Semantic Document Architecture for Desktop Data Integration and Management

Doctoral Dissertation submitted to the
Faculty of Informatics of the *Università della Svizzera Italiana*
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

presented by
Saša Nešić

under the supervision of
Prof. Mehdi Jazayeri

November 2010

Dissertation Committee

Prof. Fabio Crestani	Università della Svizzera Italiana, Switzerland
Prof. Cesare Pautasso	Università della Svizzera Italiana, Switzerland
Prof. Dragan Gašević	Athabasca University, Canada
Prof. Klaus Tochtermann	University of Kiel, Germany

Dissertation accepted on 30 November 2010

Prof. Mehdi Jazayeri
Research Advisor
Università della Svizzera Italiana, Switzerland

Prof. Michele Lanza
PhD Program Director

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Saša Nešić
Lugano, 30 November 2010

To my parents

Abstract

Over the last decade, personal desktops have faced the problem of information overload due to increasing computational power, easy access to the Web and cheap data storage. Moreover, an increasing number of diverse end-user desktop applications have led to the problem of information fragmentation. Each desktop application has its own data, unaware of related and relevant data in other applications. In other words, personal desktops face a lack of interoperability of data managed by different applications. Recent years have also seen the rapid growth of shared data in online social network communities. Desktop users have been publishing extensively data from their personal desktops to different social-networking sites. In the current data-publishing scenario, desktop data that is published to a social network becomes completely disconnected from desktops they originate. Moreover, there is no interoperability between the same desktop data published to different social networks.

A core idea of a Social Semantic Desktop vision is to enable semantic integration and data interoperability on the personal desktop by applying Semantic Web technologies, and to connect data from personal desktops into a unified information space of social network communities. This thesis introduces a new form of documents, called *Semantic Documents*, which attempts to bring desktop documents closer to this vision and provides a software architecture, namely *Semantic Document Architecture (SDArch)* that supports semantic documents. Semantic documents enable unique identification, semantic annotation, and semantic linking of fine-grained units of documents' data. Semantic links can be established between the semantically related document data units, whether they are stored on the same personal desktop or shared within social networks. Therefore, semantic documents integrate data of desktop documents into a unified desktop information space as well as fill the gap between the desktop information space and the information space of the social network communities. New processes such as the semantic document search and navigation, which are enabled by such integrated desktop information space, improve the effectiveness and efficiency of desktop users in carrying out their daily tasks.

The thesis's main contributions are the development of the *Semantic Document Model (SDM)* that describes semantic documents and the design of SDArch that provides solutions for the semantic document repository, services that support semantic documents related processes, and tools that enable desktop users to interact with semantic docu-

ments. Additionally, in order to validate the thesis I implemented the SDArch prototype, which is a fully-functional software providing the implementation of all the intended SDArch functionalities.

The thesis is validated by two evaluation studies: i) the experimental evaluation of the information retrieval in integrated collections of semantic documents, and ii) the usability evaluation of the user effectiveness, efficiency, and satisfaction in using the SDArch services and tools. The results of these two evaluation studies proved that semantic documents have potential to semantically integrate and improve interoperability of desktop data, thus improving the effectiveness and efficiency of desktop users while carrying out their daily tasks.

Acknowledgements

I would like to express my deepest gratitude to my advisor Mehdi Jazayeri for his encouragement, guidance and support from the very early stage of this research as well as for giving me extraordinary experiences throughout the work. One simply could not wish for a better and friendlier advisor. I would also like to extend my deepest gratitude to Dragan Gašević for the deep and insightful discussions and comments about my work. His wide knowledge and his logical way of thinking have been of great value for me. Special thanks to Fabio Crestani and Monica Landoni for their advice and fruitful discussions that brought an additional value to my work.

I would like to thank all the professors, Ph.D. students and administrative stuff at the Faculty of Informatics, University of Lugano for their friendship and support. In particular, thanks to my office-mates Francesco Lelli, Navid Ahmadi and Cedric Mesnage who made our office a convivial place to work.

Thanks to my parents Andja and Branko, and sister Violeta. Without your unflagging love and support this thesis would simply be impossible. I am proud to be your son and brother.

Finally, I wish to thank my beloved girlfriend Dragana for her love, support, patience and understanding during all these years.

Preface

This thesis concerns the Ph.D. work done under the supervision of Prof. Mehdi Jazayeri at the University of Lugano. The results of this work have also been published in the following papers:

[1] S. Nešić, D. Gašević, and M. Jazayeri. An ontology-based framework for author-learning content interaction. *In proceedings of the* 6^{th} *International Conference on Web-based Education - WBE*, volume 2, pp. 359 - 364, Chamonix, France, 2007.

[2] S. Nešić, D. Gašević, and M. Jazayeri. An Ontology-Based Framework for Authoring Assisted by Recommendation. *In proceedings of the* 7^{th} *IEEE International Conference on Advanced Learning Technologies - ICALT*, pp. 227 - 231, Niigata, Japan, 2007.

[3] S. Nešić, J. Jovanović, D. Gašević, and M. Jazayeri. Ontology-Based Content Model for Scalable Content Reuse. *In proceedings of the* 4^{th} *ACM SIGART International Conference on Knowledge Capture - K-CAP*, pp. 195 - 196 Whistler, Canada, 2007.

[4] S. Nešić, D. Gašević, and M. Jazayeri. Semantic Document Management for Collaborative Learning Object Authoring. *In proceedings of the* 8^{th} *IEEE International Conference on Advanced Learning Technologies - ICALT*, pp. 751 - 755, Santander, Spain, 2008.

[5] S. Nešić, D. Gašević. Extending MS Office for sharing Document Content Units over the Semantic Web. *In proceedings of the* 8^{th} *IEEE International Conference on Web Engineering - ICWE*, pp. 350 - 353, New York, USA, 2008.

[6] S. Nešić. Semantic Document Model to Enhance Data and Knowledge Interoperability. *Annals of Information Systems, Special Issue on Semantic Web and Web 2.0*, volume 6, pp. 135 - 160, Springer , 2009.

[7] S. Nešić, F. Lelli, D. Gašević, and M. Jazayeri. Towards Efficient Document Content Sharing in Social Networks, *In proceedings of the* 2^{nd} *International Workshop on Social Software Engineering and Applications - SoSEA 2009, Co-located with ESEC/FSE*, pp. 1 - 8, Amsterdam, The Netherlands, 2009.

[8] S. Nešić, F. Crestani, D. Gašević, M. Jazayeri. Concept-Based Semantic Annotation, Indexing and Retrieval of Office-Like Document Units. *In proceedings of the* 9^{th} *International Conference on Adaptivity, Personalization and Fusion of Heterogeneous Information - RIAO*. pp. 234 - 237, Paris, France 2010.

[9] S. Nešić, M. Jazayeri, D. Gašević. Semantic Document Architecture for Desktop Data Integration and Management. *In proceedings of the* 22^{nd} *International Conference on Software Engineering and Knowledge Engineering - SEKE*, pp. 73 - 78, San Francisco, USA, 2010.

[10] S. Nešić, M. Jazayeri, D. Gašević, M. Landoni. Using Semantic Documents and Social Networking in Authoring Course Material: An Empirical Study. *In proceedings of the* 10^{th} *IEEE International Conference on Advanced Learning Technologies - ICALT*. pp. 666 - 670, Sousse, Tunisia, 2010.

[11] S. Nešić, F. Crestani, D. Gašević, M. Jazayeri. Search and Navigation in Semantically Integrated Document Collections. *In proceedings of the* 4^{th} *International Conference on Advances in Semantic Processing - SEMAPRO*. pp. 55 - 60, Florence, Italy 2010.

Contents

Figures

Tables

Chapter 1

Introduction

The idea of using Semantic Web technologies to enhance data interoperability and information management on personal desktops has been widely researched over recent years and has been shaped in the vision of Semantic Desktop [27]. A number of Semantic Desktop projects such as [9, 103, 115, 32, 49, 116] have been initiated aiming at providing a semantic infrastructure that covers all desktop applications and integrates information sources that users operate on. All of these projects attempt to enhance the existing desktop infrastructures by adding an additional semantic layer providing semantic descriptions (annotations) that refer to actual desktop resources. In such scenario, the semantic integration of desktop resources should happen at the semantic layer by interlinking descriptions of semantically related resources instead of linking actual resources. The main problem here is the propagation of modifications to resources and their relationships to the semantic layer. This problem is even more distinct in case of composite resources where the semantic descriptions should refer to components of the resources instead of the whole resources.

Desktop documents (e.g., MS Office, OpenOffice and PDF) hold a significant part of the data stored on local desktops and hence they play an important role in the vision of the Semantic Desktop. However, document data is kept into format-specific elements and is hardly accessible across application boundaries. In the last few years several XML-based document formats have been developed, such as the Open Document Format for Office Applications (ODF) [120] and Microsoft Open Office XML (OOXML) [39], which opened a way towards easier document transformation and data exchange. However, establishing explicit links among semantically related data across document borders is barely possible today. The main problem lies in the fact that only entire documents are considered as uniquely identified resources which can be referenced and linked. Existing desktop documents are organized into units (e.g., sections, paragraphs, tables and figures), but these units are not uniquely identified outside the documents and can not be put in explicit relationships with other desktop resources (e.g., other documents and document units, e-mails, images, audios and videos).

Existing desktop-document annotation approaches [118, 127, 40] utilize standardized metadata and ontology-based annotations to semantically annotate documents. Most of these approaches rely on a document-centric annotation storage model which stores annotations inside an internal document representation. The document-centric model has been used as the dominant annotation model for desktop documents mainly because it overcomes the problem of keeping annotations and documents consistent. However, storing annotations inside a document usually requires an extension of the document's format, which is not always possible. Thus, the possibility of the annotation depends on the ability of a document format to be extended. In addition, only few annotation approaches that utilizes ontological-annotations address the problem of the annotation relevance, that is, try to measure semantic relatedness between document data and ontological concepts that they annotate. Finally, none of the existing annotation approaches offers a solution to semantic interlinking of document data which are annotated by the same semantic annotations.

In spite of many drawbacks, existing semantic annotation approaches have improved data search and discoverability in desktop documents. However, the lack of the quantification of annotation relevance and the lack of explicit semantic relations (links) between semantically related data hamper machine-processability of desktop document semantics that is one of the final objectives of the Semantic Desktop. Existing desktop documents are still to a great extent only for human use.

Despite great improvements in sharing personal desktops data over the Internet infrastructure, personal desktop are still 'closed-worlds' that mainly focus on individuals' data and still there is no efficient interoperability between data stored on different desktops. The Social Semantic Desktop (SSD) is a broader concept than the Semantic Desktop, which besides data interoperability on personal desktops also aims at connecting personal desktop data into a unified information space of social communities [27]. By the envisioned Web of linked data, this unified information space could be achieved by adhering to the linked data principles [12]. Therefore, in order to be able to participate in this vision, desktop document data must adhere to the linked data principles as well. However, the existing desktop documents are not capable of that, mainly because of the same reasons that hamper document data integration and interoperability on personal desktops. Accordingly, solutions for both the document data integration on personal desktops and the integration of data from the local desktop documents to the global, unified information space (i.e., the Web of Linked data) should be found within the same comprehensive solution.

This thesis attempts to bring such a solution by introducing a new form of documents, namely ***Semantic Documents*** and designing a corresponding document architecture, namely ***Semantic Document Architecture - SDArch***, that provides semantic document storage capabilities, services for managing semantic documents and tools that enable users to interact with semantic documents.

Semantic documents are composite information resources composed of uniquely identified, semantically annotated, and semantically interlinked document data units of different granularity. Each semantic document is characterized by unique permanent machine-processable (MP) representation and a number of temporal human-readable (HR) representations rendered from the MP representation. Semantic documents are described by a new document representation model called a ***Semantic Document Model - SDM***.

By providing appropriate services and tools that run on the semantically integrated desktop information space, which is also connected seamlessly to the unified information space of social communities, semantic documents have potential to improve significantly the effectiveness and efficiency of desktop users in completing their daily tasks.

1.1 Thesis Statement and Contributions

I formulate my thesis as:

> ***"Semantic documents integrate desktop documents into a unified desktop information space, and enable data from desktop documents to be integrated into a unified information space of social communities."***

In order to validate my thesis, I answer the following two research questions:

- Q1: How do semantic documents improve information finding and retrieval in semantically integrated document collections?
- Q2: How do semantic documents facilitate desktop users in completing tasks that draw data from both a personal desktop and social communities?

By answering these research questions my thesis provides the following contributions:

- **Introducing the Semantic Document Model - SDM** [86, 88, 89]. SDM integrates the semantic layer into the core of the document representation structures. It provides a globally unique identification of document units of different granularity, enables the semantic annotation of document units by ontology-based conceptualized semantics, and provides structures for establishing explicit semantic links among semantically related document units.
- **Designing the Semantic Document Architecture - SDArch** [95, 90, 87, 92, 93]. SDArch is a software architecture that supports management of semantic documents

and enables users to take benefit from new features introduced by the semantic document model. I designed SDArch as a three-tier, service-oriented architecture composed of the data layer that provides the semantic document repository, then the service-oriented middleware, and the presentation layer that provides the SDArch user interface. Semantic document authoring, semantic document search, and semantic document navigation represent main semantic document management processes for which I provided detailed description as well as designed services that realize them. In addition to these three semantic document management processes, SDArch provides services that are responsible for management of SDArch user profile data, sharing semantic documents among SDArch users, and organizing SDArch users into a social network around shared semantic documents.

- **Providing the SDArch Prototype Implementation** [91, 92, 95]. In order to validate the implementability of the proposed architecture and the underlying semantic document model, and to enable the evaluation studies that would validate the thesis, I developed the SDArch prototype. The prototype is a fully-functional software providing the implementation of all the intended SDArch functionalities.

- **Evaluating the Semantic Document Information Retrieval and the Usability of the SDArch Services and Tools** [94, 96]. I performed the two evaluation studies aiming to answer the two research questions, thus validating the thesis. The main objective of the first evaluation study was to evaluate the semantic document search by performing a set of experiments on two different test collections. In the second evaluation study, I evaluated the user effectiveness, efficiency and satisfaction in using the SDArch services and tools. The applied usability evaluation approach involved both objective quantitative measures of the user effectiveness and efficiency, and a subjective user feedback of the user satisfaction.

1.2 Structure of the Thesis

The reminder of the thesis is organized as follows:

- **Chapter 2** presents an overview of three relater research areas to my work: Document Engineering, Knowledge Engineering, and the Semantic Web. Document engineering is the research area to which this thesis aims to contribute. Knowledge engineering is the research area that provided some techniques and formalisms which are applied in my approach. The Semantic Web and the Social Semantic Desktop, which is considered as one of the Semantic Web recent application areas, are the research area that brought the motivation for my research and opened the issue that I aimed to solve.

- **Chapter 3** introduces the semantic document model (SDM) that I developed in order to enable better integration of semantically related data managed by different desk-

top applications as well as to make data from desktop documents be linkable across desktop borders to the envisioned Web of linked data.

- **Chapter 4** describes the semantic document architecture (SDArch) that I designed in order to support management of semantic documents (i.e., instances of the introduced SDM). SDArch provides storage capabilities for semantic documents, services that realize semantic document management processes, and tools that enable users to interact with these services. My focus in this chapter is on the overall design of the architecture and the detailed description of the three services, namely, the user profile management, the social network management, and the ontology management services. These services are not core to semantic document management, but provide functionalities/data that the semantic document management processes rely on.

- **Chapter 5** presents the semantic document management processes enabled by SDArch. There are three top-level processes: semantic document authoring, semantic document search, and semantic document navigation. They are realized by a number of sub-processes that are realized by specific SDArch functional modules. The functional modules responsible for the semantic document processes are encapsulated into two SDArch services: the semantic document authoring and the semantic document search and navigation services. In this chapter, I give a detailed specification of all the three top-level processes and describe the functional modules of the two services as well as their interface.

- **Chapter 6** describes the current implementation of the SDArch prototype. The SDArch prototype is a feature-complete, fully-functional software, providing the implementation of all intended SDArch functionalities. It was used for the experimental evaluation of the proposed semantic document information retrieval and for the usability evaluation of the SDArch services and tools with end-users.

- **Chapter 7** presents and discusses the results of the two evaluation studies that I conducted in order to evaluate the thesis statement. The main objective of the first evaluation study, which included a set of experiments executed against two test collections, was to evaluate the effectiveness of the proposed semantic document information retrieval and to compare it with related concept-based information retrieval approaches and the conventional full-text search. In the second evaluation study, I evaluated the usability of the SDArch services and tools considering the user effectiveness, efficiency and satisfaction in using them.

- **Chapter 8** concludes the thesis by discussing the main contributions of my work and giving an outlook on future work.

Acknowledgement: This work was supported in part by Project Nepomuk, Number 027705, supported by the European Commission in Action Line IST-2004-2.4.7 Semantic-based Knowledge and Content Systems, in the 6th Framework.

Chapter 2

Related Research Efforts

My research work spans three research areas: Document Engineering (DE), Knowledge Engineering (KE), and the Semantic Web (SW) (Figure 2.1).

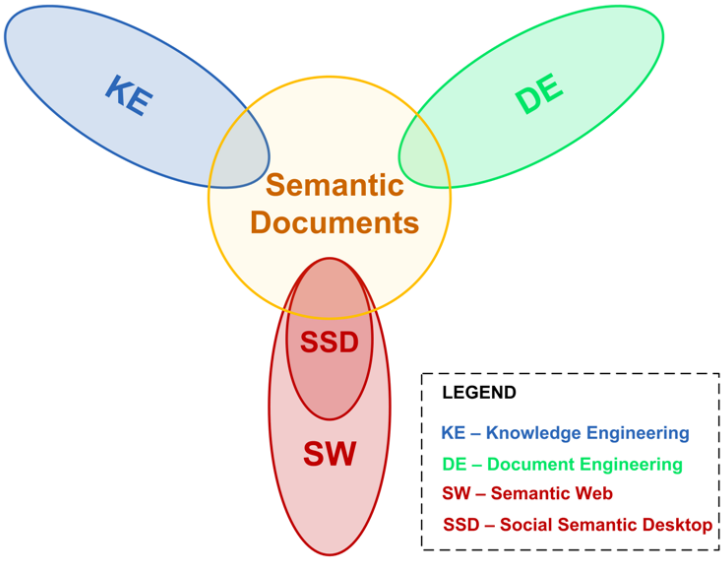


Figure 2.1. Related research areas

DE is the research area whose state of the art I want to enhance and which provides related work that could be compared to my approach. KE is the research area that provides some techniques and formalisms which are applied in my approach as well as the area in which some results of my work could have potential impact. SW is the research area that brought the motivation for my research and opened the issue that I aim to solve. It provides standards, languages and formalisms I use as basis of my approach. With respect to SW, I further position my research interests in the area of Social Semantic Desktop (SSD), which actually attempts to apply the Semantic Web

technologies to improve data and application interoperability on individual desktops as well as to extend a personal desktop into a collaborative environment that supports information and content sharing across social and organizational relations.

The rest of the chapter is organized into four sections that give overviews of the four areas, DE, KE, SW and SSD respectively.

2.1 Document Engineering

Documents play a key role in the construction of social reality and they are an important part of every aspect of human society and culture [7]. The perception and definition of documents have continuously been changing over time following the developments of human society. There have been many attempts [30, 17, 18, 130] to define a document by observing documents from different viewpoints such as the nature of document storage medium, the document representation format, the document interchange model, and the role of a document.

The International Institute for Intellectual Cooperation, an agency of the League of Nations, developed a technical definition of document: "Any source of information, in material form, capable of being used for reference or study or as an authority". In 1935 Walter Schuermeyer wrote: "Nowadays one understands as a document any material basis for extending our knowledge which is available for study or comparison" [30]. Suzanne Briet defined a document as any physical or symbolic sign, preserved or recorded, intended to represent, to reconstruct, or to demonstrate a physical or conceptual phenomenon [17]. In the context of computer communication, a document can be defined as a structured amount of information that is meant for human perception and can be interchanged among systems as a unit [18]. Recent trends tend towards the definition of a document as a knowledge model [130] consisting of interlinked information atoms as smallest document units, which can be interpreted without a document context.

In document evolution, one of the key changes happened with the introduction of '*Digital Era*', which led to the main classification of documents into the paper and digital documents. A paper document is distinguished, in part, by the fact that its content is written on paper. However, the aspect of technological medium is less helpful with digital documents. For example, wordprocessing and PDF documents exist physically in a digital environment as strings of bits; but so does everything else from a digital environment represented in the same way can be considered as a document. Buckland [18] argues that documents should be defined in terms of function rather than physical format. By fallowing this trend, everything that behaves like a document is a document. For practical purposes, people developed pragmatic definitions, such as "anything that can be given a file name and stored on electronic media" or "a collection of data plus properties of that data that a user chooses to refer to as a logical unit".

The principle differences between paper and digital documents come from different types of physical medium that is used for document storage and from the way in which documents are created, managed and communicated among people. Digital documents have many advantages over paper documents, including compact and lossless storage, easy maintenance and efficient retrieval and fast transmission. With the use of networked information systems, digital documents have become highly available and can be found more easily than paper documents. Moreover, digital documents can manifest properties that are not available in their paper counterparts. Examples of such properties are hyperlinks, virtual structures (e.g., documents whose elements are created dynamically), and inclusion of 'dynamic media' such as audio and video. Despite these valuable features, paper is still superior to the digital medium for some purposes. For example, comparing with paper documents, digital documents are less stable in time (i.e., their content can change at any point in time), and can be cited only if they are managed by trustworthy sources.

2.1.1 Computer Model of Documentation

A computer model of documentation has evolved from 80-column ASCII files, through various kinds of presentation markup (e.g., *TEX* and *troff*) to so-called structural markup (e.g., *LATEX* and *SGML*). The aim has been to enable computers to provide as rich a presentation of document content as possible and then to make that presentation as independent of the document content as possible. The increasing popularity of Personal Computers (PCs) for typesetting documents raised issues of document exchange between users at different sites. This created the need to define common document interchange formats. In order to allow documents to be interchanged among systems as a unit, document architectures should define the concepts for integrating content portions of different information types with structural information into one entity, namely a document [43].

Reid [107] defined a document model with hierarchical nesting that was used in the *Scribe* word processor. *Scribe* introduced named environments, which had the role of containers (e.g., ordered lists and tables). Environments could be nested and any kind of hierarchical structure can be defined trough relationships between environments. One of the most interesting models based on Reid's approach is *tnt* [47] which uses a forest of ordered trees to represent the different document parts. The approach of Dori et al [33] is also based on a tree concept, combining low-level elements into higher level ones. Its innovation lies in the definition of a generic logical structure that can be applied to different classes. In this model every object is defined as 'texton' or 'graphon' at the highest level of granularity. Depending on the document class instantiated and the current granularity level, textons can be classified as paragraphs, sentences, words, characters, etc., while graphons can be instantiated as lines, drawings, images, charts, tables, etc.

Tree-like document models have been developed in a range of different formats. Most of them have been inspired by *Scribe*, which influenced the development of *SGML* and is a direct ancestor to *HTML* and *LATAX*. *SGML* provides an abstract syntax that can be realized in many different concrete syntaxes. From the late 80s on, most substantial new markup languages have been based on *SGML* including *TEI*, *DocBook* and *XML*. A common feature of the majority of markup languages is that they intermix document content with markup instructions in the same data stream or file. The other option is to isolate document markup from document content using pointers, offsets and identifiers. This type of document markup is known as a *standoff markup*. However, embedded or *inline markup* is much more common elsewhere.

In the last few years, the development of document interchange formats based on *XML* demonstrated how complex structural information may be defined within modern desktop document processors. Recent standards such as the Open Document Format for Office Applications - *ODF* [97] and Microsoft's Open Office XML - *OOXML* [39] opened the way for the *XML* based exchange of documents between different office applications. As a part of the next section I will also take a closer look at these standards.

2.1.2 Desktop Document Architectures

A document architecture defines a document model that integrates different types of content such as text, graphics, audios and videos, and provides a collection of services and a user interfaces that forms a single integrated document environment [33]. Examples of desktop document architectures and formats include *OpenDoc*, *Microsoft's Object Linking and Embedding - OLE*, *Open Document Architecture - ODA*, *Multivalent Documents*, *OpenDocument Format - ODF*, *Office Open XML - OOXML*, *Compound Document Format - CDF* and *Active Documents*.

OpenDoc [2] is a set of shared libraries designed to facilitate the easy construction of compound, customizable, collaborative, and cross-platform documents. To do this, *OpenDoc* replaces application-centered user model with a document-centered one. The user focuses on constructing a document or performing an individual task, rather than using any particular application. The software that manipulates a document is hidden, and users feel that they are manipulating the parts of the document without having to launch or switch applications. *OpenDoc* envisaged a document being composed of material contributed from a variety sources such as MacWrite, Adobe Photoshop and Adobe Illustrator. Each piece of material in *OpenDoc* document would be rendered by calling on the appropriate application at the appropriate time. If the document was sent to a remote machine, not having all of the required application programs, then a system of lower-quality renders bitmap approximations to ensure that the document could at least be read. In many ways *OpenDoc* was well ahead of its time but it floundered because of the need to have a wide variety of authoring applications available and the effort needed to make each of these applications be 'OpenDoc aware' in order for them to fully participate in the framework [125].

Object Linking and Embedding - OLE [98] is a technology that allows embedding and linking to documents other objects developed by Microsoft. It is primarily used for managing compound documents and transferring data between different applications. OLE technology also enables visualization of data from other applications that the host application is not normally able to generate itself (e.g. a pie-chart in text document). It is founded on the Component Object Model - COM, which is a language-neutral way of implementing objects that can be used in environments different from the one they were created in. Although OLE objects achieved an important success, the platform dependence (i.e., they can be used only with Microsoft Windows) suppressed their broader use.

Open Document Architecture - ODA [43] is another application of compound document formats that was developed in the mid - 1980s by several standardization bodies. It represents a set of international standards for the interchange of compound documents consisting of text, images, and graphic contents [43]. *ODA* defines interchange formats, concepts to represent the structure of the information in a document, and the meaning of a set of formatting parameters. One of the main aims of the *ODA* was to allow so called 'blind document interchange'. This means that a document can be interchanged among two systems such that re-visibility and layout stay preserved just based on the knowledge that both systems comply to the international standard. However, no significant document software chose to support the format. It also took an extraordinarily long time to release the format (the pilot was financed in 1985, but the final specification not published until 1999). Given a lack of products that supported the format, only few users were interested in using it.

Multivalent Document - MVD model [101] is an architecture in which a document is composed out of distributed data and program resources, called 'layers' and 'behaviors' respectively. Layers and behaviors are assembled by an MVD compliant browser from multiple distributed sources over the network. Any media type can potentially be bridged into the multivalent model. The model exposes virtually all aspects of document processing to behaviors, and provides the means to compose layers (i.e., data) and behaviors into a single coherent document.

OpenDocument Format - ODF [97] is an open XML-based document format designed to be used for documents containing text, spreadsheets, charts, and graphical elements. The format makes transformations to other formats simple by leveraging and reusing existing standards wherever possible. From a technical point of view, *ODF* is a ZIP archive that contains collection of different XML files, as well as binary files, such as embedded images. The use of XML makes access to document content easier since content can be opened and changed with simple text editors in contrast to the previously used binary formats which were cryptic and difficult to process.

Office Open XML - OOXML [39] is a document format for representing spreadsheets, charts, presentations and word processing documents. *OOXML* documents are stored in Open Packaging Convention - OPC packages, which are ZIP files containing

XML markup files and a specification of the relationships between them. The OPC package can also include embedded binary files such as images, audios and videos. An OOXML document may contain several XML markup files encoded in specialized markup languages corresponding to applications within the Microsoft Office product line. The primary markup languages are: WordprocessingML for word-processing, SpreadsheetML for spreadsheets, and PresentationML for presentations.

Compound Document Format - CDF [23] is a document format developed by the W3C that manipulates with contents from multiple formats, such as SVG, XHTML, SMIL and XForms. As of the end of 2007, the OpenDocument Foundation, which previously supported *ODF*, switched alliances and started promoting *CDF*.

Active Documents [104] are an extension of the compound document concept. An active document is a document that acts on its computing environment or that transforms itself when it is manipulated by a user through an editor [104]. ActiveX [98] documents, formally known as 'document objects' are an example of the active documents. This approach distinguishes between a document, such as a word document or video clip, and the application that can open, edit, display, and save the document. In other words, ActiveX documents consist of two components: the 'document' itself and the ActiveX DLL or EXE server that supports it.

2.1.3 Document Annotation Models

Knowledge about documents has been traditionally managed through the use of metadata, which can concern the world around the document. Metadata, usually interpreted as 'data about data', can be considered as a mechanism for expressing semantics of information, as a means to facilitate information seeking, retrieval, understanding and use [118]. Metadata can be expressed in a diverse range of human and artificial languages and forms. Metadata languages require shared representations of knowledge as the basic vocabulary from which metadata statements can be asserted.

Semantic document annotation refers to the process of creating metadata by using ontologies as metadata vocabularies. Dublin Core (DC) [34] is an example of a lightweight ontology that is being widely used to specify the characteristics of digital documents. It specifies predefined set of concepts i.e., document features such as author, date, contributor, description and format.

The annotation storage model is one important issue regarding semantic document annotation. There are two major models: the Semantic Web model and the Document-Centric model. In the first model, annotations are stored separately from the source document content [127]. This model is primarily used for annotating Web (HTML) documents, since documents and annotations are owned by different people and organizations and stored in different places. An advantage of decoupling of semantic annotation from the document content is that no changes to a document are required. Also, embedded complex annotations would have negative impact on the volume of the content

and can complicate its maintenance. In addition, the resulting decoupling of semantics and content facilitates document reuse because it is possible to set up rules which control and automate which kinds of annotations are transferred to new documents and which are not. It also makes it easy to produce different views of a document for users regarding their interests. The drawback of separating annotations from a document is an extra overhead that is required to maintain links between a document and its annotations [127]. The second model stores annotations and their vocabularies (ontologies), inside the internal document representation. This model has been used as the dominant annotation model for desktop office-like documents (e.g., Word, PDF and Spreadsheet) [40, 124], because it overcomes the problem of keeping annotations and documents consistent. However, storing annotations inside a document usually demands extending the document format schema, which is not always possible. Thus, the possibility of the annotation depends on the ability of a document format schema to be extended.

Besides the annotation storage model the other important issue regarding document annotation is the way in which annotations are generated. A number of document annotation frameworks and tools have been developed [127], some of which rely on knowledge workers' domain knowledge while others are based on automatic content analysis. This leads to the classification of the annotation into manual and automatic. Both types have comparative advantages and drawbacks.

Manual annotation is usually done by using authoring tools which provide an integrated environment for the simultaneous document authoring and annotation. However, the use of human annotators is often fraught with errors due to factors such as annotator's familiarity with the domain, personal motivation and complexity of annotation schemas. The quality of such annotation strongly depends on the annotators' knowledge and time they are able to spend creating the annotation.

Automatic annotation provides the scalability needed to annotate existing documents, and reduces the burden of annotating new documents. The main advantage of automatic over manual annotation is the reduced workload for annotators. In particular, this is important for annotating large collections of legacy documents. Automatic annotation systems are based on the following kinds of automatic supports [127]: i) rules or wrappers written by hand that try to capture known patterns for the annotations; ii) information extraction (IE) systems incorporating supervised learning; iii) IE systems that use some unsupervised machine learning; and iv) natural language processing (NLP) systems. However, automatically generated annotations are less accurate then those generated by professional annotators; in order to have well-annotated documents, human intervention is still required. Because of the limited accuracy there are fairly few completely automated annotation tools. They are rather semi-automated with various degrees of automation and rely on human intervention at some point in the annotation process [80].

Semantically annotated documents bring the advantages of semantic search, retrieval and interoperability. However, the real use and success of semantic document

annotation strongly depends on the overhead of increased annotation effort. To minimize this overhead the semantic annotation system must be easily integrated in existing document-authoring environments (e.g., MS Office, and OpenOffice). Moreover, in order to further reduce user workload, these systems need automation to support annotation, automation to support ontology maintenance, and automation to help maintain the consistency of documents, ontologies and annotations. However, fully integrated environments are still some way off. WiCKOffice [19], AktiveDoc [77]], SemanticWord [124], and PDFTab [40] are some examples of the systems/tools that are aimed at integrating the annotation/knowledge markup process into standard office-like environments and making annotation simultaneous to authoring.

SemanticWord [124] extends Microsoft Word in several dimensions. First, MS Word GUI is augmented with toolbars that support the creation of semantic descriptions (or annotations) that are attached to text regions. The GUI is also extended to show these annotations embedded within the text and to support their direct manipulation through mouse gestures. Second, content from the Semantic Web (both ontology definitions and factual descriptions) is brought into SemanticWord to compose annotations that are later dumped back into the Semantic Web. Third, SemanticWord extends Word services by integrating AeroDAML [76], an automated information extraction system. AeroDAML analyzes and annotates the text of the document as it is being typed, appearing to the author as a service analogous to WordÕs spelling and grammar checking. Finally, SemanticWord supports the rapid composition of annotated text through template instantiation.

PDFTab [40] is a plug-in extension to Protege (ontology development environment) that supports 'semantic documents'. The semantic document approach [40] is a recent initiative which proposes much deeper integration of documents and ontologies. The ultimate goal of this approach is not merely to provide metadata for documents, such as DC descriptions, but to integrate documentation and knowledge representation to the point where they use a common structure, which provides both documentation and knowledge representation views. The PDFTab allows users to import PDF documents into Protege and to link them to ontologies using provided annotation properties. In other words, PDFTab bridges the ontology and document domains and enables users to take advantage of the rich Protege environment for creating ontologies to create semantic documents. The combined packaging of documents and ontologies is advantageous in that the semantic documents retain their ontology content throughout electronic communication and archival storage. A critical factor for semantic documents is the linkage between the printable document and the ontology.

WiCKOffice [18] is an environment that provides several knowledge services to assist authors in making knowledge/annotations an explicit part of the document representation. A 'knowledge fill-in' service and 'knowledge recall' service are motivated by the need to provide timely and convenient access to knowledge, which would otherwise have to be manually looked up on institutional intranet. A third service, 'in-line guide-

lines', also assists recall by exposing guidelines and constraints captured from a design specification that are relevant to the part of the document currently being worked on. WiCK extensions to the Microsoft Office environment utilize key computational knowledge services to assist the writing task, and to update the knowledge-bases when the writing task is completed.

AktiveDoc [77] is a system for supporting knowledge management in the process of document editing and reading. Its main feature is to support users (both readers and writers) in the timely sharing and reusing relevant knowledge/annotations. It enables the annotation of documents at three levels: ontology based content annotation, free text statements and on-demand document enrichment. AktiveDoc is a client-server application integrated in a Web based KM system and providing both manual and semi-automatic annotation. While many current systems modify the original document to add annotations, AktiveDoc saves them in a separate database. Documents are saved in KM systems that act as a knowledge base and every document is logically associated with its annotations.

2.1.4 Limitations of Existing Desktop Document Architectures

In Sections 2.1.2 and 2.1.3, I have analyzed a number of desktop document architectures and document annotation models applied in them, respectively. In this section I discuss limitations of the existing desktop document architectures, which I have identified with regard to the vision of the SSD (Section 2.4. I have grouped the identified limitations into the following six categories.

Lack of Openness: The application specific document formats keep document data closed into format specific elements so that it is hardly accessible across application boundaries. In the last few years several XML-based document formats have been developed, such as the ODF [97] and OOXML [39] (default formats of OpenOffice and MS Office documents respectively), which opened a way towards easier transformation of their native form to and from other formats, by providing export/import bridges. However, using one-to-one export/import bridges is unsuitable for the highly dynamic online world, where the number of document formats grows constantly. Developing such bridges is a difficult and costly process, as bridges must have detailed knowledge of proprietary forms and interfaces. If we have for example, N different platform specific document formats, we need N^2-N bridges in order to enable all possible transformations. Moreover, a document's data is kept in structural elements which are difficult to access without knowing the document schema definition. This limits document data reuse in different applications which is mostly done manually by 'copy-paste' practice.

Next, the ability to assemble compound/multimedia documents in a dynamic way by invoking contents from distributed sources is limited. The OpenDoc [125] framework has gone the furthest in this direction, but it floundered because of the need to have a wide variety of authoring applications available and the effort needed to make each of these applications be 'OpenDoc aware' in order to participate in the framework. More-

over, in the case of compound multimedia documents it is not possible to edit all types of document content within a single document. Usually, applications for editing multimedia document support only several formats of each content media type (e.g., image, audio and video). These applications just render approximations of content types that they do not support so that the document could at least be read.

Finally, existing document architectures are not open enough for collaborative document authoring and editing. In software development, the Concurrent Versions System (CVS) software keeps track of all work and all changes in a set of files, and allows several developers to collaborate. In office-like document management there are some similar initiatives such as Microsoft's SharePoint, but they are still significantly less effective and less utilized than CVS systems.

Lack of Granularity and Referenceability: Currently, only whole desktop documents can be considered as resources which can be identified and referenced. Document data is organized into units (e.g., paragraphs, tables and sections), but these units are not uniquely identified entities that can be put in explicit relationships with some 'outside world' resources (e.g., peoples, organizations and places). It is difficult to access and interact directly with a particular document's unit, without obtaining the whole document first. Whenever someone wants to access some of the document's units either to read or edit them the whole document has to be obtained.

Lack of Customization/Personalization: The ability of current document architectures to adapt document content/data to correspond with users' specific needs or to meet specific usage objectives is low. In general, existing desktop documents are static and cannot respond to changes to the context in which they are used (e.g., different users and different usage objectives).

Lack of Traceability: Over time documents constantly change and evolve throughout many versions. In current document architectures transparent document evolution is only possible if documents are maintained by version control systems, which is rarely the case. On the contrary, users usually maintain different versions of documents by encoding information about versions into document names or placing different document versions in different locations of the filesystems. This way they create new documents instead of new versions of the same document, since there is no explicit link between the two copies of the document. Neither the document name nor its location is reliable enough to identify document versions. In the highly dynamic networked world, this is an even less reliable solution. The document identification has to be unique and universal in order to enable transparent document evolution and document reuse in different context. What is even less transparent is the evolution of document units and their usage path. By copying from one to another document, the two copies of the same document unit stay effectively unrelated.

Limited Annotation: Document annotation is a way to add extra information to a document; in other words, to model knowledge 'about' the document. I have identified several limitations that characterize the annotation of existing office-like desktop docu-

ments. Firstly, the annotation is usually restricted to predefined annotation vocabularies such as Dublin Core (DC) [34] and Learning Object Metadata (LOM) [35]. Extending annotation vocabularies with new user-defined terms is difficult because each term from the vocabularies should have a schema defined element where its value will be stored. Therefore, in order to extend the annotation vocabulary, the document schema should be extended as well, which is tedious and not always possible. Secondly, schema defined elements for storing annotations are usually provided only for whole documents; it is rarely possible to annotate parts of the document. For example `dc:creator` is not applicable at the level of document paragraphs. Thirdly, there is no convenient solution for the annotation storage model. The two existing models (Section 2.1.3), the document-centric and the Semantic Web model have significant limitations when applied to existing desktop documents. The Semantic Web model that keeps annotations decoupled from a document, is mostly inapplicable because of the high cost of maintaining links between the document and its annotations. This is mainly due to the lack of openness and the difficult addressability of document content. On the other hand, the document-centric model that stores annotations inside internal document representation, would have negative impact on the volume of the document and can complicate its maintenance if embedding complex annotations. Also, in the document-centric model annotations can be added only by users who can access the document and have rights to edit it, while in the Semantic Web model everybody could add annotations. Fourthly, annotations are still passive elements which improve search and retrieval, but do not modify document content, appearance or runtime properties. Finally, document authoring environments suffer integrated support for the automatic annotation. The manual annotation produces more accurate annotations, but in case of a large collection of legacy documents it is almost impossible.

Absence of Knowledge Conceptualization: In contrast to document annotations that model additional knowledge about the document, the document's declarative knowledge is what the document provides about its topic. Current documents model only a human understandable variant of this knowledge; software agents can neither discover nor use it. Combining documents and domain ontologies [40] is an attempt towards the conceptualization of a document's declarative knowledge, but a pervasive solution that takes in account all aspects of the conceptualization and codification of document declarative knowledge does not exist. Conceptualization of document declarative knowledge and its codification in a machine processable form will enable intelligent software agents to infer new knowledge (i.e., new assertions that characterize a domain describe in a document). Moreover, by providing procedural knowledge that explains how users can use document data in achieving some objectives, software agents will be able to assist humans in problem-solving by recommending document units that holds appropriate information for them.

2.2 Knowledge Engineering

Many scientific disciplines including Cognitive Sciences (CS) [122] and Artificial Intelligence (AI) have been concerned with defining the notion of knowledge. However, there is no single agreed definition of knowledge today. A number of definitions have been formulated such as:

- Knowledge is understanding of a subject area [37]. It includes concepts and facts about that subject area, as well as relations among them and mechanisms for how to combine them to solve problems in that area;
- Knowledge is a fluid mix of data, experience, practice, values, beliefs, standards, context, and expert insight that provides a conceptual arrangement for evaluating and incorporating new data, information and experiences [29];
- Knowledge is richer, more structured and more contextual form of information. It is required to perform complex tasks such as problem-solving, and encompasses such things as experience and expertise [74];

Instead of defining the term knowledge precisely, some researchers [58, 5] focus on knowledge cues. A knowledge cue can be considered as any kind of symbol, pattern or artifact that evokes some knowledge in a person's mind, when viewed or used. Knowledge cues can be stored on a computer - while knowledge may not.

Knowledge engineering [5, 37, 46] is a field within AI that involves integrating knowledge into computer systems in order to solve complex problems normally requiring a high level of human expertise [42]. Currently, it refers to the building, maintaining and development of knowledge-based intelligent systems [73]. The central component of any knowledge based intelligent system is its knowledge base. In order to develop a practical knowledge base, it is necessary to acquire human knowledge (e.g., from human experts or other sources), to understand it properly, to transform it into a form suitable for applying various knowledge representation formalisms and to encode it in the knowledge base using appropriate representation techniques, languages, and tools. This process is also known as knowledge acquisition.

It has been frequently stated that the problem of knowledge acquisition is 'the critical bottleneck' of knowledge based system development [5]. There are many knowledge acquisition (KA) techniques that can be classified into manual and (semi)automated. Usually, the expert knowledge is acquired through common social science methods such as interviews, questionnaires, and discourse analysis. However, in many cases when a system requires a large knowledge base which should be constantly augmented with new knowledge, the manual techniques are not applicable. Therefore, the trend in knowledge acquisition has turned towards the use of (semi)automated knowledge acquisition techniques based on machine learning and qualitative modeling. Recently, the Web 2.0 and social network services (e.g., Facebook, MySpace and LinkedIn) have

opened the way for the acquisition of the so called 'collective knowledge' through the collaborative social tagging of web resources [56].

Once a knowledge base is populated, knowledge can be utilized. Knowledge retrieval is the inverse process of knowledge acquisition - finding knowledge when it is needed. Retrieved knowledge can serve both humans and intelligent software systems. Later one can perform reasoning by using knowledge and problem solving strategies to obtain conclusions, inferences and explanations. In the rest of the section I present common knowledge representation techniques and languages.

2.2.1 Knowledge Representation Techniques

Natural languages can express almost everything related to human experience, and hence they are the most powerful knowledge representation technique. However, the use of natural languages for knowledge representation in AI is very restricted, owing to the fact that they are extremely complex for machine processing. Even more important and more difficult is the problem of machine understanding of the meaning of natural languages.

Knowledge representation is the notation or formalism used for encoding knowledge for storage in a knowledge-based system. Different mental representation of the human mind, as proposed by cognitive theories, such as logical propositions, rules, concepts, images and analogies, constitute the basis of different knowledge representation techniques [66]. The field of AI has not produced fully intelligent machines but one of its major achievements is the development of a range of techniques for representing knowledge, which can be classified into four categories: *ladders*, *semantic networks*, *tabular representations*, and *rules*.

Ladders are hierarchical (tree-like) diagrams. Important types of ladders are: i) concept ladders, which show classes of concepts and their sub-types and models 'is a' relationships; ii) composition ladders, which show the way a knowledge object is composed and model 'has-part' or 'part-of' relationships; iii) decision ladders, which show the alternative courses of action for a particular decision; iv) attribute ladders, which show attributes and values; and v) process ladders, which show processes (tasks) and the sub-processes (sub-tasks) of which they are composed.

Semantic networks are graphs made up of objects, concepts, and situations in some specific domain of knowledge (the nodes in the graph), connected by some type of relationship (the links/arcs). All semantic networks can be represented as collections of Object-Attribut-Value (O-A-V) triplets. O-A-V triplets are a technique used to represent facts about objects/concepts and their attributes. It serves as a basic building block of any kind of semantic network. Examples of semantic networks include concept maps, process maps and state transition networks. Designed after the psychological model of human associative memory, concept maps [5] are graphs made up of concepts from specific domain knowledge, connected by some type of relationship. A process map is a way of representing information of how and when processes and tasks are performed. They

show the inputs, outputs, resources, roles and decisions associated with each process or task in a domain. The third important type of semantic networks is the state transition network. The state transition networks comprise two elements: i) nodes that represent the states that a concept can be in, and ii) arrows between the nodes showing all the events and processes/tasks that can cause transitions from one state to another.

Tabular representations make use of tables or grids for knowledge representation. The most common and the most often used form of this representation technique are frames. A frame is structure for representing stereotypical knowledge of some concept or object. Frames are similar to classes and objects in object-oriented programming. Each frame is easy to visualize using a matrix representation. The left-hand column represents the attributes associated with the concept (class) and the right-hand column represents the appropriate values.

Rules are a knowledge representation technique and a structure that relates one or more premises (conditions) or situations to one or more conclusions (consequents) or actions. The premises are contained in the *IF* part of the rule, and the conclusions are contained in the *THEN* part, so that the conclusions may be inferred from the premises when the premises are true. Some rules may include certainty factor, a numeric value assigned to both premises and conclusion that represents the degree of belief in them. The knowledge of a particular knowledge based system may be represented using a number of rules. In such a case, the rules are usually grouped into a hierarchy of rule sets, each set containing rules related to the same topic.

