping native structures onto our own mental models and representing data in a unified way. The social aspect of sharing and community building in an organization is done by connecting individual semantic desktops.

The answers to the previous questions, with the *Social Semantic Desktop* now in the picture, are thus:

A1: The user can manage and structure their personal information (mental model) via PIMO.

A2: The user can share and exchange their personal information via the *Social Semantic Desktop* network, which allows for a content-based routing and a "link routing" based on social connections.
A3: The user can find experts within their social circle by using intelligent services on top of the *Social Semantic Desktop* infrastructure. These utilize the interest profile of the users' PIMOs to detect and classify experts and communities.

In the following chapters we will learn about the foundational technology which enables the realization of the here presented general issues, *i. e.*, methodologies for the lifting of existing data onto personal information models, and the semantic backbone of the *Social Semantic Desktop*—consisting of NRL and the rest of the NEPOMUK Ontologies.

3 Semantic Lifting and Human Language Technologies for the Semantic Desktop

3.1 Background

The *Social Semantic Desktop* requires metadata represented in RDF/NRL (*cf.* Sect. 4) to operate. The RDF metadata can be the result of the following processes:

- i) Lifting of existing structured data onto RDF
- ii) Usage of Human Language Technology (HLT) to capture knowledge from text and transform that into RDF
- ii) Manual creation of metadata by linking, annotation or tagging

In this chapter we will focus on lifting and HLT. Semantic lifting is the process of capturing the semantics of various types of (semi-)structured data and/or non-semantic metadata and translating such data into relations, attributes and concepts within an ontology. Candidate data for lifting includes non-semantic metadata (*e. g.*, in XML), emails, directory structures, files on disk, IMAP mailboxes, address books and schemas such as iCalendar[2]. The core challenges are to integrate existing legacy Desktop data into the *Social Semantic Desktop* (*cf.* Fig. 2); to expose or make explicit such data to both the *Social Semantic Desktop* and the *Semantic Web*; to reuse, rather than replace, existing data; and to enhance, rather then replace, existing applications.

Human language technology (HLT), in its broadest sense, can be described as computational methods for processing and manipulating language, for instance text analysis, information extraction or controlled language. This technology has materialized on the *Semantic Desktop* in the form of integration into the user's email client, personal meeting note-taker as well an automated textual analysis of documents on their desktop.

[2] `http://en.wikipedia.org/wiki/ICalendar`

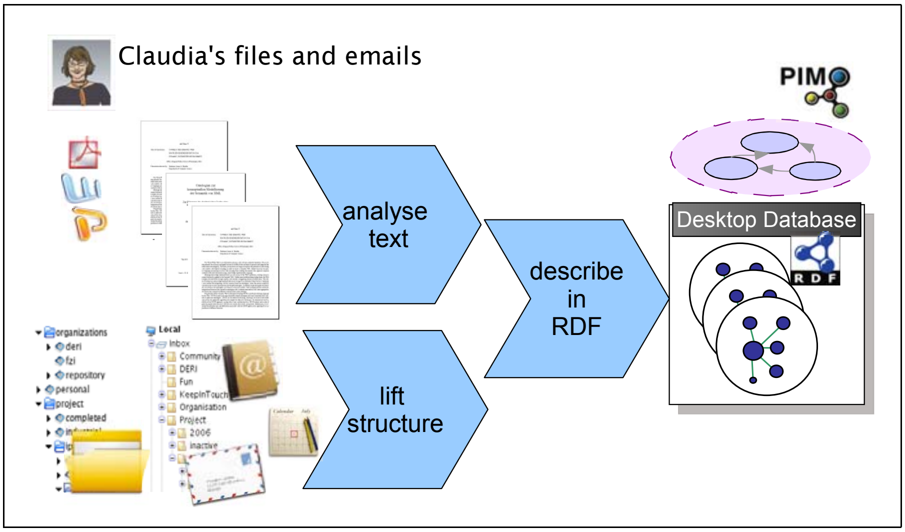

Fig. 2. Lifting unstructured data onto standard semantic representations

3.2 Lifting on the Semantic Desktop

In most cases, the process of lifting structured information onto an RDF layer indexes data that exists in desktop applications. The data is converted to standard vocabularies and stored in an RDF repository which serves as a local storage. Existing systems have implemented a lifting service for a *Semantic Desktop* up to varying degrees, *e. g.*, Aperture, which is a Java application; Beagle++, a Linux-Gnome desktop crawler; and Strigi, which is part of KDE 4. We will now have a closer look at these systems.

Aperture: Aperture[3] is a cross-platform java project. Aperture extracts data from various information sources by crawling each source. It transforms the data from the existing formats to RDF, using a set of purposely-developed Ontologies. Aperture only does the crawling and extraction, since storage is usually handled by a Sesame[4] RDF data store.

Beagle++: Beagle++[5] is based on Gnome Beagle. Beagle is a desktop and search application. Gnome Beagle consists of sets of Backends and Filters. Every backend is in charge of extracting metadata from various data sources.

Semantic extensions for extractors of Gnome Beagle towards Beagle++ are: i) Path Annotation with WordNet, ii) Web Cache Metadata Generation, and iii) Publication Metadata Generation from PDF files.

[3] `http://aperture.sourceforge.net/`
[4] `http://www.openrdf.org/`
[5] `http://beagle2.kbs.uni-hannover.de/`

Strigi/Soprano: Nepomuk-KDE[6] uses Strigi[7] and Soprano[8] as core component for data lifting. Strigi, in a similar fashion to Aperture and Beagle++, crawls the data available on the hard disk and extracts the file metadata as well as the content of the files (where it makes sense to do so). As an example, audio files often carry information about the artist in their metadata. On the other hand while PDF files can contain metadata about the author, the author can also be referred to in the content of the PDF file itself. Soprano runs in the Nepomuk storage process. Strigi reads the data out of the files and passes the information into Nepomuk/Soprano. Sesame[9] or Redland[10] are RDF repository backends for Soprano. Soprano fully supports both PIMO and NIE as valid data formats.

3.3 Human Language Technology on the Semantic Desktop

In this section we brief the application of Human Language Technology (HLT) to extract information from textual documents, and how techniques like controlled language and natural language generation can be utilized to generate user-friendly interfaces to the *Semantic Desktop*.

HLT applied on Textual Content. We first have a look at the application of HLT for information extraction from textual content in order to create NRL-based metadata from documents.

Keyphrase Extraction On the *Semantic Desktop* keyphrases are an important instrument for cataloging and information retrieval purposes, *e. g.*, Keyphrases can be used for Semantic Tagging. In literature research, they provide a high-level and concise summary of the content of textual documents or the topics covered therein, allowing humans to quickly decide whether a given text is relevant. As the amount of textual content on desktops grows fast, keyphrases can contribute to manage large amounts of textual information, for instance by marking up important sections in documents, *i. e.*, to provide increased user experience in document exploration.

As keyphrases are a description of textual data, the consideration of HLT tools in order to automate the extraction process is obvious. While shallow techniques are a long way from language understanding, in combination with statistical processing they can be helpful in many ways, providing a first stop in automatic content-metadata extraction, which then can be used as input for more sophisticated technologies.

The main idea here is to use the keyphrases as a first step to propose a Semantic Tag in order to annotate a document (*cf.* Sect. 5.1). The reduced set of keyphrase candidates will provide a less noisy summary of the topics mentioned in a document. This reduced set, in fact, does enable querying ontology

[6] `http://nepomuk.kde.org/`
[7] `http://strigi.sourceforge.net/`
[8] `http://soprano.sourceforge.net/`
[9] `http://www.openrdf.org/`
[10] `http://librdf.org/`

libraries (*e. g.*, OntoSelect[11] or Watson[12]) for good-fitting schemes, which then can be retrieved for further semantic annotation in addition to the Semantic Tags provided by the keyphrases.

Speech Act Detection One of the applied HLT on text technology of the *Semantic Desktop* is based on speech act detection in email and instant messaging conversations. The notion of a speech act pursued here is based on that defined by John Searle [31]. At its most basic definition a speech act is an utterance, understood more specifically as a performative utterance or an illocutionary act (a term introduced by John L. Austin [1]), where it is assumed that by saying something one actually is doing something. In our case, the utterances take the form of typed text. For instance, a sentence from Claudia's subordinate asking her politely to attend an important meeting, expresses the speaker's (or the sender of the sentence) wish for Claudia to attend, and sets a new requirement for Claudia—to reply to the meeting suggestion. On the contrary, the same sentence from Claudia's manager will also express the wish of the sender for Claudia to attend, but the expected requirement for Claudia will be to attend the meeting without further ado.

Semanta[13] is a fully-implemented system supporting Semantic Email, whereby we have lifted email processes to a semantic level via speech act theory and a formally-defined ad-hoc email workflow model. In our approach we considered the fact that an email has one or more purposes, or Action Items. The content of an email message can be summarized into a number of such items (*e. g.*, Meeting Request, Task Assignment, File Request, *etc.*). Once exchanged, every single action item can be seen as the start, or continuation of a separate workflow.