2.2.2 Knowledge Representation Languages

The knowledge base contains a set of sentences - the units of the knowledge represented using one or more knowledge representation techniques, i.e., assertions about the world [113]. The sentences are expressed in a knowledge representation language. Knowledge representation languages should be capable of both syntactic and semantic representation of entities, events, actions, processes, and time. Formal notation for knowledge representation allows inference and problem solving. Moreover, queries can be made to the knowledge base to obtain what the system currently knows about the world. In accordance to the knowledge representation techniques which are described above, AI researchers have developed a number of knowledge representation languages.

Logic-Based Representation Languages: The popularity of formal logics as the basis of the knowledge representation languages arises for practical reasons. They are all formally well founded and are suitable for machine implementation. Also, every formal logic has a clearly defined syntax that determines how sentences are built in the language, a semantics that determines the meanings of sentences, and an inference procedure that determines the sentences that can be derived from other sentences.

Propositional logic is a form of symbolic reasoning that assigns a symbolic variable to a proposition. A proposition is a logical statement that is either true or false. The truth-value of the variable represents the truth of the corresponding statement (the

proposition). Propositions can be linked by logical operators (AND ($\wedge$), OR ($\vee$), NOT ($\neg$), IMPLIES ($\Rightarrow$), and EQUIVALENCE ($\Leftrightarrow$) to form more complex statements and rules. Propositional logic allows formal and symbolic reasoning with rules, by deriving truth-values of propositions using logical operators and variables.

First-Order logic extends propositional logic by introducing the universal quantifier $\forall$, and the existential quantifier $\exists$. It also uses symbols to represent knowledge and logical operators to construct statements. Its symbols may represent constants, variables, predicates, and functions. Using predicates, functions, and logical operators, it is possible to specify rules. Reasoning with first order logic is performed using predicates, rules, and general rules of inference to derive conclusions. First-order logic is like an assembly language for knowledge representation [37]. Higher-order logic, modal logic, fuzzy logic, and even neural networks can all be defined in first-order logic.

Description logic is based on two components TBox and ABox. Developing a knowledge base using a description logic language means setting up terminology (the vocabulary of the application domain) in a part of the knowledge base called the TBox, and assertions about named individuals (using the vocabulary from the TBox) in a part of the knowledge base called the ABox. The vocabulary consists of concepts and roles. Concepts denote sets of individuals. Roles are binary relationships between individuals.

Frame-Based Representation Languages: In all frame-based representation languages, the central principle is a notation based on the specification of frames (concepts and classes), their instances (objects and individuals), their properties, and their relationships to each other [134]. Frame-based languages are suitable for expressing generalization/specialization, i.e., organizing concepts into hierarchies. They also enable reasoning, by making it possible to state in a formal way that the existence of some piece of knowledge implies the existence of some other, previously unknown piece of knowledge. With frame-based languages, it is possible to make classifications, that is, concepts are defined in an abstract way and objects can be tested to see whether they fit such abstract descriptions.

Rule-Based Representation Languages: Rule-based representation languages are popular in commercial AI appliactions, such as expert systems [37]. Every rule-based language has an appropriate syntax for representing the If-Then structure of rules. Vianu [129] notes that there are two broad categories of rule-based languages: declarative languages, which attempt to provide declarative semantics for programs, and production system languages, which provide procedural semantics based on forward chaining of rules. The rule-based representation formalism is recognized as an important topic not only in AI, but also in many other branches of computing. This is especially true for Web engineering. Rules are one of the core design issues for future Web development, and are considered central to the task of document generation from a central XML repository. In response to such practical demands from the world of the Web, the Rule Markup Initiative (RMI) has taken steps towards defining RuleML, a shared Rule Markup Language [111]. RuleML enables the encoding various kinds of rules in XML for

deduction, rewriting, and further inferential-transformational tasks. The Rule Markup initiative now covers a number of new developments, including Java-based rule engines, an RDF-only version of RuleML, and MOF-RuleML.

2.3 The Semantic Web

> *The Semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation [8].*

Humans are the current Web's semantic component. They are required to process the information culled from Web resources to ultimately determine their meaning and relevance for the task at hand. The Semantic Web intends to move some of that processing to software agents. In order to map Web resources more precisely, computational agents require machine-readable description (metadata) of the content and capabilities of Web accessible resources. These descriptions must be in addition to the human-readable versions of that information, complementing but not supplanting it. Therefore, the real success of the Semantic Web depends on the possibility of creating valuable semantic metadata. It can be argued that until anyone can easily create metadata about any Web resource and share that metadata with everyone, no true Semantic Web will arise [61].

> *The Semantic Web is a vision: the idea of having data on the Web defined and linked in a way that it can be used by machines not just for display purposes, but for automation, integration and reuse of data across various applications [9].*

Besides describing the available resources with metadata, the Semantic Web is also concerned about making data and metadata to be efficiently shared and reused across application, enterprise, and community boundaries, as well as providing the agency to manage them. It tries to get people to make their data available to others by adding links and following relative links. It is the next stage of linking on the Web - linking data not documents. In the Semantic Web model, both data and metadata storage are primarily distributed in adaptive virtual networks. Peer-to-Peer (P2P) architecture is envisaged as replacing centralized data storage and represents one of the pillars of the Semantic Web. Moreover, distributing and delocalizing functionality through Web services is an integral part of the Semantic Web model. Finally, by taking advantage of semantically marked up data and provided services, a variety of diverse software agents can be developed to facilitate the full span of possible collaborative processes (i.e., human to human, human to machine, and machine to machine).

In the rest of this section I first outline the main components (layers) of the Semantic Web, then briefly discuss the distributed data storage and processing on the Semantic Web and conclude the section with the discussion on Semantic Web agents.

2.3.1 Semantic Web Components

The principal technologies of the Semantic Web fit into a set of layered specifications commonly known as Tim Berners-Lee's the 'Semantic Web Layer cake' [8].

Identity (URI): Handling resources on the Web requires a strong and persistent implementation of unique identity. The Uniform Resource Identifier (URI) is the generic solution, but it was not directly implemented in the early Web. The URI subset of Uniform Resource Locator (URL) was used to base global identity on an abstract 'location' instead - as protocol plus domain plus local path or access method. However, actual location on the Web (URL) is less useful than it may seem. The unpredictable mobility or availability of Internet resources at specified URLs is an inconvenience at best, often resulting in cryptic error messages from the server instead of helpful redirection. Another URI subset, Uniform Resource Name (URN), has been under careful development by the Internet Engineering Task Force (IETF) committee for some times, with the intent to provide persistent identities based on unique names within a defined namespace such as *urn:namespace:named-resource*. A modification to the current DNS system, for example, would resolve current issues with changing resource URLs by providing dynamic translation from URN pointers to the actual server-relative locations.

Markup - XML: The syntactic component of the Semantic Web is the markup language that enables distinction between content representation and the metadata that defines how to interpret and process it. XML is a commonly accepted markup language, because among other things it fulfills the dual requirements of being self-defining and extensible document description. The visible part of the markup component is its syntax, expressed as a reserved set of text pattern 'tags' embedded in the document but invisible in human-use rendering. XML depends on URI, but is in turn the foundation for most of the higher layers in the Semantic Web 'layer-cake' model.

Descriptive Assertions - RDF and RDFS: Given identities and markup, the next step is to codify the meaning of Web content and to identify and describe relationships between data published on the Web. Since most content is published independently in a variety of formats that cannot directly be parsed and 'understood' by software agents, the Semantic Web solution is to introduce a metadata framework that provides an encoding and interpretation mechanism so that resources can be described in a way that particular software can understand it. The Resource Description Framework (RDF) is a common specification framework to express resource metadata, in a form software can readily process. The defining elements of the RDF are: *resource*, *property* and *assertion* (*statement*). A resource is anything that can be assigned a URI. A property is named entity that state relationships between resources or from resources to data values. An assertion is a statement about some relationship, as a combination of a resource, a property and a property value.

RDF properties may be thought of as attributes of resources and in this sense correspond to traditional attribute-value pairs. RDF properties also represent relationships between resources.RDF however, provides no mechanisms for describing these proper-

ties, nor does it provide any mechanisms for describing the relationships between these properties and other resources. That is the role of the RDF vocabulary description language, RDF Schema. RDF Schema defines classes and properties that may be used to describe classes, properties and other resources. To do all this, RDFS uses frame-based modeling primitives from AI, such as 'Class', 'subClassOf', 'Property' and 'subPropertyOf'. RDF and RDFS provide a standard domain-neutral model (mechanism) to describe individual resources. The model neither defines the semantics of any application domain, nor makes assumptions about a particular domain. Defining domain-specific features and their semantics requires additional facilities.

Domain Conceptualization - Ontologies: Semantic-level interoperation among Web applications is possible only if semantics of Web data are explicitly represented on the Web, in machine-understandable form. To make Web content machine-understandable, Web resources must contain semantic markup or descriptions that use the vocabulary (terminology) defined by ontologies [61]. The vocabulary defined by ontologies is different from human-oriented vocabularies such as glossaries and thesauri in that it provides logical statements that describe what the terms are, how they are related to each other, how they can or can not be related to each other. Each ontology provides a description of the concepts and relationships that can exist in its domain and that can be shared and reused among different intelligent agents and applications.

Ontologies play multiple roles in the architecture of the Semantic Web: i) they establish semantic interoperability on the Web by mapping Web data with domain concepts defined by shared ontologies; ii) they enable Web-based knowledge processing, sharing, and reuse among applications; iii) they enable intelligent services such as information brokers, semantic search agents, information filters and intelligent information integration.

Ontology Web Language (OWL) [99] is the commonly accepted language for encoding ontologies nowadays. It is direct successor of DAML+OIL [106] Semantic Web language, which resulted from merging two other Web ontology languages DAML [64] and OIL [45], both of which were heavily influenced by RDF(S). Like its predecessors, OWL provides a set of XML elements and attributes, with well-defined meanings, which are used to describe domain concepts and their relationships in an ontology. In addition, OWL enables further constraints on the relationships among concepts, including cardinality and domain and range restrictions such as union and disjunction. A distinctive feature of OWL vocabulary is its extreme richness in describing relations among classes, properties, and individuals. For example, we can specify in OWL that a property is *Symmetric*, *Transitive*, *InverseOf* another one, or *EquivalentOf* another one. Also, we can state that a class is defined to be an *IntersectionOf* or a *UnionOf* some other classes. Similarly, a class instance can be the *SameIndividualAs* another instance, or it can be required to be *DifferentFrom* a certain other instance.

Logic - Queries and Rules: Once Web content is described and ontological metadata published, focus turns at utilizing the accumulated semantics. Resource discovery is the

first step in utilizing the Web data. The basic mechanism is the query - we ask for information that fulfills particular requirements. A query is a collection of one or more rules, explicit or implied, that logically define the parameters of the information we seek [37]. Two general approaches to query RDF metadata can be distinguished: i) SQL/XQL style approach, which views RDF metadata as a relational or XML database and devises API methods to query the object classes; ii) KB/KR style approach, which views the linked structure described by RDF metadata as a Web knowledge base, and applies knowledge representation and reasoning technologies on it. Many RDF query languages have been developed such as SquishQL, N3, RDQL, SeRQL and SPARQL. Nowadays, SPARQL as a W3C recommended query language, is commonly accepted in the Semantic Web community.

Aside from resource discovery, accumulated semantics accompanied by appropriate ontologies and specified rules can act as fuel for reasoning systems. The term rules in the context of the Semantic Web refers to elements of logic programming and rule-based systems bound to Semantic Web data. Rules capture dynamic knowledge as a set of conditions that must be fulfilled in order to achieve the set of consequences of the rule. They offer a way to express constraints on the relationships defined by RDF and can be used to discover new implicit relationships. The Rule Markup Initiative (RMI) has taken steps towards defining a shared Rule Markup Language (RuleML). In the context of the Semantic Web, the Semantic Web Rule Language (SWRL) extends OWL with first-order logic (FOL) based on RuleML. SWRL covers the entire rule spectrum, from derivation and transformation rules to reaction rules [123]. It can thus specify queries and inferences in Web ontologies, mappings between Web ontologies, and dynamic Web behaviours of workflows, services, and agents.

2.3.2 Semantic Web of Linked Data

The term Linked Data refers to a set of best practices for publishing and collecting structured data on the Web [12]. Traditionally, data published on the Web has been made available as chunks of row digital content stored in XML, or marked up as HTML tables. In the conventional hypertext Web, the meaning of the relationship between two linked documents can only be implicitly distinguished, as HTML does not provide elements that enable typed links between documents nor between individual entities described in particular documents.

Over few past years, the adoption of the Linked Data best practices has lead to the creation a global information space, connecting data from different sources such as scientific publications, music stores, television and radio programs, on-line communities and business records. This global information space is usually refereed as a Linked Open Data Cloud. Upon linked data that adheres to the Linked Data principle, a number of new types of applications has been enabled. Two most extensively considered application types are the linked data browser and linked data search engines. The linked data browser allows users to start to browse data in one data source and then navigate along

links into related data sources. The linked data search engines crawls the linked data cloud and provides expressive query capabilities over linked data, similar to how a local database is queried today. What makes a significant difference between linked-data enabled applications in general and the Web 2.0 mashups is that linked data applications operate on top of unbound, global data space, contrary to mashups which operate against a fixed set of data sources.

In order to get all data published on the Web to be a part of a single global data space, data publishers have to adhere to the following 'rules' [9], which have become known as the 'Linked Data principles:

- Use URIs as names for things;
- Use HTTP URIs so that people can look up those names
- When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL)
- Include links to other URIs, so that they can discover more things

By publishing data according to the Linked Data principles allows data to be discovered and used by various applications. The process of publishing a data set as Linked Data on the Web involves: assigning HTTP dereferenceable URIs to the entities described by the data set, setting RDF links to related data sources on the Web thus enabling clients (both humans and software agents) to navigate the Web of Data by following RDF links, and providing metadata about published data. RDF links represent a key prerequisite for published data to be discovered and processed afterwards. In most cases data sources provide information about large numbers of entities. Therefore, it is common practice to use semi-automated or fully-automated approaches to generate links.

Inspire of a number of different interpretations of the Semantic Web vision [8, 65], what all of them have in common is the original goal of building a global Web of machine-readable data. According to Berners-Lee [82], "The first step is putting data on the Web in a form that machine can naturally understand, or converting it to that form. This creates what I call a Semantic Web - a web of data that can be processed directly or indirectly by machines". This actually means that while the semantic Web is a goal, Linked Data provides the means to reach that goal. Over time, with Linked Data as a foundation, some of more advanced applications associated with the Semantic Web vision, such as intelligent software agents (Section 2.3.4) are more likely to become a reality.

By following trends of publishing data, considerable efforts have also been put in building applications that exploit published data. Three main categories of applications, which have been built upon Linked Data, are: Linked Data browsers, Linked Data search engines, and domain-specific Linked Data applications. Similar to traditional Web browsers which enable users to navigate across the Web by following hypertext

links, Linked Data browser enables users to navigate between data sources by following RDF links. Linked Data search engines crawls Linked Data and provide query capabilities over aggregated data [31]. In general, they can be classified into two categories: human-oriented and application oriented. Finally, a number of services offering more domain-specific functionalities by 'mashing-up' data from various Linked Data sources. Examples of domain-specific applications include Revyu [60], DBpedia Mobile [6] and Talsi Aspire [21].

A possible problematic area of Linked Data, which might rise some concerns, is the opportunity to violate privacy of integrated data from distinct sources. Some interesting research initiatives in this domain include are Weitzner's work on the privacy paradox [132] and the work by the TAMI project on information accountability [133].

2.3.3 Semantic Web of Intelligent Software Agents

Deploying generic mechanisms on the Web to facilitate collaboration between humans is important in its own right, but the Semantic Web advocates a much broader view. By taking advantage of semantically marked up data and provided services, a variety of diverse software agents can be developed to facilitate the full span of possible collaborative processes [63]: i) human and human (mediated by the tools and protocols, often called PP in the sense of person-to-person, or even people-to-people, as a variant of the technical term P2P for peer-to-peer); ii) human and machine (person-to-agent, often called PA); and iii) machine and machine (agent-to-agent, often called AA).

What the Semantic Web brings to the situation are the predicate-based structures to express meaningful assertions, and the ontologies and rules to enable intelligent agents to parse meaning from these assertions (sentences). Intelligent agents will not be able to 'think' like their human counterparts, but will be able to reason logically around relationships (explicit and implicit) and infer valid new ones. Although there is some agreement on general characteristics of intelligent agents, the decision to call a piece of software an agent is an arbitrary one [52]. Generic features that distinguish intelligent agents from ordinary application software are: autonomous behavior, reactive behavior, proactive behavior, social ability, intelligence, cooperation and mobility. Intelligent agents can act independently on behalf of a user to perform goal-oriented tasks. They can perceive and respond to changes in their environment and communicate with the user, the system, and other agents. Moreover, agents can work with other agents to perform more complex tasks than they themselves can handle. Finally, agents can move between host systems to access remote resources or arrange local interactions with remote agents.

Although the main focus of the Semantic Web is to move some of the information processing to intelligent agents it also tries to facilitate person-to-person collaboration. Social Network Services (SNSs) (e.g., Friendster, MySpace, LinkedIn and Facebook), which have become very popular recently, provide a multitude ways for the Web users to colaborate. By using SNSs, people are now better connected and can easily access, reuse

or comment content that is authored by other people from the same social network. However, a big diversity of SNSs hampers the collaboration between people who belong to different social networks. In other words the current social networks act as isolated islands of connected people and their contents. One possibility to overcome this lack of networks interoperability is to leverage Semantic Web technologies into social networks - interconnecting both content and people in meaningful ways [16]. By using agreed-upon semantic vocabularies (ontologies) to describe people, shared content and the connections that bind them all together, SNSs can interoperate via common semantics. Thus the Semantic Web brings the potential to connect disconnected social networks into a unified network of whole Internet users. On the other hand, SNSs provide the possibility of creating valuable semantic metadata - and the Semantic Web still lacks sufficient (valuable) metadata.

2.3.4 Semantic Search

Since the very beginning of the Semantic Web story, a broad range of semantic document retrieval approaches has been developed in the context of the Semantic Web. While traditional document search mostly relies on the occurrence of words in documents, semantic search refers to a document retrieval process that exploits domain knowledge, usually formalized by means of an ontology. An increasing number of information retrieval systems have started to use domain ontologies to come up with semantic annotation of documents and to help the users clarify their information needs. It is usual that this kind of information retrieval systems are also called ontology-based or concept-based IR systems.

The majority of existing concept-based semantic search approaches [110, 44, 68, 15, 128] is focused on identifying occurring of ontological concepts in a document by utilizing concept descriptions (e.g., concept labels) and applying different NLP to analyze the document's content. This process is known as a simple syntactic concept matching [81]. Ontological concepts discovered in this way (syntactic concept matches) are then used for the ontological annotation and concept-indexing of the document. There are only few approaches that also try to assess the relevance of the syntactic matches for the document they annotate. In [100] the concepts' relevance is calculated based on the frequency of concept occurrence in the document. The approach presented in [44] calculates the concepts' relevance by analyzing the link structure of the underlying knowledge base. To the best of my knowledge, the issue of determining the concepts' relevance based on ontology features has only been addressed in the approach presented in [100]. This approach extends traditional *tf-idf* method by taking into account the global usage of concepts, individuals and triples in the annotations.

Some approaches [62, 119, 67, 131] apply lexical semantics (from lexical vocabularies such as WordNet) to extend the set of discovered syntactic matches. This process is usually referred to as a lexically-expanded syntactic concept matching. Moreover, the structure and formal semantics of the underlying knowledge bases can be used to

further modify the set of discovered syntactic matches [81]. In the literature, those approaches are known as semantic matching approaches. Most existing semantic matching approaches can be classified into two categories: approaches that relay on the graph traversal of underlying knowledge bases [57, 31, 1, 110] and approaches based on formal semantics of RDF, RDFS and OWL [78, 3, 36]

Besides semantic document annotation, ontologies have also been used in the process of enriching and disambiguating user queries. Approaches presented in [15, 128, 1, 57] use ontologies as thesauri containing synonyms, hypernyms and hyponyms for the query terms, and do not consider the context of each term, that is, every term is equally weighted. [28] presents a probabilistic query expansion model based on a similarity thesaurus which was constructed automatically. In that approach the query is expanded by adding terms that are similar to the concept of the query, rather than selecting terms that are similar to the query terms. Similarly, in [54], a query expansion is performed by selecting additional terms from those that are connected to the same concepts as the user's query terms.

2.4 The Social Semantic Desktop

As on the Web, traditional desktops face a lack of interoperability of data managed by different applications. Each desktop application has its own data, unaware of related and relevant data in other applications [55]. The idea of using Semantic Web technologies to enhance data interoperability and information management on the personal desktops has lead to the creation of the semantic desktop paradigm. The core idea of the semantic desktop is to enable data interoperability on the personal desktop by applying Semantic Web standards and technologies [27]. Formal ontologies should be employed to capture both a shared conceptualization of desktop data and personal mental models [115]. RDF (Resource Description Format) should serve as a common data representation format.

Moreover, despite great improvements in sharing personal data throughout a networked infrastructure, personal desktop environments are still Ôclosed-worldsÕ that mainly focus on individualsÕ data and still there is no efficient interoperability between data stored on different desktops. There are several new technologies, which could provide a means to build the semantic bridges necessary for data exchange and application integration as well as dramatically impact a way in which people interact and collaborate: the Semantic Web, peer-to-peer computing and online social networking. Stefan Decker presented in [27] a vision of how these different thrusts will evolve and produce so-called, Networked or Social Semantic Desktop (SSD), which "enables people and communities to directly collaborate with their peers while dramatically reducing the amount of time they spend filtering and filing information". SSD is considered as a broader concept of semantic desktop.

The objective of the social semantic desktop is twofold. Firstly, it should provide data and application interoperability on personal desktops, which is actually the objective of the semantic desktop paradigm. Secondly, it aims at connecting personal desktops into a unified information space of social communities [55], which is considered to be a social desktop paradigm. In the following two sections overview trends and recent developments in both of these two paradigms.

2.4.1 The Semantic Desktop

Over past several years a number of Semantic Desktop projects such as [103, 115, 32, 49] have the goal of providing a semantic infrastructure that covers all applications and integrates information sources that desktop users operate on. In general of these projects attempt to enhance the existing desktop infrastructures by adding an additional semantic layer, providing semantic descriptions that refer to actual resources. In such scenarios, the semantic integration of desktop resources should happen at the semantic layer by interlinking descriptions of semantically related resources instead of linking actual resources. The main problem here is the propagation of modifications to resources and their relationships to the semantic layer. To the best of our knowledge, only the approach presented in [116] offers the model that integrates the semantic layer into the actual desktop file system, enabling explicit semantic links between desktop files. However, the model recognizes only entire files as identifiable and linkable resources. In the rest of this section I briefly present several semantic desktop related projects (initiatives), which in my opinion made a significant progress regarding realization of the semantic desktop paradigm.

Haystack [103] is an integrated approach which enables individual users to manage their personal information in a way that makes the most sense to them. Integrated word-processors, email client, image manipulation, instant messaging and other functionality removed application-created barriers of information representation and accessibility. Moreover, haystack provides a complete semantic programming environment, from user interface to database. The project was ground-breaking in terms of the dynamic creation of user interfaces, but it ended before establishing any standards.

Gnowsis [115] semanic desktop was a first research project targeting a Semantic Desktop system. The main goal of Gnowsis was to complement established desktop applications and the desktop operating system with Semantic Web features, rather than replacing them. A particular focus was put on Personal Information Management (PIM) [115]. In addition, the project addressed the problem of how to identify and represent desktop resources in an unified RDF graph.

Semex (SEMantic EXplorer) [32] is another platform for personal information management. Semex has two main goals: 1) to enable browsing personal information by semantically meaningful associations, and 2) to enable the personal information space to be used as an anchor for importing external information sources, thereby offering an environment for performing on-the-fly integration tasks. An initial version of Se-

mex platform was integrated into the CALO Project , [10] which is a broader platform providing cognitive assistants that learn their users' behavior over time.

MyLifeBits [49] was a project aiming to fulfill the Memex vision first posited by Vannevar Bush in 1945. The project was supposed to produce a system for storing all of one's digital resources, including documents, images, sounds, and videos. The system had to fulfill the following four principles: 1) collections and search must replace hierarchy for organization, 2) many visualizations should be supported, 3) annotations are critical to non-text media and must be made easy, and 4) authoring should be via transclusion. The system view personal data as a graph of information. Nodes in the graph represent documents and annotation metadata, while edges represent the annotates relationship. Moreover, it integrates text and multimedia objects, allowing to annotate a file by linking it to another file, and by manually adding text annotation or audio annotation.

The Sile Model [116] is a semantic file system infrastructure for the desktop. Actually, it can be considered a complementary virtual file system that may be integrated into a Semantic Desktop. The sile model provides an integrated view on desktop resources and associated semantic annotations, and serves as an intermediate layer between applications and actual storage infrastructure.

The first problem with all above-disacussed projects is that they do not consider collaborative work and the interconnection of semantic desktops at all. They exclusively concentrate on a single user scenario. The second problem is integration. For example Haystack provides a well evaluated user interface, but it does not reuse established desktop applications that users are familiar with, thus faces the user with a new environment. Finally, none of the projects established standards which would increase interoperability and reusability. The only semantic desktop project so far, which has also considered the interconnection of semantic desktop and established a framework and standards so that components can be reused is the Nepomuk social semantic desktop project [55].

NEPOMUK project applied technologies, originally developed for the Semantic Web, on the user's desktop. On the local scale (i.e., personal desktop environment) NEPOMUK uses the Semantic Web technologies to integrate information between applications such as email, contacts, calendars, or file-manager. On the global scale, NEPOMUK uses the Semantic Web technologies to enable the exchange of artefacts among users without loosing the meaning of the data. This way, NEPOMUK adds a value to the personal desktop in three dimensions: 1) improved personal information management, 2) improved cross-media and cross-application application linking, and 3) improved information sharing and exchange across social and organisational boundaries. The main contribution of the NEPOMUK project, regarding the realization of the SSD paradigm, was an architecture design of the SSD platform. As a member of the University of Lugano, which participated on the project, the author of this thesis was involved in several tasks related to the architecture design. The APIs designed by NEPOMUK project are being

implemented on top of several frameworks. One of them is KDE implementation of NEPOMUK APIs[1].

2.4.2 The Social Desktop

Over the past several years social interaction between people has become more important than ever. Online Social Network Services (SNS) (e.g., Twitter[2], MySpace[3], Facebook[4] and LinkedIn[5]) have brought a new way of interconnecting online content and networked people for various social and professional purposes. People are now better connected and can easily access, reuse or comment on content that is authored by other people from the network. However, in spite of being well-connected within particular social networks, the diversity of SNSs hampers the interoperability between people from social networks, which adhere to different SNSs. Current social networks act as isolated islands of connected people and their contents. Moreover, still there is a big gap between an individual desktop information space and a shared, social network's information space. One possibility to overcome this lack of network interoperability, and fill-in the gap between the individual and shared information space is to leverage semantics into social networks - interconnecting both content and people in meaningful ways [85].

Combining SNSs with Semantic Web technologies gives benefits to both social networking and the Semantic Web. By using agreed-upon semantic formats to describe people, shared objects and connections that bind them all together, SNSs can interoperate via common semantics. Thus, the Semantic Web has the potential to connect disconnected social networks into a unified network potentially encompassing all Internet users. On the other hand, SNSs provide the possibility of creating valuable semantic metadata - and the Semantic Web still lacks sufficient (valuable) metadata.

Rather than building separate social networks, the Internet infrastructure should be augmented to include a social networking infrastructure, making social networking a shared component across various desktop and Web applications. The social desktop (SSD) [55] paradigm is about extending a personal desktop into a collaborative environment that supports information and content sharing across social and organizational relations. It aims to transform the conventional desktop into a seamless, networked working environment, by losing the borders between individual applications and the physical workspace of different users [115].

Microsoft SocialPC [83] is ongoing project which aims to extend Windows operating systems by adding a social desktop infrastructure. Actually, SocialPC attempts to embed the web oriented model for sharing in the center of the desktop experience,

[1] http://nepomuk.kde.org/
[2] http://twitter.com/
[3] http://www.myspace.com/
[4] http://www.facebook.com/
[5] http://www.linkedin.com/

demonstrating new ways of integrating the desktop and the web. Every desktop document should have a backing social URL for sharing without having to upload or copy or move it from its natural location. By using this link, online users are able to access not just the file, but also a built-in social experience which includes a rich preview of each item, comments, related items and tags. Whenever some users comment with this social link via the web browser, the comment is also available directly in Windows, and vice versa.

NEPOMUK social semantic desktop provides an operating system layer that extends the personal desktop into a collaboration environment that supports both personal information management and information and content sharing and exchange across social and organizational relations. NEPOMUK is based on a set of carefully designed and integrated ontologies for the social semantic desktop like PIMO (Personal Information Model Ontology) [115] and SIOC (Semantically Interlinked Online Community) [85]. While PIMO provides formal representation of the mental model of a desktop user, SIOC provides a mechanism to integrate online community information (e.g. discussion methods such as blogs, forums and mailing lists). NEPOMUK uses a P2P infrastructure as a communication medium, thus avoiding centralized SNSs whose maintenance requires a major investment. Profile and user information remains the property of individual users and multiple social software applications can crawl it.

The social desktop opens the way for a range of new social software applications that can employ all person-to-person connections and access shared objects of interests through the social desktop infrastructure.

Chapter 3

Semantic Documents Modeling

The form of documents has been changing over time following the development of the society. One of the key changes happened with the introduction of 'Digital Era', which led to the main classification of documents into paper documents and electronic documents. The principle difference between paper and electronic documents comes from different types of a physical medium that is used for document data storage. The data storage medium determines a way in which documents are created, managed and communicated among people. Paper documents are created manually or with the help of some mechanical devices (e.g., a typewriter), and are communicated among people in the same manner as any other mobile, physical object. In contrast to paper documents, electronic documents are stored on digital mediums such as hard disk drives (HDDs), CD-ROMs, DVDs and flash memories. They are computer-supported in all phases of their life cycle, starting from the creation and utilization to the archival and destruction. The most popular ways of communicating electronic documents among people are document exchange by e-mails and document publishing on the Web. The latest generations of conventional electronic documents, such as word processing documents, slide presentations, spreadsheets and PDFs are structured documents, which have a visual layer separate from their data and a logical structure. The logical structure of electronic documents enables the structuring and the organization of document data into smaller units (e.g., sections, paragraphs, illustrations and tables), which can be accessed directly, by using embedded named anchors. Moreover, some electronic documents enable readers to add extra information to document units without making any changes to the actual document's data. This extra information, also known as an annotation, may help other readers to better understand the document. Despite many differences between paper and electronic documents, the purpose of documents remained unchanged. Both paper and electronic documents serve as a medium for the information and data sharing among humans.

Electronic desktop documents hold a significant part of the data stored on local, personal desktops as well as data hosted on the Web, and represent important source of data

for the future Semantic Web. However, data held in conventional electronic documents do not meet the requirements demanded of the Semantic Web [9]. The transformation/adjustment of electronic documents data is necessary, which will lead to the new generation of documents that we refer to as semantic documents. So far, annotating electronic documents by machine-processable, meta-level descriptions [127] has been the most common strategy to adapt electronic documents to the Semantic Web. However, it is possible to state that the ultimate goal of semantic documents is not merely to provide annotations for document data, but to provide two representations of a knowledge (human readable (HR) and machine processable (MP)) stored into documents, and to provide platform/tool independent, unified view of the HR document representation. In other words, semantic documents should enable document data and knowledge to be sharable and understandable not only by humans but also by machines. If we consider a paper document as a printed copy of an electronic document, then an electronic document can be considered as the HR representation of a semantic document.

In this chapter I present my solution for the design of the semantic document model, which represents a foundation of the new document generation. I start with the design principles of semantic documents, then I describe the semantic document model in detail and continue with the discussion on the two document representations (i.e., MP and HR) of semantic document which stem from the model. I conclude the chapter with some remarks on the model's design.

3.1 Semantic Documents Design Principles

So far, the term semantic document has been referring mainly to documents annotated by concepts from ontologies [40]. The process of annotating documents by ontological concepts is also known as a semantic document annotation. Ontological concepts that annotate a document, together with the underlying ontologies that define them, intend to enable intelligent software agents to search documents in a more meaningful way, comparing to the traditional content-based search, and to discover desired information/knowledge. Over the past few years, a considerable number of ontology-based semantic annotation approaches has been developed [44, 75, 110, 15, 9]. While differing in many aspects, they all attempt to enhance documents by adding an additional, semantic layer containing conceptualized semantic descriptions (i.e., ontological concepts) that refer to actual documents.

In my vision, the term semantic document does not denote semantically annotated electronic documents, but rather a new document form in which the semantic layer is integrated inside the document representation. In other words, the semantic layer is not considered as an additional layer in a document representation, but as integral part of it. In the rest of the section I describe a set of design principles, which I identified as fundamental for semantic documents.

Document data granularity: Semantic documents are composite information resources, composed of a number of smaller resources called document units. Each document unit is characterized by the binary data and the information provided by that data. There are two types of document units: atomic document units and composite document units. Each atomic document unit has one content stream, connecting the binary data (e.g., text, image, audio and video) to the document unit. Depending on the concrete implementation of a semantic document, the binary data of document units can be kept into the actual representation of the document or stored into an external binary data repository. Composite document units aggregate a number of atomic or other composite units and add navigation to them.

Document data identity: Semantic documents are uniquely identified by globally unique resource identifiers (URIs). Moreover, each document unit within a semantic document, wheither atomic or composite, is identified by a globally unique URI.

Document data annotation: Semantic document annotations are entities which are identified by the annotation identifier, the annotation type and the annotation body. The annotation body is determined by the annotation type and can hold a data-value or reference to another entity. The annotation types, which I am focused on in this thesis, are: standardized metadata, semantic (ontological) annotations, and social-context annotations. All of these annotation types will be considered in several sections of the thesis (Section 3.2.2, Section 4.2.4, Section 5.1.2). It is important to point out that a potential, new annotation type can be introduced by anyone providing the name of the annotation type and specifying the annotation body. The semantic document annotations can be added to all levels of a document granularity: a whole document, composite document units and atomic document units (Section 3.2.1).

Document data linking: There are two main types of links which can be established between semantic document units: structural links and semantic links. Structural links are used to build a logical structure of a semantic document. Semantic links are used to explicitly represent the semantic relations between document units. Both, structural and semantic links can be established between document units which belong to the same or different semantic documents. However, only those document units connected by structural links are considered to belong to the same document. Actually, it means that the document units that are connected by structural links to document units of many different documents, belong to each of those documents. Accordingly, document units can be easily added or removed from a semantic document, simply by linking or unlinking them from the document. Document units can have an arbitrary number of semantic links, connecting them to other document units. In principle, if two document units are annotated by the same semantic annotations (ontological concepts) then the two units share some semantics and there can be identified one or more semantic relations between them. The semantic links can be then established between these document units. Finally, semantic document units can be linked with any other uniquely identified data (resources) on the Semantic Web. Considering that semantic document units are

encapsulated fractions of document data, this linking fulfills one of the main principles of the Semantic Web, that is, the transition from linking documents to linking data [12].

Document data universality: Semantic documents are not owned by a single application. They are universal and platform/tool independent. They are transferable across platforms and are easy to port and integrate to existing applications on any platform. Appropriate transformation functions which map conventional document formats to and from the semantic documents can be provided. In that case, semantic documents can also serve as an intermediate step in the document transformation from one conventional, application-specific document format to another. In this way the number of necessary transformations for N target applications (document formats) will be reduced from N^2-N to $2N$. Where N^2-N is a number of all one-to-one transformations.

Document data accessibility: Semantic documents are completely open and queryable. Humans and software agents can search semantic documents, by searching document units based on their binary content or conceptualized semantics. In search results, the retrieved document units are represented by their URIs. Applications, such as semantic document browsers use obtained document unit URIs to access the MP representations of the document units and then render the HR representations that can be perceived by a user. Moreover, semantic document browsers provide an exploratory interface through which the user can interact with the retrieved document units, and can traverse semantic documents by navigating along the semantic links between document units.

Document data traceability: Over time semantic documents change and evolve through a number of versions. In order to verify the evolution path of document units, each document unit has a version identifier in addition to its URI. Moreover, semantic documents provide mechanisms and structures for capturing and formal representation of changes that can be made to document units. Similar to the representation of the semantic document annotations, the document unit changes are represented as uniquely identified entities which are linked into the MP representation of the document units, and are characterized by a set of specific properties that hold information about the changes. By using such formalized change representations, previous versions of document units can be rebuilt and re-deployed. This is especially important in case of document revisions when previous versions do not exist any more.

In accordance with the above listed design principles, I give the following definition of semantic documents:

➥ ***Definition of Semantic Documents***
Semantic documents are composite information resources composed of uniquely identified, semantically annotated, and semantically interlinked document data/information units of different granularity. Each semantic document is characterized by unique permanent machine-processable (MP) representation and a number of temporal human-readable (HR) representations rendered from the MP representation.

Two categories of potential users, that is, humans and machines (i.e., software agents), determine two possible forms of the semantic document representation. The HR representation uses conventional content types such as text, images, audios and videos to represent information stored in a semantic document. The MP representation uses conceptualized semantics (i.e., ontological concepts) and semantic links between document units to represent document information in a conceptualized, machine processable form.

3.2 Semantic Document Model

Starting from the design principles and the given definition of semantic documents, which were discussed in the previous section, I have developed a novel document representation model called Semantic Document Model (SDM) [86]. The roots of the model come from the Abstract Compound Content Model (ACCM) [89] that I had created previously with a goal to organize digital media contents into semantically annotated, compound content units with the given usage objectives.

The formal specification of SDM is done by the SDM ontology. The reason I chose an ontology to formally specify the model lies in the fact that ontologies have recently emerged as one of the most effective approaches to the modeling of different structures used in information systems, as well as to the knowledge conceptualization and sharing among applications. Therefore, ontologies seem to be a good solution to modeling of document logical structure and the document knowledge. The SDM ontology consists of four parts: the core part, the annotation part, the semantic-linking part and the change-tracking part. I now discuss each part of the ontology and describe the main classes and properties. Appendix A contains a full specification of all classes and properties of the SDM ontology. The ontology is also available at our project web site[1].

3.2.1 SDM Ontology - the Core Part

The core part of the SDM ontology provides a vocabulary (classes and properties) that defines possible types of document units and possible structural relationships among them. Two main types of document units (DUs) present in semantic documents are atomic DUs (`sdm:AtomicDocumentUnit`) and composite DUs (`sdm:CompositeDocumentUnit`). An atomic DU contains a single unit of raw digital content that exists as a physical entity independently of the document unit it belongs to and cannot be disaggregated into smaller units. These units of raw digital content are called data objects and they are specified by `sdm:DataObject` class in the ontology. Data objects are included into atomic document units via `sdm:hasDataObject` property. Moreover, data objects can be specialized into discrete data objects (`sdm:DiscreteDataObject`) such as `sdm:graphic` and `sdm:textFragment`, and continuous data objects (`sdm:ContinuousDataObject`) such

[1] `http://www.semanticdoc.org/`

as `sdm:audio`, `sdm:video` and `sdm:animation`. A composite DU aggregates a number of atomic or other composite DUs and organizes them in a given order. `sdm:hasPart` and `sdm:isPartOf` are two inverse properties used to express containment relationships between the composite DU and the document units it is composed of. The order of DUs within the composite DUs is realized by the value `sdm:order` property, which holds the ordered list of the DUs' URIs. The properties `sdm:hasPart`, `sdm:isPartOf` and `sdm:order` enable modeling structural relationships within semantic documents. Some examples of composite document units are `sdm:section`, `sdm:paragraph`, `sdm:slide` and `sdm:table`. A whole `sdm:document` is also a composite DU that is different from other composite DUs only in that it cannot not belong to any other composite DU. Figure 3.1 gives a simplified, graphical representation of the core part of the SDM ontology. The specification of all classes and properties of the SDM core vocabulary is given in Appendix A.1

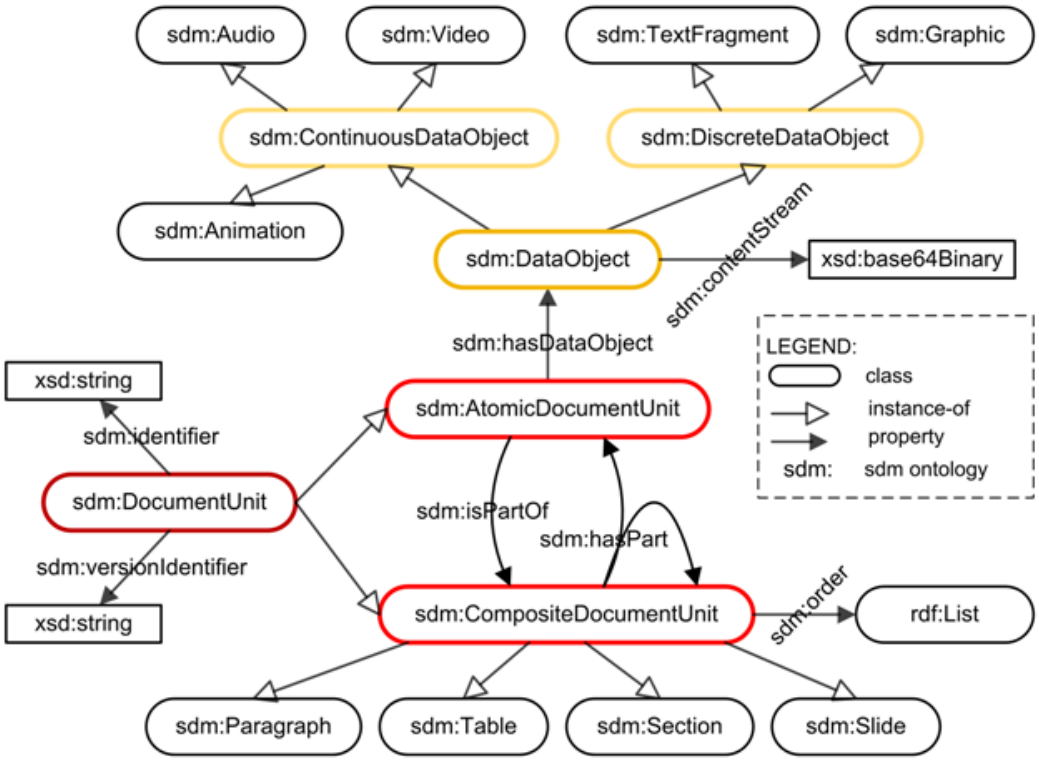


Figure 3.1. SDM Ontology - Illustration of the core part

3.2.2 SDM Ontology - the Annotation Part

The annotation part of the SDM ontology provides a vocabulary that defines the annotation types of the DUs and the interface for adding annotations to DUs. According to the design principles, which identify three possible types of the DUs annotations, the annotation vocabulary introduces the three corresponding classes `sdm:SemanticAnnotation`, `sdm:MetadataAnnotation` and `sdm:SocialContextAnnotation`, all of them being subclasses of the abstract, `sdm:Annotation` class. The annotations are linked to the DUs via `sdm:hasAnnotation` property. Figure 3.2 gives a simplified, graphical representation of the annotation part of the SDM ontology. Depending on the annotation type, the annotations may contain different properties. I now describe each of the annotation types along with their ontological properties.

The metadata annotation refers to the annotation of document units with standardized metadata, which is specified by internationally recognized vocabularies such as Dublin Core (DC) and IEEE Learning Object Metadata (LOM). These vocabularies are designed to describe any kind of resource, that is, anything that has identity. I have chosen a subset of this metadata elements, which is meaningful for document units, and incorporated it in the SDM annotation vocabulary. For example: `dc:creator`, `dcterms:created`, `dc:format`, `dc:language`, `dc:title` and `dc:description` referring to the author(s), creation date, media type, language(s), title and short description respectively. A list of all standardized metadata elements incorporated in the SDM vocabulary can be found in Appendix A.2.

The semantic annotation of DUs, is the annotation by ontological concepts that represent the conceptualization of the information/knowledge held by DUs. Viewed from the other side, DUs could be considered as the human readable description of the ontological concepts which annotate them. However, in reality it is not common that DUs contain exact description of the ontological concepts, unless they belong to documents that describe the ontology in which the concepts are specified. It is more realistic that DUs contain descriptions of some aspects of the ontological concepts. Therefore, the semantic annotations of the DUs should contain properties which determine the relevance of the annotation concepts to the DU they annotate. Accordingly, the SDM annotation vocabulary provides two properties: the `sdm:ontologicalConcept` which holds the reference to the ontological concept and the `sdm:conceptWeight` which holds the relevance weight of the concept to the DU.

In accordance with the design principles, which define semantic documents as completely open resources with easy access to DUs, semantic documents are well suited for social environments such as emerging online social networks. In fact, semantic documents are recommended to be used within social networks. The social-context annotations are introduced to capture relevant information about the user actions performed to DUs such as the browsing, reusing and modification. Every time a user interacts with a DU (i.e., performs an action on it), a new social-context annotation is added

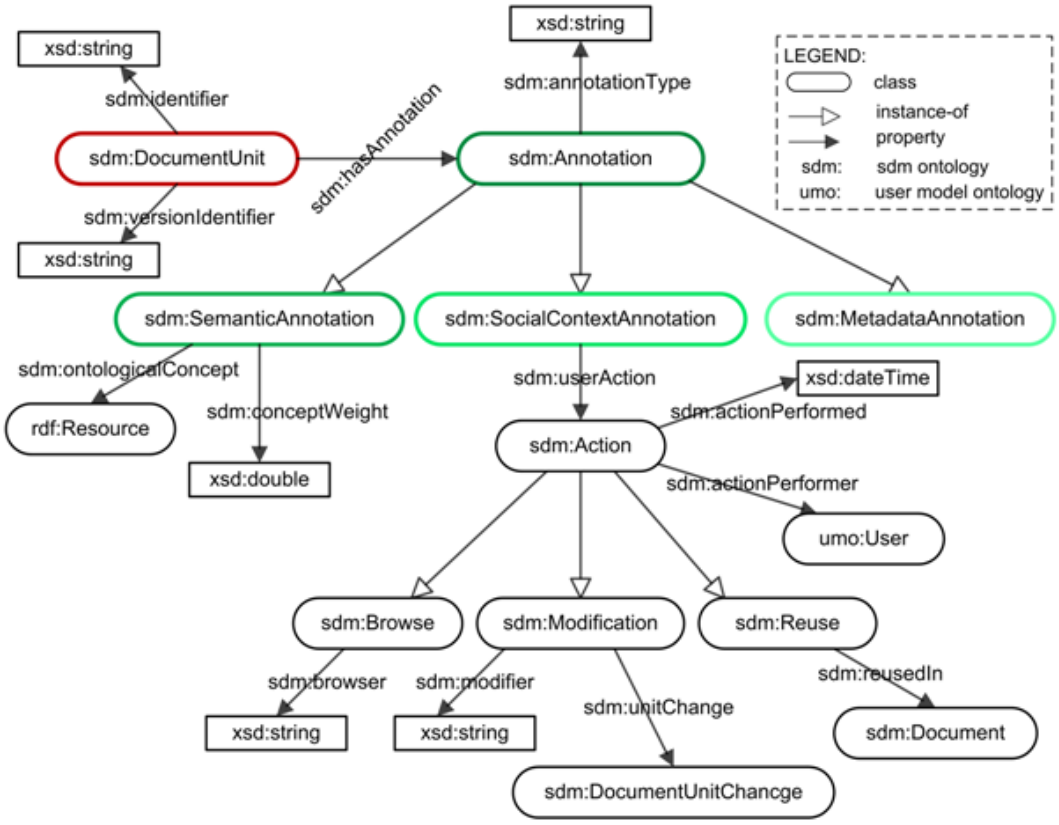


Figure 3.2. SDM Ontology - Illustration of the annotation part

to the DU. In order to capture the information about the users actions, the SDM annotation vocabulary introduces the abstact `sdm:Action` class along with two properties `sdm:actionPerformed` and `sdm:actionPerformer`. The first property holds the information about the date and time when the action has been performed, while the second property holds the information about the user who has performed the action. The `sdm:Action` class is a superclass of `sdm:Browse`, `sdm:Reuse`, and `sdm:Modification` classes, which are suited for more specific users actions, namely the document unit browsing, reusing and modification. The `sdm:hasAction` property attaches instances of these classes to the representation of the DUs on which the actions have been performed. Moreover, each of the three classes has its specific properties related to specific characteristics of the actions. For example, the `sdm:Browse` class has the `sdm:browser` property that holds information about the software used for browsing the DU's data. Moreover, the `sdm:Reuse` class has the `sdm:reusedIn` property that holds the information about the semantic document in which the DU has been reused. Finally, the `sdm:Modification` has `sdm:unitChange` property that holds the conceptualized representation of the change made to the DU, as well as the `sdm:modifier` property that

holds the information about the software used for the DU modification. The change is represented as an instance of the `sdm:DocumentUnitChange` class, which is specified in the SDM change-tracking vocabulary, which I describe later in this section.

3.2.3 SDM Ontology - the Semantic-Linking Part

The semantic-linking part of the SDM ontology provides a vocabulary that defines the interface for linking semantically related DUs (Figure 3.3). It consists of the `sdm:SemanticLink` class and the following properties: the `sdm:unitOne` and `sdm:unitTwo`, which hold references to document units linked by the semantic link; the `sdm:linkingConcept` that holds the reference to the ontological concept that conceptualizes semantics shared between the linked document units and determines the semantic relation between them; and the `sdm:linkStrength` property whose value determines the strength of the semantic relation, that is, semantic relatedness between the units. Together with the semantic annotations, the semantic links represent key elements in the machine processable representation of the document knowledge. Moreover, the semantic links enable the semantic navigation across integrated collections of semantic documents and thus help in the discovery of semantically related DUs.

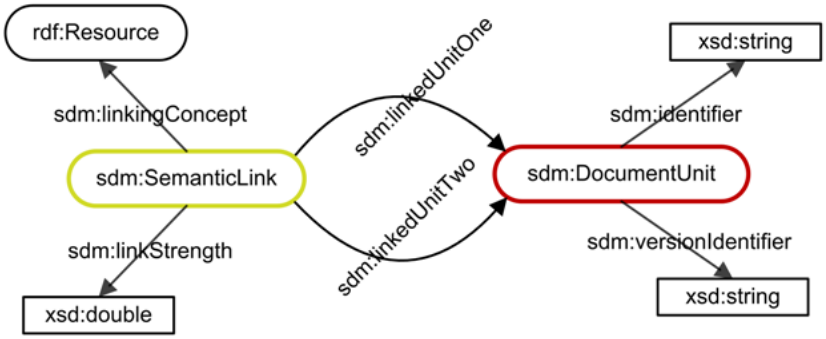


Figure 3.3. SDM Ontology - Illustration of the semantic-linking part

3.2.4 SDM Ontology - the Change-Tracking Part

The change-tracking part of the SDM ontology provides a vocabulary that defines possible changes of DUs as well as the whole semantic document (i.e., the logical structure of semantic document). Two main classes of the vocabulary (Figure 3.4) are the `sdm:AtomicUnitChange` and the `sdm:CompositeUnitChange`, which are both subclasses of the abstract `sdm:DocumentUnitChange` class. The `sdm:AtaomicUnitChange` class is used to describe a change made to an atomic DU. Since the atomic DU contains only one data object, the change to the DU is actually the change to its data object. According to the definition of the data object, which defines the data object as a unit of

raw digital content that exists as a physical entity and which can not be disaggregated into smaller units, any change made to the data object creates the new version of it. In order to keep track of this kind of changes, the presence of both the old and the new versions of the data object is necessary. Thereby, I defined the `sdm:oldDataObject` and `sdm:newDataObject` properties, which hold the references to the old and the new versions of the data object, respectively.

The `sdm:CompositeUnitChange` class is used to describe a change made to the composite DU. Based on the definition of the composite document units, I have identified the following types of possible changes: 1) addition of a new DU; 2) subtraction of a constitutive DU; 3) reordering of constitutive DUs; 4) change to a constitutive atomic DU; and 5) change to a constitutive composite DU. Changes of the first two types are captured by the `sdm:addedDocumentUnit` and `sdm:subtractedDocumentUnit` properties, which link the added and subtracted DUs to the instance of the `sdm:CompositeUnitChange` class. Changes of the third type are captured by the `sdm:oldUnitsOrder` and `sdm:newUnitsOrder` properties, which link instances of the `rdf:List` class that keep the new and old order of the constitutive DUs of the composite DU. If the change to the composite DU comes as a result of the changes to its constitutive DUs (i.e., the change of the type four and five), then instances of `sdm:AtomicUnitChange` and `sdm:CompositeUnitChange` classes are linked to the DU's instance of the `sdm:CompositeUnitChange`. The `sdm:hasAtomicUnitChange` and `sdm:hasCompositeUnitChange` properties are defined for this purpose. Since the whole semantic document is also represented as a composite DU, the changes made to it are represented in the same way as the changes made to any other composite DU.

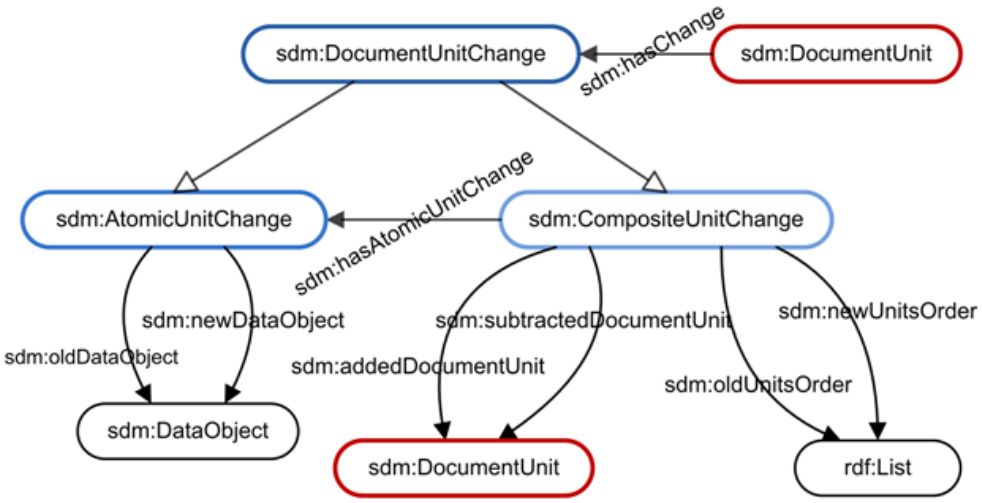


Figure 3.4. SDM Ontology - Illustration of the change-tracking part

3.3 Authoring Facts of the SDM Ontology

The SDM ontology is authored in OWL 2 (Web Ontology Language). OWL is a family of knowledge representation languages for authoring ontologies endorsed by the World Wide Web Consortium [121]. Figure 3.5 shows a portion of the RDF/XML-based serialization of the ontology containing the XML namespace declarations, the declarations of the `sdm:DocumentUnit`, `sdm:AtomicDocumentUnit`, and `sdm:CompositeDocumentUnit` classes and the declarations of some of the properties of the `sdm:DocumentUnit` class. In OWL ontologies, the XML namespaces are used to specify external vocabularies that are used in the ontology, to specify the default namespace of the ontology, and to specify the base URI of the ontology. The first namespace declaration (line 2) introduces the OWL vocabulary. Moreover, since OWL depends on constructs defined by RDF, RDFS, and XML Schema datatypes, next three namespace declarations (lines 3-5) introduce those vocabularies. The declaration from line 6 specifies the default namespace of the ontology, stating that unprefixed qualified terms, which appear in the ontology, refer to the current ontology. The last namespace declaration (line 7) specifies the base URI of the ontology, which is also the URI of the document containing the ontology's specification. Following the practice of the W3C OWL Working Group, I have chosen the ontology URI to be the same as the URL of the ontology's OWL file.

3.4 Instantiating the HR and MP Semantic Document Representations

According to the design principles, each semantic document provides two representations of the data and knowledge stored within it. One representation is intended to be used by humans, in a similar way humans use existing, conventional electronic documents, while the other representation is intended to be used by software agents, that is, machines. From the human point of view the advantage of the dual document representation is that users can continue to work with documents as before, but now can also benefit from the services, which are enabled by the use of the MP document instances. For example, the semantic document search can be used to locate and retrieve document units based on their conceptualized semantics. Moreover, by the semantic navigation the users can navigate across documents following the semantic links between related document units. On the other hand, from the machine point of view, the advantage of the dual document representation is that intelligent software agents can use and process the conceptualized document semantics in reasoning on and answering some domain-specific questions related to the documents' topics.

The MP representation, that is, the instances of the SDM described in the previous section, employs the HTTP URIs to identify document units and the Resource Description Framework (RDF) data model [9] to model document unit descriptions and structural and semantic links between the units. The use of the HTTP URIs and RDF data

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://www.semanticdoc.org/sdm.owl#"
    xml:base="http://www.semanticdoc.org/sdm.owl">
  <owl:Class rdf:ID="DocumentUnit"/>
  <owl:Class rdf:ID="AtomicDocumentUnit">
    <rdfs:subClassOf rdf:resource="#DocumentUnit"/>
  </owl:Class>
  <owl:Class rdf:ID="CompositeDocumentUnit">
    <rdfs:subClassOf rdf:resource="#DocumentUnit"/>
  </owl:Class>
  <owl:Class rdf:ID="Annotation"/>
  <owl:Class rdf:ID="DocumentUnitChange"/>
  <owl:ObjectProperty rdf:ID="hasAnnotation">
    <rdfs:domain rdf:resource="#DocumentUnit"/>
    <rdfs:range rdf:resource="#Annotation"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="hasChange">
    <rdfs:domain rdf:resource="#DocumentUnit"/>
    <rdfs:range rdf:resource="#DocumentUnitChange"/>
  </owl:ObjectProperty>
  <owl:DatatypeProperty rdf:ID="hasVersionIdentifier">
    <rdfs:domain rdf:resource="#DocumentUnit"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="hasIdentifier">
    <rdfs:domain rdf:resource="#DocumentUnit"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  .
  .
</rdf:RDF>
```

Figure 3.5. A snippet of the SDM ontology OWL file specifying DocumentUnit class and its main properties

representation model is inline with the Linked Data principles [13], so that semantic documents can be seamlessly integrated to the Linked Open Data cloud and further to the envisioned Semantic Web. Moreover, the conceptualization of the document semantics by using ontologies and the explicit semantic links between document data, can lead to the creation of a sufficient amount of semantically integrated data. This creation is

necessary for the Semantic Web to succeed.

The MP representation is actually an RDF graph, whose nodes are instances of DUs classes from the SDM ontology such as `sdm:Paragraph`, `sdm:Table`, `sdm:Illustration`, and `sdm:Slide` (see Appendix A.1) interlinked by RDF links that represent structural and semantic relationships, and to which are linked concepts from domain ontologies through the semantic annotation interface. Figure 3.6 shows a snippet of the MP representation of an example semantic document encoded in the RDF/XML syntax. The snippet contains a representation of a document unit of the 'Section' type, which is composed of five other document units of the 'Paragraph','Illustration' and 'Table' unit types, and is annotated by three semantic annotations and one social-contex annotation.