The sMail Conceptual Framework [28] applies Speech Act Theory [31] to the email communication process, in order to provide a formal structure and semantics for these action items and their workflows. Email action items like the ones above can be represented by a number of speech act instances provided in the sMail ontology[14]. The Email Speech Act Workflow model [27] is then used to support the user with handling email workflows, *e. g.*, providing them with a set of options when reacting to action items in email.

Computational linguistics technologies, namely Ontology-Based Information Extraction (OBIE) techniques, are employed to provide semiautomatic annotation of action items (speech acts) in email content. The information extraction is based on a declarative model which classifies text into speech acts based on a number of linguistic features like sentence form, tense, modality and the semantic roles of verbs. The system deploys a GATE [9] corpus pipeline consisting of a tokenizer, modified sentence splitter, POS tagger, keyphrase lookup via Finite State gazetteers and several JAPE [10] grammars. Email annotations are represented in RDF, using a number of Ontologies (sMail, NMO, NCO, *etc.*) and embedded within email messages. This enables semantic email to be used

[11] `olp.dfki.de/ontoselect/`
[12] `http://watson.kmi.open.ac.uk/WatsonWUI/`
[13] `http://smile.deri.ie/projects/semanta`
[14] `http://ontologies.smile.deri.ie/smail`

as a vessel for the transportation and also the sharing of semantics across social semantic desktops.

HLT to generate Interfaces. HLT can be applied to Controlled Language, Natural Language Generation and Document Analysis in order to provide a user-friendly interface to the *Semantic Desktop*. Below we provide examples of how these techniques were utilised.

Controlled Language Interfaces Our research investigates how HL) Interfaces, specifically Controlled Natural Languages (CNL) and applied Natural Language Generation(NLG) can provide a user-friendly means for the non-expert users or small organizations to exploit *Semantic Web* technologies specifically on the *Social Semantic Desktop*.

Roundtrip Ontology Authoring[12] (ROA) is a process that allows non-expert users to author or amend an ontology by using simple, easy-to-learn, controlled natural language. The process is a combination of Controlled Language for Information Extraction (CLIE) and Text Generation which is developed on top of GATE.

Furthermore Controlled Language (CNL) [11] offers an incentive to the novice user to annotate, while simultaneously authoring his/her respective documents in a user-friendly manner, but simultaneously shielding him/her from the underlying complex knowledge representation formalisms. A natural overlap exists between tools, used for both ontology creation and semantic annotation. However, there is a subtle difference between both processes. Semantic annotation has been described as both a process, as well as the outcome of the process. Hence it describes i) the process of addition of semantic data or metadata to the content given an agreed ontology and ii) the semantic data or metadata itself as a result of this process. Of particular importance here is the notion of the addition or association of semantic data or metadata to content.

Personalized Visual Document Collection Analysis The PIMO ontology can also be used to aid scientists and analysts alike in exploring a text collection in a personalized manner in addition to being a formal representation of parts of knowledge workers' spheres of interest.

Apart from the need to retrieve information from documents that are relevant to certain topics of interest, knowledge workers often also need to explore and analyze a collection of documents as a whole, to gain further understanding. Unlike the information retrieval activity, the information analysis activity aims to provide the users with an overall picture of a text collection as a whole, on various dimensions instead of presenting them with the most relevant documents satisfying some search criteria. Given the amount and the unstructured or weakly-structured nature of textual documents that analysts have to deal with, developments in visualization research are beneficial in helping them to gain needed insights in a timely manner.

In this context, we utilized an innovative visualization approach, called IVEA [35,36], which leverages upon the PIMO ontology and the Coordinated Multiple Views technique to support the personalized exploration and analysis of document collections. IVEA allows for an interactive and user-controlled exploration process in which the knowledge workers can gain meaningful, rapid understanding about a text collection via intuitive visual displays. Not only does it allow the users to integrate their interests into the visual exploration and analysis activity, but it also enables them to incrementally enrich their PIMO Ontologies with entities matching their evolving interests in the process. With the newly added entities, the PIMO ontology becomes a richer and better representation of the users' interests and hence can lead to better and more personalized exploration and analysis experiences in the future. Furthermore, not only can IVEA be beneficial to its targeted task, but it also provides an easy and incremental way that requires minimal effort from the users to keep their PIMO Ontologies in line with their continuously changing interests. This, indirectly, can also benefit other PIMO-based applications. The work leverages upon research in information visualization, information retrieval, and human language technology for semantic annotation. These technologies are used in support of the larger HCI goal of enabling effective personalized interactions between users and text collections.

In the next chapters, we will learn about the underlying representation formalism for the lifted knowledge—NRL and the other Nepomuk Ontologies.

4 NRL—The NEPOMUK Representational Language

4.1 Motivation

The viewpoint of the user comprehensively generating, manipulating and exploiting private as well as shared and public data has to be adequately reflected in the representational basis of a *Social Semantic Desktop*. While we think in general the assumptions of knowledge representation in the *Semantic Web* are a good starting point, the *Semantic Desktop* scenario generates special requirements. We identified two core questions which we try to tackle in the knowledge representation approach presented in this paper:

1. How can we cope with the heterogeneity of knowledge models and ontologies, esp. multiple knowledge modules with potentially different interpretation schemes?
2. How can we support the tailoring of ontologies towards different needs in various exploiting applications?

The first question is rooted in the fact that with heterogeneous generation and exploitation of knowledge there is no "master instance" which defines and ensures the "interpretation sovereignty." The second question turned out to be an important prerequisite for a clean ontology design on the semantic desktop, as many applications shall use a knowledge worker's "personal ontology."

From these general questions, we outlined the following five main requirements for knowledge representation on the *Social Semantic Desktop*:

Epistemological adequacy of modeling primitives: In the *Social Semantic Desktop* scenario, knowledge modeling is not only performed offline (*e. g.*, by a distinguished knowledge engineer), but also by the end user, much like in the tagging systems of the Web 2.0 where a user can continuously invent new vocabulary for describing his information items. Even if much of the complexity of the underlying representation formalism can be hidden by adequate user interfaces, it is desirable that there is no big *epistemological gap* between the way an end-user would like to express his knowledge and the way it is represented in the system.

Integration of open-world and closed-world assumptions: The main principle of the SW is that it is an open world in which documents can add new information about existing resources. Since the Web is a huge place in which everything can link to anything else, it is impossible to rule out that a statement could be true, or could become true in the future. Hence, the global semantic web relies on a open-world semantic, with no unique-name assumption—the official OWL and RDF/S semantics. On the other hand, the main principle on the personal *Semantic Desktop* is that it is a closed-world as it mainly focuses on personal data. While most people find it difficult to understand the logical meaning and potential inferences statements of the open-world assumption, the closed-world assumption is easier to understand for the user. Hence, the Personal Semantic Desktop requires the closed-world semantics with a unique-name assumption or good smushing techniques to achieve the same effects. The next stage of expansion of the personal semantic desktop is the *Social Semantic Desktop*, which connects the individual desktops. This will require open-world semantics (in between desktops) with local closed-world semantics (on the personal desktop). Thus the desktop needs to be able to handle external data with open-world semantics. Therefore we require a scenario where we can always distinguish between data per se and the semantics or assumptions on that data. If these are handled analogously, the semantic desktop, a closed-world in theory, will also be able to handle data with open-world semantics.

Handling of multiple models: In order to adequately represent the social dimension of distributed knowledge generation and usage [37], a module concept is desirable which supports encapsulation of statements and the possibility to refer to such modules. The social aspect requires a support for provenance and trust information, when it comes to importing and exporting data. With the present RDF model, importing external RDF data from another desktop presents some difficulties, mainly revolving around the fact that there are no standard means of retaining provenance information of imported data. This means that data is propagated over multiple desktops, with no information regarding the original provider and other crucial information like the context under which that data is valid. This can result in various situations like ending up with outdated RDF data with no means to update it, as well as redundant RDF data which cannot be entirely and safely removed.

Multiple semantics: As stated before, the aspect of distributed (and independently created) information requires the support of the open-world assumption (as we have it in OWL and RDF/S), whereas local information created on a single desktop will have closed-world semantics. Therefore, applications will be forced to deal with different kinds of semantics.

Multiple views: Also required by the social aspect is the support for multiple views, since different individuals on different desktops might be interested in different aspects of the data. A view is dynamic, virtual data computed or collated from the original data. The best view for a particular purpose depends on the information the user needs.