```
<rdf:RDF xmlns:sdm="http://www.semanticdoc.org/sdm.owl#"
 <rdf:Description rdf:about=sdm:Section_633970940538_3>
  <sdm:hasVersionIdentifier rdf:resource="V_0" />
  <sdm:hasAnnotation rdf:resource=sdm:SemanticAnnotation_633970940538_31/>
  <sdm:hasAnnotation rdf:resource=sdm:SocialContextAnnotation_633970940538_27/>
  <sdm:hasAnnotation rdf:resource=sdm:SemanticAnnotation_633970940538_32/>
  <sdm:hasAnnotation rdf:resource=sdm:SemanticAnnotation_633970940538_33/>
  <sdm:hasPart rdf:resource=sdm:Paragraph_633970940538_5/>
  <sdm:hasPart rdf:resource=sdm:illustration_633970940538_6/>
  <sdm:hasPart rdf:resource=sdm:Paragraph_633970940538_7/>
  <sdm:hasPart rdf:resource=sdm:Table_633970940538_8/>
  <sdm:hasPart rdf:resource=sdm:Paragraph_633970940538_9/>
  <sdm:order rdf:resource=sdm:633970940538_23/>
  <sdm:isPartOf rdf:resource=sdm:Document_633970940538_1/>
      .
      .
      .
 </rdf:Description>
</rdf:RDF>
```

Figure 3.6. An example snippet of the MP document representation encoded in the RDF/XML syntax

The MP representation of a semantic document is considered to be the only persistent instance of the semantic document. The HR representation is temporal and can be rendered from the MP representation whenever humans want to browse or edit the semantic document. The authors and authorized users of the semantic document can also edit it by incorporating changes from the HR to the MP document representation. The use of existing, well-established document formats, as HR document representations is preferable because the development of new formats and tools for their management requires expensive investment and it is not likely to happen immediately. Therefore, the initial success of the SDM demands the use of existing document formats for the HR rep-

resentations of semantic documents. However, overtime semantic documents will likely be accompanied with appropriate software that will be suited specifically for browsing and editing the semantic documents by humans. As a part of this thesis I developed the prototype software that is discussed in Chapter 6.

3.5 Summary

In order to enable document data to be efficiently shared and reused across desktop application and social community boundaries, desktop documents should be completely open and queryable resources, whose data are represented in a form understandable to both humans and machines. These are also the requirements that desktop documents need to satisfy in order to contribute to the visions of the Social Semantic Desktop [27] and further the Semantic Web [8]. With the aim of achieving these goals, I developed the SDM model. The formal specification of the model is done by the SDM ontology that consists of four parts (vocabularies): the core part (Section 3.2.1), the annotation part (Section 3.2.2), the semantic-linking part (Section 3.2.3), and the change-tracking part (Section 3.2.4)

Machine-processable instances of SDM are unique, platform independent and the only persistent representations of semantic documents. Human-readable instances of SDM are temporal and can be rendered from the documents' machine-processable representations whenever humans want to browse or edit semantic documents. The machine-processable semantic document representation employs globally unique, HTTP dereferenceable URIs to identify document units, concepts from domain ontologies to semantically annotate document units, and RDF to model structural, hierarchical and semantic relationships among document units.

Chapter 4

Semantic Document Architecture - SDArch

In order to support semantic document management and to enable users to take benefit from semantic documents, I have developed a new software architecture, called Semantic Document Architecture or SDArch. Since semantic documents are designed to replace conventional desktop documents, SDArch should be considered as an integral part of the envisioned social semantic desktop (SSD) architecture (Section 2.3). So far, there have been several initiatives to develop an SSD platform [55, 72, 115, 14, 103]. However, none of them was offering a mature solution at the time when I started to develop SDArch, thus I decided to develop SDArch as a self-contained software architecture, providing all necessary functionalities for the semantic document management including also those functionalities that SDArch should share with other systems of SSD. The SDArch functionalities are implemented by a number of services equipped with the standardized interface, so that they can be easily integrated into the SSD platform.

I dedicate two chapters, Chapter 4 (this one) and Chapter 5 to describe SDArch. In this chapter, I describe the overall design of the architecture. In Chapter 5, I provide the detailed discussion on the semantic document management processes and the architecture's services that enable them.

4.1 The SDArch Design

I designed SDArch as a three-tier, service oriented architecture. From the bottom up, the architecture's layers are the *data layer*, the *service-oriented middleware*, and the *presentation layer*. Figure 4.1 illustrates the layered SDArch architecture.

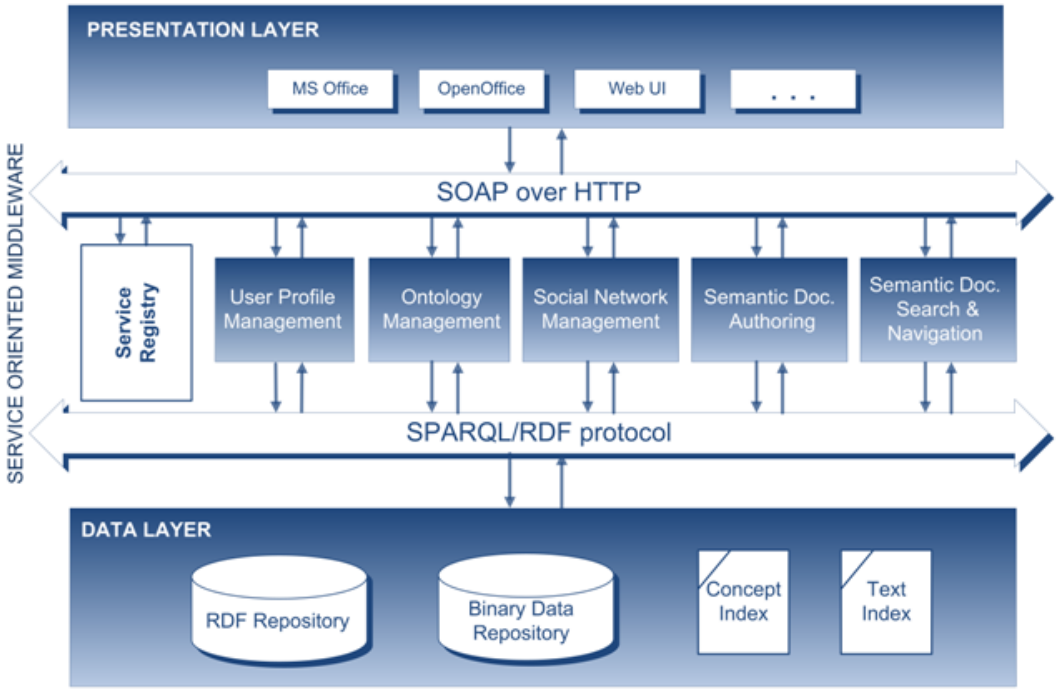


Figure 4.1. SDArch layered architecture

4.1.1 Data Layer

The data layer contains the semantic document repository that is composed of the RDF and binary data repository, and equipped with the concept and text indexes. The RDF repository stores RDF (MP) representations of semantic documents. A binary content of semantic document units is kept separately from RDF document representations in the binary data repository. Conceptually, RDF nodes representing semantic document units could hold the binary data (i.e., as a value of the `xsd:base64Binary` datatyped RDF property). However, if a large amount of binary data is managed as a part of the RDF representation, structured queries (e.g., SPARQL queries) executed against the RDF data would not be efficient. In order to have better performance of query execution and thus better search and retrieval, I decided to separate binary content of document units from their RDF representation and to store binary content in the binary data repository. As explained in Section 3.2.1, atomic document units are document units that hold some binary content. Accordingly, each atomic document unit has a property that holds the content stream to document units' binary data stored in the binary data repository. Textual contents are serialized and stored into plain text files, while graphical, audio and video contents are stored into appropriate media files. SDArch maintains: i) a single concept index that enables the semantic document search (Section 5.2.1) over RDF data; ii) a single text index that enables the full-text search over textual document

content. Both indexes are updated every time a new semantic document is added to or removed from the semantic document repository. In addition, the RDF repository exposes remotely accessible SPARQL endpoint over HTTP, so that SPARQL queries can be sent from remote machines.

4.1.2 Service-Oriented Middleware

The SDArch follows the Service Oriented Architecture (SOA) paradigm. The SDArch functionalities are implemented by the SDArch services, which are registered at the SDArch service registry. The SDArch service registry provides registration and look up functionality for the SDArch services. It further provides methods to enable and disable services. The service registry is also known as the SDArch middleware. As communication protocol between the services SDArch supports SOAP. The core SDArch functionalities are implemented through a number of functional modules, which are encapsulated into five services: 1) the Semantic Document Authoring, 2) the Semantic Document Search and Navigation, 3) the User Profile Management, 4) the Social Network Management, and 5) the Ontology Management service. All the services provide the Web interface containing methods through which the service functionalities can be accessed. Each service method can be invoked by applications from the presentation layer, or by the other SDArch services. The data layer and the five core SDArch services, together, form a software unit which can be delivered and installed, allowing processes running on different machines to interact across the network. In addition, besides the core services, SDArch can be extended by an arbitrary number of new services, which have to be developed in accordance with the SDArch design principles. In order to be used in SDArch, new services first need to be registered in the SDArch service registry. In the scope of this thesis, I only considered the design and implementation of the five core SDArch services.

4.1.3 Presentation Layer

The presentation layer is the top layer of SDArch, which provides the user interface for the SDArch services. According to the service-oriented nature of SDArch, the presentation layer is technology- and platform-independent. It can contain Web-based applications, desktop-based applications and mobile phone applications. Moreover, the presentation layer can contain completely new applications built from scratch or extended existing applications. According to my decision to use existing, well-established document formats for the human readable representation of semantic documents (Section 3.5), in the SDArch prototype (Chapter 6) developed in the work on this thesis, I have been mainly focused on extending the existing document authoring suites, instead of creating completely new applications. In that way, I wanted to let users take advantages of the SDArch functionalities, while still working within familiar environments. As a part of the usability evaluation that I conducted (Section 7.2), I also considered the

'ease-of-use' and 'ease-of-learning' of the added tools to the chosen document authoring suite (i.e., MS Office). The discussion of the results of these two usability criteria is given in Section 7.2.7.

4.2 The SDArch Services

As I mentioned in the introduction of this chapter, the SDArch functionalities can be grouped in two groups: i) the functionalities that are related exclusively to semantic document management and ii) the functionalities that SDArch should share with other services of SSD. The first group of functionalities are implemented by the two SDArch services: 1) the semantic document authoring and 2) the semantic document search and navigation services. The second group of SDArch functionalities contains: functionalities necessary for managing domain ontologies, which are used for the semantic document annotation, linking and indexing, then functionalities necessary for managing the SDArch user profile data, and functionalities necessary for managing the SDArch social network data. The second group of functionalities are implemented by the three SDArch services: 3) the User profile management, 4) the social network management, and 5) the ontology management services.

My main focus regarding the SDArch design, in the scope of this thesis, was on the overall architecture design and the detailed design of the two SDArch services related to semantic document management (i.e., services 1 and 2). For a detailed description of the functionalities provided by these two services and the corresponding processes realized by them, I dedicate the next chapter (Chapter 5). In this chapter I give just a brief overview of these two services. The design of the other three services (i.e., services 3, 4, and 5) was dedicated mainly to those functionalities that interfere the semantic document management processes.

4.2.1 Semantic Document Authoring Service

The semantic document authoring service provides the functionalities necessary for the realization of the semantic document authoring process. Figure 4.2 shows a functional model of the service including the service's functional modules, the module interdependencies and the service's interface.

The service's interface provides two methods:

`Transform()`	Takes an existing, conventional document (e.g., Word and PowerPoint) and transforms it to a semantic document.
`Conceptualize()`	Takes a document unit and retrieves a set of ontological concepts whose instances appear in the document unit's content.

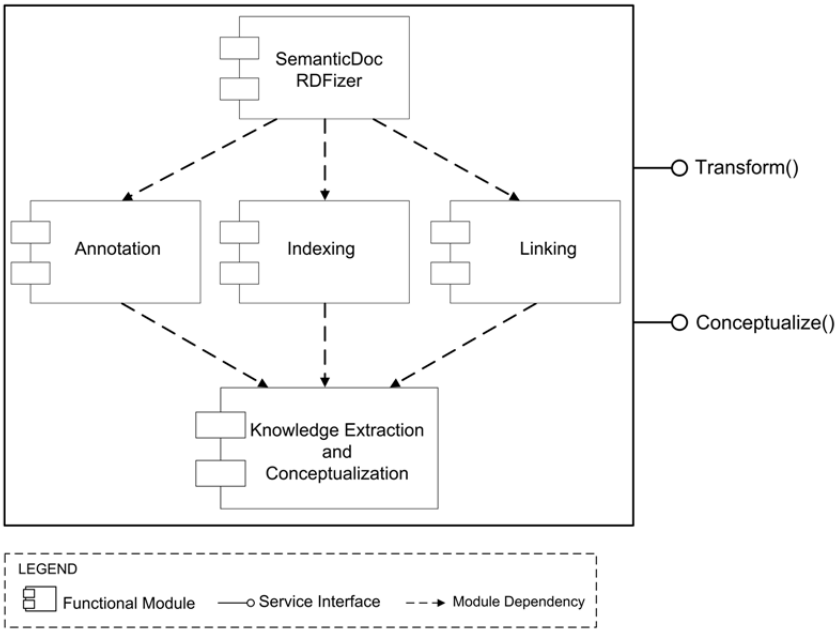


Figure 4.2. Functional model of the semantic document authoring service

While the `Transform()` method is invoked by applications from the presentation layer, the `Conceptualize()` method is invoked by the semantic document search and navigation service in the process of generating the semantic query (Section 5.2.1). The service's functionalities are implemented by five functional modules: SemanticDoc RDFizer, Annotation, Indexing, Linking, and Knowledge Extraction and Conceptualization modules. Each of the modules handles one or more internal processes that the semantic document authoring is composed of. The detailed description of the semantic document authoring process and the service's functional modules are given in Chapter 5.

4.2.2 Semantic Document Search and Navigation Service

The semantic document search and navigation service provides the functionalities necessary for the realization of the semantic document search, the full-text search and the semantic document navigation processes. Figure 4.3 illustrates the functional model of the service. The service's functionalities responsible for the semantic document search are implemented by two functional modules: the Search and the Search Personalization modules. The service's functionalities responsible for the semantic document navigation process are implemented by the Semantic Navigation module. Both of these two processes and their corresponding functional modules will be described in details in Chapter 5. In addition, the full-text search is provided as a complementary search to be used when the semantic document search does not retrieve any results. It is realized by the Search module.

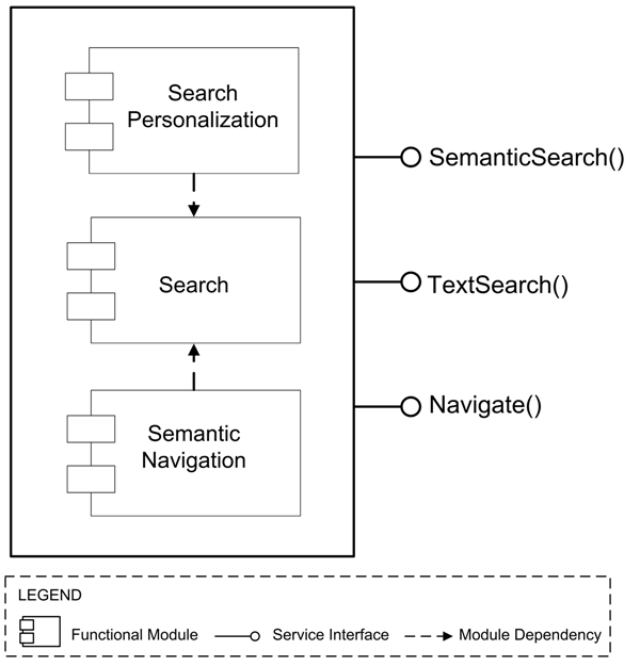


Figure 4.3. Functional model of the semantic document search and navigation service

The service's interface exposes two methods, which correspond to the two main processes that the service realizes:

`SemanticSearch()`	Takes the initial free-text user query, transforms it into a semantic query (Section 5.2.1) and executes the semantic query against the concept index of the SDArch semantic document repository.
`TextSearch()`	Executes the initial free-text user query against the text index of the SDArch semantic document repository.
`Navigate()`	Executes a navigational query (Section 5.2.3), which is generated by the user clicking on a semantic link of a retrieved document unit by the semantic search, against the SDArch RDF repository.

4.2.3 User Profile Management Service

The SDArch users are described by the SDArch user profiles and uniquely identified by an OpenID[1]. I chose OpenId for the SDArch users identification, since it is an open, decentralized standard for the authentication of online users. OpenID can be used for

[1] `http://openid.net/foundation/`

access control, allowing users to log on to different services with the same digital identity. The SDArch user profiles are specified by the SDArch user model. The same as the machine-processable representation of the semantic documents, the SDArch user profiles are represented by RDF data representation model and stored in the SDarch RDF repository. Being uniquely identified and described by machine-processable descriptions, the SDArch users are represented in accordance to the envisioned user representation of SSD and the Semantic Web.

The user profile management service provides functionalities necessary for managing information kept in the user profile. Moreover, it serves the user information to the other SDArch services which need that information (e.g., the semantic document search service in the search personalization process). Finally, the service provides access to some of the user information to the other users from the same SDArch social network. In what follows in this section, I first describe the SDArch user model, and then describe the functional model of the service including the service's interface and the service's functional modules.

The SDArch User Modeling

The SDArch user model is influenced by the notion of semantic documents as completely open resources, composed of easily accessible and reusable data/information units (i.e., document units). It intends to replace today's application-centered user model [102, 79] with a document-centered user model. In other words, the SDArch user should focus on authoring a document or performing an individual task, rather than using any particular application. Moreover, instead of authoring documents completely from scratch, SDArch encourages document authors to reuse existing, well-defined document data and potentially modify them to serve their purposes. Reuse of appropriate existing document data not only saves authors' time, but also has the potential to improve the quality of the authored documents. In particular, if the author does not possess an adequate knowledge about the topic of a document to be authored, the reuse of data from documents created by the experts on that topic will lead to a better quality document. Contrary to conventional documents, where the reuse of document data is accompanied by the loss of the proprietary information, the reuse of data from semantic documents (in the form of semantic document units), preserves the proprietary information of the reused data.

Semantic document authoring is one of the main processes that the SDArch users are involved in. Moreover, the new document architecture prioritizes the reuse of existing, well-defined document content over the creation of a new content while authoring new documents. Therefore, modeling user preferences regarding document contents to be reused was one of the main aspects that I was focused on while designing the SDArch user model. The user preferences are important for the personalization of the semantic document search. Based on the values of the user preferences, the document units retrieved by the semantic document search are re-ranked to better correspond to the user preferences. Similar to the semantic document model, the SDArch user model

is specified by an ontology, namely the SDArch user-model ontology. Figure 4.4 gives a simplified, graphical representation of the ontology. The full specification of all ontology's classes and properties is given in Appendix A.5.

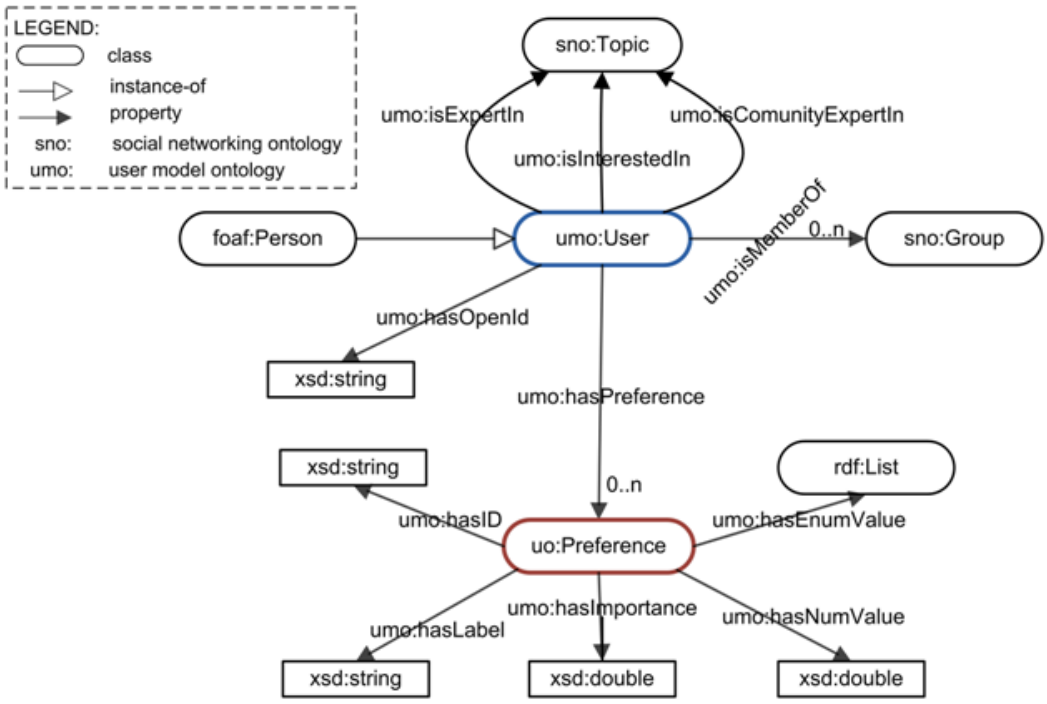


Figure 4.4. Illustration of the user model ontology

Two main classes in the ontology are `umo:User` and `umo:Preference`. The `umo:User` class is derived from the `foaf:Person` class of the friend-of-a-friend (FOAF) ontology[2]. The FOAF ontology contains classes and properties for describing people, links between them and things they create and do. As a subclass of the `foaf:Person` class, the `umo:User` class inherits properties that model a personal information of the SDArch user such as name, title, age and e-mail address. Moreover, the class inherits the `foaf:knows` property of the `foaf:Person` class, which is a property that was of particular interest for this thesis. This property enables the SDArch users to be connected to other persons that they know. In addition to the properties inherited from the `foaf:Person` class, the `umo:User` class provides a set of properties that are introduced in order to model some specific characteristics of the SDArch user. As mentioned above, each SDArch user is uniquely identified by an OpenID, which is kept as a value of the `umo:hasOpenID` property. The property `umo:isMemberOf` holds the information about a social network group that the user is member of. A detailed discussion on social network

[2] `http://xmlns.com/foaf/spec/`

modeling, I give in Section 4.2.4. Moreover, `umo:interestedIn`, `umo:isExpertIn` and `umo:isComunityExpertIn` are properties which hold the information about the topics that the user is interested in, the topics that the user self-asserts that he is an expert in, and the topics that are determined as a part of the user's expertise based on the amount of his document contents which has been reused by other members from the same SDArch social network. Finally, the `umo:User` class has the `umo:userPreference` property, which holds instances of the `umo:Preference` class that I describe next.

The `umo:Preference` class is a generic class introduced to specify the user's preferences regarding the choice of document units to be reused. The information modeled by this class plays one of the key roles in the personalization of the semantic document search discussed in Section 5.2.3. The class has the following properties: `umo:hasId`, `umo:hasLabel`, `umo:hasImportance`, `umo:hasNumValue` and `umo:hasEnumValue`. Instances of the class, that is, user preferences are uniquely identified by the preference ID and the preference label. The value of `umo:hasImportance` property determines the importance of the preference for the user. Different preferences have different importance for different users. The preference importance is specified by the user. Based on the nature of the preference, the preference value can be expressed by a numerical or enumerated (i.e., list of entities) value. The `umo:hasNumValue` and `umo:hasEnumValue` properties hold the preference's numerical and enumerated values respectively. In contrast to the preference importance, the preference value (it is either numerical or enumerated) is not specified by the user, but learned automatically over time by monitoring the user's activities.

Preferences that I introduced in the user model are correlated with the set of information that can be extracted from the social-context annotations (Section 3.2.2). I now briefly describe each of them.

1. Preferred Authors ($Pref_1$): this preference specifies an ordered list of SDArch users who share their semantic documents. The order is formed based on a number of DUs that the user has reused from each of the users. The first author in the list is the user from whom the user has reused the most DUs. Initially, the list is empty and gets populated over time. Every time the user reuses a DU, the list is updated and potentially reordered.

2. Preferred Software Applications ($Pref_2$): this preference specifies an ordered list of document authoring/editing applications. The order of an application in the list is determined by the number of reused DUs, which are authored using that application. Similarly to the list of preferred authors, the list of preferred applications is initially empty and gets populated over time. Every time the user reuse a DU, the list is updated and potentially reordered.

3. Preferred Number of Reuses ($Pref_3$): this preference specifies if the user rather reuses DUs that have been reused many times or DUs that have been rarely reused. The preference is characterized by a numerical value which falls in the range $[0, 1]$ of real numbers, 0 meaning that the user always reuses the DU that has been reused the least

number of times comparing with other DUs that are retrieved as a search result, and 1 meaning the opposite.

4. Preferred Number of Modifications ($Pref_4$): this preference specifies if the user rather reuses DUs that have been modified many times, that is, DUs with many versions or DUs that have been rarely modified. The preference takes a numerical value from the range $[0,1]$ of real numbers, 0 meaning that the user always reuses the DU that has been modified the least number of times comparing to other DUs retrieved as a search result, and 1 meaning the opposite.

5. Preferred Number of Browses ($Pref_5$): this preference specifies if the user rather reuses DUs that have been browsed many times or DUs that have been browsed rarely. The preference takes a numerical value from the range $[0,1]$ of real numbers, 0 meaning that the user always reuses the DU that has been browsed the least number of times comparing to other DUs retrieved as a search results and, 1 meaning the opposite.

6. Preferred Time of the Reuse ($Pref_6$): this preference specifies if the user rather reuses DUs that have been recently reused or DUs that have not been reused for a long time. The preference is characterized by a numerical value from the range $[0,1]$ of real numbers, 1 meaning that the user always reuses DUs that have been reused most recently of all DUs retrieved as a search result, and 0 meaning the opposite.

7. Preferred Time of the Modification ($Pref_7$): this preference specifies if the user rather reuses DUs that have been modified recently or DUs that have not been modified for a long time. The preference is characterized by a numerical value from the range $[0,1]$ of real numbers, 1 meaning that the user always reuses DUs that have been modified most recently of all DUs retrieved as a search result, and 0 meaning the opposite.

8. Preferred Time of the Browsing ($Pref_8$): this preference specifies if the user rather reuses DUs that have been browsed recently or DUs that have not been browsed for a long time. The preference is characterized by a numerical value from a range $[0,1]$ of real numbers, 1 meaning that the user always reuses DUs that have been browsed the most recently of all DUs retrieved as a search result, and 0 meaning the opposite.

The way in which the values of the enumerated preferences (i.e., values of $Pref_1$ and $Pref_2$) are learnt over time is as follows. Every time the user reuses a DU, the information about the author and the authoring application of the DU is extracted from the DU's annotations and then the system uses them to update the lists of the user's preferred authors and preferred applications. The way in which the values of the numerical preferences (i.e., values of $Pref_3$ - $Pref_8$) are learned over time is more complex. I will explain that on the example of $Pref_3$, that is, the 'Preferred numbers of reuses' preference. The other numerical preferences ($Pref_4$ - $Pref_8$) follow the same formulation and ranking approach. Let us mark $preferenceNumValue$ of $Pref_3$ as $p \in [0,1]$. The initial value of p is 0, but over time it changes based on DUs that the user has reused. With N, I denote the number of DUs that the user has reused. Suppose now that the user wants to create a new document. In order to do that he searches the existing semantic

documents for DUs to reuse. The search engine retrieves a set of DUs, $\mathbb{D} = \{d_1, ..., d_n\}$, each of them being reused a certain number of times $\mathbb{N} = \{n_1, ..., n_n\}$. After previewing the retrieved DUs, the user decides to reuse document unit d_i. The reuse of document unit d_i triggers the update of the preference value p and the system calculates a new value p' as follows:

$$p' = \frac{N * p + Pref_3(d_i)}{N + 1} \tag{4.1}$$

$Pref_3(d_i) \in [0, 1]$ is a weight of the document unit d_i for the preference $Pref_3$ in the scope of the set of retrieved document units $\mathbb{D}$. It is calculated as:

$$Pref_3(cu_i) = \frac{n_i - n_{min}}{n_{max} - n_{min}} \tag{4.2}$$

n_{min}, n_{max} represent minimum and maximum elements of the $\mathbb{N}$. In the case of $n_{min} = n_{max}$, meaning that the user does not actually have different choices regarding the given preference, the value p of the preference remains unchanged.

Functional Model of the User Profile Management Service

Figure 4.5 illustrates the functional model of the user profile management service. The service's functionalities are implemented by two functional modules. The first module, namely the Read and Write module, provides functions for reading and writing data to the RDF representation of the user profile. The second module, called the Update module, provides functions for updating the user profile data. The updating of the user profile data depends on the user's own activities but also on the activities of the other members of the SDArch social network. The service's interface exposes four methods:

`CreateProfile()`	Creates a default user profile (i.e., RDF instance of the user model ontology).
`SetProfileData()`	Sets the user's data into the profile.
`GetProfileData()`	Retrieves data from the user's profile.
`UpdateProfileData()`	Updates the user profile based on data created by the Update module.

The service realizes the following three processes: 1) manual editing of the user profile, 2) automatic update of the user profile, and 3) personalization of the semantic document search. I now briefly explain all of them.

Manual editing of the user profile: The SDArch users are characterized by the set of basic information that has to be specified by users themselves. Moreover, regarding the user preferences, users are responsible to specify the importance (e.g., value of the `umo:preferenceImportance` property) of each preference included in the profile. In the

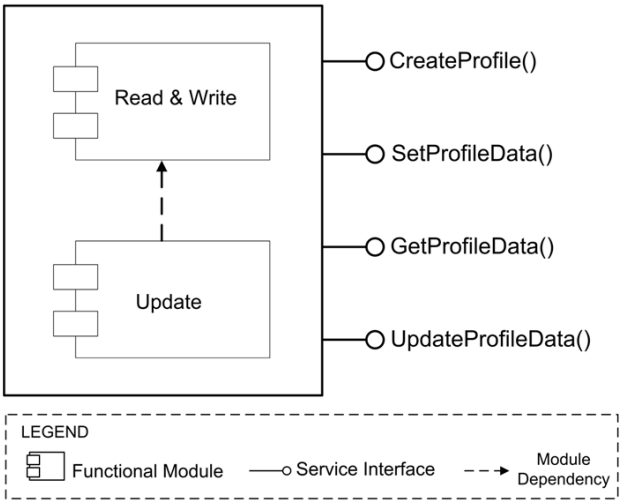


Figure 4.5. Functional model of the user profile management service

current design of the service, users can choose between low, medium and high as a preference importance value. The default value of the preference importance is medium for each preference. The process of the manual editing of the profile data requires an appropriate application at the SDArch presentation layer, which would provide the graphical user interface for browsing and editing the profile data. This application would actually invoke `SetProfileData()` and `SetProfileData()` methods of the service.

Automatic update of the user profile: Besides the user basic information, the SDArch user is characterized by the set of user preferences whose values evolves over time based on the user's own activities as well as the activities of the other users from the SDArch social network. For example, the automatic update of the profile will be initiated every time the user reuses a document unit. This user action is captured by an event-based, monitoring system that monitors the SDArch user behavior and which is a part of the SDArch presentation layer. When the user reuses a document unit (e.g., by simple clicking on an appropriate GUI element for adding document units from the semantic document search and navigation results), the `UpdateProfileData()` method is called and the service receives the URI of the reused document unit. After that, the update module acquires relevant information from the document unit's annotations (i.e., the standardized metadata annotations and the social-context annotations). By utilizing this acquired information and the user profile data, the update module calculates the new values of the user preferences. The way the new preference values are calculated, I already explained in the previous section. The automatic update is finished when the new preference values are written back into the user profile. For reading and writing data to the user profile, the service utilizes the read and write service module.

Personalization of the semantic document search: The main purpose of the user

preferences, specified in the user profile, is to enable re-ranking of document units that are retrieved by the semantic document search, so that the new order better corresponds with the user's preferences. The personalization of the semantic document search is considered as a subprocess of the semantic document search, which I discuss in detail in Section 5.2.1. Here, I just want to indicate the role of the user profile service within the semantic document search process. In one sentence, the role is to provide the semantic document search and navigation service, with data from the user profile that are used in the search personalization process.

4.2.4 Social Network Management Service

One of the design principles of semantic documents is to enable interlinking of document data (i.e., document units) with other data on the Semantic Web. Semantic documents are uniquely identified entities, which can be interlinked by structural or semantic links (Section 3.2) with other uniquely identified resource on the Semantic Web. This feature of semantic documents, implies the main difference in a way we reuse content from semantic documents and conventional documents. While reusing data from conventional documents assumes copying and pasting existing document data into a new document, reusing data from semantic documents is realized simply by linking desired document units (i.e., their RDF nodes) to the specified locations in the RDF representation of a new semantic document. On an individual desktop, each semantic document is represented by only one persistent, machine-processable RDF representation. This representation is integrated into a unified desktop information space (i.e., SDArch RDF repository). However, more copies of the RDF document representation are possible in the shared RDF space (repository). More copies of the RDF document representation does not mean that there are more copies of the actual semantic document. They all describe the same document. Accordingly, sharing semantic documents is equal to publishing RDF document representations to the shared RDF space. In addition, regardless of the number of copies of the RDF document representation, the binary content of each semantic document unit has only one physical copy, which resides on the location whose address is encoded into the document unit URI.

The social network management service provides functionalities that enable the SDArch users to share their semantic documents by publishing the RDF representations of the documents into shared RDF repositories. Moreover, the service provides functionalities for organizing SDArch users into a SDArch social network formed around shared semantic documents. Finally, the service provides functionalities for the creation of the social-context annotation of the shared semantic documents. Figure 4.6 illustrates a network of the SDArch enabled users. In the rest of the section, I first discuss how I modeled the SDArch social network. After that, I describe the service's functional model.

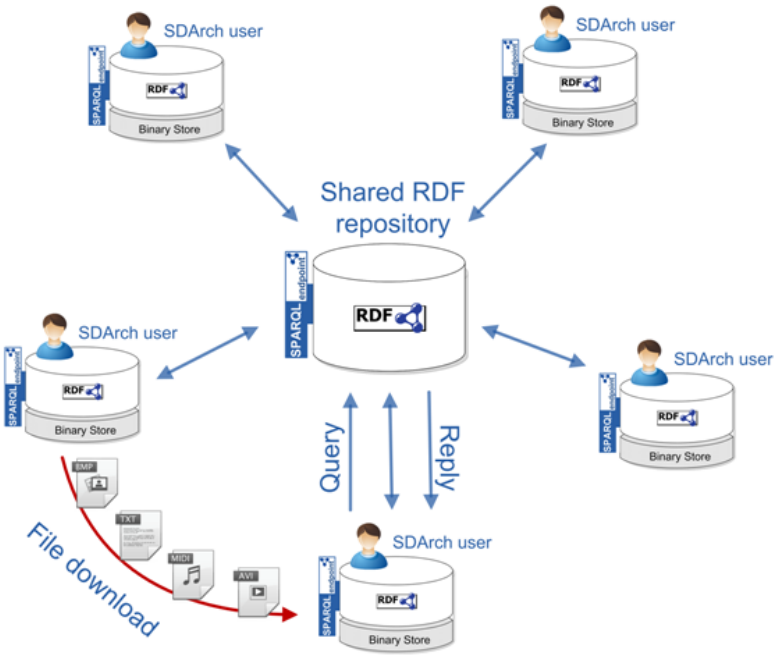


Figure 4.6. Networked SDArch users - illustration

Modeling the SDArch Social Network

Similar to the semantic document model and the SDArch user model, I chose ontologies to formally specify the model of the SDArch social network. The model is specified by the social network ontology. Figure 4.7 gives a simplified, graphical representation of the ontology. The full specification (all classes and properties) of the ontology is given in Appendix A.6. The main class in the ontology is the `sno:Group` class. This class models a group of SDArch users who are organized around a given topic of interest and share semantic documents related to that topic. Each SDArch user group is determined by the group's identifier and topic of interest. The group's topic of interest is specified by an instance of the `sno:Topic` class. Moreover, each group has members (i.e., SDArch users) who are further represented as instances of the `umo:User` class. A group member who has initiated a group is also identified as the group's creator. Each group maintains a number of semantic documents published by the group's members. Finally, each group is equipped with an online forum which is specified by the `sno:Forum` class and its properties. A forum contains posts (instances of the `sno:Post` class) that are created and published by group members.

One SDArch user can be a member of many groups. Also, the same semantic document can be published in more than one group. However, publishing the same document in more than one groups, does not mean many copies of the document's RDF representation in the shared RDF space. The RDF representations of the documents

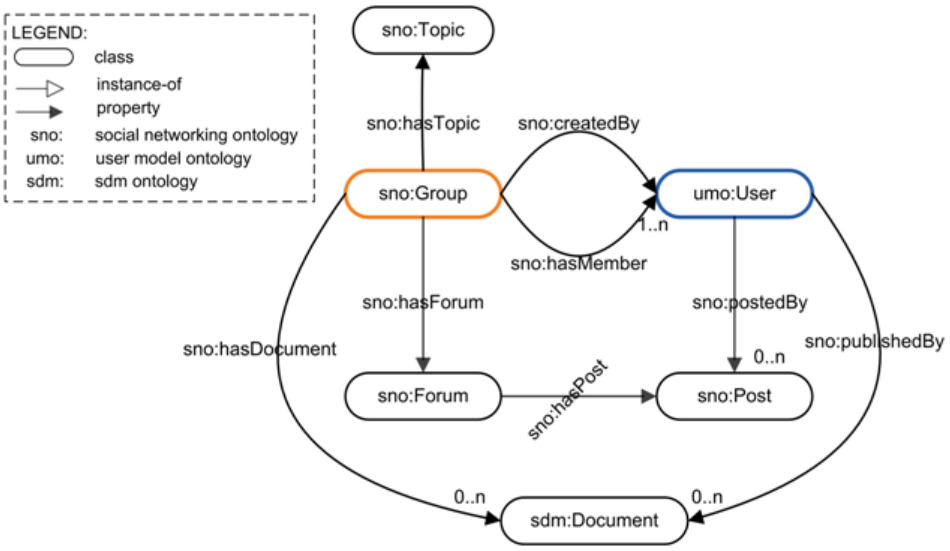


Figure 4.7. Illustration of the social network ontology

from all existing groups are stored into the same shared RDF space. Moreover, data about the social network, that is, the RDF instance of the SDArch social network ontology, is stored into the shared RDF space, so that all members of the network have access to the network's data.

Interoperability with other Social Networking Services

Social Network Services (SNSs) provide a multitude of ways for users to interact and have been promoted as central to the second generation of the Web (i.e., Web 2.0), which is characterized by mass participation and collaboratively generated Web content. The SNSs have brought a new way of interconnecting online content and networked people for various social and professional purposes. People are now better connected and can easily access, reuse or comment on content that is authored by other people from the network. However, in spite of being well connected within particular social networks, the diversity of existing SNSs hampers the interoperability between members of social networks that are managed by different SNSs.

The existing SNSs form bounded communities (i.e., social networks) of users and their shared data, usually referred to as shared objects of interest. Although most of the SNSs provide a common set of functionalities such as personal profile management, social network management, private and public messaging, social tagging of shared objects, discussion forums, and events management, each SNS represents the users, shared objects and relationships among them in an application specific way. This hampers the interoperability among the social networks and, ultimately, data creation and sharing

on the Web. Users of different social networks cannot interact between each other and can not access, comment, or reuse shared objects that belong to other social networks. In other words, today's SNSs do not have agreed-upon semantics to describe shared objects, that is, there is no shared understanding of the objects' meaning in different SNSs.

One possibility to overcome this lack of network interoperability is to leverage semantics into social networks - interconnecting both content and people in a meaningful way [16]. By providing the SDM ontology we attempted to address this problem in case of document units as shared objects of interest. Moreover, the SDArch complies with the Semantic Web principles for integration of data originating from diverse sources. Using globally unique URIs (i.e., openIDs) for the identification of both SDArch users and document units, enables them to be identified independently of particular SNSs. Once an URI is assigned to a document unit, the document unit can be linked to other uniquely identified document units or resources, regardless the social networking services that they are managed by. The RDF is a standard model for data interchange on the Semantic Web. It allows structured and semi-structured data to be mixed, exposed and shared across different applications. By publishing shared document units' descriptions as RDF instances, the SDArch enables them to be easily interchanged with other existing SNSs. However, in order to understand the meaning of the shared document units, other SNSs should use the agreed upon semantics that lie behind the document units descriptions, that is, shared domain ontologies. The interoperability between the existing SNSs that use domain ontologies different than those used by the SDArch could potentially be solved by the ontology mapping [41].

Functional Model of the Social Network Management Service

The functionalities of the social network management service are grouped into three functional modules namely the Group Management, the Document Publishing, and the Social-Context Annotation (SCA) modules. Figure 4.8 gives a high-level graphical illustration of the service's functional model.

The group management module provides functionalities that enable SDArch users to create or delete a group, join and leave a group, list all groups in the network, and list groups' details (e.g., group members and documents). These functionalities are accessible through the following service's methods:

`CreateGroup()`	Creates a new user group within the SDArch social network dedicated to a given topic.
`DeleteGroup()`	Deletes an existing user group from the SDArch social network.
`JoinGroup()`	Enables the user to join a given social network group.
`LeaveGroup()`	Enables the user to live a given social network group.
`GetAllGroups()`	Retrieves all the groups from the SDArch social networks.
`GetGroupData()`	Retrieves details of a given group.

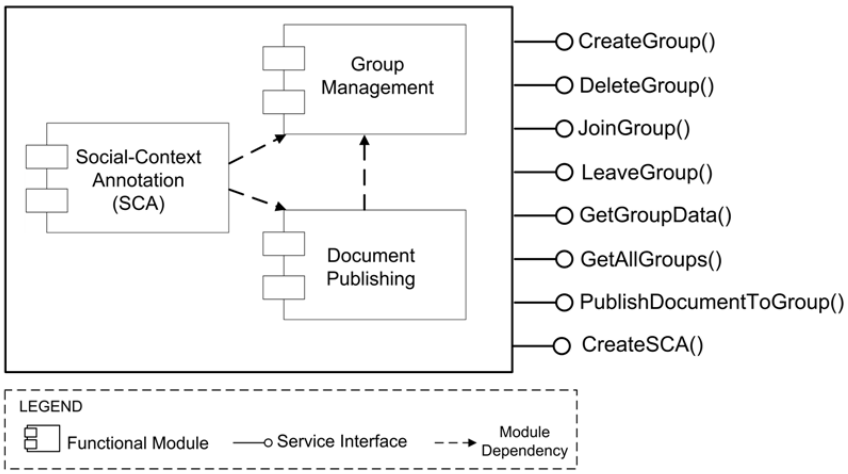


Figure 4.8. Functional model of the social network management service

The SDArch social network's data are stored in the shared RDF repository, which is equipped with a publicly available SPARQL endpoint.

The document publishing module provides functionalities that enable SDArch users to publish their semantic documents to the shared RDF repository. Before publishing a semantic document, the users have to choose a group to which they want to publish the document. To obtain information about available groups as well as to update the group's data (e.g., a list of the group's documents) after the successful document publishing, the document publishing module utilizes the functions of the group management module. The functionalities of the publishing module are accessible through the `PublishDocumentToGroup()` method of the service's interface.

`PublishDocumentToGroup()`	Publishes a desktop semantic document into a shared semantic document repository of the SDArch social network.

The social-context annotation module is responsible for the creation of the social-context annotations for the shared semantic document units. The functionalities of this module are accessible through the `CreateSCA()` service method.

`CreateSCA()`	Creates the social-context annotation for a semantic document unit that the user performs an action to.

This method is invoked by appropriate applications of the SDArch presentation layer, which are equipped with an event-based monitoring system that monitors the SDArch user behavior. Whenever the user performs an action such as browsing, reusing and modifying a document unit, the monitoring system invokes the method, which then creates a SCA for the document unit that the user performed the action to.

4.2.5 Ontology Management Service

Domain ontologies play a very important role in the authoring of semantic documents as well as in the semantic document search. In order to semantically annotate document units, and then to identify and link semantically related document units, we first conceptualize document units' semantics and represent them by concepts for domain ontologies. Moreover, in the semantic search, which is described in Section 5.2.1, I use domain ontologies to transform the initial, keyword-based user query to the semantic query that is represented by a set of ontological concepts and their corresponding relevance weights. The availability and the quality of domain ontologies, used by the SDArch services determine the quality of the semantic annotation, linking and semantic document search and navigation. The ontology management service is responsible for managing domain ontologies that are used by in SDArch. SDArch places ontologies, that is, their OWL files, into the SDArch ontology repository, which is a part of the desktop SDArch RDF repository.

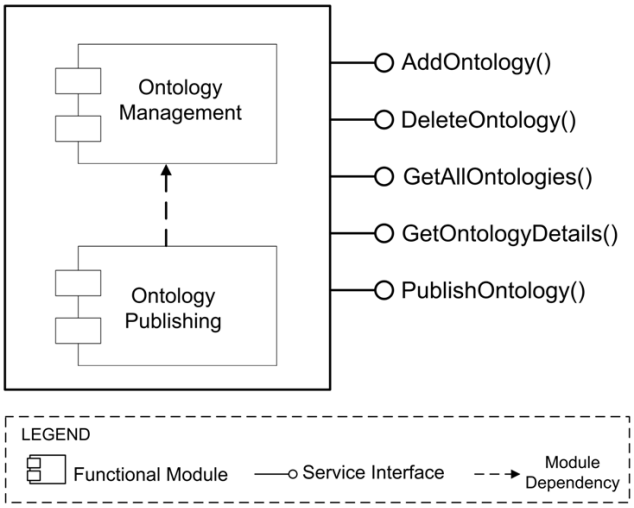


Figure 4.9. Functional model of the ontology management service

Figure 4.9 gives a high-level, graphical illustration of the service's functional model. The service's functionalities can be achieved through the service's interface consisting of the following methods:

`AddOntology()`	Adds a new ontology to the ontology repository.
`DeleteOntology()`	Deletes an exiting ontology from the ontology repository.
`GetAllOntologies()`	Retrieves all the ontologies from the ontology repository.
`GetOntologyDetails()`	Retrieves details of a given ontology.
`PublishOntology()`	Publishes an ontology to the ontology repository of the shared semantic document repository.

The service's methods are also invoked by other SDArch services in the scope of the execution of several SDArch processes. Within the semantic annotation, linking and indexing processes, as well as the semantic document search process, the service is called by the semantic document authoring service (Section 4.2.1). The role of the service in these processes is to extract concepts (concepts' URIs) and concepts' descriptions from a given domain ontology, and to serve them to the knowledge extraction and conceptualization module (Section 5.1.1) of the semantic document authoring service. Within the semantic document publishing process, the service is called by the social networking management service (Section 4.2.4). The role of the service within this process is to check if the ontologies, which have been used for the semantic annotation of the document to be published, have already been published to the shared RDF space and if not to publish them. Shared semantic documents need their underlying domain ontologies to be shared as well.

4.3 Summary

In this chapter, I presented a software architecture, called SDArch, that I developed aiming to support the management of semantic documents and provide users with the new services enabled by the new form of documents. I designed SDArch as a three-tier, service oriented architecture, composed of the data layer (Section 4.1.1), service-oriented middleware (Section 4.1.2) and presentation layer (Section 4.1.3).

In the actual design, the SDArch functionalities are provided by the set of five services, two of which are exclusively related to the semantic document management (i.e., the semantic document authoring and the semantic document search and navigation services), and the other three are responsible for the management of the SDArch user profile, the SDArch social network and domain ontologies that SDArch employs for semantic annotation, indexing and linking of semantic document units. The SDArch middleware provides the service registry for registering or unregistering services from the architecture, and defines the communication protocol at the service layer and between the service and presentation layers.

My focus in this chapter was on the overall design of SDArch and the detailed description of the user profile management service, the social network management service, and the ontology management service. For the detailed description of the semantic document management processes and the two SDArch services that realizes them, I dedicate the next chapter.

Chapter 5

Semantic Document Management

The SDArch enabled processes can be classified in two groups. The first group contains processes that are directly related to the semantic document management. The second group contains processes that are related to the management of the other entities (i.e., SDArch users, the SDArch social network, and domain ontologies) of the proposed new document architecture. In the previous chapter, I described the overall design of SDArch and discussed the SDArch services and processes related to the SDArch user profile management, the domain ontologies management, and the SDArch social network management. I dedicate this whole chapter to the semantic document management processes and the SDArch services that realize them, as these are the core of SDArch.

Semantic document management consists of the processes responsible for the management of semantic documents throughout their life-cycle. The top-lavel semantic document management processes, which I have identified as particularly important and which I have designed and implemented in my thesis are: semantic document authoring, semantic document search and semantic document navigation. These top-level processes are composed of a number of low-level, sub-processes, some of which are part of more than one top-level processes. For example, a knowledge extraction and conceptualization process, which I explain later, is shared by the semantic document authoring and semantic document search processes. Low-level semantic document management processes are implemented by functional components which I refer to as functional modules. In the SDArch middleware, the functional modules responsible for the semantic document management processes are encapsulated into two services: the semantic document authoring service and the semantic document search and navigation service. Figure 5.1 illustrates these two services and the three top-level semantic document management processes that they realize.

This chapter consists of two sections: 1) semantic document authoring and 2) search and navigation in semantic documents. In the first section, I describe the semantic document authoring process along with its constituent sub-processes: the knowledge extraction and conceptualization and the semantic annotation, indexing and linking. In

the second section, I describe the semantic document search, the search personalization, and the semantic document navigation processes.

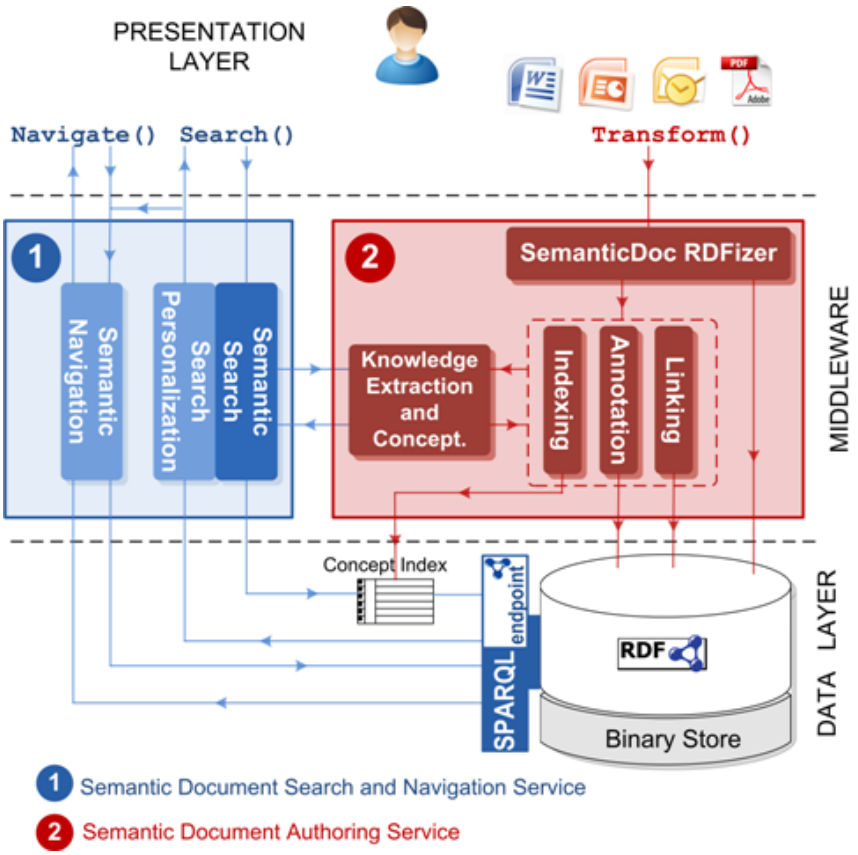


Figure 5.1. Semantic document management - services and related processes

5.1 Authoring of Semantic Documents

The life-cycle of a new product, whether a software or general-market product, starts with the creation process [53]. In case of semantic documents, it is the process of the semantic document authoring. Moreover, an important thing to consider when introducing a new product is whether such kind of products already exist and whether the new product intends to replace them or coexists with them. My intention with semantic documents is to gradually replace conventional documents. In the initial phase, conventional document formats should stay in use and play a role of the human-readable representation of semantic documents. There should be only one, persistent representation of semantic documents in a form of the machine-processable, RDF representation, while for human browsing and editing of semantic documents there should be rendered tem-

poral document representations in the form of some of existing conventional document formats. Sometime in the future there might be designed a completely new software, especially suited for human browsing and editing semantic documents without using the existing document formats.

I have distinguished between two possible scenarios of semantic documents authoring. The first one is about the transformation of documents implemented in conventional document formats (e.g., MS Word, MS PowerPoint or PDF documents) into semantic documents. The second one is about the use of an application software that is developed specifically for authoring semantic documents completely from scratch. In order to let users continue to work in familiar environments, such as existing document authoring suites, while taking advantage of the new document model as well as to enable conversion of existing collections of documents into semantic documents, in the current SDArch design I considered the first authoring scenario. Such authoring scenario is also inline with my hypothesis that semantic documents can be browsed and edited in a number of existing document formats, that is, semantic documents can have a number of human readable document views. Moreover, what I also wanted to achieve with the proposed authoring scenario is to minimize user efforts with as much automation as possible. Accordingly, the generation of RDF document representation and the semantic annotation, indexing and linking of the document units, which are actually the processes that semantic document authoring is composed of, are completely automated. Whenever SDArch users want to transform a conventional document to a semantic document, all they need to do is to select an appropriate domain ontology and to initiate the transformation. The rest of the transformation is completely automated. The users should select those domain ontologies that correlates well with the topic of the document. The ontology selection step could be automated in case of the transformation of a large number of documents of the same topic, when the system is supposed to use the same, predefined domain ontology for all documents. I suggest the use of existing domain ontologies published on the Semantic Web. There already exists a considerable number of widely used domain ontologies, most of which could also be applicable for semantic annotation, indexing and linking of desktop documents. By using existing, shared domain ontologies, we will enable different applications, on either the same or networked desktops, to interpret data of different semantic documents in a semantically coherent way.

Semantic document authoring is realized by the semantic document authoring service. The authoring process starts by the *SemanticDoc RDFizer* module (Figure 5.1), which scans a document to be transformed, recognizes the document's units (e.g., sections, paragraphs, tables, illustrations and slides) and generates their RDF descriptions, that is, RDF nodes that represent the document units, URIs that identify document units, and RDF links that represent structural relationships between the document units. The *SemanticDoc RDFizer* is also responsible to generate the standardized metadata describing document units. I have chosen a subset of internationally standardized metadata ele-

ments, which are appropriate for the annotation of document units, such as `dc:creator`, `dcterms:created`, `dc:format`, `dc:language`, `dc:title` and `dc:description` referring to the author(s), creation date, media type, language(s), title and short description of a document unit respectively. The SemanticDoc RDFizer derives most of the document units' metadata from the metadata of the document that is being transformed. Some of the metadata is also generated based on the available formatting information. For example, values of `dc:creator`, `dcterms:created`, `dc:format` and `dc:language` metadata elements are derived from the document's metadata, while a value of a `dc:title` element is generated based on the formatting information. On the other hand, a text fragment whose font style is 'title' or 'heading' is taken to be a value of the `dc:title` element of all the consecutive document units up to the next such formated text fragment. Moreover, a value of the `dc:description` element is generated out of the values of the previously explained elements, by applying the following pattern: 'A document unit of {dc:format} media type with a title {dc:title} authored by {dc:creator} on {dcterms:created}'. After having generated the document units' metadata, the SemanticDoc RDFizer links the metadata to the document units' RDF nodes.

Finally, SemanticDoc RDFizer stores the generated RDF document representation into the RDF repository. In addition, for each identified document unit SemanticDoc RDFizer calls the semantic *annotation*, *indexing* and *linking* functional modules. The processes realized by these three modules all rely on the process of knowledge extraction and conceptualization from document units, which is realized by the separate functional module of the semantic document authoring service. Being fundamental for the semantic annotation, indexing and linking, in the rest of this section I first explain the knowledge extraction and conceptualization process.

5.1.1 Knowledge Extraction and Conceptualization

The process of the document unit knowledge extraction and conceptualization is about the discovery of ontological concepts that conceptualize information/knowledge stored in the document unit, and the calculation of relevance weights of the discovered concepts for the document unit. As a result of this process, human-readable information stored in the document unit is represented by a set of ontological concepts and a corresponding concept weight vector. The concept weight vector is composed of the calculated relevance weights of the discovered concepts.

The knowledge extraction and conceptualization, that I propose, combines syntactic matching of the lexically expanded set of concept labels against the document unit's content, and semantic matching based on the concept exploration algorithm that I developed. In the rest of the section, I first describe syntactic matching along with the lexical expansion of the concept descriptions. Then, I describe semantic matching and give a detailed explanation of the concept exploration algorithm. The algorithm plays the core role in semantic matching and thus it is very important for the whole knowledge extraction and conceptualization process.

Syntactic Matching

Both syntactic and semantic matching require the use of an appropriate domain ontology that conceptualizes the domain(s) of information stored in considered document units. The objective of syntactic matching is to analyze the content of the document unit and to try to identify the occurrence of concept labels from the used domain ontology. The concepts whose labels occur in the document unit, are considered as the document unit's syntactic matches. In order to achieve better readability, I replace the term 'document unit' with the acronym 'DU' in the rest of the section.

Any domain ontology can be represented as a graph $O := (\mathbb{C}, \mathbb{R}, H^C, H^R)$ where $\mathbb{C} = \{c_1, c_2, c_3, ..., c_n\}$ is a set of concepts, $\mathbb{R} = \{r_1, r_2, ..., r_m\}$ is a set of relations and H^C, H^R are hierarchies defining a partial order over concepts and relations respectively. Moreover, each concept is described with a set of labels. For example, the set of labels of the concept c_i is $\mathbb{L}_i = \{l_{i1}, l_{i2}, ..., l_{im}\}$. In practice, however, ontology engineers provide only one label per ontology concept or even neglect to label concepts, considering human-readable parts of concept URIs as concept labels [121]. To solve the problem of insufficient concept labels, and to achieve more efficient syntactic matching, we start syntactic matching by performing a lexical expansion of concept labels of concepts from the used domain ontology.

The objective of the lexical expansion is to expand a set of concept labels of each concept from the ontology with related terms from lexical dictionaries such as WordNet[1], in order to enhance syntactic matching afterwards. In our approach we consider three dimensions of the lexical expansion: *synonym*, *hyponym*, and *hypernym*. The result of the lexical expansion is the expanded sets of concept labels. For example, for concept $c_i \in \mathbb{C}$ it is:

$$\mathbb{L}_i^e = \{l_{i1}, l_{i2}, ..., l_{im}, l_{im+1}, ..., l_{ik}\} \tag{5.1}$$

where the first m labels are original labels from the ontology, and the following k are terms which came from the lexical expansion and which were discovered by following the *synonym*, *hypernym* and *hyponym* relations, respectively. In order to make a distinction between the original labels and those coming from the lexical expansion, I introduced a label relevance factor $rFactor(l)$ and form a concept label relevance vector. The concept label relevance vector of the concept c_i is:

$$\overrightarrow{R_L}(c_i) = [rFactor(l_{i1}), ..., rFactor(l_{ik})] \tag{5.2}$$

where $rFactor(l_{ij}) \in \mathbb{R}$ has value 1 if l_{ij} is the original label, has value δ_{syn} if l_{ij} is the synonym, has value δ_{hyper} if l_{ij} is the hypernym and has value δ_{hypo} if l_{ij} is the hyponym of the original label. In the evaluation of the semantic document search and retrieval that I have conducted (Section 7.1), I used the values $\delta_{syn} = 0.7$, $\delta_{hyper} = 0.47$, $\delta_{hypo} = 0.84$, which had been determined in the experimental studies reported in [50].

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

Having the lexical expansion done, the next step is to analyze the content of a *DU* and check if some of the concept labels (including those from the lexical expansion) occur in the *DU*. The concepts whose labels occur in the *DU*, I refer to as the DU's syntactic matches. Moreover, for those concepts we calculate their relevance weight for the *DU* by taking into account the following: 1) the concept labels' relevance factor (determined in the lexical expansion), 2) the labels' frequency in the *DU* and 3) the inverse document unit frequency of the concept labels in a collection of all *DUs* conceptualized by the same domain ontology. Since *DUs* are defined as small pieces of a document content, I do not use the length normalization factor [114], which is used in case of the text categorization and indexing of large text documents, while determining the concept relevance weight.

Let us consider again the example concept $c_i \in \mathbb{C}$ and an example document unit d. Firstly, for each label l_{ij} from the expanded set of the concept's labels (5.1) we count the label frequency $LF(l_{ij})$ in the document unit d. Secondly, we calculate the inverse document frequency $IDF(l_{ij})$ [112] as $log\frac{N}{1+n}$, where n is a number of *DUs* to which the label l_{ij} is assigned and N is a total number of *DUs* in the collection. Finally, when we have $LF(l_{ij})$ and $IDF(l_{ij})$ calculated, we calculate the weight of the concept label l_{ij} for document unit d as follows:

$$w_{l_{ij}} = LF(l_{ij}) * IDF(l_{ij}) \tag{5.3}$$

The weights of all the concept labels (5.1) of the concept c_i regarding the document unit d form a concept labels weight vector:

$$\overrightarrow{W_L}(c_i|d) = [w_{l_{i1}}, w_{l_{i2}}, ..., w_{l_{ik}}] \tag{5.4}$$

Based on the concept labels weight vector (5.4) and the concept labels relevance vector (5.2), we calculate the weight of the concept c_i regarding the document unit d as a scalar product of these two vectors:

$$w_{c_i} = \overrightarrow{W_L}(c_i|d) * \overrightarrow{R_L}(c_i) \tag{5.5}$$

If $w_{c_i} > 0$, then the concept c_i annotates the document unit d and w_{c_i} determines the relevance of this annotation.

In the same way as for the concept c_i, we calculate the weights of all other concepts from the ontology regrding the document unit d. Concepts whose weight is greater then zero form the concept vector of the document unit d:

$$\overrightarrow{d} = [c_1, c_2, ..., c_r]; \quad c_i \in \mathbb{C} \quad \wedge \quad w_{c_i} \geq 0 \tag{5.6}$$

The weights of the concepts from the concept vector $\overrightarrow{d}$ form a concept weight vector of the document unit d:

$$\overrightarrow{W_C}(d) = [w_{c_1}, w_{c_2}, ..., w_{c_r}] \tag{5.7}$$

As a result of syntactic matching we got the initial set of concepts, that is, syntactic matches of the considered document unit d.

Semantic Matching

The objective of semantic matching is to extend the set of syntactic matches with semantically related concepts from the considered domain ontology. For this purpose I introduce the Concept Exploration Algorithm (CEA) that is explained in detail in the next section. In short, the algorithm takes an input concept and traverses the ontology to discover concepts which are semantically related to it. For the discovered concepts the algorithm also calculates the semantic distance between them and the input concept.

By applying the algorithm to the syntactic matches (5.6) of our example document unit d, we discover a set of the document unit's semantic matches and form the expanded concept vector $\overrightarrow{d}^{\,e} = [c_1, c_2, ..., c_r, c_{e1}, ..., c_{em}]$ of the document unit. For each of the semantic matches c_{ej}, the algorithm calculates the semantic distance $SDist^c(c_{ej}, c_i)$ from the initial syntactic match $c_i \in \overrightarrow{d}$. The weight $w_{c_{ej}}$ of the semantic match c_{ej} for the document unit d is then calculated by the following function:

$$w_{c_{ej}} = w_{c_i} * \beta^{-SDist^c(c_{ej}, c_i)}; \quad \beta > 1 \tag{5.8}$$

where w_{c_i} is the weight of the syntactic match c_i and β is a generic coefficient. I devised the function (5.8) so that it satisfies boundary conditions regardless of the value of coefficient β. For the first boundary condition $SDist^c(c_{ej}, c_i) = 0$, meaning that the concepts c_{ej} and c_i are semantically identical, $w_{c_{ej}} = w_{c_i}$, that is, the weight of the semantic match is the same as the weight of the initial syntactic match. For the second boundary condition $SDist^c(c_{ej}, c_i) \to \infty$, meaning that the concepts c_{ej} and c_i are semantically unrelated, $w_{c_{ei}} \to 0$, that is, the weight of the semantic match tends towards zero. For $SDist^c(c_{ej}, c_i) \in (0, \infty)$, the optimal value of coefficient β has to be experimentally determined. My hypothesis is that coefficient β strongly depends on the chosen domain ontology. In the evaluation which results I report in Section 7.1, I demonstrate how this parameter was experimentally determined for the domain ontologies used in the evaluation.

Concept Exploration Algorithm - CEA

The main assumption on which the algorithm is based is the possibility to associate numerical values to ontological relations, which we refer to as *the relation semantic distances ($SDist^r$)*, thus forming the weighted ontology graph. I distinguish between two types of the relation semantic distances: $SDist^r_{\mathscr{D}\to\mathscr{R}}(r)$ determining semantic distance of the concepts belonging to the domain $\mathscr{D}(r)$ of relation r from the concepts belonging to the range $\mathscr{R}(r)$ of r, and $SDist^r_{\mathscr{R}\to\mathscr{D}}(r)$ determining the semantic distance of the concepts belonging to the range $\mathscr{R}(r)$ of r from the concepts belonging to the domain $\mathscr{D}(r)$ of r.

Measuring the semantic distance/relatedness has received a great deal of attention in the field of lexical semantics [109]. In the field of ontology engineering, however, the

focus has been on the formal representation of relations between concepts rather than the measurement and quantification of the relational semantic distances. To the best of my knowledge none of the existing ontology representational languages has built-in constructs/attributes that could be used to express the value of the semantic distances between ontological concepts linked by a given relation. In general, the values of the relational semantic distances can be: 1) specified at design time of the ontology by the domain experts, 2) experimentally devised by using a controlled knowledge/data or 3) learned over time by exploiting the ontology in real world applications within the ontology domain. Based on my experience, the choice between these three strategies is strongly domain-dependent. A combination of the strategies is also possible. In the evaluation experiments that I conducted (Section 7.1), I used the values of the relational semantic distances which were experimentally devised.

The general idea of the algorithm (see Algorithm 1) is to explore the ontology graph starting from the input concept to find all concepts which satisfy the given semantic distance constraint (SD_c) and the given path length constraint (PL_c). SD_c is the maximum allowed semantic distance between the input and target concepts. PL_c is the maximum number of hops (i.e., ontology relations) allowed to belong to a path between the input and target concepts.

Algorithm 1 Concept Exploration Algorithm

1: INPUT O_w, c, SD_c, PL_c
2: OUTPUT $\overrightarrow{C'_e}$, $\overrightarrow{SD_e}$
3: $\mathbb{P} = Paths1(O_w, c, PL_c) = \{p_1, ..., p_m\}$ {finds all paths from c with a length $\leq PL_c$}
4: $\mathbb{C} = Concepts(\mathbb{P}) = \{c_1, ..., c_n\}$ {extracts all concepts from the set of paths $\mathbb{P}$}
5: **for all** c_i such that $c_i \in \mathbb{C}$ **do**
6: $\quad \mathbb{P}_i = Paths2(c, c_i, \mathbb{P}) = \{p_{i1}, ..., p_{ik}\}$ {finds a set of acyclic paths $\mathbb{P}_i \subset \mathbb{P}$ between c to c_i}
7: $\quad$ **for all** p_{ij} such that $p_{ij} \in \mathbb{P}_i$ **do**
8: $\quad\quad SDist^p(p_{ij})$ {calculates the semantic distance of path p_{ij}}
9: $\quad$ **end for**
10: $\quad SDist^c(c_i, c)$ {calculates the semantic distance of the concept c_i from c}
11: **end for**
12: $\overrightarrow{C'} = [c'_1, ..., c'_p]$, $c'_i \in \mathbb{C}$ and $SDist^c(c'_i, c) \leq SD_c$
13: $\overrightarrow{SD} = [SDist^c(c'_1, c), .., SDist^c(c'_p, c)]$

The algorithm takes the following inputs: a weighted ontology graph O_w formed by associating values of the relation semantic distances to the ontology relations; an input concept c; a semantic distance constraint SD_c and a path length constraint PL_c. The output consists of a vector of discovered concepts $\overrightarrow{C'}$ and a vector of the semantic distances $\overrightarrow{SD}$ between the discovered concepts and the input concept. The algorithm

starts by the $Paths1(O_w, c, PL_c)$ function (line 3), which constructs a set $\mathbb{P}$ of all possible acyclic paths, starting from the input concept c, whose length is $\leq PL_c$. Next, (line 4) the $Concepts(\mathbb{P})$ function extracts all concepts from the set of paths $\mathbb{P}$ and forms a distinct set of extracted concepts $\mathbb{C}$. Next, (line 6) for each concept $c_i \in \mathbb{C}$ function $Paths2(c, c_i, \mathbb{P})$ returns a set of paths $\mathbb{P}_i$ ($\mathbb{P}_i \subseteq \mathbb{P}$) which start in concept c and end in concept c_i. Next, (line 8) for each path $p_{ij} \in \mathbb{P}_i$ between c and c_i, function:

$$SDist^p(p_{ij}) = \sum_{k=1}^{n} SDist^r_{\mathscr{R}\rightarrow\mathscr{D}}(r_k)\Big|_{\text{if direction of } r_k \text{ is } c \rightarrow c_i} \vee \vee \quad SDist^r_{\mathscr{D}\rightarrow\mathscr{R}}(r_k)\Big|_{\text{if direction of } r_k \text{ is } c \leftarrow c_i} \tag{5.9}$$

calculates the semantic distance on the path that I refer to as *the path semantic distance ($SDist^p$)*. For those $r_k \in p_{ij}$ of the same direction as a direction $c \rightarrow c_i$, function (5.9) takes $SDist^r_{\mathscr{R}\rightarrow\mathscr{D}}(r_k)$ while for r_k of the direction $c \leftarrow c_i$, it takes $SDist^r_{\mathscr{D}\rightarrow\mathscr{R}}(r_k)$.

After the algorithm calculates the path semantic distances of all paths $\mathbb{P}_i$, it calculates the semantic distance of concept c_i from the input concept c by applying function (5.10). I call this distance *the concept semantic distance* ($SDist^c$). $SDist^c(c_i, c)$ can also be considered as the relation semantic distance $SDist^r_{\mathscr{R}\rightarrow\mathscr{D}}(r(c, c_i))$ of a new single relation $r(c, c_i)$ from the concept c to c_i.

$$SDist^c(c_i, c) = SDist^r_{\mathscr{R}\rightarrow\mathscr{D}}(r(c, c_i)) = \frac{1}{\sum_{j=1}^{k} \frac{1}{SDist^p(P_{ij})}} \tag{5.10}$$

I designed function (5.10) so that it prioritizes the impact of paths with the small path semantic distances in determining the concept semantic distance. Finally, the algorithm discards all concepts from the set $\mathbb{C}$ that do not satisfy the SD_c constraint, forming in that way the output vector of the discovered related concepts $\overrightarrow{C'}$ and the vector of their semantic distances $\overrightarrow{SD}$ from the input concept c.

5.1.2 Semantic Annotation, Indexing and Linking

According to the design principles of semantic documents, I identified three main types of the document units' annotation: the standardized metadata annotation, the social-context annotation and the semantic annotation. The standardized metadata-annotations are generated by the *SemanticDoc RDFizer* module at the beginning of the authoring process as it was described in Section 5.1. The social-context annotations are generated over time based on the interaction of the SDArch social network members with the

semantic document units. I already explained the process of the social-context annotation as one of the processes realized by the social network management service (Section 4.2.4). This section is dedicated to the third annotation type, that is, the semantic annotation.

The fundamental part of the semantic annotation process is the knowledge extraction and conceptualization, which I explained in the previous section. Having the document unit's concepts discovered, the rest of the semantic annotation process is all about the linking of the discovered concepts and their relevance weights to the document unit's RDF node of the document's RDF representation. The concepts and their weights are linked to the concept's RDF node via the instances of the semantic annotation interface (`sdm:ontologicalConcept` and `sdm:conceptWeight`) defined by the annotation part of the SDM ontology (Section 3.2.2). The semantic annotation process is realized by the annotation module of the semantic document authoring service.

Besides semantic annotation, the discovered concepts are also used for concept indexing of the document units. SDArch supports a single concept index for all semantic documents stored in the SDArch semantic document repository, that is, a single concept index for the entire SDArch RDF repository. The concept index is updated any time a semantic document is added to or deleted from the repository. The concept index holds a list of ontological concepts (i.e., concept identifiers) that annotate the semantic documents stored in the repository, each of which has assigned a list of the document units it annotates. Moreover, the concept index also holds the concepts' relevance weights for the indexed document units. The main role of the concept index is to enable the semantic document search that I discuss in the next section. The SDArch also employs the concept index during the process of the semantic linking of document units. The concept indexing is realized by the indexing module of the semantic document authoring service.

One of the design principles of the semantic documents is the possibility to link document data not only by structural but also by explicit semantic links. In the semantic document model I enabled this design principle by the notion of the explicit semantic links that can be established between RDF nodes of the semantically related document units. As semantically related document units, I consider document units which share the same conceptualized semantics. The shared conceptualized semantics determine implicit semantic relationships between the document units. The role of semantic linking is to identify document units between which there are implicit semantic relationships and to set up the explicit semantic links between the document units' RDF nodes. The semantic links are implemented as instances of the linking interface that is specified by the linking part of the SDM ontology (Section 3.2.3). Figure 5.2 shows the definition of the linking interface in OWL. According to this definition, the semantic link is an instance of the `sdm:SemanticLink` class, and it is determined by two document units (`sdm:unitOne` and `sdm:unitTwo`) it links, the annotation concept (*sdm:annotationConcept*) that annotates both document units and determines the semantic relationship between them, and

the link strength (*sdm:linkStrength*). The link strength is a measure of the semantic relatedness of the linked documents. It is calculated as a product of the concept's relevance weights of the linked document units. This actually means that the more relevant the concept is for the document units the stronger is the semantic link.