In the next section, we will briefly discuss the state of the art which served as input for the NEPOMUK Representation Language (NRL, [33,34])[15]. Sec. 4.3 gives an overview of our approach. The following sections elaborate on two important aspects of NRL, the *Named Graphs* for handling multiple models (Sec. 4.4) and the *Graph Views* for imposing different semantics on and application-oriented tailoring of models (Sec. 4.5). In Sec. 4.6, we present an example which shows how the concepts presented in this paper can be applied. Sec. 6 summarizes the NRL approach and discusses next steps.

4.2 State of the Art

The Resource Description Framework [17] and the associated schema language RDFS [5] set a standard for the *Semantic Web*, providing a representational language whereby resources on the web can be mapped to designated classes of objects in some shared knowledge domain, and subsequently described and related through applicable object properties. With the gradual acceptance of the *Semantic Web* as an achievable rather than just an ideal World Wide Web scenario, and adoption of RDF/S as the standard for describing and manipulating semantic web data, there have been many attempts to improve some RDF/S shortcomings to handling such data. Most where in the form of representational languages that extend RDF/S, the most notable of which is OWL [2]. Other work attempted to provide further functionalities on top of semantic data to that provided by RDF/S by revising the RDF model itself. The most successful idea perhaps is the named graph paradigm, where identifying multiple RDF graphs and naming them with distinct URIs is believed to provide useful additional functionality on top of the RDF model. Given that named graphs are manageable sets of data in an otherwise structureless RDF triple space composed of all existent RDF data, most of the practical problems arising from dealing with RDF data, like dealing with invalid or outdated data as well as issues of provenance and trust, could be addressed more easily if the RDF model supports named graphs. The RDF recommendation itself does not provide suitable mechanisms for talking about graphs or define relations between graphs [3,17,5,14]. Although the extension of the RDF model with named graph support has been proposed [7,32,19], and the motivation and ideas are clearly stated, a concrete extension to the RDF model supporting named graph has not yet materialized. So far, a basic syntax and semantics that models minimal manipulation of named graphs

[15] Full specifications available at `http://www.semanticdesktop.org/ontologies/nrl/`

has been presented by participants of the Semantic Web Interest Group.[16] Their intent is to introduce the technology to the W3C process once initial versions are finalized. The SPARQL query language [19], currently undergoing standardization by the W3C, is the most successful attempt to provide a standard query language for RDF data. SPARQL's full support for named graphs has encouraged further research in the area. The concept of modularized RDF knowledge bases (in the spirit of named graphs) plus views that can be used to realize the semantics of a module (with the help of rules), amongst other things, has been introduced in the *Semantic Web* rule language TRIPLE [32]. Recently, [30] introduced the concept of Networked Graphs, which are a declarative mechanism to define views over distributed RDF graphs with the help of SPARQL rules.

Since the existing approaches are incomplete wrt. the needs of NEPOMUK and most *Semantic Web* scenarios in general, we propose a combination of named graphs and TRIPLE's view concept as the basis for NRL, the representational language we are presenting. In contrast to TRIPLE, we will add the ability to define views as an extension of RDF and named graphs at the ontological level, thus we are not dependent on a specific rule formalism as in the case of TRIPLE.

In the rest of the NRL section, we will give a detailed description of the named graphs and views features of NRL. Other features of NRL (which consist of some RDFS extensions mainly inspired by Protégé and OWL) will not be discussed.

4.3 Knowledge Representation on the Social Semantic Desktop: The NRL Approach

NRL was inspired by the need for a robust representational language for the *Social Semantic Desktop*, that targets the shortcomings of RDF/S. NRL was designed to fulfill requirements for the NEPOMUK *Social Semantic Desktop* project,[17] hence the particular naming, but it is otherwise domain-independent.

As discussed in the previous section, the most notable shortcoming of the RDF model is the lack of support for handling multiple models. In theory Named Graphs solve this problem since they are identifiable, modularized sets of data. Through this intermediate layer handling RDF data, *e. g.*, exchanging data and keeping track of data provenance information, is much more manageable. This has a great influence in the social aspect of the *Social Semantic Desktop* project, since the success of this particular aspect depends largely on how to successfully deal with these issues. All data handling on the semantic desktop including storage, retrieval and exchange, will therefore be carried out through RDF graphs. Alongside provenance data, more useful information can be attached to named graphs. In particular we feel that named graphs should be distinguished by their roles, *e. g.*, Ontology or Instance Base.

Desktop users may be interested in different aspects of data in a named graph at different times. Looking at the contents of an image folder for instance, the user might wish to see related concepts for an image, or any other files related to

[16] `http://www.w3.org/2004/03/trix/`

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

it, but not necessarily both concurrently even if the information is stored in the same graph. Additionally, advanced users might require to see data that is not usually visible to regular users, like additional indirect concepts related to the file. This would require the viewing application to realize the RDF/S semantics over the data to yield more results. The desktop system is therefore required to work with extended or restricted versions of named graphs in different situations. However, we believe that such manipulations over named graphs should not have a permanent impact on the data in question. Conversely, we believe that the original named graph should be independent of any kind of workable interpretation executed by an application, which can be discarded if and when they are no longer needed.

For this reason, we present the concept of Graph Views as one of the core concepts in NRL. By allowing for arbitrary tailored interpretations for any established named graph, graph views fulfill our idea that named graphs should not innately carry any realized semantics or assumptions, unless they are themselves views on other graphs for exactly that purpose, and that they should remain unchanged and independent of any view applied on them. This means that different semantics can be realized for different graphs if required. In practice, different application on the semantic desktop will require to apply different semantics, or assumptions on semantics, to named graphs. In this way, although the semantic desktop operates in a closed-world, it is also possible to work with open-world semantic views over a graph. Importing a named graph with predefined open-world semantics on the semantic desktop is therefore possible. If required (and

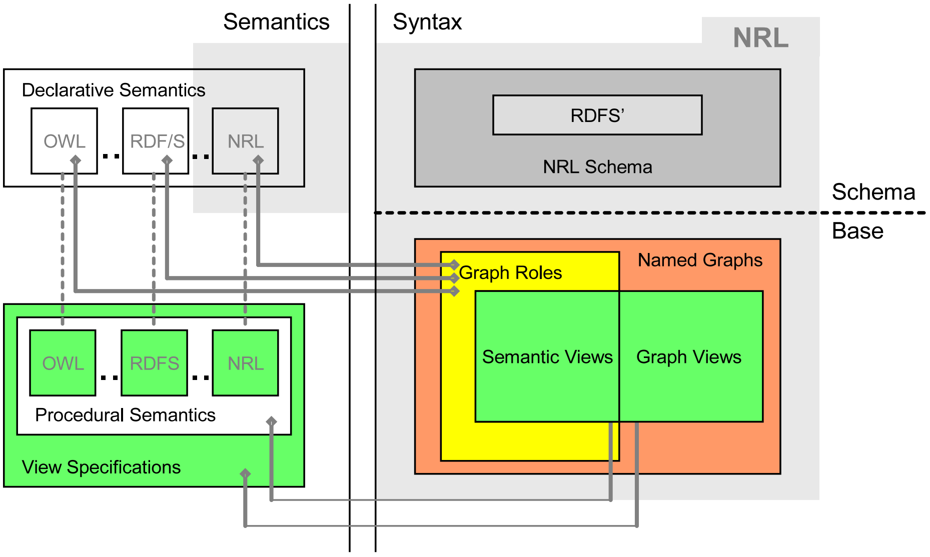

Fig. 3. Overview of NRL—Abstract Syntax, Concepts and Semantics

meaningful), closed-world applications can then work with a closed-world semantics view over the imported graph.

Fig. 3 gives an overview of the components of NRL, depicting both the syntactical and the semantic blocks of NRL. The syntax box contains, in the upper part, the NRL Schema language, which is mainly an extension of (a large subset of) RDFS. The lower part shows how named graphs, graph roles, and views are related, which will be explained in detail in the rest of this paper.

The left half of the figure sheds some light on the semantics of NRL, which has a declarative and a procedural part. Declarative semantics is linked with graph roles, *i. e.*, roles are used to assign meaning to named graphs (note that not all named graphs or views must be assigned some declarative semantics, *e. g.*, in cases when the semantics is (not) yet known or simply not relevant). Views are also linked to view specifications, which function as a mechanism to express procedural semantics, *e. g.*, by using a rule system. The procedural semantics has, of course, to realize the declarative semantics that is assigned to a semantic view.

4.4 Handling Multiple Models: NRL Named Graphs

Named graphs (NGs) are an extension on top of RDF, where every distinct RDF graph is identified by a unique name. NGs provide additional functionality on top of RDF particularly with respect to metametadata (metadata about metadata), provenance, and data (in)equivalence issues, besides making data handling more manageable. Our approach is based on the work described in [7] excluding however, the open-world assumption stated there. As stated earlier (*cf.* Sec. 4.3) we believe that named graphs should not innately carry any realized semantics or assumptions on the semantics. Therefore, despite being designed as a requirement for the Semantic Desktop, which operates under a closed-world scenario, NRL itself does not impose closed-world semantics on data. This and other semantics can instead be realized through designated views on graphs.

A named graph is a pair (n, g), where n is a unique URI reference denoting the assigned name for the graph g. Such a mapping fixes the graph g corresponding to n in a rigid, non-extensible way. The URI representing n can then be used from any location to refer to the corresponding set of triples belonging to the graph g. A graph g' consistent[18] with a distinct graph g named n cannot be assigned the same name n.

An RDF triple can exist in a named graph or outside any named graph. However, for consistency reasons, all triples must be assigned to some named graph. For this reason NRL provides a special named graph, `nrl:DefaultGraph`. Triples existing outside any named graph are considered part of this default graph. This ensures backward compatibility with triples that are not based on named graphs. This approach gives rise to the term RDF Dataset as defined in [19]. An RDF dataset is composed of a default graph and a finite number of distinct named

[18] Two different datasets asserting two unique graphs but having the same URI for a name contradict one another.

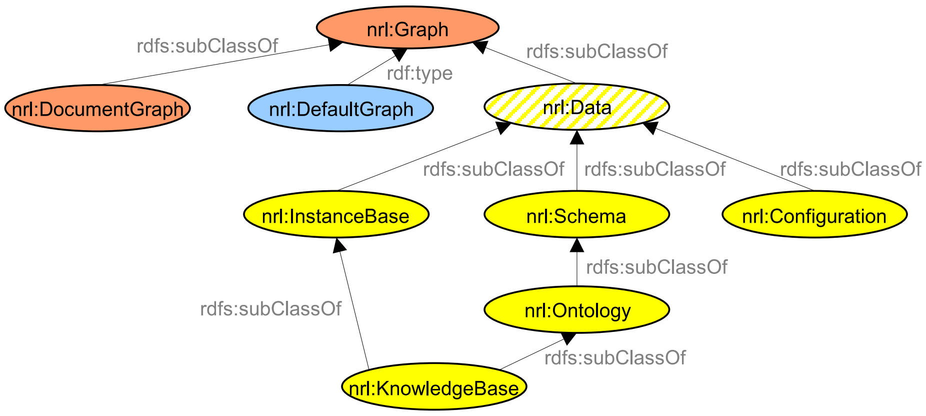

Fig. 4. NRL Named Graph Class Hierarchy

graph, formally defined as the set $\{g, (n_1, g_1), (n_2, g_2), ..., (n_n, g_n)\}$ comprising of the default graph g and zero or more named graphs (n_i, g_i).

NRL distinguishes between graphs and graph roles, in order to have orthogonal modeling primitives for defining graphs and for specifying their role. A graph role refers to the characteristics and content of a named graph (*e. g.*, simple data, an ontology, a knowledge base, *etc.*) and how the data is intended to be handled. NRL provides basic Graph Metadata Vocabulary for annotating graph roles, which vocabulary is extended in the Nepomuk Annotation Ontology (NAO)[19]. Graph metadata is attached to roles rather than to the graphs themselves, because its more intuitive to annotate an ontology, for example, rather than the underlying graph. Roles are more stable than the graphs they represent, and while the graph for a particular role might change constantly, evolution of the role itself is less frequent. An instantiation of a role represents specific type of graph and the corresponding triple set data.

Fig. 4 depicts the class hierarchy supporting NGs in NRL. Graph roles are defined as specialization of the general graph representation `nrl:Data`. A special graph, `nrl:DocumentGraph`, is used as a marker class for graphs that are represented within and identified by a document URL. We now present the NRL vocabulary supporting named graphs. General graph vocabulary is defined in Sec. 4.4 while Sec. 4.4 is dedicated entirely to graph roles.

Graph Core Vocabulary

nrl:Graph and nrl:DocumentGraph. Instances of these classes represent named graphs. The name of the instance coincides with the name of the graph. The graph content for a `nrl:DocumentGraph` is located at the URL that is the URIref for the `nrl:DocumentGraph` instance. This allows existing

[19] `http://www.semanticdesktop.org/ontologies/2007/08/15/nao`

RDF files to be re-used as named graphs, avoiding the need of a syntax like TriG[20] to define named graphs.

nrl:subGraphOf, nrl:superGraphOf, and nrl:equivalentGraph. These relations between named graphs have the obvious semantics: they are defined as $\subseteq$, $\supseteq$, and $=$ on the bare triple sets in these graphs.

nrl:imports is a subproperty of `nrl:superGraphOf` and models graph imports. Apart from implying the $\supseteq$ relation between the triple sets, it also requires that the semantics of the two graphs is compatible if used on, *e. g.*, graphs that are ontologies.

nrl:DefaultGraph. This instance of `nrl:Graph` represents the graph containing all triples existing outside any user-defined named graph. Since we do not apply any semantics to triples automatically, this allows views to be defined on top of triples defined outside of all named graphs analogously to the named-graph case.

Graph Roles Vocabulary

nrl:Data. This subclass of `nrl:Graph` is an abstract class to make graph roles easy-to-use marker classes. It represents the most generic role that a graph can have, namely that it contains data.

nrl:Schema and nrl:Ontology are roles for graphs that represent data in some kind of conceptualization model. `nrl:Ontology` is a subclass of `nrl:Schema`.

nrl:InstanceBase marks a named graph to contain instances from schemas or ontologies. The properties `nrl:hasSchema` and `nrl:hasOntology` relate an instance base to the corresponding schema or ontology.

nrl:KnowledgeBase marks a named graph as containing a conceptual model plus instances from schemas or ontologies.

nrl:GraphMetadata is used to mark graphs whose sole purpose is to store metadata about other graphs. Data about a graph (Graph Metadata) is thus stored in a corresponding graph having this role. The property `nrl:graphMetadataFor` binds a metadata graph to the graph being annotated. Although a graph can have multiple metadata graphs describing it, there can only be one unique metadata graph which defines the graph's important core properties, e.g. whether it is updatable (through `nrl:updatable`) or otherwise. NRL provides the `nrl:coreGraphMetadataFor` property for this purpose, as a subproperty of `nrl:graphMetadataFor`, to identify the core metadata graph for a graph.

nrl:Configuration is used to represent technical configuration data that is irrelevant to general semantic web data within a graph. Other additional roles serving different purposes might be added in the future.

nrl:Semantics. Declarative semantics for a graph role can be specified by referring to instances of this class via `nrl:hasSemantics`. These will usually link (via `nrl:semanticsDefinedBy`) to a document specifying the semantics in a human readable or formal way (*e. g.*, the RDF Semantics document [14]).

[20] `http://sites.wiwiss.fu-berlin.de/suhl/bizer/TriG/`

4.5 Imposing Semantics on Graphs: NRL Graph Views

A named graph consists only of the enumerated triples in the triple set associated with the name, and does not inherently carry any form of semantics (apart from the basic RDF semantics). However in many situations it is desirable to work with an extended or restricted interpretation of simple syntax-only named graphs. These can be realized by applying some algorithm (*e. g.*, specified through rules) which enhances named graphs with entailment triples, returns a restricted form of the triple set, or an entirely new triple set. To preserve the integrity of a named graph, interpretations of one named graph should never replace the original. To model this functionality and retain the separation between original named graph and any number of their interpretations, we introduce the concept of *Graph Views*.

Views are different interpretations for a particular named graph. Formally, a view is an executable specification of an input graph into a corresponding output graph. Informally, they can be seen as arbitrary wrappings for a named graph. Fig. 5 depicts graph view support in NRL. Views are themselves named graphs. Therefore one can have a named graph that is a different interpretation, or view, of another named graph. This modeling can be applied recurrently, yielding a view of a view and so on.

View specifications can execute the view realization for a view, via a set of queries/rules in a query/rule language (*e. g.*, a SPARQL query over a named graph[21]), or via an external application (*e. g.*, an application that returns the transitive closure of `rdfs:subClassOf`). As in the latter example, view realizations can also realize the implicit semantics of a graph according to some language or schema (*e. g.*, RDFS, OWL, NRL *etc.*). We refer to these as *Semantic Views*, represented in Fig. 5 by the intersection of `nrl:GraphView` and graph roles. One can draw a parallel between this figure and Fig. 3. In contrast to graph roles, which have only declarative semantics defined through the `nrl:hasSemantics` property, semantic views also carry procedural semantics, since the semantics of these graphs are always realized, (through `nrl:realizes`) and not simply implied.

Views Vocabulary. In this section we briefly present the NRL vocabulary supporting graph view specifications.

nrl:GraphView represents a view, modeled as a subclass of named graph. A view is realized through a view specification, defined by an instance of `nrl:ViewSpecification` via `nrl:hasSpecification`. The named graph on

[21] A way for using SPARQL to realize view definitions (called Networked Graphs) has been described in [30]. While Networked Graphs allow views to be defined in a declarative way (in contrast to NRL's somewhat procedural way), they lack many of the features we think are important for a view language, *e. g.*, they do do not allow access to the underlying RDF graphs without any interpretation, and they only allow views to be defined via SPARQL which excludes languages with more advanced semantics like OWL and also languages that do not have a declarative semantics.