```
<owl:Class rdf:ID="SemanticLink"/>
<owl:Class rdf:ID="DocumentUnit"/>
<owl:ObjectProperty rdf:ID="unitOne">
   <rdfs:domain rdf:resource="#SemanticLink"/>
   <rdfs:range rdf:resource="#DocumentUnit"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="unitTwo">
   <rdfs:range rdf:resource="#DocumentUnit"/>
   <rdfs:domain rdf:resource="#SemanticLink"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="annotationConcept">
   <rdfs:range rdf:resource="#Concept"/>
   <rdfs:domain rdf:resource="#SemanticLink"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="linkStrength">
   <rdfs:range rdf:resource="#float"/>
   <rdfs:domain rdf:resource="#SemanticLink"/>
</owl:DatatypeProperty>
```

Figure 5.2. OWL definition of the semantic linking interface

The semantic linking process is realized by the linking module of the semantic document authoring service. The semantic linking is performed after the semantic annotation and indexing, so that the linking module can easily obtain indexing (annotation) concepts of each document unit by reading the concept index. For each document unit of a semantic document that is being authored, the linking module checks the concept index to see if there are some other document units indexed by the same concepts. Then, for each found document unit, the linking module first calculates the strength of a potential semantic link, and if the strength is above a given threshold value, then it generates the link. Finally, the generated semantic link is added to the SDArch RDF repository.

By establishing the explicit semantic links not only between document units that belong to the same semantic document, but also between document units that belong to different semantic documents, SDArch can bring data of all desktop documents into an integrated information space. Moreover, by publishing semantic documents to shared RDF repositories and linking their document units with semantically related document units of other SDArch users, SDArch has potential to semantically integrate distant, desktop information spaces into the globally unified information space. Following the semantic links, the SDArch user can easily navigate across such unified information

space and discover semantically related data. The semantic navigation is one of the topics described in the next section.

5.2 Search and Navigation in Semantic Documents

The search and navigation in semantic documents is realized by the semantic document search and the semantic document navigation processes. The role of these two processes is to enable SDArch users to easily discover, access, and reuse desired data from semantic documents stored on local desktops as well as from shared collections of semantic documents.

The main feature that distinguishes the semantic document search from the document search of conventional documents is the possibility to search semantic documents by considering not only their contents, but also their conceptualized semantics. Accordingly, my main focus in developing the semantic document search was to find a way how to utilize the conceptualized semantics, so that I get optimal search results. However, by being aware that it will not always be possible to have a sufficient amount of conceptualized semantics, I also considered the full-text search as an alternative search of semantic documents. The full-text search is performed against the document units' binary content, which is stored into the SDArch binary data repository, and it should be triggered whenever the semantic document search does not retrieve any results for the given user query. Since the full-text search in semantic documents is implemented in the same way as the full-text search in existing, conventional documents, I will not further explain it.

The semantic document navigation is another process that makes distinction between conventional and semantic documents. Most of existing conventional document formats provide very limited support for the document navigation. Usually, they provide 'the table of contents - TOC' navigation based on the hierarchical relationships between document parts. Some document formats also support 'bookmarks' (i.e., marked locations in a document) that enable users to jump to marked locations within the document. Bookmarks improve the document navigation in the sense of faster access to some document parts, but still they do not help with the navigation across semantically related document parts. In contrast to conventional documents, semantic documents provide explicit links that connect semantically related document units, so that besides the navigation by following hierarchical relationships, semantic documents also enable the navigation by following semantic relationships between document units.

Besides the semantic search and the semantic navigation, which are founded on the utilization of the conceptualized document semantics, SDArch also employs the available social-context annotations while searching semantic documents. After the semantic document search, the social-context annotations of the retrieved document units are used together with the values of the user preferences from the user profile to reorder the search results so that they better correspond with the user's preferences. In other words,

SDArch uses the social-context annotations and the user preferences to personalize the semantic document search.

In the rest of the section I give detailed descriptions of the semantic document search, the search personalization process, and the semantic document navigation.

5.2.1 Semantic Document Search

According to the semantic document model, semantic document units are uniquely identified, semantically annotated units of document data that can be searched and retrieved. Accordingly, the semantic document search is the search of semantic documents for semantic document units. The semantic document search is realized by the search module of the semantic document search and navigation service (Figure 5.1).

The search process normally starts by the user constructing a query that reflects his information needs. The initial form of the user query in the semantic document search is a free-text query. Constructing free-text queries overcomes the problem of knowledge overhead, as it does not require the end user to be familiar with any particular knowledge schema (e.g., ontologies and taxonomies). However, the price of free-text queries is ambiguity. Keywords may have multiple meanings (lexical ambiguity) and a complex expression can have multiple underlying structures (structural ambiguity) [4]. Therefore, the first step in the semantic document search that I propose is *'making sense of the user query'*, that is, finding out the semantic meaning of the user query. In my approach, I model the semantic meaning of the query by means of a query concept vector, which is composed of the concepts from domain ontologies, and a corresponding query concept weight vector. However, it is not always possible to find the exact semantic meaning of the query, as there may be more than one concept which matches a single query keyword. My solution to this is to find out all the concept matches for each query keyword and calculate their relevance weights. Actually, the way I form semantic queries from free-text queries is quite similar to the conceptualization of document units' semantics (Section 5.1.1). In other words, I treat a free-text query in the same way as a document unit is treated in the knowledge extraction and conceptualization process. After syntactic and semantic matching of ontological concepts descriptions (labels) against the initial free-text query, we get the semantic query represented by the query concept vector and the query concept weight vector.

Having formed the semantic query, the rest of the search process is as follows. From the concept index we find all document units that are indexed by at least one concept from the query concept vector. After that, we calculate the similarity between such found document units and the query, and then rank them. The similarity between the query and the document units is measured by computing the similarity between the query concept weight vector and the document units' concept weight vectors previously reduced to the dimension of the query concept vector (i.e., to the number of concepts in the query concept vector). Let us suppose now that we have a query q, represented by the

concept vector $\overrightarrow{q} = [c_{q_1}, ..., c_{q_n}]^n$ and the concept weight vector $\overrightarrow{W_q} = [w_{q_1}, ..., w_{q_n}]^n$, and the document unit d, represented by the concept vector $\overrightarrow{d} = [c_1, ..., c_m]^m$ and the concept weight vector $\overrightarrow{W_d} = [w_1, ..., w_m]^m$. The reduced document unit's concept weight vector $\overrightarrow{W'_d} = [w'_1, ..., w'_n]^n$ is formed so that $w'_i = w_j$ if c_i exists in $\overrightarrow{d}$ and $c_i = c_j$ if c_i does not exist in $\overrightarrow{d}$ then $w'_i = 0$. The similarity between vectors $\overrightarrow{W_q}$ and $\overrightarrow{W'_d}$ is computed as the cosine of the angle between them:

$$Similarity(\overrightarrow{W_q}, \overrightarrow{W'_d}) = \frac{\overrightarrow{W}_q * \overrightarrow{W}'_d}{|\overrightarrow{W}_q||\overrightarrow{W}'_d|} \tag{5.11}$$

Finally, the search result is formed by ranking the document units based on their similarity to the query. Before retrieving the document units, the search module obtains the information about the document units' content streams, by querying their RDF descriptions, and then fetches their binary contents from the binary data repository. When the user wants to see the annotation data of some of the retrieved document units, the search module performs an additional step in which it executes a predefined set of SPARQL queries against local SDArch RDF repository in case of document units from local semantic documents, or against the shared RDF repository otherwise.

The effectiveness of the semantic document search in terms of precision and recalls [20] has been evaluated with two test collections. In Section 7.1), I describe the test collections and discuss the evaluation results. Moreover, I compared the effectiveness of the semantic document search with the effectiveness of the conventional full-text search and related concept-based searches.

5.2.2 Personalization of the Semantic Document Search

Two general approaches to the search personalization are: 1) query modification or query expansion based on the user profile information [102] and 2) re-ranking the search results using the user profile information [79]. In the search personalization approach that I propose, namely the socially-enhanced search personalization, I utilize the information extracted from the social context annotations of the document units and the values of the user preferences from the user profile. The approach is called the socially-enhanced search personalization because of the social context annotations that it utilizes and which are the result of the SDArch users collaboration in the SDArch social network. The core of the approach is the personalized ranking algorithm, which I describe in the rest of the section. The algorithm is implemented by the search personalization module of the semantic document search and navigation service.

The two types of input data, that the personalized ranking algorithm is founded on, are the social context annotations and the values of the user preferences. The social context annotations are specified by the annotation part of the SDM ontology (Section 3.2.2). How these annotations are generated was discussed as a part of the SDArch so-

cial network management process (Section 4.2.3). The personalized ranking algorithm is where the social context annotations are utilized. Moreover, how I specify the user preferences in the SDArch user model and how the preference values are set, was discussed as a part of the SDArch user profile management (Secdtion 4.2.2). Before I start explaining the algorithm, I first discuss a set of information that can be extracted from the social-context annotations, and the list of the user preferences that the algorithm takes in consideration.

By mining the social context annotation of each DU from the semantic document search results, we can extract the following set of information, which is of interest for the personalization process:

1. **Number of Reuses:** $N_{reuses} \in \mathbb{N}$
2. **Number of Modifications:** $N_{modifications} \in \mathbb{N}$
3. **Number of Browses:** $N_{browses} \in \mathbb{N}$
4. **List of Users:** $L_{users} = \{u_1, u_2, ..., u_m\}$ where u_i is the URI of the i^{th} user from the list;
5. **List of Applications:** $L_{applications} = \{app_1, app_2, ..., app_p\}$ where app_i is the ID of the i^{th} application from the list;
6. **Reuse Times:** $T_{reuses} = \{t_{r_1}, t_{r_2}, ..., t_{r_n}\}$ where $t_{r_i} \in \mathbb{N}$ is a time of the i^{th} reuse of the DU, represented as a number of UNIX timestamps;
7. **Modification Times:** $T_{modifications} = \{t_{m_1}, t_{m_2}, ..., t_{m_q}\}$ where $t_{m_i} \in \mathbb{N}$ is a time of the i_{th} modification of the DU, represented as a number of UNIX timestamps;
8. **Browse Times:** $T_{browses} = \{t_{b_1}, t_{b_2}, ..., t_{b_r}\}$ where $t_{b_i} \in \mathbb{N}$ is a time of the i^{th} browse of the DU, represented as a number of UNIX timestamps;

Regarding the user preferences, the personalized ranking algorithm distinguishes between two types:
preferences with enumerated values:

- $Pref_1$ - Preferred Authors;
- $Pref_2$ - Preferred Software Applications;

and, preferences with numerical value:

- $Pref_3$ - Preferred Number of Reuses;
- $Pref_4$ - Preferred Number of Modifications;
- $Pref_5$ - Preferred Number of Browses;

- $Pref_6$ - Preferred Time of the Reuse;
- $Pref_7$ - Preferred Time of the Modification;
- $Pref_8$ - Preferred Time of the Browsing;

All of the above listed user preferences are explained and discussed within the user profile management (Section 4.2.3).

The general idea of the personalized ranking algorithm is the following. First, the algorithm calculates the rank of each DU with respect to each of the preferences ($Pref_3$ - $Pref_8$) separately. Then, it calculates the final rank of the DU by summing up its ranks regarding each preference, previously multiplied by the preference importance value that comes from the user profile. For each of the user preferences, the algorithm provides the corresponding function for calculating the rank value of the document unit.

The algorithm (see Algorithm 2) starts by extracting the information (1), (2), ..., (8) for each $d_i \in \mathbb{D}$ (line 3-5), where $\mathbb{D}$ is a set of retrieved document units by the semantic document search. This is achieved by the search personalization module (Figure 5.1) executing a pre-defined set of SPARQL queries against the semantic context annotations. Next (lines 6-13), the algorithm calculates the rank value of each document unit $d_i \in \mathbb{D}$ with respect to the enumerated preferences ($Pref_1$ and $Pref_2$). First (line 8), it calculates the similarity between the preferences' lists (i.e. the list of preferred authors in case of the preference $Pref_1$ and the list of preferred applications in case of the preference $Pref_2$) and the document unit's list of users and list of applications.

$$Simil(\overrightarrow{P_j}, \overrightarrow{W_i}) = \frac{\overrightarrow{P_j} * \overrightarrow{W_i}}{|\overrightarrow{P_j}||\overrightarrow{W_i}|} \tag{5.12}$$

The $Simil$ function (5.12) calculates the similarity between the lists as the cosine of the angle between vectors $\overrightarrow{P_j}$ and $\overrightarrow{W_i}$. Vector $\overrightarrow{P_j} = [p_{j1}, ..., p_{jm}]$, $j \in \{1, 2\}$ is the weight vector of the preference $Pref_j$. For the preference $Pref_1$, the weights in the preference weight vector represent numbers of DUs that the user has reused from each of the authors. For the preference $Pref_2$, the weights in the preference weight vector $\overrightarrow{P_j}$ represent numbers of DUs that the user has reused from documents authored by each of the applications. Vector $\overrightarrow{W_i} = [w_{i1}, ..., w_{im}]$, $w_{ij} = 0 \vee 1$ is the weight vector of the document unit d_i. For $Pref_1$, the vector $\overrightarrow{W_i}$ is formed so that for each author from the preference list, who is also in the list of d_i's users, it has a weight 1. Otherwise, the weight is 0. For $Pref_2$, the vector $\overrightarrow{W_i}$ is formed so that for each application from the preference list, which is also in the list of d_i's applications, it has a weight 1 and 0 otherwise. In the next step (line 11), the algorithm calculates the rank value (5.13) for each DU regarding the two enumerated preferences. To do that, it takes into account the similarities between all DUs and the preference: $Simil(\mathbb{D}|Pref_j) = \{Simil(\overrightarrow{W_1}, \overrightarrow{P_j}), ..., Simil(\overrightarrow{W_n}, \overrightarrow{P_j})\}$.

Algorithm 2 Personalized Ranking Algorithm

1: INPUT $\mathbb{D} = \{d_1, ..., d_n\}, \mathbb{P} = \{Pref_1, ..., Pref_8\}$
2: OUTPUT $\mathbb{D}'$ {re-ranked $\mathbb{D}$ set}
3: **for all** d_i such that $d_i \in \mathbb{D}$ **do**
4: FIND: (1),(2),...,(8) by searching SCA
5: **end for**
6: **for** $j = 1$ to 2 **do**
7: **for all** cu_i such that $d_i \in \mathbb{D}$ **do**
8: $Simil(d_i|Pref_j)$
9: **end for**
10: **for all** d_i such that $d_i \in \mathbb{D}$ **do**
11: $RankValue1(d_i|Pref_j)$
12: **end for**
13: **end for**
14: **for** $j = 3$ to 8 **do**
15: **for all** d_i such that $d_i \in \mathbb{D}$ **do**
16: $Diff(d_i|Pref_j)$
17: **end for**
18: **for all** d_i such that $d_i \in \mathbb{D}$ **do**
19: $RankValue2(d_i|Pref_j)$
20: **end for**
21: **end for**
22: **for all** d_i such that $d_i \in \mathbb{D}$ **do**
23: $RankValue(d_i)$
24: **end for**
25: SORT $\mathbb{D}$ in decreasing order of $RankValues$

$$RankValue1(d_i|Pref_j) = \frac{Simil(\overrightarrow{W_i}, \overrightarrow{P_j})}{max(Simil(\mathbb{D}|Pref_j))} \tag{5.13}$$

The function (5.13) is devised so that the rank value of the document unit with the maximum $Simil$ is 1, and the rank values of the other document units fell in the range $(0, 1)$. Next (lines 14-21), the algorithm calculates the rank values for each $d_i \in \mathbb{D}$ with regard to the numerical preferences $(Pref_3 - Pref_8)$. First (line 16), it calculates the difference $Diff(Pref_j(d_i)|P_j)$ between the weight of $d_i \in \mathbb{D}$ for the preference $Pref_j$ in the scope of the retrieved document units $\mathbb{D}$, and the preference value P_j that comes from the user profile.

$$Diff(Pref_j(d_i)|P_j) = |P_j - Pref_j(d_i)| \tag{5.14}$$

I explain with the example of the preference $Pref_3$ (i.e., preferred number of reuses) how we calculate the weight of $d_i \in \mathbb{D}$ for the given numerical preference $Pref_j$ in the scope of the retrieved set of document units. The other numerical preferences ($Pref_4$ - $Pref_8$) follow the same approach. Let us denote the weight of the document unit d_i for the preference $Pref_3$ in the scope of the retrieved set of document units $\mathbb{D}$ as $Pref_3(d_i) \in [0,1]$. Moreover, let us suppose that the number of retrieved document units $\mathbb{D} = \{d_1, d_2, ..., d_m\}$ is m, each of which has been reused a certain number of times $\mathbb{N} = \{n_1, n_2, ..., n_m\}$. The weight $Pref_3(d_i) \in [0,1]$ is then calculated by the following formula:

$$Pref_3(d_i) = \frac{n_i - n_{min}}{n_{max} - n_{min}} \tag{5.15}$$

where n_{min} and n_{max} represent minimal and maximal element of $\mathbb{N}$. In the case of $n_{min} = n_{max}$, meaning that the user does not actually have different choices of document units regarding this preference, the rank value of document units regarding this preference is not calculated and consequently it is not considered in the calculation of the general rank of the document units. Formula (5.15) is also used for the preferences $Pref_4$ - $Pref_8$, but then the set $\mathbb{N}$ holds a number of modifications and a number of browses of each document unit $d_i \in \mathbb{D}$ for $Pref_4$ and $Pref_4$ respectively, while for the preferences $Pref_6$, $Pref_7$ and $Pref_8$, the set $\mathbb{N}$ holds times (i.e., numbers of timestamps) of the last reuse, last modification and last browsing of each $d_i \in \mathbb{D}$ respectively.

Having calculated differences for all document units $Diff(Pref_j(\mathbb{D})|P_j)$, in the next step (line 19) the algorithm calculates the rank value of each document unit with regard to the given numerical preference:

$$RankValue2(d_i|Pref_j) = \frac{min(Diff(Pref_j(\mathbb{D})|P_j))}{Diff(Pref_j(d_i)|P_j)} \tag{5.16}$$

The rank value of the document unit with the minimum $Diff$ is 1 and the rank values of the others fell in a range $(0,1)$. Finally, when the rank values of the all DUs, with regard to each of the preferences, are calculated the algorithm calculates the general rank of each of them (line 23):

$$\begin{aligned} RankValue(d_i) = &\sum_{j=1}^{2} RankValue1(d_i|Pref_j) * if_j \\ &+ \sum_{j=3}^{8} RankValue2(d_i|Pref_j) * if_j \end{aligned} \tag{5.17}$$

where if_j represents the preference importance factor of the preference $Pref_j$. The value of this factor comes from the user profile (Section 4.2.2).

The personalized ranking algorithm ends by sorting the DUs in the decreasing order with regard to their general ranks. This is also the end of the search process that includes the semantic document search and the search personalization. To sum up, the semantic document search is an objective search that retrieves the same results regardless of the user who executes the query. It is based on the use of the conceptualized semantic from the semantic document units. The search personalization is the supplement to the semantic document search, which takes into account the values of user preferences and the social context annotations of semantic document units to make the search results be more suitable for the user.

5.2.3 Semantic Document Navigation

In the traditional hypertext Web, browsing and searching are often seen as the two dominant modes of interaction [12]. While web browsers provide the mechanisms for navigating the Web space, search engines are the place at which that navigation process begins.

It is possible to make a correlation between the navigation on the Web and the navigation in integrated collections of semantic documents, that I refer to as semantic document navigation. In semantic document navigation, document units play a role of linked documents on the Web, while hyperlinks are replaced by the semantic links. As well as the Web navigation, the semantic document navigation begins after the initial search, in this case the semantic document search. Together, the semantic document search and the semantic document navigation should enable SDArch users to explore collections of semantic documents, whether they are stored in local or shared semantic document repositories.

```
PREFIX sdm: <http://www.semanticdoc.org/sdm.owl#>
SELECT ?targetUnit ?strenght
WHERE {
         ?link sdm:annotationConcept sdm:concept_32154
         ?link sdm:unitOne sdm:unit_42177
         ?link sdm:unitTwo ?targetUnit
         ?link sdm:linkStrength ?strength
      }
ORDER BY ?strength
```

Figure 5.3. An example navigational SPARQL query

The semantic document navigation is realized by the semantic navigation module of the semantic document search and navigation service. The navigation process requires the existence of a navigation interface, which should be a part of the SDArch presentation layer, through which the user can interact with the navigation module and

navigates across the semantic documents by following the semantic links between document units. As an example of the semantic document navigation interface, I developed a tool called the semantic document browser (Section 6.4.6), which is a part of the SDArch prototype. Figure 6.7 gives a snapshot of the tool. The navigation starts by the user browsing the details of the selected document unit from the search results, and then clicking on one of the document unit's annotation concepts. The clicked annotation concept determines the set of concrete semantic links (established over the same semantic relationship) that connect the document unit with other units. This user action activates the semantic navigation module, which takes as input the URI of the document unit and the concept's URI, forms the navigational SPARQL query, and executes the query against the RDF repository. The navigational query is formed in accordance to the semantic link specification (Figure 5.2).

Figure 5.3 shows an example navigational query. This example query returns all document units (i.e., units' URIs) that are linked to the initial document unit (`sdm:unitOne`) via semantic links determined by the clicked annotation concept (`sdm:concept_32154`). The returned document units are then ordered by the strength of the semantic links. Similar to the end of the search process, in order to obtain the content of the retrieved document units, the navigation module performs an additional step in which it first obtains the information about the content streams of the document units, and then obtains the actual document units' contents. This is the point when the navigation module finishes its work. For obtaining the annotation data of some of the retrieved document units on the user's demand, the search service is being employed. By browsing and clicking on some of the annotation concepts of the retrieved document units, the navigation process will continue. The navigation interface provides the user the possibility to go back to the previous list of document units as well.

5.3 Summary

In this chapter, I described the semantic document managed processes enabled by SDArch. The three top-level processes comprise the semantic document authoring (Section 5.1), semantic document search (Section 5.2.1), and semantic document navigation (Section 5.2.3). Each of these processes is composed of a number of low-level, sub-processes, which are realized by appropriate SDArch functional modules. The SDArch functional modules whose functionalities are involved into the semantic document management processes are encapsulated into two SDArch services, namely the semantic document authoring service and the semantic document search and navigation service.

The semantic document authoring is based on the automatic transforming conventional desktop documents into semantic documents by utilizing domain ontologies. The authoring process includes generation of document units' RDF descriptions for document units of a document to be transformed as well as their semantic annotation, indexing and linking (Section 5.1.2). Semantic annotation, indexing and linking of a

document unit all relay on the conceptualized document unit's semantics that are obtained through the knowledge extraction and conceptualization process (Section 5.1.1). The main role in this process plays the CEA algorithm that utilizes selected domain ontologies to discover ontological concepts whose instances appear in document units and to measure the relevance of such discovered concepts for the document units.

The semantic document search and navigation processes are founded on the utilization of conceptualized document unit semantics and semantic links among document units, respectively. The semantic document search starts by an initial free-text user query, which is then transformed by the semantic document search and navigation service into the semantic query (i.e., query concept vector), and executed against the SDArch concept index. After the semantic document search, the service utilizes social-context annotations of the retrieved document units and the preferences specified in the user's profile to reorder the search results so that the new order better correspond to the user. This process I called the personalization of the semantic document search (Section 5.2.2). The semantic document navigation is realized by the navigational queries executed against RDF representation of semantic links connecting document units.

Chapter 6

The SDArch prototype

In this chapter, I present the prototype of the proposed semantic document architecture, which I developed in order to prove the architecture's implementability and to enable evaluations that would validate the thesis's statement. The prototype development has followed the evolutionary prototyping approach [120]. When developing a prototype using this approach, the prototype is continually refined and rebuilt. It also allows developers to first focus on parts of the system that they understand the best, instead of developing a whole system.

The SDArch prototype has gone through several iterations in its development, following the incremental refining of the architecture's design. In this chapter, I describe a beta version of the SDArch prototype, whose development finished at the beginning of 2010. The prototype is a fully-functional software, providing the implementation of all three layers from the SDArch architecture (Figure 4.1). While the implementation of the SDArch data layer was based mainly on the deployment of an existing RDF repository, I implemented the SDArch service and user interface layers completely from scratch. Both, the SDArch services and the SDArch user interface tools are developed under open source projects hosted at SourceForge.Net[1]. The SDArch prototype was awarded second prize for innovation in 2009-2010 by Ated-ICT Ticino[2].

The main objectives of developing the SDArch prototype are the following:

- Firstly, the prototype validates that the new document architecture (SDArch) and the underlying semantic document model (SDM) are implementable and provide the intended functionality.
- Secondly, the prototype is used in an experimental (formal) validation of the architecture, including real-world experiments with end users. Conducted experiments and their results are described in Chapter 7.

[1] http://sourceforge.net/
[2] http://www.ated.ch/

- Thirdly, by being developed as an open source software, the prototype has been available for other researchers, thus allowing them to verify my findings and to extend my work with their own ideas. As an example, some of the services provided by the prototype were reused in the development of another prototype system designed within IntelLEO project[3]. IntelLEO is a scientific project, supported by the European Commission under the 7th Framework Programme, which aims to explore supportive technologies for learning and knowledge-building activities of learners in intelligent, learning extended organizations.

- Fourthly, the prototype was used personally by myself to support my personal document management and to help me refine the design principles of the architecture by 'eating my own dogfood'. 'Eating your own dog food', also called 'dogfooding', is when a company uses the products that it makes [59]. Dogfooding can be a way for a company to demonstrate confidence in its own products, and hence a kind of testimonial advertising.

The rest of the chapter is organized as follows. I start with a brief outline of the software (i.e., existing programing libraries) that I used in the prototype implementation (Section 6.1). Then, I describe the implementation of each of the three architecture's layers starting with the SDArch data layer (i.e., RDF repository, text index and concept index - Section 6.2), then the SRArch services (Section 6.3), and finishing with some example applications of the SDArch user interface (Section 6.4).

6.1 Used Software

In the prototype implementation, I used a number of existing, open source, software libraries for the implementation of some functionalities intended by SDArch. Most of these libraries had already been used in the implementation of a number of scientific projects, and had become de-facto standards in their application areas. In this section I just list the used libraries and briefly describe each of them. The concrete role of each library in the scope of the SDArch prototype as well as in which functional module of the prototype the library is involved, is discussed later in this chapter.

C# is the programing language used to develop the SDArch prototype. I used C# version 3.0 and the Microsoft .NET 3.5 framework.

Open XML SDK 2.0 for Microsoft Office[4] is a .NET API for working with the Open XML file formats. The SDK is built on the System.IO.Packaging API and provides strongly-typed classes to manipulate documents that adhere to the Office Open XML File Formats

[3] http://www.intelleo.eu/

[4] http://openxmldeveloper.org/default.aspx

Specification. The Office Open XML File Formats specification is an open, international, ISO/IEC 29500 standard. Starting with Microsoft Office 2007, the Office Open XML file formats have become the default file format of Microsoft Office.

SemWeb.NET[5] is a Semantic Web/RDF library written in C# for Mono or Microsoft's .NET. The library can be used for reading and writing RDF (XML, N3), keeping RDF in a persistent storage (memory, MySQL, etc.), querying the persistent storage via a simple graph matching and SPARQL, and making SPARQL queries to remote endpoints.

Sesame[6] is a semantic web API which features an RDF storage layer, inference support, querying using SPARQL, an RDF parser, and other features. In practice, Sesame was used as the RDF repository, and SemWeb.NET as a frontend API to interact with it.

Windows Communication Foundation - WCF is a part of the .NET Framework that provides a unified programming model for building service-oriented applications that communicate across the web and the enterprise.

Lucine.Net[7] is a source code, class-per-class, API-per-API and algorithmatic port of the Java Lucene full-text indexing and search engine to the C# and .NET platform.

WordNet.Net[8] is a .Net library for WordNet. WordNet is a lexical database for the English language, which groups English words into sets of synonyms called synsets and provides various semantic relations between these synonym sets.

GATE[9] is an integrated development environment for natural language processing, including components for diverse natural language processing tasks, e.g. parsers, tagging tools, information retrieval tools, information extraction tools for various languages, and many others. GATE tools are implemented in Java. In order to use the GATE functionalities in the SDArch modules that are implemented in .NET, I have converted the GATE libraries of my interest to .NET DLL libraries using IKVM.NET utilities.

Visual Studio Tools for Office - VSTO[10] is a set of development tools available in the form of project templates and runtime components that allow extensions to the Office applications to be written in CLI compliant language (such as C#) as well as to use functionality and user interface constructs from Office applications in .NET applications.

[5] http://razor.occams.info/code/semweb/
[6] http://www.openrdf.org/
[7] http://lucene.apache.org/lucene.net/
[8] http://opensource.ebswift.com/WordNet.Net/
[9] http://gate.ac.uk/
[10] http://msdn.microsoft.com/en-us/vsto/default.aspx

6.2 Implementation of the SDArch RDF Repository, Text Index and Concept Index

The SDArch RDF repository provides storage and access capabilities to RDF representations of semantic documents. It encapsulates the Sesame 2 RDF repository and uses the SemWeb RDF Library to interact with it. The set of features provided by the SDArch RDF repository includes:

- reading and writing RDF statements,
- MySQL DB-backed persistent RDF storage,
- SPARQL query support,
- full-text query support.

Resources whose RDF descriptions are stored in the SDArch RDF repository (in our case, semantic document units) have to be identified by globally unique URIs. Therefor, for the SDArch implementation, a URI scheme and naming convention had to be created. I created the SDArch URI schema as follows:

```
http://semanticdoc.org/[user OpenID]/[resource ID]
```

The parts of the schema are:

- **semanticdoc.org** - a reserved DNS domain name for semantic documents;
- **user OpenID** - an OpenID identifier identifying the SDArch user. OpenID is an open, decentralized standard for the authentication of online users. Each SDArch user must be identified by an OpenID (Section 4.2.3);
- **resource ID** - a document unit identifier which is generated based on the following pattern '{document unit label} + # + {timestamp of the creation time}'. Document unit labels are the same as the labels of the document unit concepts defined by the SDM ontology (e.g., `sdm:paragraph`, `sdm:section`, `sdm:table` and `sdm:slide`). A timestamp of the creation time is generated by the *SemanticDoc RDFizer* module (Section 5.1) of the semantic document authoring service, during the authoring of the document unit's RDF representation;

The SDArch URI schema ensures both global and local uniqueness of the document units' URIs. Global uniqueness is achieved by the combination of the reserved DNS domain name (`semanticdoc.org`) and the user OpenID. Local uniqueness, that is, uniqueness of document unit URIs within the local desktop environment, is achieved by the timestamp of the document units' creation time. Here is an example document unit URI:

```
http://semanticdoc.org/sasanesic.myid.net/paragraph#1281022990
```

The SDArch provides two ways of searching data stored in the RDF repository: the semantic document search and the full text search. However, my primary focus in the prototype implementation was on the semantic document search. The full-text search is considered only as an optional search employed in case of ineffective semantic document search. Ineffective semantic document search can be caused by insufficient conceptualized semantics. The semantic document search utilizes the concept index and executes appropriate SPARQL queries against RDF data to obtain the search results. The SDArch handles a single concept index for the whole SDArch RDF repository. The concept index is realized as an in-memory hash table. The full-text search utilizes Lucene to index the binary content of the semantic document units, which is stored into the SDArch binary data repository. Similar to the concept index, the SDArch handles a singular text index for the whole binary data repository. Lucene indexing operates on the level of the atomic document units, as the atomic document units are holders of the binary data (Section 3.2.1) in semantic documents.

6.3 Implementation of the SDArch Services

For the implementation of the SDArch middleware services, I used Windows Communication Foundation or WCF framework. WCF is an application programming interface in the .NET Framework for building connected, service-oriented applications. WCF is designed in accordance with service oriented architecture principles to support distributed computing where services are consumed by consumers. WCF services provide a WSDL interface (Web Services Description Language), which any WCF client can use to consume the service, regardless of which platform the service is hosted on. WCF implements many advanced web services (WS) standards such as WS-Addressing, WS-ReliableMessaging and WS-Security. Moreover, WCF includes predefined bindings for most common communication protocols such as SOAP over HTTP, SOAP over TCP, and SOAP over Message Queues. Interaction between a WCF service endpoint and a WCF client is done using a SOAP envelope. SOAP envelopes are in simple XML form that makes WCF platform independent.

All SDArch middleware services are implemented as WCF services. The communication among the services and between the services and the SDArch presentation layer is realized by exchanging SOAP messages over HTTP. The internal functionalities of the SDArch services are grouped into 15 SDArch functional modules, which are implemented as .NET code assemblies. The code is organized into 14 namespaces, containing altogether 77 .NET Framework types (i.e., classes and interfaces). In .NET an assembly provides a fundamental unit of physical code grouping, while a namespace provides a fundamental unit of logical code grouping. Moreover, a single assembly may contain many types whose hierarchical names have different namespace roots, and a namespace

may span multiple assemblies. Each assembly is stored as an .exe (executable) or .dll (dynamic-link-library) file. The assemblies of the SDArch functional modules are stored as .dll files. The programing language that I used for the implementation of functional modules is C#. Table 6.1 exhibits code statistics of the implementation of the SDArch services.

Number of Services	5
Number of Functional Modules (.NET Assemblies)	15
Number of .NET Namespaces	14
Number of .NET Types (Classes and Interfaces)	77

Table 6.1. SDArch services - implementation statistics

The SDArch services are developed as an open source project hosted at SourceForge[11]. The MSDN-style API documentation of all the code assemblies modules is also available online[12]. I used Microsoft's Sandcastle[13] to generate the API documentation. For the evaluation purposes, I have deployed the services on the University's research server[14] and all of them are publicly accessible. In the rest of the section I briefly describe the implementation of each of the SDArch services.

Semantic Document Authoring Service: The current implementation of the semantic document authoring service has support for the transformation of MS Office documents (i.e., Word and PowerPoint) to semantic documents. For the next generation of the service, I plan to add support for other document formats such as PDF and OpenOffice documents as well. The service's functionalities are grouped into five modules: SemanticDoc RDFizer, Knowledge Extraction and Conceptualization, Annotation, Indexing and Linking modules.

In order to access and manipulated data from MS Office documents (i.e., Word and PowerPoint), SemanticDoc RDFizer utilizes the Open XML SDK 2.0 for MS Office. To generate RDF representations of document units, RDFIzer utilizes SemWeb RDF library. Knowledge Extraction and Conceptualization module performs the lexical expansion of ontological concept descriptions (concept labels,) by utilizing WordNet.Net lexical library, then performs the syntactic matching by utilizing GATE, a natural language processing library, and finally performs the semantic matching by applying the concept exploration algorithm (Section 5.1.1). The indexing module performs the concept indexing and the full-text indexing. The full-text indexing is realized by using the Lucine.Net

[11] http://sourceforge.net/projects/sdarch/
[12] http://www.semanticdoc.org/sdms/Help/
[13] http://sandcastle.codeplex.com/
[14] http://www.semanticdoc.org/

library. The implementation of the semantic annotation and linking modules did not rely on the use of any external libraries. I implemented these modules completely from scratch.

Semantic Document Search and Navigation Service: The internal functionalities of the semantic document search and navigation service are realized by three functional modules: the semantic document search, the search personalization and the semantic document navigation modules. The semantic document search module realizes the semantic document search and the full-text search of semantic documents. The module utilizes the SemWeb RDF library for executing SPARQL queries employed during the semantic document search. For the full-text search, the module utilizes the Lucene.Net library. The search personalization module realizes the personalized ranking algorithm (Section 5.2.2). The implementation of this module did not relay on the use of external libraries. I provided my own implementation for all the functionalities of the module. The semantic document navigation module realizes the semantic navigation process by executing navigational SPARQL queries (Figure 5.3) against the linked RDF data. Similar to the search module, the navigation module utilizes SemWeb RDF library for the execution of the SPARQL queries.

Ontology Management Service: The SDArch prototype currently supports only OWL ontologies. Any domain ontology from the Semantic Web, which is an OWL ontology, can be used by the SDArch prototype. However, in order to be able to use an existing ontology, the SDArch must obtain a copy of the ontology file (RDF/XML ontology representation) and store it in the ontology repository. The ontology repository is a part of the RDF repository. The ontology management in SDArch is realized by the ontology management service. The internal functionalities of the service are grouped into two functional modules: the ontology management and the ontology publishing modules. The ontology management module utilizes the SemWeb RDF library to interact with ontological data from the repository. The ontology publishing module publishes the local ontologies (i.e., ontology files stored on the local desktop RDF repository) into the shared RDF repository.

Social Network Management Service: The social network management service provides three groups of functionalities: functionalities responsible for the managements of the user groups within the SDArch social network, functionalities that enable SDArch users to publish and share their semantic documents with other members of the SDArch social network, and functionalities that enable the generation of social-context annotations of the shared semantic documents. For each group of functionalities I dedicated one functional module in the service implementation.

The social network data, that is, data describing user groups in the SDArch social network, is represented as RDF instances of the social network ontology (Section 4.2.4);

and those instances are stored in the shared RDF repository. The social-context annotations are stored in the shared RDF repository too. The shared RDF repository is implemented in the same way as the local desktop SDArch RDF repository, except that it exposes remotely accessible SPARQL endpoint over HTTP, and is deployed on the server machine. Publishing and sharing semantic documents within the SDArch social network, means uploading document RDF representations into the shared RDF repository, while the binary contents of the documents stays on the local desktop (Figure 4.6).

User Profile Management Service: The functionalities of the user profile management service are realized by two functional modules. The first module, the profile read&write, is responsible to create the RDF representation of the user profile, to store such RDF representation of the profile into the RDF repository, and to read and write profile data when it is requested. To manipulate with the RDF data of the user profile, the module utilizes the SemWeb RDF Library. The second module, the profile update, is responsible for updating the values of the profile's dynamic parameters. As I explained in Section 4.2.3, these values are learnt over time by monitoring the user activities. The update procedure, that is, the way the profile update module calculates new values of the parameters, is also explained in Section 4.2.3.

6.4 Implementation of the SDArch User Interface

The SDArch presentation layer is platform independent and can contain desktop-based, Web-based, and mobile phone applications. As a long-term goal I plan to develop a completely new application suite for authoring, searching, browsing, and editing semantic documents. In the implementation of the current version of the SDArch prototype, my strategy regarding the presentation layer was to extend some well-known, document authoring suites by adding support for semantic documents. The main motivation for that was to allow users still working in familiar environment to take advantage of the new document architecture. Moreover, I wanted to show the interoperability of the proposed semantic document model and existing, conventional document formats, since SDArch uses conventional document formats as the human-readable representation of semantic documents. Accordingly, I have decided to extend MS Office 2007 with a set of tools/applications that I called 'SemanticDoc' tools. I have chosen MS Office mostly because of its wide usage and popularity, thus avoiding a potential problem of recruiting a sufficient number of participants for the followed usability evaluation (Section 7.2).

SemanticDoc tools extend MS Office with a set of tools that enable users to deal with semantic documents. In other words, they provide access to the SDArch services from within MS Office (i.e., MS Word and MS PowerPoint). Since SDArch enables users to share their semantic documents and to form a social network around shared documents, SemanticDoc tools actually turned MS Office into a social environment.

SemanticDoc tools are implemented and integrated into MS Office as MS Office add-ins. For their development I used the Visual Studio Tools for Office (VSTO). VSTO enables the integration of the added office add-ins into the MS Office programing object model. Similar to the implementation of the SDArch services, the implementation of SemanticDoc tools is done in the C# programing language. In addition, all SemanticDoc tools are developed under the open-source project hosted at SourceForge[15] .

SemanticDoc tools can be installed and run on Windows XP/Vista/7 with installed MS Office 2007. The following is a list of the installation prerequisites:

- Windows Installer 3.1;
- .NET Framework 3.0;
- Visual Studio Tools for Office System 3.0 Runtime;

When a user starts to install SemanticDoc tools, the setup program will check if the prerequisites are installed on the system, and if not it will automatically download them from the vendors' web sites and install them. In order to get and install the installation prerequisites, the installation process requires an internet connection.

Figure 6.1. SemanticDoc MS Office ribbon menu tab

After successful installation, when the user opens MS Office (i.e., MS Word or MS PowerPoint) SemanticDoc tools will be automatically loaded and a new 'SemanticDoc' ribbon menu tab (Figure 6.1) will appear in MS Office. A graphical design of the added ribbon tab follows the main design principles of MS Office. SemanticDoc tools are grouped and accessible through several toolboxes. Each toolbox contains tool(s) which provide the interface for accessing a certain group of SDArch services. In the rest of this section I briefly describe and illustrate each of the SemanticDoc tools. More detailed information, snapshots and demos of the tools can be found at the project's Web cite[16].

6.4.1 User Account and Profile Tools

In order to use the SDArch services and to participate in the SDArch social network, a user first needs to open an SDArch account and get a default user profile. The user can open the account by using the account manager tool and providing an OpenID.

[15] `http://sourceforge.net/projects/semdoc/`
[16] `www.semanticdco.org`

The account manager validates the OpenID by sending the validation request to the OpenID's provider, and if the OpenID is valid, creates a default user profile. The role of the OpenID is twofold. Firstly, as a part of the SDArch URI schema (Section 6.2), it contributes to the unique identification of the user's resources (i.e., semantic document units). Secondly, it is used to uniquely identify the SDArch user within the SDArch social network. By opening the account the user gets an empty, default profile.

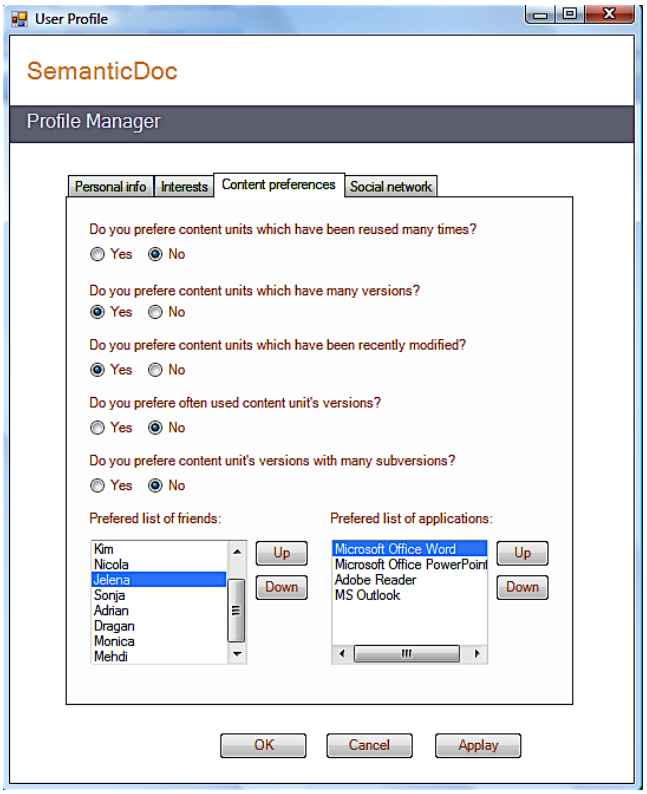


Figure 6.2. User profile manager

The user profile manager is another tool that enables the user to edit and manage the user profile. The user profile manager invokes the SDArch user profile management service. Through the user profile manager, the user can specify: a user basic info (e.g., name, occupation, and e-mail address), a list of user interests and projects, and a list of the SDArch social network members with whom the user intends to share semantic documents. Moreover, for each of the user preferences (Section 4.2.3), the user can specify the initial value, which will be adjusted/learned automatically over time. I enabled the user to set the initial values of the preferences in order to avoid so-called 'cold start' and enable the search personalization (Section 5.2.2) from the very beginning. Figure 6.2 shows a snapshot of the user profile manager displaying a user interface for setting up the user preferences.

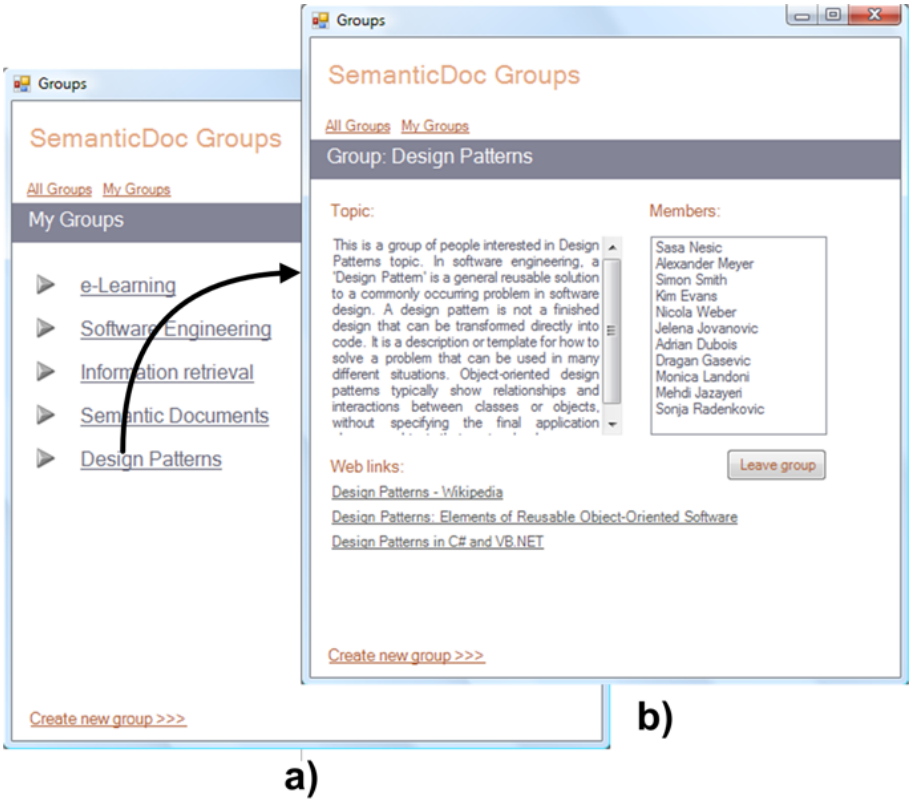


Figure 6.3. Social network manager: a) a list of all social groups; b) a detailed view of a selected group

6.4.2 Social Network Manager

Members of the SDArch social network can organize themselves into groups dedicated to particular topics of interest. By using the social networking manager tool, every SDArch user can join or leave an existing group, or initiate a new group. To initiate a new group, the user needs to specify the group's topic and to provide some topic-related information (e.g., the topic's short description and the list of the topic's Web references). Figure 6.3 shows a snapshot of the tool displaying: a) a list of all existing groups, and b) group details of a selected group. In the current prototype implementation, there is no restriction for joining existing groups, that is, all groups are available to all members of the SDArch social network. To access and manage groups' data, the social network manger invokes the methods of the SDArch social network management service.

The motivation for forming separate user groups was not to get a number of separate semantic document repositories, but to better integrate documents of the same topic of interest. Documents from the same group share the same conceptualization vocabulary (i.e., ontology), which is the key precondition of the semantic linking and the semantic

document navigation. All members of the same group use the same domain ontology for the semantic annotation, indexing and linking of the group's documents.

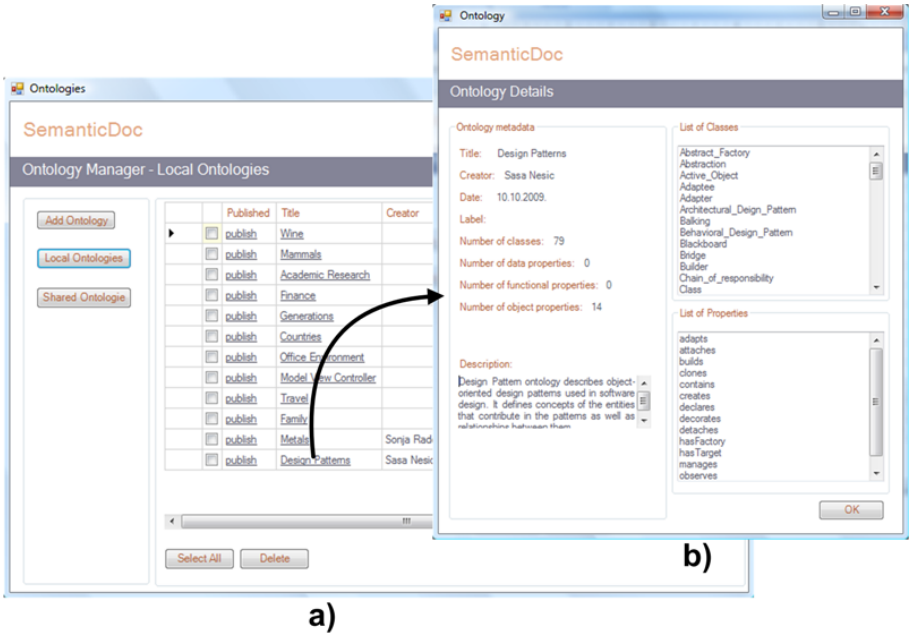


Figure 6.4. Ontology manager: a) a list of all ontologies; b) a detailed view of a selected ontology

6.4.3 Ontology Manger

The ontology manager tool enables SDArch users to manage ontologies that are used by SDArch. First, the ontology manager enables users to add a desired domain ontology to the SDArch ontology repository. The only limitation is that the ontology must be an OWL ontology stored in the RDF/XML file format. Then, for each ontology from the ontology repository, the user can browse the ontology details: the ontology's metadata (e.g., creator, short description, and creation time and data) and the list of the ontology's concepts and properties. Moreover, if the user is a member of the SDArch social network, the ontology manager enables him to publish the ontologies from his local, desktop ontology repository to the shared ontology repository as well as to browse the shared ontologies. Figure 6.4 gives snapshots of the tool displaying: a) a list of all ontologies from the repository and b) details of a selected ontology. The ontology manager invokes the methods of the ontology management service to access and manipulate the ontologies.

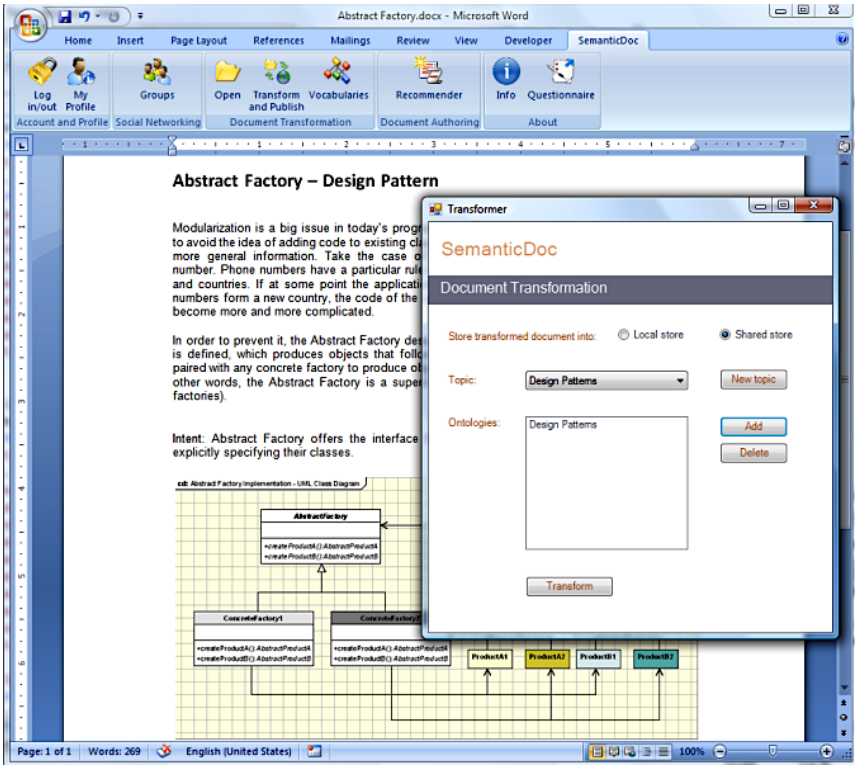


Figure 6.5. Document transformer and publisher

6.4.4 Document Transformer and Publisher

This tool enables the SDArch users to transform an active office document (i.e., a document that is opened in MS Word or MS PowerPoint) to a semantic document. The semantic document obtained during the transformation can be stored into a local, desktop semantic document repository or published to a shared semantic document repository. The transformation process is completely automated. All the user needs to do, before initiating the transformation, is to select (an) appropriate domain ontology/ies (domain ontology/ies that conceptualize the document's topic) and the destination repository for the semantic document. If the user selects the shared repository, then he can also specify the user group to which document collection to add the semantic document. The document transformer and publisher invokes methods of two SDArch services, the semantic document authoring and the social network management services. The first service is deployed for the document transformation and the other one for the document publishing. Figure 6.5 shows a snapshot of the tool together with a sample Word document to be transformed.

6.4.5 Document Recommender

The document recommender tool provides the user interface for the personalized semantic document search. This tool is a starting point from where the user starts to explore semantic documents, and the semantic documents can be stored in either a local or shared repository. The exploration process, initiated by the personalized semantic document search done by the document recommender, is then continued by the semantic document navigation in the semantic document browser. The tool is called 'document recommender' because it retrieves the search results (i.e., document units) in an order which is adjusted (recommended) to the user's personal preferences.

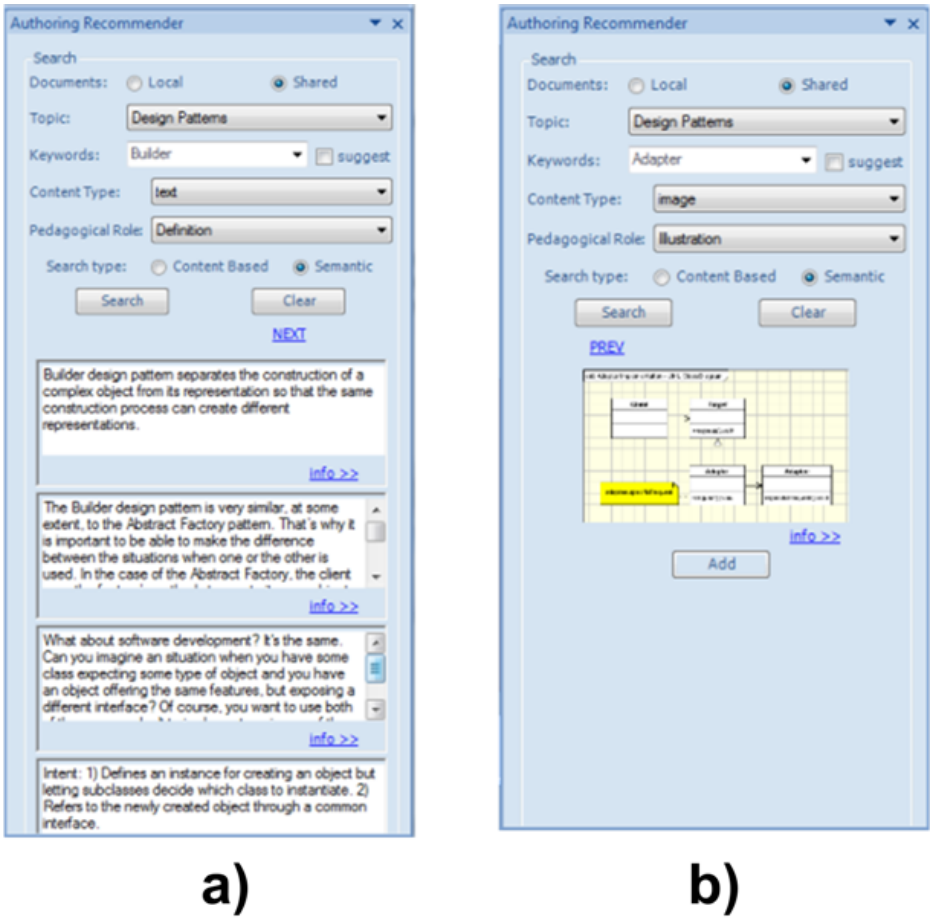


Figure 6.6. Document recommender: a) an example search for textual document units, and b) an example search for document units of the image content type

The user interface of the document recommender enables the user to specify the following search parameters. Firstly, the user specifies which semantic document repository will be searched (i.e., local or shared). Secondly, the user specifies the user query

in a form of the free-text keyword query. The tool offers the auto-completion keyword-suggestion support, which helps the user while specifying the query. Suggested terms are concept labels from domain ontologies that have been used for the semantic annotation and indexing of the semantic documents from the specified repository. The suggested terms, if used, result in a better quality of the semantic query (Section 5.2.1), that the search service generates from the initial keyword query. Thirdly, the user selects the content type of desired document units (i.e., text, image, audio or video). Finally, the user specifies the search type: the semantic document search or the full-text search. As it was explained in Section 5.2, these are the search types which are supported by the semantic document search service. If the user selects the semantic document search, the keyword query will be transformed into a semantic query and then executed by the service. Otherwise, the service executes the initial keyword query. In addition, during my work on this thesis I have been also investigating applicability of the proposed semantic document search in e-learning domain [90, 87, 96]. For that application domain, I realized that in addition to the mentioned querying parameters, it would be useful to support the pedagogical role a desired document unit needs to play. Thus, if the user searches for e-learning content, the search interface provides an additional element for specifying a pedagogical role [71] of the document units (e.g., definition, example and illustration).

When the search is done, the document recommender displays previews of the retrieved document units to the user. Figure 6.6 gives a snapshots of the document recommender displaying: a) the search form and previews of top-ranked textual document units, and b) the search form and previews of top-ranked document units of image content type. For each of the retrieved document units, the user can see the detailed view including document unit content and document unit annotation data. The detailed view is shown in the semantic document browser, which is another tool launched from the document recommender.

6.4.6 Semantic Document Browser

The semantic document browser enables the user to browse details of document units and to navigate across semantic documents following semantic links between document units. In the current implementation, the browser can be launched from the document recommender by clicking on previews of the search results. For the next generation of the tool, I plan to enable its individual launching and the possibility to start the semantic navigation not only from the search results but also by entering the URI of an initial document unit.

The main window of the semantic document browser is composed of two panels (Figure 6.7). The right panel displays the document unit' content, metadata annotations (e.g., creator and creation date), and information extracted from the document unit's social-context annotations (e.g., the number of the document unit's reuses and

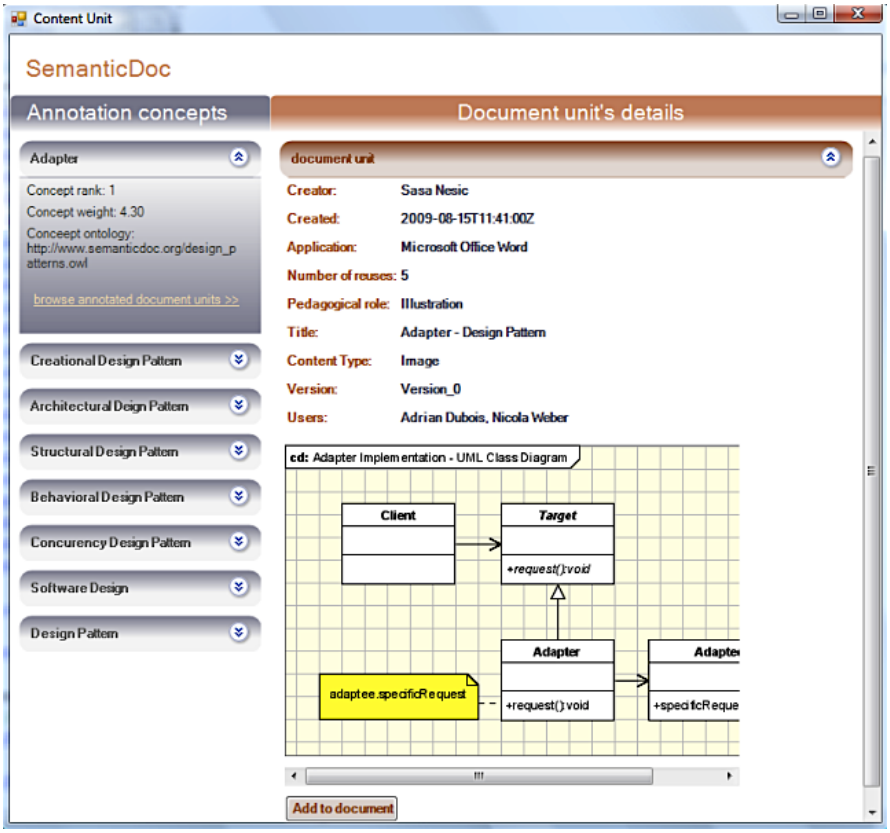


Figure 6.7. Semantic document browser

the list of SDArch users who have reused the document unit). The left panel displays an ordered list of ontological concepts that annotate the document unit. For each annotation concept, the user can see the concept's rank, the concept's relevance weight for the document unit, and the ontology in which the concept is defined. Moreover, if there exist document units that are linked to the document unit via semantic links determined by the annotation concept, the browser displays the link labeled as 'browse annotated document units'. By the user clicking on this link, the browser initiates the semantic document navigation process (Section 5.2.3) and invokes the SDArch semantic document navigation service. The navigation results (i.e., discovered document units) are ordered by the strength of the semantic links between them and the initial document unit, and are displayed on the right panel of the browser's window.

6.5 Summary

By developing the SDArch prototype, I aimed to validate that SDArch is implementable and to enable experimental (Section 7.1) and usability (Section 7.2) evaluations of the

intended SDArch functionalities.

The prototype has gone through several versions in its development, following the incremental refining of the architecture's design. The actual version, which I presented in this chapter, is the feature-complete beta release whose development finished at the beginning of 2010. The prototype is a fully-functional, providing the implementation of all the three SDArch layers (i.e., the semantic document repository, services and tools). It has been developed as an open source software hosted under two SourceForge.net projects. One of the projects provides the implementation of the SDArch semantic document repository (Section 6.2) and the SDArch services (Section 6.3), while the other one provides the implementation of the SDArch tools (Section 6.4).

Chapter 7

Evaluation

In this chapter, I report on the results of the evaluation that I conducted in order to validate the thesis statement:

> ***"Semantic documents integrate desktop documents into a unified desktop information space, and enable data from desktop documents to be integrated into a unified information space of social communities."***

I designed the evaluation so that it answers the two research questions that stem from the thesis statement:

- **Q1:** How do semantic documents improve information finding and retrieval in semantically integrated document collections?
- **Q2:** How do semantic documents facilitate desktop users in completing tasks that draw data from both a personal desktop and social communities?

The evaluation included two studies: 1) the evaluation of the semantic document information retrieval (corresponding to the research question **Q1**), and 2) the usability evaluation of the SDArch services and tools (corresponding to the research question **Q2**). The main objective of the first evaluation study was to evaluate the effectiveness of the semantic document search by comparing it with related concept-based search approaches and the conventional full-text search. Since the semantic document annotation is fundamental for the semantic integration (i.e., semantic linking and indexing) of desktop documents, and thus, also for the semantic document search and navigation, the effectiveness of the semantic document annotation was also considered in the study. The main objective of the second evaluation study was to show that SDArch services and tools can improve the effectiveness and efficiency of desktop users in completing their daily tasks that rely on both data from a personal desktop and shared data of social network communities.

The chapter contains two main sections (Section 7.1 and 7.2) that correspond to the conducted evaluation studies, and a section providing some summary remarks on the evaluation (Section 7.3).

7.1 Evaluation of Semantic Document Information Retrieval

The standard approach to the evaluation of information retrieval systems [20] requires a test collection consisting of:

- a document collection,
- a test suite of information needs, expressible as a query set, and
- a set of relevance assessments, standardly a binary assessment of either *relevant* or *nonrelevant* for each query-document pair.

The essence of the evaluation revolves around the notion of *relevant* and *nonrelevant* documents with respect to a user information need. A document is relevant if it addresses the stated information need. A document in the test collection is usually given a binary classification as either relevant or nonrelevant. However, in practice, a document relevance can be considered as a scale, with some documents highly relevant and others marginally relevant. Accordingly, some evaluation approaches employ multiple degree relevance assessments such as Cumulative Gain (CG) and Discounted Cumulative Gain (DCG) [69]. Collecting relevance assessments is a time-consuming and expensive process involving human beings. Therefore, both the document collection and the query set should be of a reasonable size. Moreover, a human is not a device that reliably performs a judgment of document relevance, rather human relevance judgments are quite subjective and variable. One solution to avoid eventual problems related to the relevance assessment, is to use standard test collections that provide document relevance judgments for predefined sets of queries. Some of the most often used standard test collections include the Cranfield collection, TREC - Text Retrieval Conference Collection, NTCIR - NII Test Collections for IR Systems, and CLEF - Cross Language Evaluation Forum. Another solution is *pooling*, an approach where document relevance is assessed over a subset of the collection that is formed from the top K documents returned by a number of different IR systems (usually the ones to be evaluated) or documents found by expert searchers in an interactive process.

Having a test collection ready, the next issues in the evaluation is how to measure the effectiveness of the information retrieval system. The two most frequently used measures for information retrieval effectiveness are *Precision* and *Recall* which are defined as follows.

Precision (P) is the fraction of retrieved documents that are relevant:

$$Precision = \frac{\#relevant\ items\ retrieved}{\#retrieved\ items} = P(relevant|retrieved) \tag{7.1}$$

Recall (R) is the fraction of relevant documents that are retrieved:

$$Recall = \frac{\#relevant\ items\ retrieved}{\#relevant\ items} = P(retrieved|relevant) \tag{7.2}$$

In a ranked retrieval context, appropriate sets of retrieved documents are naturally given by the top K retrieved documents. For each such set, precision and recall values can be plotted to give a precision-recall (P-R) curve. It is useful to remove potential jags from the curve, and the standard way to do that is by forming the interpolated precision at certain recall levels. The most often used is the 11-point (0.0, 0.1, ... , 1.0) interpolated average precision. For each recall level, there should be calculated the arithmetic mean of the interpolated precision at that recall level for each query in the test collection. Such calculated values are then used for plotting the P-R curve.

The evaluation approach, which I used for the evaluation of semantic document information retrieval, relies on the standard evaluation approach outlined above. However, specifics of the proposed semantic document search as well as capabilities of the SDArch prototype required some adjustments of the standard evaluation approach. Firstly, besides the document collection, the query set, and document relevance assessments, the test collection should have also included an appropriate domain ontology (or a set of domain ontologies) that is used for the semantic annotation, indexing and linking of the document collection. Secondly, the transformation capabilities of the SDArch prototype restricted the document collection to MS Office (i.e., Word and PowerPoint) documents. As a consequence of these two 'preconditions', I was not able to identify any standard test collection that would be suitable for evaluation of semantic document information retrieval. In fact, the use of some of the existing, standard test collections would require a lot of document transformation efforts, but what would be even bigger problem is finding appropriate domain ontologies. Creation of new domain ontologies, conceptualizing domains of the standard document collections, was not an acceptable solution too, since the engineering of domain ontologies requires adequate domain knowledge and is a time consuming task. It also would not be in accordance with the semantic document design principles (Section 3.1), which suggest the reuse of existing, well-defined domain ontologies instead of creating new ones. Therefore, I decided to form my own test collections, which were a tradeoff between available domain ontologies and document collections that corresponded to the domains of those ontologies.

In the rest of the section, I first discuss the evaluation goals and explain the evaluation procedure. After that, I present the evaluation results obtained by conducting the evaluation on two test collections that I formed for the purpose of this evaluation. The discussion on the evaluation results concludes the section.

7.1.1 Evaluation Goals

Over the last decade a considerable number of ontology-driven information retrieval approaches has been developed with a goal to enhance the search and retrieval by making use of ontological annotations [110, 44, 68, 15, 128]. A central problem of ontology-driven information retrieval so far, has been a problem of having a substantial amount of accurate ontological annotations. Most existing ontology-based annotation approaches rely on syntactic matching [81] of ontological concept descriptions (i.e., concept labels) against a document content. In spite of the advanced data mining and NLP techniques applied in these approaches, incomplete and ambiguous concept descriptions usually lead to the insufficient and inaccurate annotations. Some approaches, such as [57, 31, 1, 110], try to enlarge the amount of the ontological annotations by considering ontological concepts which are related to those concepts discovered by the syntactic matching. Such concepts are usually referred to as semantic matches and the process of their discovery as semantic matching [81]. Combinations of syntactic and semantic matching can increase the amount of semantic annotations, but it opens the problem of the annotations' relevance. Therefore, one of the most important issues in this scenario is how to assess the relevance of the discovered semantic matches and to use the most relevant of them for the annotation.

In the semantic document annotation and indexing approach (Section 5.1.2) that I proposed, I combine lexically-expanded syntactic concept matching and semantic concept matching realized by the concept exploration algorithm (CEA) to generate ontological annotations. As explained in Section 5.1.1, the general idea of the CEA algorithm, is to explore the given domain ontology(ies) starting from an input concept (i.e., a concept coming from the syntactic matching) to find ontological concepts which satisfy the given semantic distance constraint (SD_c) and path length constraint (PL_c). SD_c represents the maximum semantic distance from the input concept up to which the algorithm can traverse the ontology graph. PL_c is the maximum path length, that is, a number of hops (i.e., ontology relations) from the input concepts that the algorithm is allowed to perform while traversing the ontology graph. My hypothesis regarding these two CEA parameters is the following:

➥**H1:** *There exist optimal the values of the SD_c and PL_c parameters for each semantic document collection (i.e., applied domain ontology), with respect to the optimal semantic document search*

Once the semantic concept matches are retrieved by the CEA, the next issue is to determine their relevance for the document unit they should annotate. The function (Formula (5.8)), which I introduced for this purpose takes into account the relevance weight of the initial concept (i.e., the syntactic match) and the semantic distance between the syntactic and the semantic match calculated by the CEA to calculate the relevance of the semantic matches. This function is generic (with a generic parameter β),

which satisfies the boundary conditions (i.e., when the semantic distance is equal to 0 and tends towards infinity) regardlessly of the value of the parameter β. Similar to the parameters SD_c and PL_c, my hypothesis regarding the parameter β is:

➥**H2:** *There exists the optimal value of the parameter β for each semantic document collection (i.e., applied domain ontology), with respect to the optimal semantic document search.*

Accordingly, the first evaluation goal was to validate the above stated hypotheses (H1 and H2) by determining the optimal values of the SD_c, PL_c, and β parameters for test collections that were considered in this evaluation study.

The second goal of the evaluation, which is also considered as the main evaluation goal, was to measure the effectiveness of the semantic document search and to compare it with the effectiveness of other ontology-driven, search approaches as well as the conventional full-text search. Full-text search is enabled in the SDArch prototype by deploying the Lucine text indexing and search library. The ontology-driven (concept-based) search approaches, which I find important to compare to the semantic document search, mainly implemented some variations of the syntactic concept matching [81] enhanced by the use of lexical vocabularies such as WordNet. However, I experienced several problems while trying to determine which of the existing, ontology-driven approaches to consider in the evaluation. First, most of the existing approaches were only conceptually described and did not provide any prototype implementation. Second, for those approaches, for which it was specified that they provide some prototype implementations, the prototypes were either unavailable or too complex to be installed and run. So, the compromise solution that I chose was to categorize the ontology-driven search approaches, into two general categories: 1) the concept-based searches that apply the simple syntactic matching, and 2) the concept-based searches that apply the lexically-expanded syntactic matching, and then to try to provide prototype simulations for them. These prototype simulations I realized by modifying the actual SDArch prototype. The modifications on the SDArch prototype were easily done, since the syntactic concept matching, lexical expansion of the syntactic matching matching, and semantic matching were all implemented by separate methods in the SDArch object model. In other words, the modifications were actually localized to the exclusion of some method calls.

In addition to measuring the effectiveness of the semantic document search, these two additional prototype implementations enabled me to evaluate the proposed semantic document annotation and indexing approaches, by comparing them with the semantic annotation and indexing approaches that relay on the syntactic concept matching and lexically-expanded syntactic concept matching. Two main aspects I was focused on, were the amount and the quality (relevance) of ontological concepts used for the annotation and indexing. Table 7.1 summarizes the evaluation goals.

Table 7.1. Summary of the evaluation goals

First goal: Determining parameters of the concept exploration algorithm
- Validation of the hypotheses H1 and H2 and determining the optimal values of the SD_c, PL_c and β parameters of CEA algorithm for the two test collections used in this evaluation.
Second (main) goal: Measuring effectiveness of the semantic document search
- Measuring the effectiveness of the semantic document search in terms of precision and recall, and its comparison with the effectiveness of: 1) concept-based search approaches based on the simple syntactic matching, 2) concept-based search approaches based on the lexically-expanded syntactic matching, and 3) the conventional full-text search. - Evaluation of the semantic document annotation in terms of the annotation amount and annotation quality (relevance).

7.1.2 Evaluation Procedure

My activities on the evaluation were divided into two phases: 1) preparation of the evaluation, and 2) conducting the evaluation and analyzing the evaluation results. The first phase comprised the preparation of the test collections and the design of the evaluation experiments. The preparation of the test collections included the choice of domain ontologies, acquisition of the document collections, formulation of the query sets, and relevance assessment for the query sets. The second phase, evaluation conducting, comprised the execution of the evaluation experiments and the analysis of the evaluation results by applying appropriate evaluation measures and metrics.

In the rest of this section I provide more details on the preparation phase, considering both the preparation of the test collections and the design of the evaluation experiments. Section 7.1.3 and Section 7.1.4 report the results of the evaluation conducting on the two test collections which were considered in this evaluation study.

Preparation of the Test Collections

The choice of domain ontologies was the initial task in the preparation of the test collections and thus the initial task of the evaluation, too. I decided to form two test collections with two different domain ontologies in order to validate the evaluation results on more than one domain, and thus prove their external validity [117]. This actually meant that I needed to choose two domain ontologies. The ontology choice was not limited to any particular domain. The only constrains that somehow influenced the choice were

the quality of the ontology and the availability of documents of the ontology's domain, which I had to acquire afterwards. The assessment of the ontology quality was not an easy task, since there are no precisely defined criteria for that. What I considered, with respect to the ontology quality were the richness of the ontology (i.e., a number of concepts and properties), the applicability of the ontology to real-world scenarios, and the expertise of the ontology's authors in the ontology engineering and the domain that the ontology conceptualizes.

Over recent years, a considerable number of domain otologies has been created and published on the Web. Moreover, a number of ontology search engines such as Swoogle[1], has been developed and deployed on the Web. After a comprehensive search for ontologies on the Web and a literature review, I identified the Ontology of Mammals of the World (MAMO) [126] as an ontology appropriate for my evaluation. The ontology was not publicly available on the Web, but its authors were kind to share it with me. The second domain ontology, namely the Metals ontology, I obtained from the IntelLEO project[2], in whose initial prototype implementation some of the SDArch services were reused. The ontology was created by the project's industry partner (Key-To-Metals company[3]), which owns one of the world's largest metals database. After choosing the ontologies, the next task was the acquisition of the document collections. For the first ontology (MAMO), I acquired the document collection out of Wikipedia articles related to mammals. The document collection for the second ontology (Metals) I obtained from the same source as the ontology itself. The document collections were of the similar size. The main difference between the collections laid in a fact that one was composed of documents coming from an 'open' source, such as Wikipedia, created by the general Web audience, while the other one was composed of documents created exclusively by domain (Metals) experts. The details of both the ontologies and their corresponding document collections, I provide in Section 7.1.3 and Section 7.1.4, in which I explain the evaluation conducting on the two test collections.

Having chosen the ontology and acquired the document collection, the next step in the preparation of the test collection, was to form the evaluation query sets. As it was explained in Section 5.2.1, the initial form of the user query supported by the SDArch is a free-text query. During the semantic document search, the initial free-text query is first transformed to a semantic query and then executed by the SDArch semantic document search service (Section 5.2.1). While forming the query sets for the evaluation test collections, I had to limit a number of queries to a reasonable number, in order to be able to collect relevance assessments afterwards. Collecting relevance assessments is a time-consuming and expensive process involving humans, desirably domain experts.

[1] http://swoogle.umbc.edu/

[2] http://www.intelleo.eu/

[3] http://www.keytometals.com/

Moreover, by taking into account that in the semantic document search the relevance assessment had to be done at the level of document units (not at the level of whole documents), this process was even more demanding. Accordingly, I limited the query sets to five queries for both test collections. For the document collection of the first test collection, which was composed of Wikipedia articles, I asked four of my colleagues (3 PhD students and 1 Postdoc fellow) to perform the relevance assessment. For the second test collection, I was assisted by three Key-To-Metals engineers who performed the relevance assessment. Performing the relevance assessment was the final step in the preparation of the test collections.

Design of the Evaluation Experiments

I designed the evaluation experiments with regard to the evaluation goals. In total, there were three experiments and all of them were supposed to be executed twice, once with one test collection and once with another one. ***Experiment 1*** and ***Experiment 2*** were dedicated to the first evaluation goal, while ***Experiment 3*** was dedicated to the second evaluation goal (Table 7.1). All experiments were executed on a PC equipped with 4GB RAM and a Quad-Core 2.9GHz processor. What follows in this section are the descriptions of the experiments.

Experiment 1: The objective of this experiment was to validate the hypothesis H2 (p.113), and to determine the optimal value of the parameter β of the CEA algorithm (Section 5.1.1) for the test collections used in this evaluation study. The experiment included the execution of the evaluation query set against the collections of semantic documents that were obtained by transforming the documents of the evaluation document collection into semantic documents with applied the pre-estimated values of the SD_c and PL_c parameters and the parameter in question (β). The idea was to find the combination of these parameters' values that produces the optimal precision and recall. The pre-estimated values of the considered parameters, were determined based on my theoretical understanding of the CEA algorithm. Accordingly, the parameter SD_c took the values $\{0.5, 1, 1.5, 2\}$, and the parameter PL_c took the values $\{1, 2, 3\}$. This resulted in 12 different SD_c - PL_c value-pairs. Each of these value-pairs was then combined with the pre-estemated values of the parameter $\beta \in \{1.5, 2, 2.5, 3, 3.5\}$, which resulted in creation of 60 semantic document collections in total. The semantic document collections were grouped into 12 groups of 5 documents, where all documents of the same group were characterized by the same SD_c - PL_c value-pair. Having the semantic document collections ready, the queries of the test collection's query set were executed against each of the 12 groups of the semantic document collections. For each group, I created interpolated P-R curves for all semantic document collection from the group, and based on them I determined the optimal value of the parameter β for the group. As the optimal β value for a given group was taken the β value that corresponded to the optimal

P-R curve. Having determined the optimal β values for all the 12 groups, as the optimal β value for the test collection (i.e., the applied domain ontology) was taken a mean value of them.

Experiment 2: The objective of this experiment was to validate the hypothesis H1 (p.112), and to determine the optimal values of the SD_c and PL_c parameters of the CEA algorithm (Section 5.1.1) with respect to the optimal precision and recall. Similar to experiment 1, the evaluation query set was executed against the set of semantic document collections that was obtained by combining the pre-estimated values of $SD_c \in \{0.5, 1, 1.5, 2, 2.5\}$ and $PL_c \in \{1, 2, 3\}$ parameters, and the optimal β value previously determined in experiment 1. The only difference between the pre-estimated values of SD_c, PL_c used in this experiment from those used in experiment 1, was an additional value 2.5 in the SD_c value set. The experiment's execution was as follows. First, I determined the optimal values of SD_c regarding each value of PL_c values separately. I actually grouped the semantic document collections into 3 groups of 5 document collections, where all collections of the same group had the same value of the PL_c parameter and a different value of the SD_c parameter. I executed the query set against each of the groups and generated their corresponding P-R curves. The SD_c values that corresponded to the optimal P-R curves of each group were considered as optimal values of SD_c regarding each of the three PL_c values. Then, I put together the optimal P-R curves of all three groups and determined the optimal P-R curve of all semantic document collection considered in the experiment. The values of SD_c and PL_c that corresponded to such determined optimal P-R curve, were taken as the optimal values of these parameters for the test collection (i.e., the applied domain ontology).

Experiment 3: Having determined optimal values of the β, SD_c and PL_c parameters, in experiments 1 and 2, the CEA algorithm was ready for the second evaluation goal, that is, for measuring the effectiveness of the semantic document search in terms of precision and recall and its comparison with the effectiveness of: 1) the concept-based search based on the simple syntactic matching, 2) the concept-based search based on the lexically-expanded syntactic matching, and 3) the conventional full-text search. For measuring the precision and recall of the semantic document search I executed the query set against the semantic document collection obtained by transforming the evaluation document collection with optimal values of SD_c, PL_c and β. For measuring the effectiveness of the full text search, I executed the query set against the SDArch enabled text-index of the same semantic document collection. As described in (Section 4.2.2), the SDArch search service supports both the semantic document search and the full-text search. For measuring the effectiveness of the concept-based search based on the simple syntactic matching and the concept-based search based on the lexically-expanded syntactic matching, I executed the query set against the semantic document collections generated by the two modified versions of the SDArch prototype, which I implemented so that they realize these two types of searches. The comparison of the considered searches was achieved by comparing their corresponding P-R curves.

7.1.3 Conducting the Evaluation with Test Collection 1: Mammals of the World

The initial step in the preparation of the test collection was the choice of the ontology. The ontology (MAMO), was chosen as a result of my survey of the ontology literature and the search of a number of ontology repositories on the Web. Some of the main reasons why I chose the MAMO ontology were: the ontology richness (it contains over 5,000 domain concepts), the credibility and expertise of the ontology's authors (the authors are well known in the ontology engineering community after a number of high-rated scientific articles that they published), the use of the ontology in the real-world scenarios (the ontology is used by the national library of Finland, for categorization and search of the library's arctics related to mammals), and finally, the availability of sources for the acquisition an adequate document collection. Mammals attract a lot of attention of both domain experts and wide audience, so that a large number of articles about mammals has been published on the Web.

The MAMO ontology is created as an OWL ontology, which was the only prerequisite for its use by the SDArch prototype. Besides being an OWL ontology, MAMO also conforms to the SKOS (Simple Knowledge Organization System)[4] specification. In short, SKOS defines a family of ontological properties, such as the `skos:narrower`, `skos:broader` and `skos:related` used for expressing relations between concepts within an ontology. The ontology is hosted by the Finnish Ontology Library Service, called ONKI[5], which is a centralized ontology library providing services for the global access to the hosted ontologies. The ONKI service, however, does not provide the posibility to export the whole ontology as an RDF/XLM file that I needed, so that I had to contact the authors and ask them to provide me the ontology file.

The use of domain ontologies (in this case the MAMO ontology) within the semantic document authoring and semantic document search is localized to the knowledge extraction and conceptualization module of the semantic document authoring service. This module implements the CEA algorithm (Section 5.1.1), which actually utilizes the ontology. The main assumption on which the algorithm runs is the possibility to associate numerical values of the relational semantic distances to the relations (properties) in the ontology graph, thus transforming the ontology graph into the weighted ontology graph. As explained in (Section 5.1.1), I distinguished between two types of the relational semantic distances: $SDist^{r}_{\mathscr{R}\rightarrow\mathscr{D}}(r)$ denoting the semantic distance of concepts belonging to the domain of the relation r from the concepts belonging to the range of r, and $SDist^{r}_{\mathscr{D}\rightarrow\mathscr{R}}(r)$ denoting the semantic distance of the concepts belonging to the range of r from the concepts belonging to the domain of r.

Table 7.2 shows a subset of ontological relations that I considered in this evaluation (including both test collections), along with their SKOS and OWL representations and