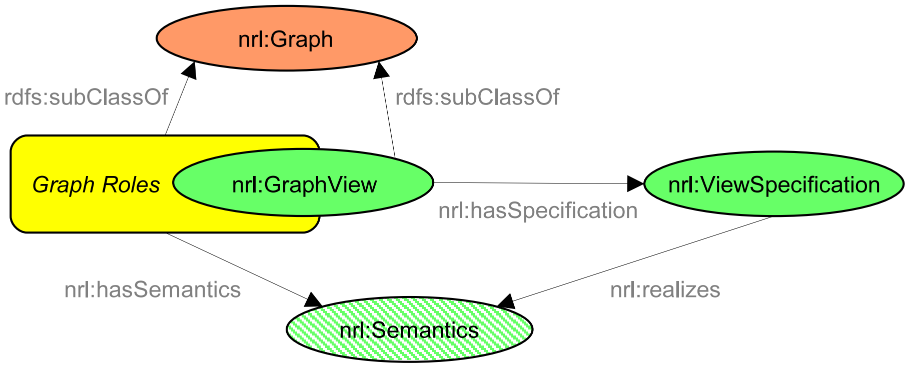

Fig. 5. Graph Views in NRL

which the view is being generated is linked by `nrl:viewOn`. The separation between different interpretations of a named graph and the original named graph itself is thus retained.

nrl:ViewSpecification. This class represents a general view specification, which can currently take one of two forms, modeled as the two subclasses `nrl:RuleViewSpecification` and `nrl:ExternalViewSpecification`. As discussed earlier, semantic views realize procedural semantics and are linked to some semantics via `nrl:realizes`. This is however to be differentiated from `nrl:hasSemantics`, which states that a named graph carries (through a role) declarative semantics which is not necessarily (explicitly) realized via a view specification.

nrl:RuleViewSpecification. Views can be specified by referring to a rule language (via `nrl:ruleLanguage`) and a corresponding set of given rules (via `nrl:rule`). These views are realized by executing the rules, generating the required output named graph.

nrl:ExternalViewSpecification. Instances of this class map to the location of (via `nrl:externalRealizer`) an external application, service, or program that is executed to create the view.

4.6 Example: NRL in Use

In this section, we demonstrate the utilization of the various NRL concepts in a more complex scenario: Ella is a biologist and works as a senior researcher at Institute Pasteur in central Paris. She would like to compile an online knowledge base describing animal species for her students to access. She knows that a rather generic ontology describing the animal species domain, O_1, is already available (which, technically speaking, means it exists as a named graph). Someone else had also supplied data consisting of a vast amount of instances for the animals ontology as a named graph with the role of instance base, I_1. However this combined data does not provide extensive coverage of the animal kingdom as required by Ella. Therefore Ella hires a SW knowledge engineer to model another ontology that defines further species not captured in O_1, and this is stored as another named graph, O_2. Since Ella requires concepts from both ontologies, the engineer merges O_1 and O_2 in the required conceptualization by creating a named graph O as an ontology and defining it as supergraph of O_1 and O_2. Furthermore, a number of real instances of the new animal species defined in O_2 is compiled in an instance base, I_2.

Ella now requires to use all the acquired and generated data to power a useful service for the students to use. Schematic data from the graph O, and the instances from I_1 and I_2 are all imported to a new graph, KB, acting as a knowledge base. Ella would like the students to be able to query the knowledge base with questions like 'Are flatworms Deuterostomes or Platyzoa?'. Although by traversing the animals hierarchy it is clear that they are Platyzoa, the statement is not innately part of the graph KB. This can be discovered by realizing the semantics of `rdfs:subClassOf` as defined in the RDFS semantics. However KB might be required as is, with no assumed semantics, for other purposes. Directly enriching KB with entailment triples permanently would make this impossible.

Therefore the knowledge engineer creates a view over KB for Ella, consisting of the required extended graph, without modifying the original KB in any way. This is done by defining a view specification that computes the procedural semantics for KB. The specification uses a rule language of choice that provides a number of rules, one of which computes the transitive closure of `rdfs:subClassOf` for a set of RDF triples. Executing that rule over the triples in KB results in the semantic view $V_1(KB)$, which consists of the RDF triples in KB plus the generated entailment triples. The separation between the underlying model and the model with the required semantics is thus retained and through simple queries over $V_1(KB)$, students can instantly get answers to their questions.

Ella later on decides to provide another service for younger students by using 'Graph Taxonomy Extractor', a graph visualization API that generates an interactive graph depicting the animal hierarchy within $V_1(KB)$. However this graph contains other information in addition to that required (*e. g.*, properties attributed to classes). Of course, Ella does not want to discard all this useful information from $V_1(KB)$ permanently just to generate the visualization. The knowledge engineer is aware of a *Semantic Web* application that does exactly what Ella requires. The application acts as an external view specification and generates a view, consisting of only triples defining the class hierarchy, over an input named graph. The view generated by this application, $V_2(V_1(KB))$, is fed to the API to effectively generate the interactive graph for the students to explore.

It is worth to note that all seven named graphs on which this last view is generated upon are still intact and have not been affected by any of the operations along the way. If the knowledge engineer requires to apply some different semantics over KB, it may still be done since generating $V_1(KB)$ did not have an impact on KB. However, the content of KB needs to be validated, or generated, each time it is used since one of its subgraphs (O_1, O_2, I_1 and I_2) can change. Although from a practical point of view this might sound laborious, from a conceptual point of view it solves problems regarding data consistency and avoids other problems like working with outdated data that can't be updated because links to underlying models have been lost.

Fig. 6 presents the "dataflow" in our example scenario, demonstrating how the theoretical basis of NRL can be applied in practice to effectively model data for use in different scenarios in a clear and consistent way.

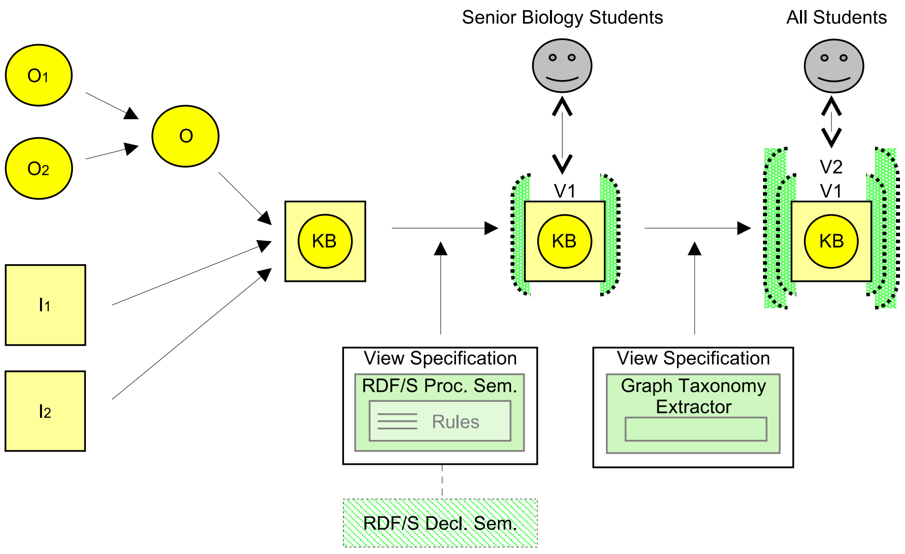

Fig. 6. NRL Dataflow Diagram

We now model the dataflow in Fig. 6 in TriG syntax.[22] TriG is a straightforward extension of Turtle.[23] Turtle itself is an extension of N-Triples[24] which carefully takes the most useful and appropriate things added from Notation3[25] while keeping it in the RDF model. TriG is a plain text format created for serializing NGs and RDF Datasets. Fig. 7 demonstrates how one can make use of the named graph paradigm and the syntax for named graphs:

- `[1]` namespace declarations
- `[2-5]` ontology graphs (`ex:o1` and `ex:o2` are defined and then imported into `ex:o`)
- `[6-8]` instance/knowledge base definitions
- `[9]` contents of ontology `ex:o2`, defining extended animal domain

[22] `http://sites.wiwiss.fu-berlin.de/suhl/bizer/TriG/`
[23] `http://www.dajobe.org/2004/01/turtle/`
[24] `http://www.w3.org/TR/rdf-testcases/#ntriples`
[25] `http://www.w3.org/DesignIssues/Notation3`

```
[1] @prefix nrl: <http://semanticdesktop.org/ontology/nrl-yyyymmdd#> .
    @prefix ex: <http://www.example.org/vocabulary#> .
[2] ex:o2 rdf:type nrl:Ontology .
[3] <http://www.domain.com/o1.rdfs> rdf:type nrl:Ontology ,
     nrl:DocumentGraph .
[4] ex:o1 rdf:type nrl:Ontology ;
         nrl:equivalentGraph <http://www.domain.com/o1.rdfs> .
[5] ex:o rdf:type nrl:Ontology ;
         nrl:imports ex:o1, ex:o2 .
[6] ex:i2 rdf:type nrl:InstanceBase ;
         nrl:hasOntology ex:o2 .
[7] http://www.anotherdomain.com/i1.rdf> rdf:type nrl:InstanceBase,
                                                  nrl:DocumentGraph .
[8] ex:kb rdf:type nrl:KnowledgeBase ;
      nrl:imports ex:o, ex:i2, <http://www.anotherdomain.com/i1.rdf> .
[9] ex:o2 {
      ex:Animal rdf:type rdfs:Class .
             ## further Animal Ontology definitions here ## }
[10]ex:i2 {
     ex:CandyCaneWorm rdf:type ex:Flatworm ;
             ## further Animal Instance definitions here ## }
[11] ex:v1kb rdf:type nrl:KnowledgeBase, nrl:GraphView ;
            nrl:viewOn ex:kb ; nrl:superGraphOf ex:kb ;
            nrl:hasSpecification ex:rvs .
[12] ex:rvs rdf:type nrl:RuleViewSpecification ;
            nrl:realizes ex:RDFSSemantics ; nrl:ruleLanguage "SPARQL" ;
            nrl:rule "CONSTRUCT {?s rdfs:subClassOf ?v} WHERE ..." ;
            nrl:rule "CONSTRUCT {?s rdf:type ?v} WHERE ..." .
[13] ex:RDFSSemantics rdf:type nrl:Semantics ; rdfs:label "RDFS" ;
            nrl:semanticsDefinedBy "http://www.w3.org/TR/rdf-mt/" .
[14] ex:v2v1kb rdf:type nrl:GraphView, nrl:KnowledgeBase ;
            nrl:viewOn ex:v1kb ; nrl:hasSpecification ex:evs .
[15] ex:evs rdf:type nrl:ExternalViewSpecification ;
            nrl:externalRealizer "GraphTaxonomyExtractor" .
```

Fig. 7. NRL Example—TriG Serialization

`[10]` contents of instance base `ex:i2`, defining instances of animals in (`ex:o2`

`[11-13]` `ex:v1kb` is defined as a view on `ex:kb` via the view specification `ex:rvs`; furthermore, `ex:v1kb` is a super graph of `ex:kb` as it realizes the RDFS semantics and thus contains the original graph plus the inferred triples; the view specification is realized (as an example) with some SPARQL-inspired `CONSTRUCT` queries (for this to work, a real rule language is required)

`[14-15]` similar to `[11-13]`, but here we define `ex:v2v1kb` with the help of an external tool, the "GraphTaxonomyExtractor"

5 NEPOMUK Ontologies

Being the representational language, NRL (see Sect. 4) serves as the language required to define the vocabulary with which other lower-level Nepomuk ontologies are represented. All these ontologies can be understood as being instances of NRL, which conceptually is to be found at the representational layer of the Nepomuk ontologies. As such we differentiated between three main ontology layers, as depicted in the Ontologies Pyramid Fig. 8, in order of decreasing generality, abstraction and stability:

1. Representational Layer
2. Upper Level Layer
3. Lower Level Ontologies

Other examples of vocabularies/schemas in the uppermost layer are RDF/S and OWL. Whereas the representational ontologies include abstract high-level classes and properties, constraints, *etc.*; upper-level ontologies provide a framework by which disparate systems may utilize a common knowledge base and from which more domain-specific ontologies may be derived. They are high-level, domain-independent ontologies, characterized by their representation of common sense concepts, *i. e.*, those that are basic for human understanding of the world. Concepts expressed in upper-level ontologies are intended to be basic and universal to ensure generality and expressivity for a wide area of domains. In turn, lower level ontologies (which are further layered into group and personal ontologies) are domain-specific and provide more concrete representations of abstract concepts found in the upper ontologies.

After discussing NRL in detail, we now provide an overview of the engineered ontologies in the upper-level ontology layer. Lower-level ontologies have not been designed by the Nepomuk ontologies task force, but by groups or individuals developing domain-specific applications for the *Social Semantic Desktop*. NRL, together with the upper-level ontologies discussed hereunder, form a central pillar in the *Social Semantic Desktop* system, as they are used to model the environment and domain of the applications. All the described ontologies have been published[26] and although a few of them are open for future adjustments, they are considered to be stable. The modularization of these ontologies in itself was a challenge, especially given the layered approach described earlier. Even though they are all upper-level ontologies, some of them require concepts and/or relationships from others, and therefore dependency relationships exist also within the same level. The design of the upper-level ontologies sought to:

– Represent common desktop entities (objects, people, *etc.*)
– Represent trivial relationships between these entities, as perceived by the desktop user
– Represent a user's mental model, consisting of entities and their relationships on their desktop

[26] `http://www.semanticdesktop.org/ontologies/`

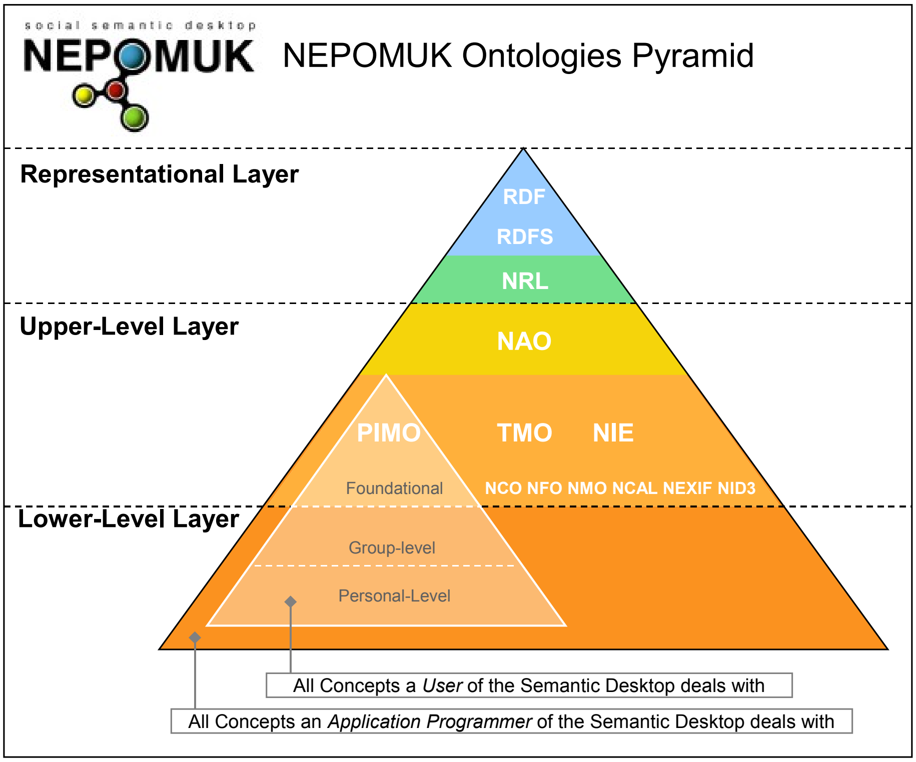
social semantic desktop
NEPOMUK
NEPOMUK Ontologies Pyramid
Representational Layer
RDF
RDFS
NRL
Upper-Level Layer
NAO
PIMO
TMO
NIE
Foundational
NCO NFO NMO NCAL NEXIF NID3
Lower-Level Layer
Group-level
Personal-Level
All Concepts a *User* of the Semantic Desktop deals with
All Concepts an *Application Programmer* of the Semantic Desktop deals with

Fig. 8. The Layered Nepomuk Ontologies Pyramid

Whereas the representation of high-level concepts like 'user', 'contact', 'desktop', 'file' was fairly straightforward, we also needed to leverage existing information sources in order to make them accessible to semantic applications on the *Social Semantic Desktop*. This information is contained within various structures maintained by the operating system and a multitude of existing 'legacy' applications. These structures include specific kinds of entities like messages, documents, pictures, calendar entries and contacts in address books. van Elst coined the term 'native structures' to describe them and 'native resources' for the pieces of information they contain [25].