[4] `http://www.w3.org/2004/02/skos/`

[5] `http://www.yso.fi/`

the assessed values of their relational semantic distances. The values of the relational semantic distances were assessed based on the results of the experimental study presented in [50]. In that study, the authors measured the semantic similarity/relatedness between WordNet terms connected via the *hypernymy*, *hyponymy*, *holonymy*, *meronymy* and *synonymy* relations, and obtained the following values: $\delta_{hyper} = 0.47$, $\delta_{hypo} = 0.84$, $\delta_{holo} = 0.12$, $\delta_{mero} = 0.16$ and $\delta_{syn} = 0.70$. A value $\delta_r = 0$ means that two terms are semantically unrelated via the relation r, while $\delta_r = 1$ means that the terms are semantically identical with respect to the relation r. By taking into account these values, I calculated the values of the relational semantic distances as $1 - \delta_r$. Moreover, I utilized the fact that *hypernymy* and *hyponymy* as well as *holonymy* and *meronymy* are mutually inverse relations. In addition to the above listed relations, I also considered the `owl:sameAs` ontological relation. It is an ontological relation that links two semantically identical concepts, so that for both types of the relational semantic distances the assessed values were 0.

Semantic relation	Representation	$SDist^r_{\mathscr{R}\to\mathscr{D}}(r)$	$SDist^r_{\mathscr{D}\to\mathscr{R}}(r)$
hypernym	$skos : broader$	$1 - \delta_{hyper} = 0.53$	$1 - \delta_{hypo} = 0.16$
hyponym	$skos : narrower$	$1 - \delta_{hypo} = 0.16$	$1 - \delta_{hyper} = 0.53$
holonym	$skos : relatedPartOf$	$1 - \delta_{holo} = 0.88$	$1 - \delta_{mero} = 0.84$
meronym	$skos : relatedHasPart$	$1 - \delta_{mero} = 0.84$	$1 - \delta_{holo} = 0.88$
synonym	$owl : equivalentClass$	$1 - \delta_{syn} = 0.30$	$1 - \delta_{syn} = 0.30$
identical	$owl : sameAs$	0	0

Table 7.2. The ontological relations considered in the evaluation along with their SKOS and OWL representations and the assessed values of relational semantic distances

The document collection that I used in the evaluation together with the MAMO ontology was composed of Wikipedia articles from the series *List of mammals of World*[6]. I selected 150 articles from this series and copied their content into the same number of Word documents. This set of Word documents represented the initial form of the evaluation document collection, which was then transformed by the SDArch semantic document authoring service into a number of semantic document collections (obtained by appalling different transformation options and values of the CEA parameters). Regardless of the transformation option and the parameters' values, each of the generated semantic document collections contained a total of 2130 semantic document units of interest for the evaluation. As a query set of the test collection, I created five queries, each of which being germane to the topic of mammals. The size of both, the document collection and the query set, had to be reasonable since the relevance assessment process was supposed to be done completely manually by human assessors. By taking into

[6] `http://en.wikipedia.org/wiki/Mammal`

account that the document collection came from the source such as Wikipedia, which is suited for a wide audience, and that the topic is widely known, I decided to ask four of my colleagues (three PhD students and one Postdoc) to perform the relevance assessment. In the rest of the section, I present the results of the three evaluation experiments performed on the mammals test collection.

Results of Experiment 1:

Table 7.3 contains the measured, optimal values of the parameter β for all SD_c-PL_c value pairs, which are formed by combining the pre-estimated sets of the $SD_c \in \{0.5, 1, 1.5, 2\}$ and $PL_c \in \{1, 2, 3\}$ values. The way I measured β values, I illustrate on the example of the $SD_c = 1.5$, $PL_c = 2$ value pair. Figure 7.1 shows the P-R curves of the queries' execution against five semantic document collections that were obtained by transforming the initial collection of Word documents, and applying the selected SD_c-PL_c value pair along with five different $beta \in \{1.5, 2, 2.5, 3, 3.5\}$ values. The P-R curves are formed measuring the interpolated precisions at standard recall points. As can be seen from the figure, the optimal β value, with respect to optimal precision and recall, is 3.

SD_c	0.5	1	1.5	2	0.5	1	1.5	2	0.5	1	1.5	2
PL_c	1	2	3	1	2	3	1	2	3	1	2	3
β	2	2.5	2	3	2.5	3.5	2.5	3	3.5	3	3	3.5

Table 7.3. Optimal values of the β parameter for the pre-estimeted SD_c-PL_c value pairs

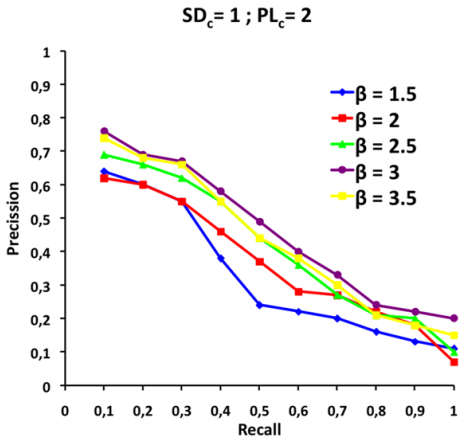
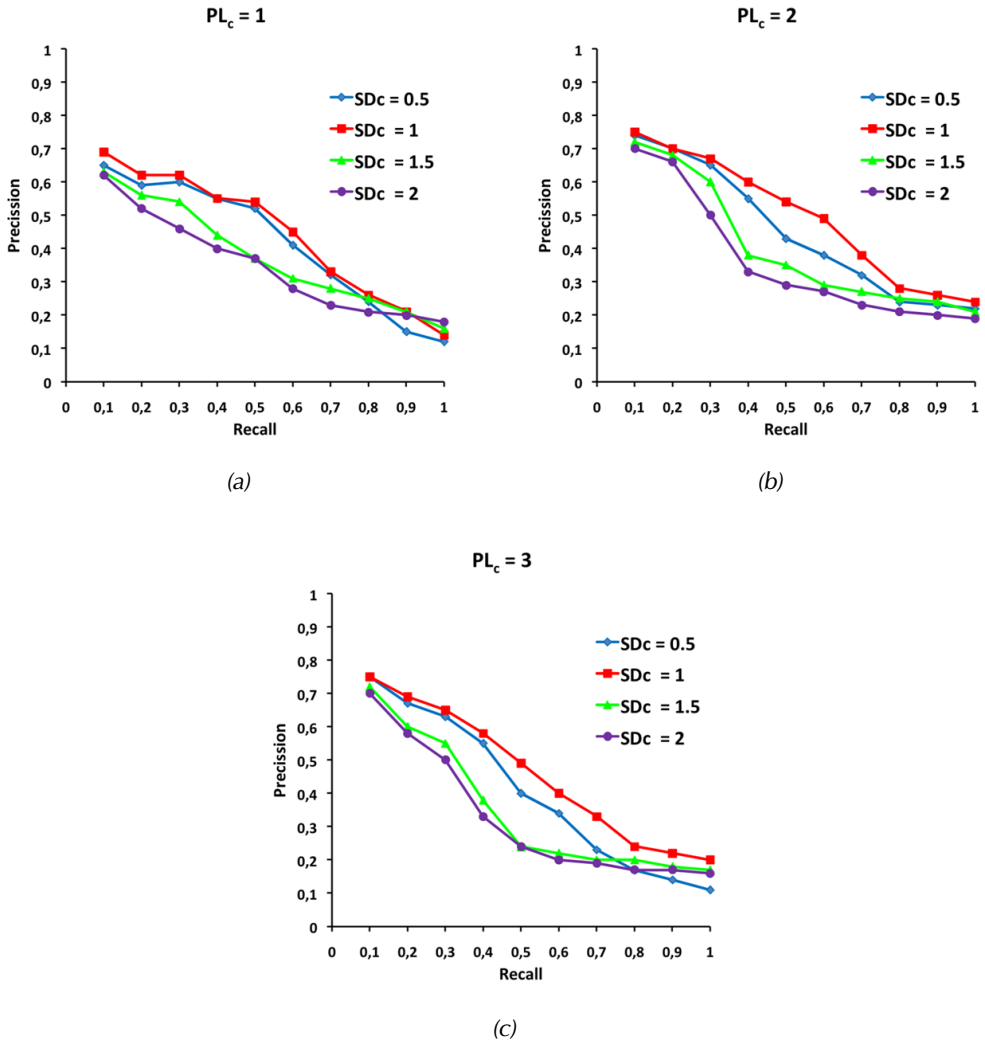


Figure 7.1. Determining optimal value of the β parameter for to the given SD_c-PL_c value pair

Having measured optimal values of the parameter β for all the SD_c-PL_c value pairs, the overall, optimal value of $\beta = 2.83$ for test collection 1 (regardless of SD_c and PL_c values) is calculated as a mean value of the optimal β values for all SD_c-PL_c value pairs (Table 7.3). That value was afterwards used as a value of β in the experiments 2 and 3.



(a) *(b)*

(c)

Figure 7.2. P-R curves of the query set execution against the three groups of semantic document collections : *(a)* $PL_c = 1$; *(b)* $PL_c = 2$; *(c)* $PL_c = 3$;

Results of Experiment 2:

Figures 7.2 (a), (b), and (c) show the P-R curves of the query set execution against the three groups of semantic document collections used in this experiment. The seman-

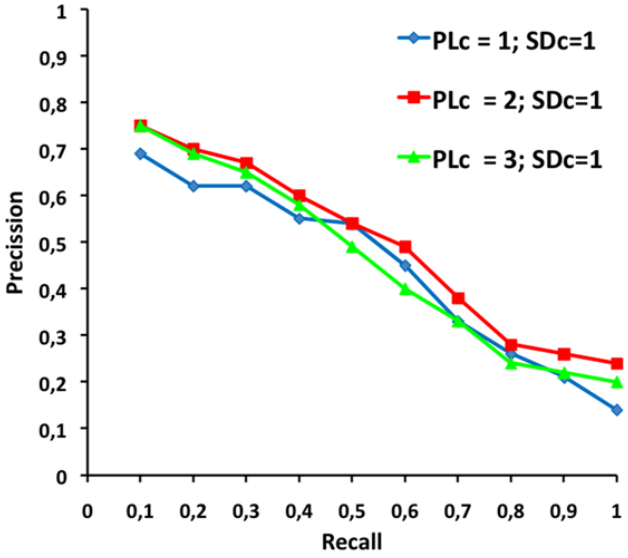


Figure 7.3. Determining optimal values of the PL_c and SD_c parameters

tic document collections belonging to the same group are characterized by the same value of $PL_c \in \{1, 2, 3\}$ parameter and the different value of $SD_c \in \{0.5, 1, 1.5, 2, 2.5\}$ parameter. Figure 7.3 puts together the optimal P-R curves of all the three groups. By comparing them, it can be seen that the semantic document collection with $SD_c = 1$ and $PL_c = 2$ showed the optimal performance with respect to the measured precision and recall. Accordingly, I took these values of the SD_c and PL_c parameters as the optimal values of these parameters for test collection 1.

Results of Experiment 3:

In this experiment, I measured the effectiveness of the semantic document search in terms of precision and recall, and compared it with the effectiveness of: 1) the concept-based search based on the simple syntactic matching, 2) the concept-based search based on the lexically-expanded syntactic matching, and 3) the conventional full-text search. Moreover, this experiment was used to evaluate a quality of the semantic document annotation in terms of the annotation amount and annotation relevance.

For examining the two concept-based searches, the initial collection of the Word documents was transformed two times, by utilizing the modified versions of the SDArch prototype, which corresponded to those searches. For examining the semantic document search, the initial collection of the Word documents was transformed by utilizing the actual SDArch prototype and applying previously determined, optimal values of $\beta = 2.83$,

$SD_c = 1$ and $PL_c = 2$ parameters. As a result of these transformations, we obtained three semantic document collections. Let us mark the transformations as:

- T_1 - for the concept-based search based on the simple syntactic matching;
- T_2 - for the concept-based search based on the lexically-expanded syntactic matching;
- T_3 - for the semantic document search (the lexically-expanded syntactic matching plus the semantic matching (Section 5.2.1)).

For each of transformations T_1 - T_3, Table 7.4 shows: 1) a distinct number of concepts of the annotation ontology (MAMO) that have been discovered in all document units of the document collection, 2) total numbers of syntactic and semantic matches, that is, the numbers of document units in which the concepts has been discovered by the syntactic and the semantic matching respectively, and 3) the average weights of the syntactic and semantic matches, calculated based on 20 randomly chosen document units.

Transformation	# of concepts	# of syn. matches	# of sem. matches	Avg. weight of syn. match.	Avg. weight of sem. match.
T_1	211	1524	-	2.56	-
T_2	343	3182	-	3.62	-
T_3	672	3182	2437	3.62	2.96

Table 7.4. Transformation results of the transformations T_1 - T_3 that correspond to the semantic document collections examined in experiment 3

By comparing results of the transformations T_1 and T_2, which implement simple and lexically-expanded syntactic matching respectively, we can see that the lexical expansion (implemented in T_2) increased the number of discovered concepts from 211 to 343, while the total number of syntactic matches was increased from 1524 to 3182. Moreover, the lexical expansion also increased the average relevance weight of syntactic matches from 2.56 to 3.62. In other words, the lexical expansion of concept labels showed potential to improve both the quantity and quality of the annotation. The transformation T_3 produced the same number of syntactic matches as T_2 (i.e., 3182), since the syntactic matching stayed intact, but it increased the total number of matches by adding 2437 semantic matches. The average weight of the added semantic matches (2.96) is less than the average weight of the lexically expanded syntactic matches (3.62), but greater than the average weight of the simple syntactic matches (2.56). These results suggest that the semantic document annotation approach that I proposed has a potential to increase the amount of semantic annotations, yet preserving the high quality of the annotations.

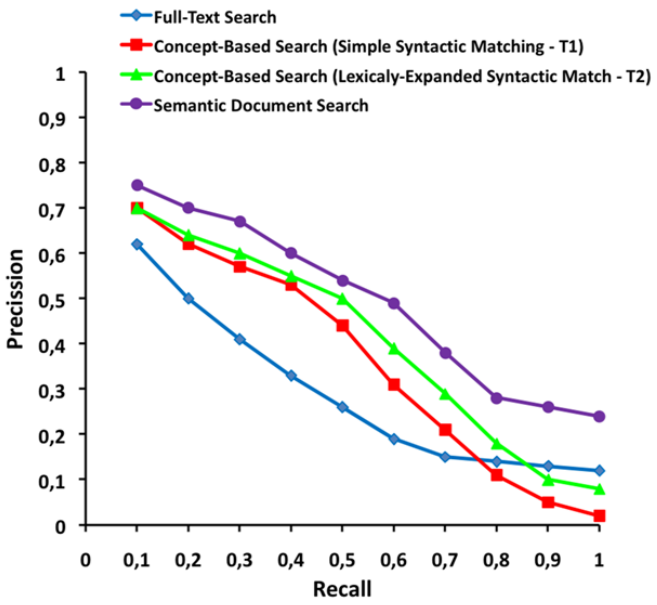


Figure 7.4. Interpolated precision at standard recall points for compared search approaches

To compare the effectiveness of the semantic document search with the concept-based searches, the query set of the test collection was executed against each of the three semantic document collections (i.e., the collections obtained as a result of T_1, T_2 and T_3 transformations). Moreover, I also considered the full-text search, which was performed by executing the query set against the text index obtained in the transformation T_3. Basically, I could use either of the three text indexes (indexes formed in T_1, T_2, or T_3), since the text indexing was the same in all three transformations. Figure 7.4 shows interpolated precision at standard recall points for the considered search approaches. As can be seen from the figure, the semantic document search was the most effective among the considered searches, with respect to both recall and precision. Comparing to the full-text search, the performance improvement of the semantic document search is more significant for the lower recall values (high precision - a left half of the P-R graph) than for the higher recall values (high recall - a right half of the P-R graph). Comparing to the concept-based searches T_1 and T_2, the situation is opposite, that is, the performance improvement of the semantic document search is less significant for the lower recall values than for the higher ones. Comparing the two of the concept-based searches between each other, we can see that the lexically expanded syntactic matching (T_2) outperforms from the simple syntactic matching (T_1). Moreover, they both outperform from the full-text search for lower recall values, while for the higher recall values, the situation is opposite.

7.1.4 Conducting the Evaluation with Test Collection 2: Metals and their Alloys

In contrast to the first test collection for which I obtained the ontology and the document collection from two different sources, for the second test collection I obtained both of them from the same source, the Key-To-Metals[7] company. Key-To-Metals is a company that maintains one of the world's most comprehensive metals database. It contains over 4 millions property records for steel, aluminum, copper, titanium, and other metals, originating from more than 40 countries and standards. The company's development team has been working on the development of a new generation, search engine that should be deployed for searching their metals database. The new search engine is suppose to use a number of semantic web technologies. In that regard, the company created the Metals ontology that contains over $1,800$ concepts about metals and their alloys. The ontology also conceptualizes a considerable number of metallurgical processes such as hardening, annealing and tempering as well as a considerable number of metals' applications in different industries.

The Key-To-Metals company is involved in the IntelLEO project, as an industry partner. This project is funded by the European Commission under the 7th Framework Programme. The goal of the IntelLEO project is to explore supportive technologies for learning and knowledge building activities of learners in Intelligent Learning Extended Organizations (IntelLEO). The project includes the design of a novel system and the implementation of its prototype. One of the services, which the new system should provide, is a Content and Knowledge Provision service. For the initial implementation of this service, the project's partners who are responsible for the service implementation, used the SDArch semantic document authoring (Section 4.2.1) and the semantic document search (Section 4.2.2) services that I developed. That was actually the way how Key-To-Metals people got to know my work, and then offered me their resources for my evaluation study. Besides the Metals ontology, they provided me a collection of 240 Word documents containing records from their metals database. After the transformation into semantic documents, a total number of 3312 semantic document units of the interest for the evaluation were identified. Moreover, Key-To-Metals engineers assisted me in the formulation of the query set as well as they performed the relevance assessment for the queries.

The same as the MAMO ontology, the Metals ontology is an OWL ontology which adheres to the SKOS specification. Accordingly, in the evaluation experiments I considered the same set of ontological relations (Table 7.2), as the one that was considered with test collection 1. The query set contained 5 queries about metals, metal alloys and their applications. In the rest of the section I present the results of the evaluation experiments 1, 2, and 3 conducted with this test collection. Since the methodology of the experiments' execution was completely the same as for test collection 1, I do not repeat

[7] http://www.keytometals.com/

it again. I am focused only on the experimental results and their explanation.

SD_c	0.5	1	1.5	2	0.5	1	1.5	2	0.5	1	1.5	2
PL_c	1	2	3	1	2	3	1	2	3	1	2	3
β	2.5	2	2	2.5	2.5	3	2.5	2.5	3.5	3	3	3

Table 7.5. Optimal values of the β parameter for the pre-estimeted SD_c-PL_c value pairs

Results of Experiment 1:

Table 7.5 contains the measured optimal values of the parameter β for each of the pre-estimeted SD_c-PL_c value pairs considered in the experiment. The optimal value of parameter β for the test collection, determined as a mean value of the optimal β values for all SD_c-PL_c value pairs, is $\beta = 2.66$.

Results of Experiment 2:

Figure 7.5 shows the P-R curves of the three groups of semantic document collections, each of the groups being characterized by the same value of $PL_c \in \{1, 2, 3\}$ and different values of $SD_c \in \{0.5, 1, 1.5, 2, 2.5\}$. Figure 7.6 collects the optimal P-R curves of all the three groups. As can be seen from the figure , the P-R curve of the semantic document collection with $SD_c = 1.5$ and $PL_c = 2$ is optimal among the shown P-R curves. Accordingly, I took these values of SD_c and PL_c as the optimal values of these parameters for test collection 2.

Results of Experiment 3:

Table 7.6 shows the results of the transformation of the initial Word document collection (i.e., document collection that I obtained from Key-To-Metals company) to semantic document collections that correspond to the compared search approaches: the concept-based search based on the simple syntactic matching (T_1), the concept-based search based on the lexically-expanded syntactic matching (T_2), and the semantic document search (T_3). For the full text search, the text index generated during the T_3 transformation was used. The transformation results show that the lexical expansion of concept descriptions (labels) increased the amount of annotation concepts (i.e., syntactic matches) from 2153 to 2879 and the average relevance weight of the annotation concepts from 1.73 to 2.43. Moreover, the semantic matching further increased the amount of annotation concepts with 1024 concepts (i.e., semantic matches), whose average relevance weight was 2.14.

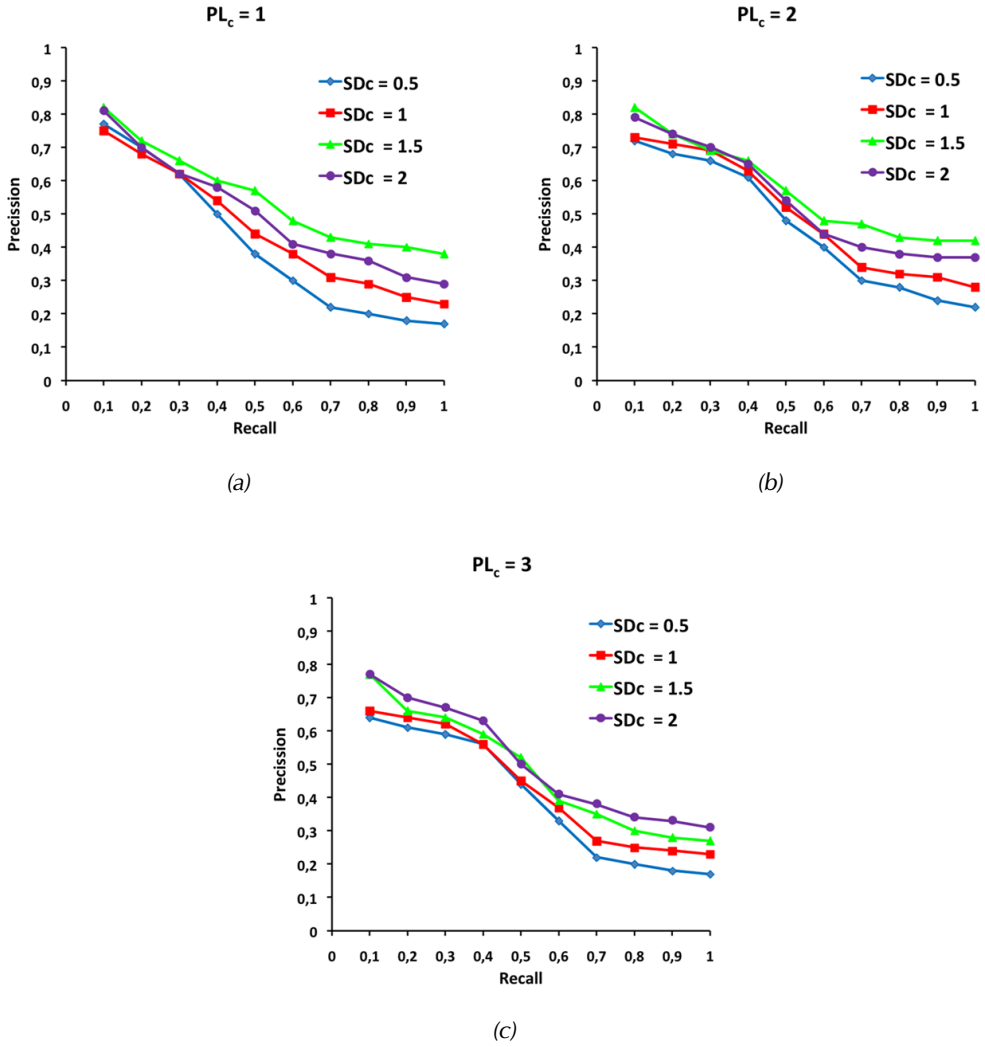


Figure 7.5. P-R curves of the query set execution against the three groups of semantic document collections : *(a)* $PL_c = 1$; *(b)* $PL_c = 2$; *(c)* $PL_c = 3$;

Figure 7.7 shows the P-R curves of the compared search approaches, formed based on the execution of the test collection's query set against the semantic document collections (i.e., concept indexes) obtained through transformations T_1 - T_3, and against the text index obtained through transformation T_3. As can be seen from the figure, the precision of the semantic document search outperforms from the precision of the other three approaches for all recall values. However, the performance improvement is more significant for the higher recall values. The same as it was with the first test collection, the concept search based on the lexically-expanded syntactic matching outperforms from

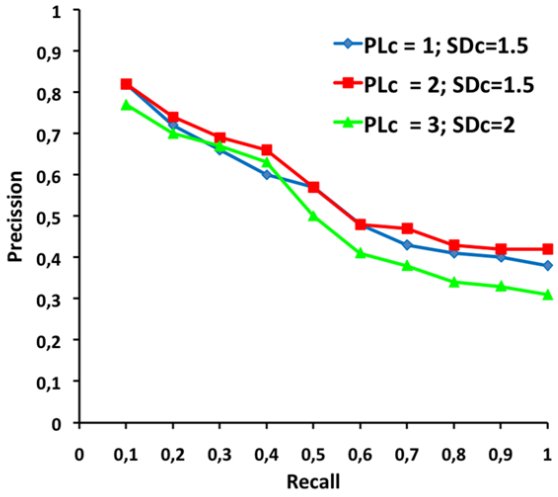


Figure 7.6. Determining optimal values of the PL_c and SD_c parameters

Transformation	# of concepts	# of syn. matches	# of sem. matches	Avg. weight of syn. match.	Avg. weight of sem. match.
T_1	117	2153	-	1.73	-
T_2	221	2879	-	2.43	-
T_3	456	2879	1024	2.43	2.14

Table 7.6. Transformation results of the transformations T_1 - T_3 that correspond to the semantic document collections examined in experiment 3

the concept search based on the simple syntactic matching for all recall values. The performance of the full-text search was the worst of the all searches, especially in case of the high recall values.

7.1.5 Discussion of the Evaluation Results

*The results of experiments 1 and 2 validated hypotheses **H1** and **h2** that there exist optimal values of of the parameters β, SD_c, and PL_c, with respect to the optimal semantic document search (i.e., precision and recall).*

The different optimal values of these parameters for the two test collections ($\beta = 3$, $SD_c = 1$, and $PL_c = 2$ for test collection 1 and $\beta = 2.5$, $SD_c = 1.5$, and $PL_c = 2$ for test collection 2), indicated that each semantic document collection has specific optimal values of these parameters and that they can be experimentally determined.

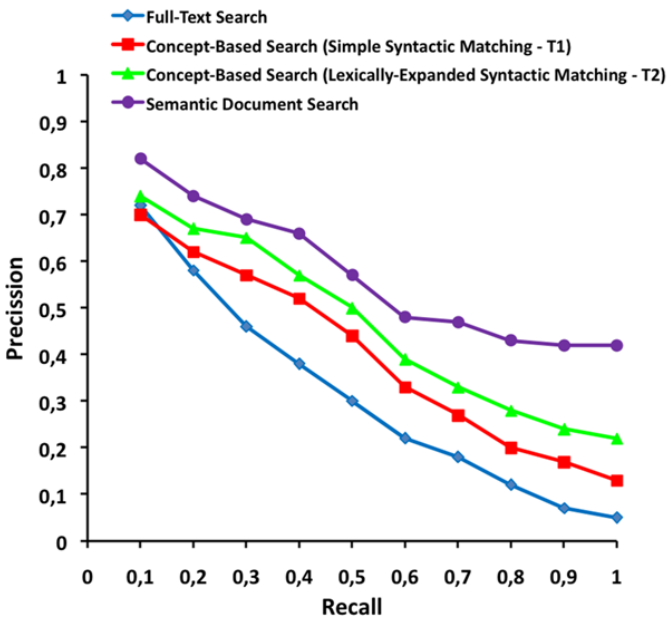


Figure 7.7. Interpolated precision at standard recall points for compared search approaches

Regarding the semantic document annotation, by analyzing the semantic document collections generated in experiment 3, I showed that the proposed semantic document annotation approach increases the amount of the semantic annotations (i.e., number of ontological concepts used to annotate document units), comparing to existing, semantic annotation approaches which are based on the syntactic matching (i.e., the simple syntactic matching and the lexically expanded syntactic matching). It is also important to indicate that the quality of the added semantic annotations by the semantic matching (i.e., the annotation relevance), is only slightly less than the quality of the corresponding syntactic matches. In case of test collection 1 (i.e., Mammals of the World, the amount of the semantic annotations was increased by 2437 semantic matches (Table 7.4) or 76% compared to the semantic annotation which realizes the lexically-expanded syntactic matches. Considering the annotation relevance, the average relevance weight of the added semantic matches was 18% less than the relevance weight of their corresponding syntactic matches. In case of test collection 2 (i.e., Metals and their Alloys), the amount of the semantic annotations was increased by 1024 semantic matches (Table 7.5) or 35%, while the relevance weight of added semantic matches was 12% less than the relevance weight of their corresponding syntactic matches. Comparing the two test collections, we can see that the proposed semantic annotation approach resulted in a higher increase of the annotation amount in the case of test collection 1. At the

same time, a decrease of the average relevance weight of the added annotations was less in the case of the second test collection. The explanation for such results I tried to find by analyzing and comparing the ontologies of the test collections. Considering the number of concepts, the MAMO ontology (around 5,000 concepts) is significantly reacher than the Metals ontology (around 1,800 concepts). Considering the level of the interconnection between concepts in the ontologies, the situation is opposite. Concepts of the Metals ontology are better interconnected (more ontological relations) than the concepts of the MAMO ontology. I drew this conclusion by calculating the average numbers of ontological relations per concept. For the MAMO ontology it was 1.8 relations, while for the Metals ontology it was 2.6 relations. In calculating these values, I took into account only the ontological relations that were treated in this evaluation (Table 7.2). A general conclusion, which I drew based on the annotation results achieved in both test collections and by analyzing the used domain ontologies is the following:

➥ ***Semantic Document Annotation - Evaluation Outcome***
The amount of semantic annotations (i.e., a number of semantic matches) depends on the number of concepts in the used ontology, while the relevance of the semantic annotations depends on the concepts' interconnection within the ontology. In other words, the more and better interconnected concepts, the better annotation results.

Regarding the effectiveness of the proposed semantic document search, the results of experiment 3 obtained for both test collections proved the following:

➥ ***Semantic Document Search - Evaluation Outcome***
The proposed semantic document search outperforms from the considered search approaches (i.e., the concept-based search based on the simple syntactic matching, the concept-based search based on lexically-expanded syntactic matching, and the full-text search) in terms of both better precision and better recall.

In spite of the fact that these results were achieved in a controlled system, in which all documents of the test collections were documents about the same given topic and the applied domain ontology perfectly matched that topic, they indicate that the semantic document search has potential to improve the document search and retrieval. The availability of well-defined domain ontologies and the application of the same, shared domain ontologies within one domain, are the only preconditions for a wider application of the proposed semantic document annotation and search approaches.

7.2 Usability Evaluation of the SDArch Services and Tools

The usability evaluation is a very important component of the user-centered system design. It is a measure of the effort needed to learn and use software tool/system for a specific purpose, and refers to the effectiveness, efficiency and satisfaction with which users can perform tasks with a system [48].

The usability evaluation of SDArch was tied closely to the SDArch prototype development and was conducted in two iterations: a formative evaluation and a summative evaluation. The formative evaluation [105] was conducted by an alpha release of the SDArch prototype, whose development was finished at the beginning of 2009. The summative evaluation was conducted by a beta release[8][9] of the SDArch prototype, whose development was finished one year after the alpha release, that is, at the beginning of 2010. The goal of the formative evaluation was twofold. Firstly, it was supposed to discover possible failures in the prototype implementation and to identify missing functionalities in the user interface of the SDArch tools. Secondly, and more importantly, it was intended to reveal deficiencies in the evaluation design and to assess the feasibility of the evaluation procedure. Moreover, a part of the evaluation material used in the formative evaluation, was reused later in the summative evaluation. However, since the evaluation design of the summative evaluation differed significantly from the design of the formative evaluation, the results of the formative evaluation were not incorporated into the results of the summative evaluation. The results of the formative evaluation were reported in [96], for which the authors received the best paper award. In this thesis, regarding the formative evaluation, I provide only a brief summary in Appendix B.1. My main focus in this thesis is on the summative evaluation and its results.

The rest of the section is organized as follows. I first discuss the goals of the SDArch usability evaluation (Section 7.2.4). Then, I elaborate the evaluation design, including a choice of the evaluation methods and metrics (Section 7.2.2), a motivational scenario of a conducted case study (Section 7.2.5), the case study participants (Section 7.2.4), a document collection used in the evaluation session that concluded the case study (Section 7.2.3), and a usability test conducted during the evaluation session (Section 7.2.6). Then, I explain how the evaluation session of the summative evaluation was conducted (Section 7.2.7) and discuss the evaluation results (Section 7.2.8).

7.2.1 Goals of the Usability Evaluation

The goal of the SDArch usability evaluation was to show to which extent SDArch improves the document management on local desktops as well as the management of shared document data within online social networks. I formulated the hypothesis of the SDArch usability evaluation as follows.

[8] http://sourceforge.net/projects/sdarch/
[9] http://sourceforge.net/projects/semdoc/

➥ **Usability Evaluation - Hypothesis**
*Using SDArch results in a more **effective**, **efficient**, and **satisfactory** user experience when authoring, exploring (i.e., searching and navigating) and utilizing documents in carrying out tasks on local desktops as well as within online social networks.*

1. With respect to user **effectiveness**, I intended to measure the accuracy and completeness with which SDArch users complete document management tasks such as document authoring, document search and document sharing. In other words, how many and what tasks the users can complete successfully using the SDArch services and tools.

2. With respect to user **efficiency**, I intended to measure the resources expended in relation to the accuracy and completeness with which SDArch users complete the document management tasks. In other words, how much effort the users spend for completing these tasks using the SDArch services and tools.

3. With respect to user **satisfaction**, I intended to measure the freedom from discomfort, and positive attitudes towards the use of the SDArch services and tools.

In addition, by the usability evaluation I also wanted to obtained the user feedback on the underlying SDArch models: the semantic document model (Section 3.2), the SDArch user model (Section 4.2.3), and the SDArch social network model (Section 4.2.4). Since most of the features of these models are hidden from end-users, I planned to familiarize the evaluation participants with them by conducting a training session, in which I presented the SDArch models and provided some samples of instances of the models.

7.2.2 Choosing the Right Evaluation Methods

There exist multiple methods of evaluating usability depending on available resources and evaluators' experience, ability, and performance. In general, usability evaluation methods can be classified into three general categories: *inspection*, *testing*, and *inquiry* [85]. In the usability inspection methods, usability specialists examine usability-related aspects of a system. Commonly used inspection methods are: Cognitive Walkthroughs, Feature Inspections, and Heuristic Evaluation [105, 51, 85]. The usability testing and inquiry methods are usually applied as a part of case studies, where representative users work on typical tasks using the system. Usability testing is usually used to obtain quantitative data about users' performance when they perform the tasks of a usability test, for example, the time that users take to complete the specific task, the number of tasks of various kinds that can be completed within the given time limit, and the number of user errors. In usability inquiry methods, evaluators obtain information about users' likes and dislikes, needs, and understanding of the system by talking to them, observing them using the monitoring systems, or letting them answer questions verbally or in a written

form. Commonly used inquiry methods are: *focus groups*, *interviews* and *questionnaires*. Regardless of the type of evaluation method (i.e., *inspection*, *testing* or *inquiry*), a usability evaluation should always be guided by an evaluation plan that includes information about data, participants, tasks and evaluation metrics [84].

With respect to the duration of a usability evaluation, evaluation approaches can be classified into two categories: a short and a long-term evaluation. The short-term evaluation approach is to setup an experiment in a lab environment and invite test users to use the evaluation system in a supervised way. This kind of evaluation is typically done in a few hours or one day, because of the lab resources, which can not be occupied for a long time, and the participants who cannot devote more time for the evaluation. The long-term evaluation approach is about conducting usability evaluation not only in a laboratory setting, but also at the users' workplace. This kind of evaluation is feasible only in case of minimal efforts and costs of the evaluation setup.

The usability evaluation approach that I undertook was greatly determined by the twofold objective of SDArch, that is, improving document data management on a local desktop environment and improving document data sharing in an online, social network environment. In order to address both aspects of the SDArch objective, besides the use the SDArch services and tools for carrying out tasks on local desktops, I had to create an online social network whose members would use the SDArch services and tools to manage their shared documents. Taking into account these issues, I decided that a case study would be the most suitable approach for the SDArch usability evaluation. The case study was supposed to include creating the SDArch social network, collecting a shared collection of semantic documents, and performing a usability evaluation session with a usability test and follow-up questionnaires. Moreover, the case study involved the training session, in which I presented the underlying SDArch models and demonstrated the SDArch services and tools to the participants of the case study. The duration of the case study was not pre-defined. It was left to be determined by the speed of the social network growth and the acquisition of shared semantic documents. Actually, when the number of the network members and the number of shared semantic documents reached desired values, I organized the evaluation session and after that concluded the case study. In the rest of the section, I describe the usability evaluation methods that I used for each of the considered usability components (i.e., user *effectiveness*, *efficiency* and *satisfaction*).

The evaluation session included the task-based usability test [135], complemented with a set of objective, quantitative measures and several follow-up questionnaires. The tasks of the usability test were designed so that they covered most of the SDArch enabled processes, including the semantic document authoring (Section 5.1), the semantic document search (Section 5.2.1), and the semantic document navigation (Section 5.2.3) as well as the SDArch user profile management (Section 4.2.3) and the SDArch social network management (Section 4.2.4). The tasks were grouped into three use cases:

- setting up the user profile and the social network;
- authoring and publishing semantic documents;
- searching and navigating across semantic documents;

I describe the use cases in Section 7.2.6. The use cases were executed one by one, each of them being accompanied with a proper follow-up questionnaire (Questionnaires A, B, and C - Section 7.2.8). The objective of those questionnaires was to acquire the user subjective opinion on the SDArch enabled processes and features of the underlying SDArch models. Regarding the SDArch enabled processes, the participants were supposed to form their opinion based on the tasks of the usability test. Regarding the SDArch models, the participants' opinion was supposed to be formed based on the models' features that they could observe while completing the tasks, and based on the information about the models that I provided in the training session organized before the evaluation session. All of the three follow-up questionnaires contained both a set of open-ended questions and a set of statements that were designed as positive statements to be rated by using a five-level Likert scale [108], starting from 1 (strongly disagree) to 5 (strongly agree).

The first two use cases were supposed to be performed on a computer equipped with the SDArch services and tools, which I call a 'SDArch system' hereafter. The third use case had to be performed twice, once by using the SDArch system and once by using a 'conventional system' (i.e., a computer equipped with a conventional MS Office). Moreover, for the third use case, the participants were divided into two control groups; the first control group performed the use case by using the conventional system first, and then by using the SDArch system, and the second control group performed the use case by using the systems in the opposite order. Finally, screen activities of the participants while performing the tasks of the third use case were recorded by a screen recording software.

By applying appropriate quantitative measures on information extracted from the screen recordings, I intended to gather some indications about the user *effectiveness* and *efficiency*. With respect to the user *effectiveness*, I measured how effective participants were in completing the tasks of the use case. Basically, I tracked how many and which tasks the participants could complete successfully by using the two compared systems. In other words, I measured the task success rates for both systems. With respect to the user *efficiency*, I measured the task completion times, the numbers of mouse clicks, and the numbers of window switches for all tasks for both systems.

At the end of the evaluation session, once the usability test was finished, the participants were asked to fill in a user satisfaction questionnaire (Section 7.2.8). I created this questionnaire, to evaluate the third usability component, that is, the user *satisfaction*. The questionnaire contained statements corresponding to the following four dimensions of the user satisfaction: the system *usefulness*, *ease-of-use*, *ease-of-learning* and *overal satisfaction*. All the statements were created as positive statements and the five-level Likert

scale (LS) was applied for their rating. Table 7.7 summarizes the chosen evaluation methods and metrics for each of the three usability components.

Evaluation Criteria	Evaluation Method	Evaluation Metric
Effectiveness	Objective - Quantitative Measure	Task Success Rates
Efficiency	Objective - Quantitative Measure “ “	Task Completion Times Number of Mouse Clicks Number of Window Switches
Satisfaction	Subjective - Questionnaires	five-level Likert scale

Table 7.7. Considered usability components with the assigned evaluation methods and metrics

7.2.3 A Motivational Scenario of the Case Study

Authoring of course material (learning content) completely from scratch has always been a difficult and time-consuming task. Current research has shown that most course instructors reuse and modify existing content, available in their own archives or on the Web [22], rather than authoring a course material from scratch. The reuse process requires a meaningful way to search and retrieve the appropriate content. In practice, authors usually reuse pieces of document content, which are related to certain concepts and play certain pedagogical roles (e.g., illustration, definition and example) [70]. Common selective reuse of document content is a cumbersome task, requiring copy-and-paste, which is a laborious and error prone process.

Extensive research has been carried out lately to enhance the reusability of learning content by leveraging semantic technologies for standardization and semantic annotation of learning content components [38, 70]. While these efforts have demonstrated some significant potential to improve the current state of the authoring of learning content, there are still some important issues to be addressed. Firstly, ontology-based semantic annotation and retrieval approaches [110, 44, 68, 15, 128] represent a step ahead in annotation, search and discovery of learning content, comparing to the approaches based on the standardized metadata annotation [35]. However, the full potential of semantic technologies will be achieved when learning content can be efficiently searched by utilizing not only semantic annotations but also semantic relationships between learning content components. Thus, not only semantic annotation, but also an appropriate semantic framework that enables linking of semantically related learning content components is necessary. Secondly, in most cases existing learning contents are isolated in huge, centralized repositories with a restrictive access, which is the opposite of trends of the emerging Web 2.0 [9] and a distributed data storage. Thirdly, despite

the fact that some authoring tools support some collaborative activities, most of them are designed primarily for individual users and pay little attention to collaborative and social activities of content authors.

Let us now suppose the following scenario. Mark is a university professor who teaches a 'Software Architecture' course. For each topic in the course, Mark prepares a PowerPoint presentation which he uses during the lecture. The next topic to be presented in the course is 'Software Design Patterns'. Mark has his own presentation on this topic from the last year, but he wants to create a new, better presentation with up to date information. In order to prepare as good a presentation as possible, he plans to consider the existing presentation, then presentations on the same topic used by his colleagues from other universities, and all other relevant articles/documents from his own archive as well as archives of his colleagues. As usual, Mark is going to use MS PowerPoint to prepare the presentation, since he is the most familiar with it. However, this time his computer is equipped with the SDArch services (Section 4.2) and tools (Section 6.4). By using the document transformer and publisher tool (Section 6.4.4), Mark has already transformed the existing presentation and some other documents from his archive relevant for the topic into semantic documents. Moreover, by using the social network manager tool (Section 6.4.2), Mark has joined the SDArch social network, which among others also include his colleagues whose course materials he plans to consider for the new presentation. In addition, the members of the SDArch social network have initiated a user group interested in software design patterns, and started to share semantic documents on this topic. Finally, Mark created the new presentation by reusing fine-grained document units, which he discovered by the semantic document search and navigation across his own collection of semantic documents as well as the shared semantic documents. To search and navigate across the semantic documents from within MS PowerPoint he employed the document recommender (Section 6.4.5) and the semantic document browser (Section 6.4.6) tools.

The above-described scenario, represents the motivational scenario of the case study that I organized in order to perform the SDArch usability evaluation. The tasks of the usability test (Section 7.2.6), which was conducted in the evaluation session at the end of the case study, corresponded to the tasks that Mark experienced in the described motivational scenario. Successful completion of those tasks, by the participant of the case study, resulted in a sample PowerPoint presentation of the given topic.

7.2.4 Participants

A crucial element of the usability evaluation process is the selection and recruitment of participants, whose background and abilities are representative of intended users of the system to be evaluated [85]. The evaluation results will only be valid if the participants are typical users of the system, or as close to that criterion as possible. If one conducts an evaluation with the 'wrong' people, regardless of how much effort one puts into the rest of the evaluation preparation, the results will be questionable and of limited

external validity [51]. Another issue regarding the selection of the participants, which has attracted a lot of attention in the HCI community, is what should be a sufficient number of participants of the usability study. In terms of quality, Nielsen [11] argues that five expert users are sufficient to discover 85% of the usability problems in a system under evaluation.

In the usability evaluation of SDArch, a total of 18 participants, from four different universities, took part. There were 7 participants from the Simon Fraser University[10], Canada, 2 participants from the Athabasca University[11], Canada, 2 participants from the University of Belgrade[12], Serbia, and 7 participants from the University of Lugano[13], Switzerland. The roles of the participants in their institutions were as follows: 3 of them were professors, 4 were post-docs and 11 were PhD students. I recruited the participants based on social and scientific connections, which I made in the scientific community during my PhD studies. All participants participated voluntarily, and showed genuine motivation in using the SDArch system. Moreover, all of them were involved in teaching at their universities, either as lecturers or teaching assistants, so that they were familiar with the authoring of course material, which represented the motivational scenario of the conducted case study.

The recruitment process was as follows. First, I initiated the SDArch social network and sent an invitation (i.e., an e-mail with instructions how to join the network) to potential participants. In total, I sent 24 invitations. As explained in Section 6.4.1, potential participants could join the SDArch social network by using the the SDArch tools. However, in that case they should have already had installed the SDArch tools on their computers. In order to avoid asking them to install the tools at that initial phase, I created a simple, Web-based application[14] for joining the SDArch social network. As shown in Section 7.2.3 and Section 7.2.7, the participants did not necessarily need to install SDArch tools on their computers in order to participate in the evaluation. After receiving the invitation request, 18 contacted persons responded positively and joined the SDArch network within the following two weeks. All of the SDArch network members also participated in the evaluation session conducted at the end of the case study. At the beginning of the evaluation session, before starting the usability test, the participants were asked to fill up the Entrance Questionnaire (Appendix B.2), containing demographic questions and questions about a participant's familiarity with technologies and tools of interest for the SDArch usability evaluation. In the rest of this section, I present some of the data collected by this questionnaire, which I find as important to be discussed.

Firstly, I wanted to see how familiar the participants were with MS Office (e.g., MS Word and MS PowerPoint in particular). Figures 7.8 *(a)* and *(b)* show: *(a)* how often

[10] http://www.sfu.ca/
[11] http://www.athabascau.ca/
[12] http://www.bg.ac.rs/
[13] http://www.usi.ch/en
[14] http://www.semanticdoc.org/OpenIdCheck/

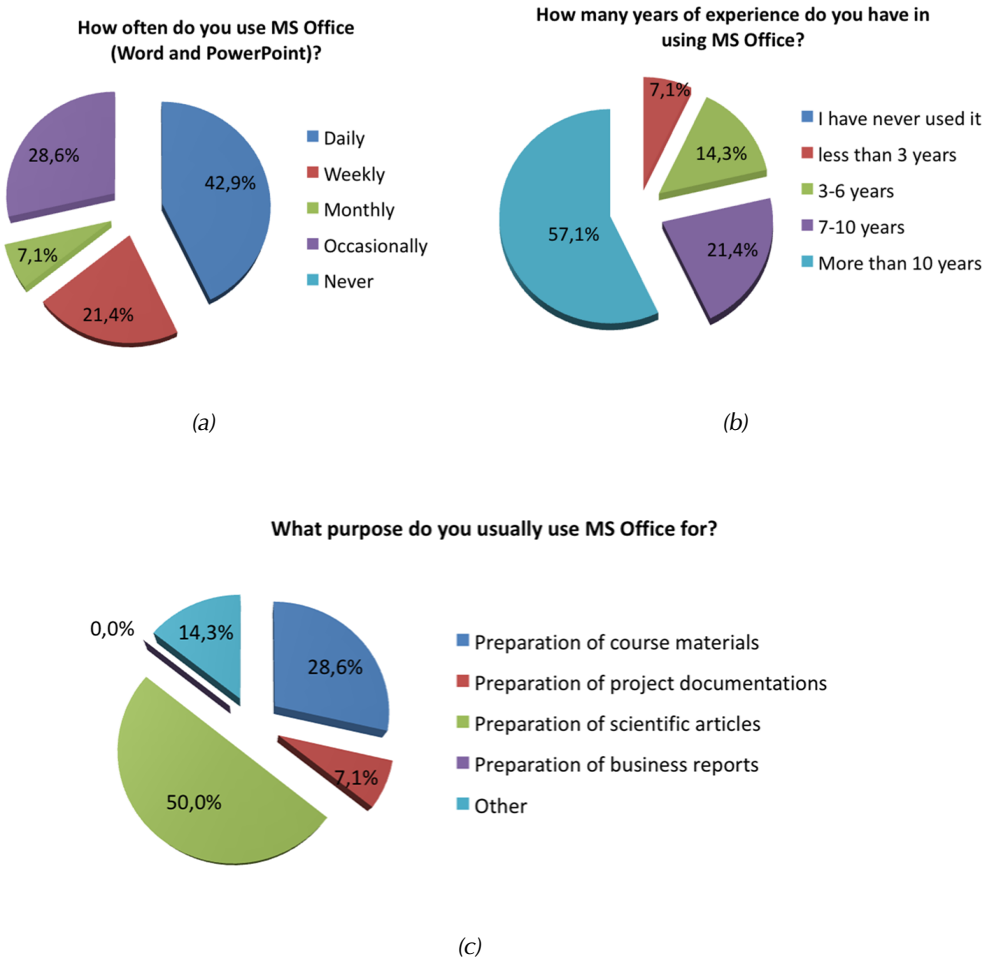
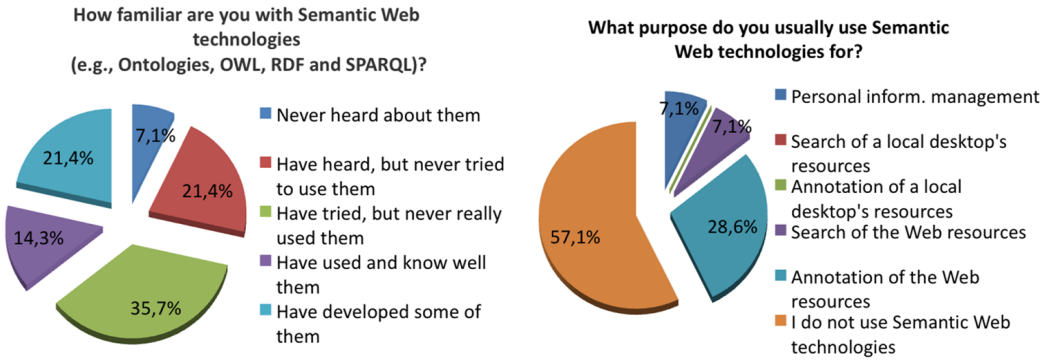


Figure 7.8. Participants' familiarity with MS Office: *(a)* how often they use MS Office; *(b)* how experienced MS Office users they are; and *(c)* what purpose they use MS Office for

the participants used MS Office and *(b)* how much experience they had in using it. As we can se from the figure, the biggest portion (49.2%) of the participants used MS Office daily, and there were no participants who 'Never' used it. Moreover, most of the participants 57.1% had more than 10 years of experience in using MS Office. Figure 7.8 *(c)* shows what purpose the participants used MS Office for. The two most significant groups were the participants who usually use MS Office for a preparation of scientific articles (50%), and the participants who usually use MS Office for a preparation of course material (28.6%).

Secondly, Figures 7.9 *(a)* and *(b)* illustrate familiarity of the participants with semantic web technologies (e.g., ontologies, RDF and SPARQL), and for what purposes they use them. As can be seen from Figure 7.9 *(a)*, there was a whole span of participants, starting from those who were completely unfamiliar with the semantic web technologies to the domain experts (i.e., participants who participated in the developed of some semantic web technologies). Familiarity with semantic web technology was an important criterion, which I considered while selecting participants. Since intended users of SDArch do not necessarily need knowledge of semantic web technologies, I wanted to have participants who were not too familiar with these technologies. On the other hand, I wanted to have some domain experts too, since I planned to evaluate the underlying SDArch models which relay extensively on semantic web technologies.



(a) *(b)*

Figure 7.9. Participants' familiarity with Semantic Web technologies *(a)* and for what purpose they use them *(b)*

Finally, I wanted to find out how familiar the participants were with online social networking. The results collected by the questionnaire showed that all of the participants were members of at least one online social network.

7.2.5 Acquisition of the Evaluation Document Collection

After participants had registered to the SDArch social network, I organized a training session using Web conferencing in which I described the underlying SDArch models (i.e., the semantic document model, the user profile model, and the social network model), and demonstrated the main functionalities of the SDArch services and tools. Moreover, I explained the installation procedure for the SDArch tools to those participants who wanted to install them on their computers. The installation of the tools was not mandatory. For all the participant I created accounts on our PC-server and enabled them to access it remotely. The only software that the participants needed to install/enable

on their computers was a remote desktop connection software. This kind of software is supported as an official feature on all new-generation, Windows operating systems (e.g., Windows XP/Vista/7), so that the participants using Windows only needed to enable it, unless they had used it before. A total of 7 participants decided to install the SDArch tools on their computers. None of them experienced any serious problem during the installation, and all of them managed to run the tools successfully.

After the training session, I initiated the 'Software Design Patterns' user group within the SDArch social network, and published an initial 30 semantic documents from my archive to the group's semantic document collection. Then, I invited the participants to join the group and asked them to check if they had some Office (i.e., Word and PowerPoint) documents related to software design patterns in their archives, and to publish them to the shared semantic document collection. Moreover, I created a simple Web-based application providing a file upload form, and asked the participants to upload the original documents, which they published to the shared semantic document collection. In that way, besides the shared semantic document collection I also obtained the collection of original MS Office documents, which I needed for the third use case in the usability test (Section 7.2.6).

Two weeks after I initiated the design patterns group, the total number of the shared semantic documents reached 70 documents. According to my personal experience as a teaching assistant, and based on the results of the formative evaluation (Appendix B.1), I judged this number of documents as sufficient for the usability test (Section 7.2.6), which was implemented in the evaluation session. Therefore, I decided to organize the evaluation session and conduct the usability test. In addition, not all the participants contributed in the acquisition of the shared semantic document collection. There were 11 of them who published one or more documents to the collection.

7.2.6 Task-Based Usability Test

The usability test was designed as a task-based [135] usability test whose tasks were organized into three use cases: 1) setting up the user profile and the social network; 2) authoring and publishing semantic documents; and 3) searching and navigating across semantic documents. The use cases involved tasks which were supposed to familiarize the participants with SDArch services and tools, and the corresponding processes they realize. While the first use sase covered the SDArch functionalities related to the user profile and social network management, the other two considered the SDArch functionalities related to the three main semantic document processes, that is, the semantic document authoring, the semantic document search, and the semantic document navigation. In the rest of this section, I discuss the three use cases focusing on their objectives and the SDArch services and tools that were deployed within them. In Appendix B.3, I provide step-by-step guides of all three use cases. Those guides were given to the participants at the beginning of the evaluation session.

Use Case 1: Setting Up the User Profile and the Social Network

The objective of this use case was to familiarize the participants with the SDArch user profile management and social network management, and then to collect the subjective user feedback on these two processes. By performing the tasks of the use case, the participants were supposed to interact with the user profile management service (Section 4.2.3) and the social network management service (Section 4.2.4). This interaction was achieved by using the user account and profile tools (Section 6.4.1), and the social network manager tool (Section 6.4.2) respectively. The feedback was collected through the follow-up questionnaire (Questionnaire A, Section 7.2.7).

Regarding the user profile management, the participants were asked to set their personal user information and the values of the user preferences that are defined in the SDArch user model (Section 4.2.3). The values of the user preferences are used for the personalization of the semantic document search (Section 5.2.2). The feedback that I wanted to collect concerned the participants' opinion of the overall usefulness of the SDArch user preferences as well as their opinion of each preference individually. In addition, I asked the participants to try to identify potential aspects, which were not considered by the existing SDArch user preferences, but which might be useful for the search personalization.

Regarding the social network management, since the participants had already been members of the SDArch social network, within this use case I wanted them to try the remaining features of the social network management, such as initiating a new user group about a given topic of interest and joining an existing group. In the feedback that I collected afterwards, my focus was on the participants' opinion of the SDArch social network model and the idea to organize users around shared semantic documents.

Use Case 2: Authoring and Publishing Semantic Documents

The objective of this use case was to familiarize the participants with the processes of authoring and publishing semantic documents to the shared semantic document repository. Each participant was given an example Word document and was asked to transform that document to a semantic document and publish it to a shared collection of semantic documents belonging to the given SDArch user group. The participants were supposed to use the document transformer and publisher tool (Section 6.4.4) to execute the tasks of the use case. This tool actually enabled the users to interact with two SDArch services: the semantic document authoring and the social network management services. The first one was deployed for the semantic document authoring and the other one for the document publishing. Similar to the previous use case, the follow-up questionnaire (Questionnaire B, Section 7.2.7). was used to collect participants' feedback on the semantic document authoring and publishing process. The focus in the questionnaire was on the following two aspects.