A multitude of relationships exist between entities on the desktop. Although in the user's mind the majority of these relationships is considered trivial (files belonging to the same folder, objects related to the same topic, file copies, professional and social contacts), they remain implicit and undefined. Exposing these relationships to semantic applications is the key to making the desktop truly semantic. The majority of the most basic relationships can be mined through the user's actions—organizing files in folders, rating desktop objects, saving files from a contact into specific folders, *etc.*Applications on the desktop will also provide means for enabling the user to define less trivial relationships (*e. g.*, through semantic annotation, tagging). During the design of the upper-level ontologies; all kinds of relationships—from the most trivial to ones that are more specific, were taken into account. The user's mental models of how information is stored and organized on their desktop are based on these entities and their relationships—their desktop, files on their desktop, folder structures, contacts who share data, *etc.*The design of a conceptual representation of personal information models was based on the way the users are used to express their knowledge, such that the concepts and relationships in the ontologies reflect the world as seen by the desktop users. This eases the process of knowledge acquisition from the user's desktop structures and activities to their machine-processable representation.

After providing our motivation, in the remaining subsections we will provide an overview of each of the ontologies in the upper-level ontology pyramid layer. The uppermost ontology in the upper-level layer (Fig. 8) is the Nepomuk Annotation Ontology (NAO) [29] which defines trivial relationships between desktop entities as conceived by the user. The Personal Information Model Ontology (PIMO) [26] can be used to express personal information models of individuals, whereas the Task Management Ontology (TMO) [6] is used to describe personal tasks of individuals. The Nepomuk Information Element set of ontologies (NIE) [18] defines common information elements that are to be found on the desktop, together with a number of more specific ontologies whose aim is to represent legacy data in its various forms.

5.1 Nepomuk Annotation Ontology (NAO)

The meaning of the term annotation is highly contextual. Depending on the context, anything can be considered as annotation within a data set (or a named graph). On the SSD, the average user is frequently seen creating representations of objects on their desktop, while the more experienced user is also frequently creating representations of concepts and their relationships. Within this context, we consider annotation to be anything that goes further than creating resources and defining their elementary relationships. A user can create an instance of a 'Person', and provide values for all the elementary properties that an instance of 'Person' can have. The user can then go one step ahead and annotate the resources with more information, of a textual (*e. g.*, custom human-readable descriptions) or non-textual (*e. g.*, links to related resources) nature. In a typical scenario there may be a number of domain-centric properties for the classes 'Person' (*e. g.*, name, address, knows *etc.*) and 'Document' (*e. g.*, author, title, *etc.*). Via vocabulary in the annotation ontology the user can provide personalized, user-friendly labels and descriptions for a resource, as well as additional information like tags and ratings. Generic relationships exist between resources across multiple domains, and making these relationships explicit would be of great benefit for the user. For example, a user may want to state that a 'Document' is about some instance of 'Person'. However this shallow kind of relationship exists between other concepts in other domains. Vocabulary that is able to express these generic relationships are provided by the annotation ontology.

Although this information is optional and does not reflect the elementary nature of a ‘Document’, it contributes to improved data unification and information retrieval via user search.

Graph Metadata is a particular form of annotation, where instead of annotating general resources, one annotates instances of named graphs, *e. g.*, to define the type of graph role. The major difference is that while generic annotation can be stored within any graph the user is working with (*e. g.*, the graph where the annotated resource is defined), metadata about a graph should always be stored outside that graph, in a separate special named graph that is aptly represented in NRL by the nrl:GraphMetadata role. Detailed specifications for this ontology are available in full online [29].

5.2 Nepomuk Information Element (NIE)

The abbreviation NIE may refer to the NIE Ontology Framework as a whole or to the NIE Core Ontology. The motivation for NIE lies in representing the multitude of applications and data formats. Previous semantic desktop projects (*e. g.*, Haystack Haystack [20] or Gnowsis [23]) had to develop their solutions. Some attempts at standardization have been made (*e. g.*, Adobe XMP[27], Freedesktop.org XESAM[28]) but a definite standard had not emerged before NIE’s conception. Apart from large metadata description frameworks there exists a considerable number of smaller single-purpose ontologies aimed at specific types of resources (*e. g.*, ICAL[29] or VCARD[30]). A broad array of utilities has been developed for extracting RDF metadata from desktop sources[31].

Various problems have been identified with the pre-existing vocabularies. They are expressed in many languages and the level of detail often leaves much to be desired. The NIE Framework is an attempt to build upon that experience, to provide unified vocabulary to describe typical native resources that may be of interest to the user. These resources are intended to serve as raw data for other semantic applications. They can be browsed, searched, annotated and linked with each other. Data represented in NIE has three roles. First, NIE data is intended to be generated by an extraction process. Second, RDF-based systems can create NIE structures natively, without building on existing applications. Third, data expressed in NIE can be imported back to native applications. Thus, the resulting ontologies serve as a mediator between semantic and native applications. The full specifications for this ontology can be accessed online [18].

5.3 Personal Information Model Ontology (PIMO)

The scope of PIMO is to model data that is within the attention of the user and needed for knowledge work or private use. The focus is on data that is accessed

[27] `http://www.adobe.com/products/xmp/`
[28] `http://xesam.org/main/XesamAbout`
[29] `http://www.ietf.org/rfc/rfc2445.txt`
[30] `http://www.ietf.org/rfc/rfc2426.txt`
[31] `http://simile.mit.edu/wiki/RDFizers`

through a *Semantic Desktop* or other personalized *Semantic Web* applications. We call this the Personal Knowledge Workspace [15] or Personal Space of Information [16], embracing all data needed by an individual to perform knowledge work. Today, such data is typically stored in files, in Personal Information Management or in groupware systems. A user has to cope with different formats of data, such as text documents, contact information, e-mails, appointments, task lists, project plans, or an Enterprise Resource Planning system. Existing information that is already stored in information systems is in the scope of PIMO, but abstract concepts can also be represented, if needed. PIMO is based on the idea that users have a mental model to categorize their environment. Each concept in the environment of the user is represented as a Thing in the model, and mapped to documents and other entities that mention the concept. Things can be described via their relations to other Things or by literal RDF properties.

In PIMO, Things are connected to their equivalent resources using directed relations. The design rationale was to keep the PIMO ontology itself, as well as the data needed to create a PIMO for a user as minimal as possible. Inside a user's PIMO, duplication is avoided. PIMO builds on NRL, NIE and NAO. By addressing all key issues: precise representation, easy adoption, easy to understand by users, extensibility, interoperability, reuse of existing ontologies and data integration; PIMO provides a framework for creating personal information management applications and ontologies. The detailed specifications for this ontology are available online [26].

5.4 Task Model Ontology (TMO)

The TMO is a conceptual representation of tasks for use in personal task management applications for the knowledge worker (KWer). It represents an agreed, domain-specific information model for tasks and covers personal task management use cases. As a domain model the TMO models the tasks a KWer deals with in the context of the KWer's other personal information. It thereby represents an activity-centric view on the KWer's personal information, as it models underlying tasks as well the relations to other personal information that is relevant to that task. The KWer regards all personal information as a single body of information [21], a personal information cloud including tasks. Information-wise, the use cases focus on an individual KWer's personal tasks and further related personal information. The full specifications for this ontology are available online [6] and they presents the state-of-the art in task models and the semi-formal description of the ontology with links to the supported use cases.

6 Summary and Outlook

The *Social Semantic Desktop* as presented in this material provides a universal platform for:

– Personal Information Management
– Distributed Information Management
– Social Expansion and Community Creation

In order to operate, the *Social Semantic Desktop* requires metadata, which can be extracted by:

- i) Lifting of existing structured data onto RDF
- ii) Usage of Human Language Technology (HLT) to capture knowledge from text and transform that into RDF
- ii) Manual creation of metadata by linking, annotation or tagging

In order to soften the border between the *Semantic Web* and the *Social Semantic Desktop*, we have applied *Semantic Web* knowledge representation for the desktop. Aligning knowledge representation on a *Social Semantic Desktop* with the general *Semantic Web* approaches (RDF, RDFS, OWL, *etc.*) promises a comprehensive use of data and schemas and an active, personalized access point to the *Semantic Web* [24]. In such a scenario, ontologies play an important role, from very general ontologies stating which entities can be modeled on a *Semantic Desktop* (*e. g.*, people, documents, *etc.*) to rather personal vocabulary structuring information items. One of the most important design decisions is the question of the *representational ontology*, constraining the general expressivity of such a system. In this paper, we concentrated on those parts of the NEPOMUK Representational Language (NRL) which are rooted in the requirements which arose by the *distributed knowledge representation and heterogeneity aspects* of the *Semantic Desktop* scenario, and which we think cannot satisfactorily be dealt with by the current state of the art. In a nutshell, the basic arguments and design principles of NRL are as follows:

- Due to the heterogeneity of the data-creating and data-consuming entities in the social semantic desktop scenario, a single interpretation schema cannot be assumed. Therefore, NRL aims at a *strict separation between data (sets of triples, graphs) and their interpretation/semantics.*
- Imposing specific semantics to a graph is realized by generating *views* on that graph. Such a generation is directed by an (executable) *view specification* which may realize a declarative semantics (*e. g.*, the RDF/S or OWL semantics specified in a standardization document).
- Graph views cannot only be used for semantic interpretations of graphs, but also for application-driven tailoring of a graph.[32]
- Handling of multiple graphs (with different provenance, ownership, level of trust, *etc.*) is essential. *Named graphs* are the basic means targeting this problem.
- Graphs can play different roles in different contexts. While for one application a graph may be an ontology, another one may see it as plain data. These *roles* can explicitly be specified.

While originally designed as a NEPOMUK internal standard for the *Social Semantic Desktop*, we believe that the arguments also hold for the general *Semantic Web*. This is especially true when we review the current trends which increasingly

[32] This corresponds to a database-like view concept.

show a shift from the view of "the Semantic Web as one big, global knowledge base" to "a Web of (machine and human) actors" with local perspectives and social needs like trust, ownership, *etc.*

Within NEPOMUK, we have developed the approach technically, by complementing the NRL standard with tools that facilitate its use by the application programmer, as well as conceptually, by the development and integration of a number of accompanying ontology standards [33]; *e. g.*, the annotation vocabulary referenced earlier, an information element ontology, and an upper-ontology for *Personal Information Models.*

Acknowledgements. This work was supported by the European Union IST fund (Grant FP6-027705, Project NEPOMUK); by the German Federal Ministry of Education, Science, Research and Technology (bmb+f), (Grant 01 IW F01, Project Mymory: Situated Documents in Personal Information Spaces) and by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2). The authors would especially like to thank all contributors to NEPOMUK's ontology taskforce.

References

1. Austin, J.L.: How to do things with words. Harvard U.P., Cambridge (1962)
2. Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinnes, D., Patel-Schneider, P., Stein, L.: OWL web ontology language reference (2004)
3. Beckett, D.: RDF/XML syntax specification (revised). W3C recommendation, W3C (February 2004), http://www.w3.org/TR/rdf-syntax-grammar/
4. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American 89 (May 2001)
5. Brickley, D., Guha, R.: RDF vocabulary description language 1.0: RDF Schema. Technical report, W3C (February 2004), http://www.w3.org/TR/rdf-schema/
6. Brunzel, M., Grebner, O.: Nepomuk task model ontology specification. Technical report, NEPOMUK Consortium (2008)
7. Carroll, J.J., Bizer, C., Hayes, P., Stickler, P.: Named graphs, provenance and trust. In: WWW 2005: Proceedings of the 14th international conference on World Wide Web, pp. 613–622. ACM Press, New York (2005)
8. Cheyer, A., Park, J., Giuli, R.: Iris: Integrate. relate. infer. share. In: Decker, S., Park, J., Quan, D., Sauermann, L. (eds.) Proc. of Semantic Desktop Workshop at the ISWC, Galway, Ireland, November 6, vol. 175 (2005)
9. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: GATE: A framework and graphical development environment for robust NLP tools and applications. In: Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (2002)
10. Cunningham, H., Maynard, D., Tablan, V.: JAPE: a Java Annotation Patterns Engine (2nd edn.). Research Memorandum CS–00–10, Department of Computer Science, University of Sheffield (November 2000)

[33] http://www.semanticdesktop.org/ontologies/

11. Davis, B., Handschuh, S., Cunningham, H., Tablan, V.: Further Use of Controlled Natural Language for Semantic Annotation. In: Proceedings of the 1st Semantic Authoring and Annotation Workshop (SAAW 2006) at ISWC 2006, Athens, Georgia, USA (2006)
12. Davis, B., Iqbal, A., Funk, A., Tablan, V., Bontcheva, K., Cunningham, H., Handschuh, S.: RoundTrip Ontology Authoring. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 50–65. Springer, Heidelberg (2008)
13. Decker, S., Frank, M.: The social semantic desktop. In: Proc. of the WWW 2004 Workshop Application Design, Development and Implementation Issues in the Semantic Web (2004)
14. Hayes, P.: RDF semantics. W3C recommendation, W3C (February 2004), http://www.w3.org/TR/rdf-mt/
15. Holz, H., Maus, H., Bernardi, A., Rostanin, O.: From lightweight, proactive information delivery to business process-oriented knowledge management. Journal of Universal Knowledge Management 0(2), 101–127 (2005)
16. Jones, W.P., Teevan, J.: Personal Information Management. University of Washington Press (October 2007)
17. Manola, F., Miller, E.: RDF primer. W3C recommendation, W3C (February 2004), http://www.w3.org/TR/rdf-primer/
18. Mylka, A., Sauermann, L., Sintek, M., van Elst, L.: Nepomuk information element framework specification. Technical report, NEPOMUK Consortium (2007)
19. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF. W3C working draft, W3C (2005), http://www.w3.org/TR/rdf-sparql-query/
20. Quan, D., Huynh, D., Karger, D.R.: Haystack: A platform for authoring end user semantic web applications. In: Fensel, D., Sycara, K., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 738–753. Springer, Heidelberg (2003)
21. Ravasio, P., Tscherter, V.: Users' theories of the desktop metaphor, or why we should seek metaphor-free interfaces. In: Kaptelinin, V., Czerwinski, M. (eds.) Beyond the desktop metaphor: designing integrated digital work environments, pp. 265–294. MIT Press, Cambridge (2007)
22. Richter, J., Völkel, M., Haller, H.: DeepaMehta – A Semantic Desktop. In: Decker, S., Park, J., Quan, D., Sauermann, L. (eds.) Proc. of Semantic Desktop Workshop at the ISWC, Galway, Ireland, November 6, vol. 175 (2005)
23. Sauermann, L.: The gnowsis—using semantic web technologies to build a semantic desktop. Diploma thesis, Technical University of Vienna (2003)
24. Sauermann, L., Dengel, A., Elst, L., Lauer, A., Maus, H., Schwarz, S.: Personalization in the EPOS project. In: Bouzid, M., Henze, N. (eds.) Proceedings of the International Workshop on Semantic Web Personalization, Budva, Montenegro, June 12, pp. 42–52 (2006)
25. Sauermann, L., van Elst, L., Dengel, A.: PIMO—a framework for representing personal information models. In: Tochtermann, K., Haas, W., Kappe, F., Scharl, A., Pellegrini, T., Schaffert, S. (eds.) Proceedings of I-MEDIA 2007 and I-SEMANTICS 2007 (2007)
26. Sauermann, L., van Elst, L., Moeller, K.: Nepomuk personal information model ontology specification. Technical report, NEPOMUK Consortium (2007)
27. Scerri, S., Handschuh, S., Decker, S.: Semantic Email as a Communication Medium for the Social Semantic Desktop. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 124–138. Springer, Heidelberg (2008)

28. Scerri, S., Mencke, M., Davis, B., Handschuh, S.: Evaluating the Ontology underlying sMail - the Conceptual Framework for Semantic Email Communication. In: Proceedings of the 6th International conference of Language Resources and Evaluation (LREC), Marrakech, Morocco (2008)
29. Scerri, S., Sintek, M., van Elst, L., Handschuh, S.: Nepomuk annotation ontology specification. Technical report, NEPOMUK Consortium (2007)
30. Schenk, S., Staab, S.: Networked graphs: A declarative mechanism for SPARQL rules, SPARQL views and RDF data integration on the web. In: Proceedings of the 17th International World Wide Web Conference, Bejing, China (2008)
31. Searle, J.R.: Speech Acts. Cambridge University Press, Cambridge (1969)
32. Sintek, M., Decker, S.: *TRIPLE*–A query, inference, and transformation language for the semantic web. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, p. 364. Springer, Heidelberg (2002)
33. Sintek, M., van Elst, L., Scerri, S., Handschuh, S.: Distributed knowledge representation on the social semantic desktop: Named graphs, views and roles in NRL. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 594–608. Springer, Heidelberg (2007)
34. Sintek, M., van Elst, L., Grimnes, G., Scerri, S., Handschuh, S.: Knowledge representation for the distributed, social semantic web: Named graphs, graph roles and views in nrl. In: Cuenca-Grau, B., Honavar, V., Schlicht, A., Wolter, F. (eds.) Second International Workshop on Modular Ontologies, WoMO 2007 (2007)
35. Thai, V., Handschuh, S., Decker, S.: IVEA: An information visualization tool for personalized exploratory document collection analysis. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 139–153. Springer, Heidelberg (2008)
36. Thai, V., Handschuh, S., Decker, S.: Tight coupling of personal interests with multi-dimensional visualization for exploration and analysis of text collections. In: IV 2008: Proceedings of the 12th International Conference on Information Visualisation, pp. 221–226. IEEE Computer Society, Los Alamitos (2008)
37. van Elst, L., Dignum, V., Abecker, A.: Towards agent-mediated knowledge management. In: van Elst, L., Dignum, V., Abecker, A. (eds.) AMKM 2003. LNCS (LNAI), vol. 2926, pp. 1–31. Springer, Heidelberg (2004)