Firstly, I wanted to obtain the participants' opinion of the semantic document model (Section 3.2) and its main features including unique identification, semantic annotation, and semantic linking of small, self-contained document data units. Moreover, I asked the participants for their opinion about the use of concepts of domain ontologies for the conceptualization of knowledge/information stored in document units and the use of such conceptualized knowledge for the semantic annotation, indexing and linking of document units.

Secondly, I asked the participants for their opinion about the proposed semantic document authoring approach that is based on the transformation of existing, conventional documents into semantic documents. The idea of publishing semantic documents to the shared semantic document repository was also considered in the questionnaire.

Use Case 3: Searching and Navigating across Semantic Documents

The objective of this use case was to familiarize the participants with the semantic document search and navigation processes. The participants were supposed to experience these processes by performing a set of tasks designed to produce a short PowerPoint presentation on the given topic (i.e., software design patterns). Contrary to the first two use cases, the participants were asked to perform the tasks of this use case twice, once using the SDArch system and once using the conventional one. When performed by using the SDArch system, this use case actually corresponded to the motivational scenario of the case study (Section 7.2.5). Basically, the participants were asked to create three slides, by exploring (i.e., searching and navigating across) the evaluation document collection (Section 7.2.3) and providing:

- a general definition of design patterns (slide 1) - **TASK 1**;
- a definition of the first pattern example (slide 2; item 1) - **TASK 2**;
- a graphical illustration of the first pattern example (slide 2; item 2) - **TASK 3**;
- a definition of the second pattern example(slide 3; item 1) - **TASK 4**;
- a graphical illustration of the second pattern example (slide 3; item 2) - **TASK 5**;

The presentation was supposed to contain exclusively data from the evaluation document collection. As explained in (Section 7.2.3), the document collection was available in two forms: 1) as the collection of original MS Office (Word and PowerPoint) documents, which was intended to be used in the authoring scenario realized by the conventional system, and 2) as the shared collection of semantic documents, which was intended to be used in the authoring scenario realized by the SDArch system. All participants took part in both authoring scenarios, but they were divided into two control groups of 9 participants. The first group was asked to use the conventional system first and then the SDArch system. The second group was asked to used the systems in the

opposite order. As explained in Section 7.2.5 the evaluation document collection was available in two forms: the collection of semantic documents and the collection of original, Office (i.e., Word and PowerPoint) documents. The semantic document collection was stored into the shared SDArch semantic document repository, and the participants could interact with the semantic documents by using the SDArch search (Section 6.4.5) and navigation tools (Section 6.4.6). The collection of original, Office documents was stored into a directory within a file system of our server used for the evaluation. Document file names were the original names given by their authors. Most of the file names were semantically related with document contents. The reason why I kept the original file names was that I wanted to have the authoring scenario by using the conventional system as realistic as possible. The participants were supposed to use Windows Explorer to browse, filter and sort documents of the document collection as well as to use Windows Search to search the documents for a particular content.

The screen activities of the participants, while they were performing the tasks were recorded by Camtasia[15] screen recorder. The recorded materials were analyzed after the evaluation session, and the chosen evaluation methods and metrics (Table 7.7) were applied to measure the user effectiveness and efficiency when using the two systems.

Besides information extracted from the screen recordings, the use case also provided subjective user feedback on the semantic document search and navigation processes, collected trough the follow-up questionnaire (Questionnaire C, Section 7.2.7).

7.2.7 Conducting the Evaluation Session

The evaluation session, including the usability test and the follow up questionnaires, was organized at the end of the case study. Putting together the time spent for the recruitment of participants, the time spent for the organization of the training session, and the time spent for the acquisition of the evaluation document collection, the evaluation session was actually organized five weeks after I sent out the invitations for participation in the SDArch evaluation. The evaluation session was conducted asynchronously, by one participant at a scheduled time. Although the evaluation session was not time-limited, it was not supposed to last longer than 4 hours, which was a minimal time distance between two participants. The reason why the evaluation session had to be conducted asynchronously was because all the participants were asked to used the same PC-server for the evaluation session. As mentioned in (Section 7.2.3), there were 7 participants who successfully installed the SDArch tools on their computers, and who could use them for the evaluation session. However, to avoid asking participants to install the screen recording software, and to simplify the manipulation of the recorded material, I asked all the participants to use our PC-server for the evaluation session. I gave the participants a time-frame of one week for the evaluation session. Scheduling the time for

[15] http://www.techsmith.com/camtasia.asp

the evaluation session was done by using Tungle.me[16], the on-line meeting scheduling software. The given time-frame was respected, and I conducted the evaluation session with all the participants within that week.

Each evaluation session started with a short skype call (approximately 5-10 minutes) in which I gave to a participant some verbal instructions regarding the usability test and provided him the written, step-by-step instructions for the use case of the usability test. Moreover, I explained to the participant when and how to start and stop the screen recording software, and how to save the recorded material. Finally, I gave the participant instructions how and where to save the two PowerPoint presentations resulting from the third use case.

I used SurveyMonkey[17], a popular, online questionnaire tool for the questionnaires conducted during the evaluation session. For each of the five questionnaires (the entrance questionnaire, the follow-up questionnaires A, B, and C of the use cases, and the user satisfaction questionnaire) I created a SurveyMonkey. These links were provided in the written, step-by-step instructions. In addition, I was available on Skype all the time during the evaluation session, so that the participants could contact me in case of any problems they experienced during the session. In case some problems occurred while the participant's activities were recorded, I asked them to first pause recording and then contact me. A total of five participants contacted me during the session, asking for some additional clarifications of the tasks, while only one participant contacted me to report a failure in the software. In the latter case, I took over control of the PC-server and after I resolved the problem, the session was continued. At the end of the session, few participants also reported that they had lost the connection to the PC-server during the session, but managed to reconnect without need for any help.

After the initial skype call, the participants first completed the entrance questionnaire and then executed the use cases of the usability test one after another accompanied by the corresponding follow-up questionnaires. At the end of the session the participants filled in the satisfaction questionnaire and after that disconnected from the PC-server.

7.2.8 Evaluation Results and Discussion

The results of the entrance questionnaire were considered in Section 7.2.4, in the scope of the discussion on the evaluation participants, and will not be further discussed hereafter. In this section, I discuss the subjective user feedback obtained through the follow-up questionnaires (i.e., questionnaires A, B, C) of the three use cases, then I discuss the results of the quantitative measures applied to evaluate the user effectiveness and efficiency, and I conclude the section with the discussion on the results of the user satisfaction evaluation, which were obtained through the user satisfaction questionnaire.

[16] http://www.tungle.me/Home/

[17] http://www.surveymonkey.net/

User Feedback on the SDArch User Profile and Social Network Management

In this section, I present the results of the questionnaire A (Table 7.8), which was designed to collect the user feedback on the SDArch user profile (Section 4.2.3) and social network management (Section 4.2.4) processes and the underlying user profile and social network models. The first column in the table contains question IDs, which I introduced just for clearer presentation of the questionnaire's results. The second column provides the information about question types. The questionnaire was composed of two types of questions/statements. The first type were positive statements rated by the five-level Likert scale (LS): 1 - Strongly Disagree; 2 - Disagree Somewhat; 3 - Uncertain; 4 - Agree Somewhat; and 5 - Strongly Agree. It actually means that the participants had to specify their level of agreement with a statement from 'strongly disagree' to 'strongly agree', with the neutral option 'uncertain', that prevents the participant from randomly selecting one option and thus causing bias for the evaluation. The second question type were open-ended questions (OE), which were mandatory to answer (i.e., the online questionnaire tool was set to allow the questionnaire to be finished only if all the OE questions were answered). The questionnaire contained 17 questions, 4 of which were OE questions and 13 were LC questions. In addition, all of the LC questions/statements were formulated as positive statements.

Table 7.8. Questionnaire A

Id	Type	Question
A1	LC	I find that the idea of the globally unique identification of SDArch users (e.g., OpenID) is useful.
A2	LC	I find that the possibility to specify my preferences, regarding document units to be reused, is useful.
A3	LC	I find that the preference specified as 'Do you prefer to reuse document units that have been reused many times?' is important.
A4	LC	I find that the preference specified as 'Do you prefer to reuse document units that have many versions?' is important.
A5	LC	I find that the preference specified as 'Do you prefer to reuse document units that have been recently modified?' is important.
A6	LC	I find that the preference specified as 'Do you prefer to reuse the most often used versions of the document unit?' is important.
A7	LC	I find that the preference specified as 'Do you prefer to reuse document units' versions which has many subversions?' is important.
A8	LC	I find that the preference that determines the list of preferred document authors is useful.

Continued on next page

Table 7.8 – Continued

Id	Type	Question
A9	LC	I find that the preference that determines the list of preferred document software is useful.
A10	OE	Are there any user preferences regarding the reuse of document data units that is currently not supported by the SemanticDoc user model and that you find important? If so, could you please describe them.
A11	LC	Overall, I find the management of the user profile information effortless.
A12	OE	What changes would you make to enhance the user profile management?
A13	LC	I find that the possibility of sharing fine-grained document units such as paragraphs, illustrations and tables (instead of whole documents) in online social networks is useful.
A14	LC	I find that the idea of organizing SDArch users into groups sharing documents of a given topic is useful.
A15	LC	Overall, I find the management of the SDArch social network effortless.
A16	OE	What changes would you make to enhance the management of the SemanticDoc social network?
A17	OE	Additional comments/suggestions?

The results of the LC statements (Table 7.9) are given in percentage of participants who ticked each of the scale items. Moreover, for each statement I also calculated the response average and standard deviation, by taking into account numerical counterparts of the Likert scale items (e.g., 'Strongly Disagree' = 1 and 'Strongly Agree = 5'). The top rated scale item of each statement is marked **bold**; if there is more than one top rated item, all of them are **bold**.

The first part of the questionnaire (LC statements A1 - A9 and A1, and OE questions A10 and A12) was related to the SDArch user profile management. The first statement (A1) evaluates the idea of unique identification of SDArch users. The top rated answer (50%) was 'Agree Somewhat' and the average rating was 4.21. The statements A2 - A9 were related to the user preferences defined by the SDArch user model (Section 4.2.4). The general idea of introducing user preferences in the user model (A2), was rated as highly positive with the average rating of 4.42. The usefulness of each of the six user preferences defined in model, was evaluated separately (A3 - A9). The average ratings of the user preferences range from 3.78 (for the preference specified as 'Do you prefer to reuse document units' versions which has many subversions?' - statement A7) to 4.71 (for the preference specified as 'Do you prefer to reuse document

units that have been reused many times? - statement A3). The highest rated preference (A3), was also the preference which scored the highest percentage (71.4%) of 'Strongly Agree' answers. The average rating of the statement A11, which stated that the user profile management process was effortless, was 4.35. In the OE questions A10 and A12, I asked the participants to identify possibly missing user preferences, and to suggest changes that might improve the user profile management. An interesting proposal for a new user preference, which sublimates several suggestions, could be specified as 'Do you prefer to reuse document units that have been reused many times in a specific time-frame?'. Regarding the improvement of the user profile management, there were several suggestions about connecting SDArch to some popular social network services such as Facebook, MySpace, LinkedIn, in order to facilitate interoperability of the profile data.

The second part of the questionnaire(LC statements A13 - A15, and OE questions A16 and A17) was related to the SDArch social network management. The idea of sharing fine-grained document units in online social networks (statement A13), was rated as highly positive with 56.4% of 'Strongly Agree' and 45.6% of 'Agree Somewhat' answers, and the average rating of 4.7. The following two statements, which evaluated the idea of organizing SDArch users into groups sharing semantic documents on a given topic (A14) and the overall user opinion of the SDArch social network management (A15), scored the average ratings of 4.58 and 4.41 respectively. The participants' suggestions, which were collected through the OE questions A16 and A17, were related mostly to the browsability and search of the social network groups and members. Since currently there is only support for a simple (linear) browsing of social network groups and members, the participants showed concerns that the navigation of both would be cumbersome if users face with a huge social network.

As a summary, the highest percent of answers for each of the LC statements of the questionnaire A were either 'Strongly Agree' and 'Agree Somewhat'. Only three statements (A4, A5, and A7) had also some 'Disagree Somewhat' answers and none of the statements had any 'Strongly Disagree' answer. Moreover, the comments and suggestions obtained through the OE questions, indicated that most of participants became familiar with the main characteristics of the user and social network management processes by performing the tasks of the first use case.

Table 7.9. Results of the Questionnaire A

Id	Strongly Disagree	Disagree Somewhat	Uncert.	Agree Somewhat	Strongly Agree	Response Average	SD
A1	0.0%	0.0%	14.3%	**50.0%**	35.7%	4.21	0.69
A2	0.0%	0.0%	0.0%	**57.1%**	42.9%	4.42	0.51
A3	0.0%	0.0%	0.0%	28.6%	**71.4%**	4.71	0.46

Continued on next page

Table 7.9 – Continued

Id	Strongly Disagree	Disagree Somewhat	Uncert.	Agree Somewhat	Strongly Agree	Response Average	SD
A4	0.0%	7.1%	28.6%	**35.7%**	28.6%	3.85	0.94
A5	0.0%	14.3%	21.4%	28.6%	**35.7%**	3.85	1.05
A6	0.0%	0.0%	7.1%	42.9%	**50.0%**	4.42	0.64
A7	0.0%	7.7%	30.8%	**38.5%**	23.1%	3.78	0.89
A8	0.0%	0.0%	8.3%	**58.3%**	33.3%	4.21	0.57
A9	0.0%	0.0%	14.3%	35.7%	**50.0%**	4.35	0.74
A10	*- open-ended question -*						
A11	0.0%	0.0%	7.1%	35.7%	**57.1%**	4.35	0.49
A12	*- open-ended question -*						
A13	0.0%	0.0%	0.0%	45.6%	**56.4%**	4.7	0.65
A14	0.0%	0.0%	7.1%	35.7%	**57.1%**	4.58	0.61
A15	0.0%	0.0%	14.3%	35.7%	**50.0%**	4.41	0.69
A16	*- open-ended question -*						
A17	*- open-ended question -*						

User Feedback on the Semantic Document Authoring and Publishing

In this section, I present the results of the questionnaire B (Table 7.10), which I designed to collect the user feedback on the semantic document authoring (Section 5.1) and publishing (Section 4.2.4) processes and the underlying semantic document model (Section 3.2).The participants answered the questionnaire after the second use case (i.e., 'Semantic Document Authoring and Publishing'). The questionnaire contained 12 questions, 10 of which were defined as positive statements adhered to five-level Likert scale and 2 were open-ended questions. The first 6 questions (LC statements B1 - B5 and OE question B6) were designed to evaluate the features of the semantic document model, which represents the foundation of the semantic document authoring and publishing processes. The following 6 questions (LC statements B7 - B11 and OE question B12) were designed to evaluate the main features of the authoring and publishing processes.

Table 7.10. Questionnaire B

Id	Type	Question
B1	LC	I find that the semantic document model, which enables the unique identification, semantic annotation and semantic linking of small, self-contained document data units, is promising solution for new generation of documents that will be able to fully contribute to the visions of the Semantic Web and the Social Semantic Desktop.
B2	LC	I find that the semantic annotation of document units by concepts from domain ontologies, which conceptualize the information kept in document units, is useful for the document unit's search and discovery.
B3	LC	I find that the explicit semantic links that connect semantically related document units are useful for the document unit's navigation and discovery.
B4	LC	I find that the possibility to establish explicit semantic links between semantically related document units that belong to different documents is useful for desktop data integration and management.
B5	LC	I find that the possibility to establish explicit semantic links between document units of the local desktop documents and other uniquely identified resources on the Web is useful for the success of the envisioned Semantic Web.
B6	OE	Do you have any suggestions for how to further enhance the proposed semantic document model?
B7	LC	I find that the possibility to transform documents of conventional desktop document formats (e.g., Word, PowerPoint, PDF) into semantic documents is useful.
B8	LC	I find that the possibility to decide which domain ontologies will be used for the semantic annotation, linking and indexing of the document to be transformed is useful.
B9	LC	I find that the possibility to browse the ontology details (e.g., classes and properties) before make the decision which ontology to use, is useful.
B10	LC	I find that the possibility to choose the location where the transformed document will be stored (i.e., local or shared semantic document repository) is useful.
B11	LC	The authoring and publishing processes are completely automated and require minimal user effort.
B12	OE	What changes would you make to enhance the transformation process?

Similar to questionnaire A, the results of the LC statements of the questionnaire B (Table 7.11), are given as percentage of participants who ticked each of the five possible scale items. The response average and standard deviation of each statement are also presented. The first statement (B1), which evaluated the core idea of the semantic document model, scored the average rating of 4.57 with 64.3% of 'Strongly Agree' answers. The following 4 statements (B1 - B5), which evaluated main features of the model, all scored the average rankings higher than 4 and had 'Strongly Agree' as the most ticked answer. Suggestions of how to improve the model, collected through the OE question B6, were mainly about the possibility to manually assess the relevance of the semantic annotations, that is, the relevance of the annotation ontological concepts for document units they annotate. In the currently supported authoring scenario, the annotation relevance is automatically calculated. However, the semantic document model itself does not limit the possibility to have the annotation relevances assessed by the domain experts instead of being automatically calculated. The statements B7 - B9 evaluated the aspects of the authoring process such as the possibility to transform documents of conventional formats into semantic documents (B7), the possibility to decide which domain ontologies will be used for the semantic annotation, linking and indexing (B8), and the possibility to browse the ontology details while making a decision which ontology to use in the authoring process. The average ratings of these three statements were 4.42, 4.14 and 3.39 respectively. The statement (B10) evaluated the possibility to store generated semantic documents either in the local (desktop) or shared semantic document repositories, where choosing the shared repository actually means publishing semantic documents. The statement B11 evaluated the degree of the automation in the authoring and publishing processes. The average rating of this statement was high (4.21), but it was the only statement in the questionnaire which also scored some negative answers (e.g.,'Strongly Disagree' = 7.1% and 'Disagree Somewhat' = 7.1%). Finally, comments obtained through OE question B12 suggested that the authoring process could be further improved if the system would provide the automatic selection of ontologies.

Table 7.11. Results of the Questionnaire B

Id	Strongly Disagree	Disagree Somewhat	Uncert.	Agree Somewhat	Strongly Agree	Response Average	SD
B1	0.0%	0.0%	7.1%	28.6%	**64.3%**	4.57	0.57
B2	0.0%	0.0%	7.1%	42.9%	**50.0%**	4.42	0.64
B3	0.0%	0.0%	7.1%	28.6%	**64.3%**	4.57	0.52
B4	0.0%	0.0%	8.3%	41.7%	**50.0%**	4.07	0.65
B5	0.0%	0.0%	0.0%	46.2%	**53.8%**	4.21	0.51
B6	*- open-ended question -*						

Continued on next page

Table 7.11 – Continued

Id	Strongly Disagree	Disagree Somewhat	Uncert.	Agree Somewhat	Strongly Agree	Response Average	SD
B7	0.0%	0.0%	7.1%	42.9%	**50.0%**	4.42	0.64
B8	0.0%	0.0%	28.6%	28.6%	**42.9%**	4.14	0.86
B9	0.0%	0.0%	28.6%	**50.0%**	21.4%	3.92	0.73
B10	0.0%	0.0%	23.1%	30.8%	**46.2%**	4.21	1.25
B11	7.1%	7.1%	0.0%	28.6%	**57.1%**	4.21	1.1
B12	*- open-ended question -*						

User Feedback on the Semantic Document Search and Navigation

In this section I present the results of the follow-up questionnaire C (Table 7.12), which I designed to collect the user feedback on the semantic document search (Section 5.2.1) and navigation (Section 5.2.3) processes. The participants were asked to fill in the questionnaire after the third use case. The questionnaire contained 10 questions, 6 of which were defined as positive LC statements and 4 OE questions.

Table 7.12. Questionnaire C

Id	Type	Question
C1	LC	I find that the possibility to search and reuse fine-grained document units based on their conceptualized semantics (i.e., annotation ontological concepts) is useful.
C2	LC	I find that the possibility to see the information about the document unit's context of use (e.g., number of reuse and list of previous users) is useful.
C3	LC	I find that the possibility to see the annotation concepts of the document unit is useful.
C4	LC	The relevance weight of the annotation concepts for the document unit they annotate, is useful information for me.
C5	LC	I find that the possibility to navigate among semantically related document units, that is, the document units annotated by the same annotation concepts is useful.
C6	LC	The possibility to reuse desirable document units from the search results, with only one mouse click, is useful and saves my time.
C7	EO	What changes would you make to enhance the proposed semantic document search and navigation?

Continued on next page

Table 7.12 – Continued

Id	Type	Question
C8	EO	What is the feature of the semantic document search and navigation tools that you liked the most?
C9	EO	Do our tools provide some new features that you have never before met in conventional Office environments (e.g., MS office or OpenOffice)? If so, please specify them.
C10	EO	Additional comments/suggestions?

The results of the questionnaire (Table 7.13) are given in the same way as the results of the follow-up questionnaires A and B. The first statements (C1) that evaluated the core idea of the semantic document search obtained 78.6% of 'Strongly Agree' answers and scored the average rating of 4.78. It was the best rated statement in this questionnaire. The statements C2, C3 and C4 were designed to evaluate the possibility to browse the social context annotations, the annotation ontological concepts, and the concept relevance weights for retrieved document units respectively. The average rating of each of them was higher than 4, with the great majority of answers being 'Agree Somewhat' and 'Strongly Agree'. The idea of the semantic document navigation (C5) was also rated as highly positive, with 64.3% of 'Strongly Agree' answers and the average rating 4.57. The statement C6 that evaluated the possibility to reuse document units from the retrieved search results with only one mouse click obtained 71.4% of 'Strongly Agree' answers and the average rating of 4.64. This statement, could be interpreted as a subjective evaluation of the user efficiency in using the semantic document search and navigation tools. The last four questions C7 - C10 were OE questions designed to collect participants' suggestions of how to potentially enhance the semantic document search and navigation processes, and to collect their opinion of what was the best feature of the evaluated tools and whether there were some novel features that they had never before met in conventional document systems. Collected suggestions were mostly about how to improve the presentation of the search and navigation results. Regarding the semantic document search and navigation features, almost all participants said that the possibility to navigate among semantically interlinked document units was the feature that they liked the most. This feature and the possibility to specify a user query by selecting suggested concept labels from domain ontologies were the features that most of the participants indicated as something novel for them.

Table 7.13. Results of the Questionnaire C

Id	Strongly Disagree	Disagree Somewhat	Uncert.	Agree Somewhat	Strongly Agree	Response Average	SD
C1	0.0%	0.0%	0.0%	21.4%	**78.6%**	4.78	0.42
C2	0.0%	0.0%	7.1%	**50.0%**	42.9%	4.35	0.63
C3	0.0%	0.0%	0.0%	**50.0%**	**50.0%**	4.5	0.51
C4	0.0%	0.0%	2.4%	**50.0%**	28.6%	4.07	0.73
C5	0.0%	0.0%	7.1%	28.6%	**64.3%**	4.57	0.63
C6	0.0%	0.0%	7.1%	21.4%	**71.4%**	4.64	0.71
C7	*- open-ended question -*						
C8	*- open-ended question -*						
C9	*- open-ended question -*						
C10	*- open-ended question -*						

User Effectiveness and Efficiency Measurements

The user effectiveness and efficiency were evaluated by performing the objective, quantitative measures (Table 7.7) of user performance in doing the tasks of the third use case. These measures were actually applied on information/data extracted from the screen recordings and the resulting PowerPoint presentations.

The evaluation metric applied for the evaluation of the user effectiveness was the task success rate. For each of the 5 considered tasks (i.e., the tasks of the third use case), Table 7.14 exhibits a number of participants who successfully completed the task by using the conventional system and by using the SDArch system. The table also provides the tasks success rates (expressed in percentages) for each task for both systems. It is important to clarify that a task was considered as successfully completed if a participant provided (i.e., managed to find and reuse) appropriate data/information required by the task. As can be seen from the table, the success rate of task 1, for both systems, was 100%. In case of the other four tasks, the task success rate was higher for the SDArch system than the conventional one. For tasks 2 and 4, the difference between the task success rates was 5.66% in favor of the SDArch system, while for the tasks 3 and 4 it was 11.11% again in favor of the SDArch system. What tasks 3 and 4 have in common is that in both of them the participants were asked to search for and provide graphical illustrations. On the other hand, in tasks 2 and 4, the participants had to search for and provide textual definitions.

Task	Conventional System		SDArch system	
	Successful Completions	(%)	Successful Completions	(%)
1	18	100%	18	100%
2	17	94.44%	18	100%
3	15	83.33%	17	94.44%
4	17	94.44%	18	100%
5	14	77.77%	16	88.88%

Table 7.14. Task success rates

➥ ***User effectiveness - Evaluation Outcome***
The task success rates for the compared systems indicate that for the given evaluation document collection and considered tasks, the user effectiveness when using the SDArch system was better that the user effectiveness when using the conventional system.

The evaluation metrics applied for the evaluation of the user efficiency were the task completion time, the number of windows switches, and the number of mouse clicks. Table 7.15 and Figure 7.10 show the average and median task completion times of all five considered tasks for the two compared systems. Moreover, Table 7.14 provides the relative performance of the participants when using the SDArch system with respect to the conventional system. For example, the relative performance of 60% indicates that the participants using the SDArch system needed 60% of the time that the participants using the conventional system needed. Finally, I conducted a *T-Test* [136] to investigate on the statistical significance of the difference in the task completion times between two control groups (i.e., the participants using the SDArch system and the participants using conventional system). The results of the *T-Test* (i.e., *p-values*), are shown in the last column of the table. In our case, *p-values* represents the probability that the measured task completion times for the two control groups are part of the same distribution. In general, *p-values* below 0.05 are considered statistically significant. In other words, *p-values* of 0.05 or greater indicate that there is no statistically significant difference between the results of two control groups.

The measured times reported in Table 7.15 show that for each of the considered tasks, the participants needed less time by using the SDArch system than the conventional one. The values of the relative user performance range from 82.99% to 90.83%. Moreover, calculated *p-values* for all tasks were statistically significant (i.e., < 0.05), which actually means that by using the SDArch system the participants needed significantly less time than by using the conventional system for all tasks.

Table 7.16 and Table 7.17 show the same statistics for the number of mouse clicks and the number of window switches, respectively. For both of these metrics, regarding all the tasks, the relative performance of the participants when using the SDArch

Task	Convent. System		SDArch system		Rel. Performance		T-Test
	Average	Median	Average	Median	Average	Median	P(T<=t)
1	4 : 10	3 : 46	3 : 28	3 : 24	90.25%	90.48%	0.00071
2	3 : 00	2 : 53	2 : 43	2 : 45	90.83%	95.10%	0.00011
3	3 : 30	3 : 25	2 : 55	2 : 45	82.99%	83.05%	$9.17 * 10^{-6}$
4	2 : 57	2 : 45	2 : 42	2 : 46	91.34%	93.46%	0.00034
5	3 : 30	3 : 19	2 : 57	2 : 52	84.13%	82.93%	$2.69 * 10^{-6}$

Table 7.15. Task completion times, relative user performance, and T-Test results

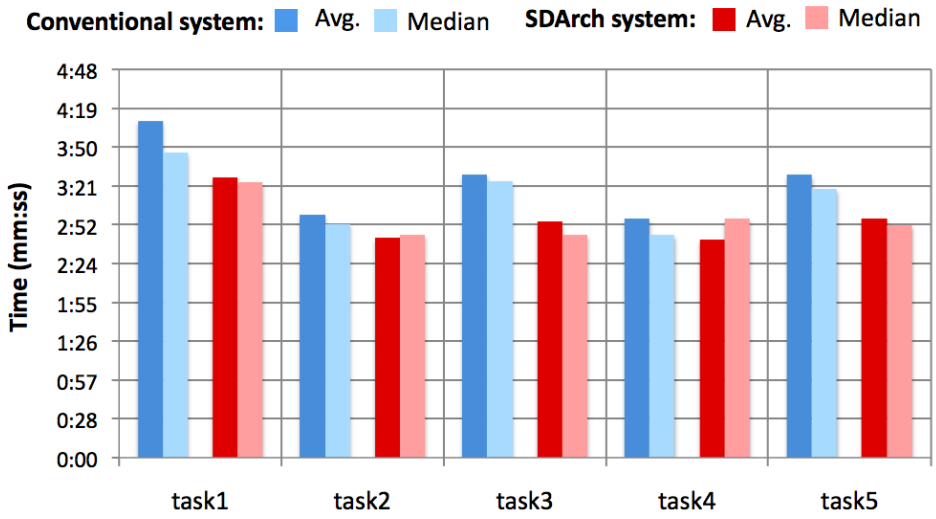


Figure 7.10. Avgerage and median task completion times

system with respect to the conventional system was less than 100%. In other words, the participants performed less mouse clicks and window switches by using the SDArch system than by using the conventional one, regarding all considered tasks. In particular, a number of window switches for each task was significantly (more than two times) less with the SDArch system. Finally, *T-Test* showed that the difference between the results of the two control groups regarding the applied evaluation metrics are also statistically significant (*p-values* of all the tasks for both metrics are much less than 0.05).

➥ ***User Efficiency - Evaluation Outcome***
The measured task completion times (Table 7.15), numbers of mouse clicks (Table 7.16) and numbers of window switches (Table 7.17) indicate that the user efficiency when using the SDArch system outperforms from the user efficiency when using the conventional system, with respect to these three applied evaluation metrics.

Task	Convent. System		SDArch system		Rel. Performance		T-Test
	Average	Median	Average	Median	Average	Median	P(T<=t)
1	20.57	21	16.35	16	79.51%	76.19%	0.0000458
2	13.21	13.5	11.14	11.5	84.32%	85.18%	0.0041
3	18.71	18	14.21	14	75.95%	77.77%	0.00016
4	14.35	14	10.28	9.5	71.64%	67.85%	0.0000983
5	16.35	15.5	13.21	12.5	80.78%	80.64%	0.0000435

Table 7.16. Number of mouse clicks

Task	Convent. System		SDArch system		Rel. Performance		T-Test
	Average	Median	Average	Median	Average	Median	P(T<=t)
1	11.42	11.5	5.07	5	44.37%	43.47%	$1.64 * 10^{-12}$
2	8.57	9	4.14	4	48.33%	44.44%	$1.22 * 10^{-7}$
3	10.42	10	4.78	4.5	45.89%	45%	$6.91 * 10^{-8}$
4	6.87	7	3.42	3	49.72%	42.85%	$3.67 * 10^{-7}$
5	10.42	10.5	4.21	4	40.41%	38.09%	$4.82 * 10^{-10}$

Table 7.17. Number of window switches

User Satisfaction

The user satisfaction is the third usability component, besides the user effectiveness and efficiency, which was considered in the SDArch usability evaluation. The following user satisfaction dimensions were considered: the system *usefulness*, the system *ease-of-use*, the system *ease-of-learning*, and the user *overall satisfaction* with the system. The user satisfaction was evaluated based on the subjective user feedback collected through the satisfaction questionnaire (Table 7.18). I created this questionnaire relying on a widely accepted, the Perceived Usefulness, Ease of Use, and Ease of Learning questionnaire [26]. All questions were formulated as positive statements and were grouped into four sections corresponding to the above listed user satisfaction dimension. Statements D1.1 - D1.9 were dedicated for the system *usefulness*, statements D2.1 - D2.10 for the system *ease-of-use*, statements D3.1 - D3.3 for the system *ease-of-learning*, and statements D4.1 - D4.4 for the user *overal satisfaction* with the system. Similar to the follow-up questionnaires A, B and C, the statements of the user satisfaction questionnaire were rated by using the five-level Likert scale.

Table 7.18. User satisfaction questionnaire

Id	Type	Question
Usefulness		
D1.1	LC	Using SDArch tools improves my job performance.
D1.2	LC	SDArch tools address my job-related needs.
D1.3	LC	SDArch tools supports critical aspects of my job.
D1.4	LC	Using SDArch tools enables me to accomplish tasks more quickly.
D1.5	LC	Using SDArch tools reduces the time I spend on unproductive activities.
D1.6	LC	Using SDArch tools enhances my effectiveness on the job.
D1.7	LC	Using SDArch tools increases my productivity.
D1.8	LC	Using SDArch tools improves the quality of the work that I do.
D1.9	LC	Using SDArch tools makes it easier to do my job.
Ease of Use		
D2.1	LC	SDArch tools are user friendly.
D2.2	LC	Using SDArch tools is effortless.
D2.3	LC	SDArch tools are intuitive enough, so that I can use them without relying on written instructions.
D2.4	LC	I find it easy to get SDArch tools to do what I want them to do.
D2.5	LC	Interaction with SDArch tools is clear and understandable.
D2.6	LC	I do not notice any inconsistencies as I use SDArch tools.
D2.7	LC	I can recover from mistakes in SDArch tools quickly and easily.
D2.8	LC	It is easy for me to remember how to perform tasks using SDArch tools.
D2.9	LC	I think that both occasional and regular users of SDArch tools would like to use them.
D2.10	LC	I quickly became skillful with SDArch tools.
Ease of Learning		
D3.1	LC	I learned to use SDArch tools quickly.
D3.2	LC	I easily remembered how to use SDArch tools.
D3.3	LC	Learning to operate SDArch tools was easy for me.
Overall Satisfaction		
D4.1	LC	I am satisfied with SDArch tools.
D4.2	LC	I feel I need to have SDArch tools installed on my computer.
D4.3	LC	I would recommend SDArch tools to a friend.
D4.4	LC	It is fun to use SDArch tools.

Having the questionnaire results collected, I first analyzed internal consistency (reliability) of the statements used for each of the considered user satisfaction dimension. In statistics, internal consistency is a measure based on the correlations between different items that measure the same concept. It actually measures whether several items that measure the same concept produce similar scores. Cronbach's α is a statistic which is commonly used as a measure of the internal consistency or reliability of a psychometric test score [25]. It can take on any value less than or equal to 1. Recommended values of α are those higher than 0.7 [24]. Table 7.19 shows the values of Crombach's α for each of the four considered user satisfaction dimension. As can be seen from the table, the questionnaire's results showed high internal consistency of all considered user satisfaction dimension, as indicated by high α values, which exceed the recommended level of 0.70.

Satisfaction Dimension	Cronbach's α
Usefulness	0.85
Ease-of-Use	0.78
Ease-of-Learning	0.92
Overall Satisfaction	0.83

Table 7.19. Internal consistency (reliability) of considered user satisfaction dimensions

The results of the user satisfaction questionnaire are shown in Table 7.20. Regarding the system usefulness (D1.1 - D1.9), the average ratings range from 3.42 to 4.57, where 3.42 is the rating of the statement D1.3 ('SDArch tools support critical aspects of my job') and 4.57 is the rating of the statement D1.4 ('Using SDArch tools enable me to accomplish tasks more quickly'). The statement D1.4 has also scored the most of 'Strongly Agree' answers (64.3%) among the statements used for the evaluation of the system usefulness. Regarding the system ease-of-use (D2.1 - D2.10), the average ratings are in the range of 3.64 to 4.71, where 3.64 is the rating of the statement D2.3 ('SDArch tools are intuitive enough, so that I can use them without relying on written instructions') and 4.71 is the rating of the statement D2.8 ('It is easy for me to remember how to perform tasks using SDArch tools'), which is also the statement with the most 'Strongly Agree' answers (71.4%). Regarding the system ease-of-learning, which was evaluated by the thee statements (D3.1 - D3.3), the statement D3.3 ('I learned to use SDArch tools quickly') scored the lowest average rating (4.42) and the statement D3.2 ('I easily remembered how to use SDArch tools') scored the highest average rating (4.57). Finally, regarding the overall satisfaction with the system (D4.1 - D4.4), the first two statements ('I am satisfied with SDArch tools' and 'I feel I need to have SDArch tools installed on my computer') are the lowest rated statement (4.21),while the third statement D4.3 ('I would recommend SDArch tools to a friend') is the highest rated statement (4.5).

Table 7.20. Results of the user satisfaction questionnaire

Id	Strongly Disagree	Disagree Somewhat	Uncert.	Agree Somewhat	Strongly Agree	Response Average	SD
Usefulness							
D1.1	0.0%	0.0%	7.1%	**57.1%**	35.7%	4.28	0.61
D1.2	0.0%	0.0%	21.4%	**57.1%**	21.4%	4	0.67
D1.3	0.0%	0.0%	**71.4%**	14.3%	14.3%	3.42	0.75
D1.4	0.0%	0.0%	7.1%	28.6%	**64.3%**	4.57	0.91
D1.5	0.0%	0.0%	14.3%	**42.9%**	**42.9%**	4.28	0.72
D1.6	0.0%	0.0%	14.3%	**50.0%**	35.7%	4.21	0.69
D1.7	0.0%	0.0%	7.1%	**57.1%**	35.7%	4.28	0.61
D1.8	0.0%	7.7%	23.1%	30.8%	**38.5%**	4	0.96
D1.9	0.0%	7.1%	7.1%	**42.9%**	**42.9%**	4.21	0.89
Ease of Use							
D2.1	0.0%	7.1%	14.3%	**64.3%**	14.3%	3.85	0.51
D2.2	7.1%	0.0%	7.1%	**64.3%**	21.4%	3.92	0.77
D2.3	0.0%	21.4%	14.3%	**42.9%**	21.4%	3.64	0.99
D2.4	0.0%	0.0%	0.0%	**76.9%**	23.1%	4.21	1.08
D2.5	0.0%	7.1%	7.1%	**57.1%**	28.6%	4.07	0.42
D2.6	7.1%	0.0%	28.6%	**35.7%**	28.6%	3.78	0.82
D2.7	0.0%	0.0%	35.7%	**50.0%**	14.3%	3.78	1.12
D2.8	0.0%	0.0%	0.0%	28.6%	**71.4%**	4.71	0.69
D2.9	0.0%	**7.1%**	14.3%	57.1%	21.4%	3.92	0.46
D2.10	0.0%	0.0%	14.3%	**64.3%**	21.4%	4.07	0.82
Ease of Learning							
D3.1	0.0%	0.0%	7.1%	35.7%	**57.1%**	4.5	0.61
D3.2	0.0%	0.0%	7.1%	28.6%	**64.3%**	4.57	0.65
D3.3	0.0%	7.1%	7.1%	21.4%	**64.3%**	4.42	0.64
Overall Satisfaction							
D4.1	0.0%	0.0%	0.0%	**76.9%**	23.1%	4.21	0.93
D4.2	0.0%	0.0%	15.4%	**46.2%**	38.5%	4.21	0.42
D4.3	0.0%	0.0%	0.0%	46.2%	**53.8%**	4.5	0.69
D4.4	0.0%	0.0%	15.4%	30.8%	**53.8%**	4.35	0.51

In order to compare the considered user satisfaction dimensions between themselves, I calculated the average rating of each of them by averaging all individual answers of their corresponding statements. Figure 7.11 exhibits a graphical representation

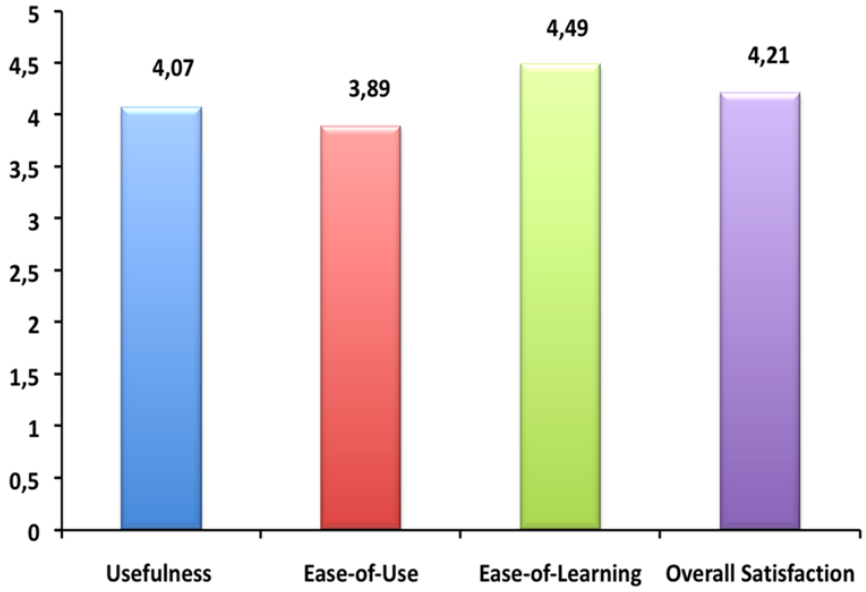


Figure 7.11. Average ratings of the considered user satisfaction dimensions

of the overall ratings of the considered user satisfaction dimensions.

➥ ***User Satisfaction - Evaluation Outcome***
As can be seen from Figure 7.11, the user satisfaction with the SDArch system (i.e., services and tools), with respect to system usefulness, ease of use, ease of learning, and overall satisfaction was highly positive. Among the considered dimensions, ease-of-learning was the highest rated dimension.

7.3 Summary

In this chapter, I presented the results of the two evaluation studies that I conducted in order to validate the thesis statement. These two studies were actually designed to answer on the two research questions that steam from the thesis statement.

The main goal of the first evaluation study, namely, the evaluation of semantic document information retrieval (Section 7.1), was to measure the effectiveness of the semantic document search(Section 5.2.1) in terms of precision and recall, and to compare it with the related concept-based searches and the full-text search. For both test collections (i.e., 'Mammals of the World' and 'Metals and their Alloys') used in this evaluation study, the experimental results showed that the semantic document search outperformed from the compared search approaches. Moreover, in this study I also evaluated the semantic document annotation approach (Section 5.1.2) by considering the amount and quality

of generated semantic annotations. In case of both test collections, the experimental results showed that the proposed semantic annotation approach increased the amount of generated semantic annotations, without loosing too much on the annotations' relevance. Finally, this evaluation study was used to determine optimal values of the generic parameters (SD_c, PL_c, and β) of the CEA algorithm for both test collections, thus validating my hypotheses that there exist optimal values of these parameters for each semantic document collection.

In the second evaluation study, namely, the usability evaluation of the SDArch services and tools (Section 7.2), my goal was to evaluate the user *effectiveness*, *efficiency*, and *satisfaction* with the SDArch services and tools. The usability evaluation was conducted through the case study, with a total of 18 participants. The case study included the creation of the SDArch social network, the acquisition of the shared collection of semantic documents, and the evaluation session with the task-based usability test and the follow-up questionnaires. The usability test was composed of three use cases containing tasks designed to familiarize the participants with most of the SDArch supported processes. The evaluation of the user *effectiveness* and *efficiency* was achieved by objective quantitative measures of the participants' performance in executing the tasks of the usability test, once by using the SDArch system and once by using the conventional system. The results of these measures showed that the participants were significantly more effective and efficient in executing the tasks by using the SDArch system than by using the conventional one. Regarding the user *satisfaction* with the SDArch system (i.e., services and tools), the participants were asked to fill in the user-satisfaction questionnaire, which was concerning the four dimensions of the user satisfaction: the system usefulness, the system ease-of-use, the system ease-of-learning, and the user overall satisfaction with the system. All questions in the questionnaire were defined as positive statements, and the five-level Likert scale was applied for their rating. The results of the questionnaire showed that the participants' satisfaction with the SDArch services and tools were highly satisfying with respect to all four considered dimensions. However, the system ease-of-learning scored the highest overall rating among the four dimensions, while the overall rating of the system ease-of-use was the lowest.

Chapter 8

Conclusions

In this thesis, I proposed a new desktop document architecture providing a new form of documents and a supporting software architecture. The new document architecture aims to improve the interoperability and integration of desktop data managed by different applications, and to enable integration of desktop data into a unified information space of social network communities [114]. It is also in line with the vision of the Social Semantic Desktop, and represents a significant contribution on the way to its realization. The actual architecture's design, which I presented in this thesis, includes the new document representation model, the document storage infrastructure for new documents, the services that realize processes enabled by new documents, and the tools that enable users to interact with the services. In addition, the thesis also offers solutions for the user modeling and modeling of a social network that is enabled by the new document architecture.

8.1 Contributions

By introducing the new document model and designing the supporting software architecture, this thesis made several contributions to the state of the art in document engineering and management. In this section, I summarize the main contributions:

- **Introducing the Semantic Document Model - SDM**. The existing document annotation models attempt to enhance conventional documents by adding an additional semantic layer, which provides semantic descriptions that refer to the documents' data. In contrary, SDM integrates the semantic layer into the core of the document representation structures. It provides a globally unique identification of document units of a different level of granularity, enables the semantic annotation of document units by ontology-based conceptualized semantics, and provides structures for establishing explicit semantic links between semantically related document units. The explicit semantic links can also be established between a document unit of a semantic docu-

ment and other, uniquely identified resources whether they are located on a personal desktop or on the Web.

- **Designing the Semantic Document Architecture - SDArch**. In order to support semantic document management and to enable users to take benefit from semantic documents, I designed the supporting software architecture called the semantic document architecture (SDArch). It is a three-tier, service-oriented software architecture composed of the data layer that contains the semantic document repository equipped with the concept and full-text indexes, then the service-oriented middleware, and the presentation layer that contains the SDArch user interface. The service-oriented middleware provides the service registry and defines the communication protocol among the SDArch services and between the SDArch services and the user interface. The actual SDArch functionalities are encapsulated into five services: the semantic document authoring, the semantic document search and navigation, the user profile management, the social network management, and the ontology management services. Potential new functionalities can be added to SDArch by registering new services into the SDArch middleware.

- **Specifying the Semantic Document Management Processes**. I specified and provided the detailed descriptions of the following top-level semantic document management processes: the semantic document authoring, the semantic document search, and the semantic document navigation. The semantic document authoring is based on the transformation of conventional desktop documents by utilizing appropriate domain ontologies and includes the generation document units' RDF descriptions, their semantic annotation, indexing and linking. The semantic document search includes the transformation of an initial, free-text user query into a concept-based, semantic query and the execution of the semantic query against the concept index of the SDArch semantic document repository. The semantic document authoring and the semantic document search are both based on the utilization of ontology-based conceptualized semantics extracted from document units and the free-text user query, respectively. Finally, the semantic document navigation is realized by executing the navigational queries against the RDF descriptions of semantic links between document units. The navigational queries are generated by an adequate exploratory user interface, through which the user interacts with semantic documents.

- **Modeling the Social Network of SDArch Users**. In order to capture social interaction between SDArch users who share their semantic documents, I introduced and modeled the SDArch social network. The SDArch social network is enabled by the social network management service that provides methods for the creation and management of the SDArch social network as well as the methods for capturing the user actions performed to shared semantic document units (e.g., document unit browsing, editing, and reusing), thus generating the document units' social-context annotations.

The personalized ranking algorithm, which I developed, utilizes the social-context annotations of semantic document units along with the user's preferences specified in the user profile to personalize the semantic document search.

- **Providing the SDArch Prototype**. I implemented the SDArch prototype as a fully-functional software providing the implementation of all the intended SDArch functionalities. The prototype was used for the thesis validation through the experimental evaluation of the semantic document information retrieval and the usability evaluation of the SDArch services and tools.

- **Evaluating the Semantic Document Information Retrieval**. In this evaluation study, which included the set of experiments conducted on the two test collections, I evaluated the effectiveness of the semantic document search by comparing it with related concept-based searches and the full-text search. In case of both test collections, the experimental results showed that the semantic document search outperformed from the compared search approaches, in terms of both better precision and better recall.

- **Evaluating the Usability of the SDArch Services and Tools**. In this evaluation study, I evaluated the user effectiveness, efficiency and satisfaction in using the SDArch services and tools. The applied evaluation approach included objective quantitative measures concerning the user effectiveness and efficiency and the user subjective feedback concerning the user satisfaction. The results obtained through the applied quantitative measures showed that the SDArch users were significantly more effective and efficient than the users using the conventional system while executing the same set of evaluation tasks. The results obtained through the subjective feedback (i.e., questionnaires) showed that the user satisfaction with the SDArch services and tools was highly positive with respect to all considered user satisfaction's dimensions (i.e., usefulness, ease-of-use, ease-of-learning, and overall satisfaction).

8.2 Open Issues and Future Directions

The research conducted during the realization of this thesis has opened several issues, which were not addressed in the thesis, but which deserve future investigation. I also encountered several possible continuation paths for the work presented in this thesis.

I start this discussion by outlining the open issues concerning the introduced semantic document model that the semantic document architecture is founded on.

- **Binary content's decoupling from RDF**. In the semantic document model, binary contents of document units are modeled as instances of the `smd:DataObject` class. This class provides the `sdm:contentStream` property which holds an atomic unit of binary content as a value of the `xsd:base64Binary` datatyped literal. Although this datatyped literal is supported in RDF, the existing RDF repositories are not meant

to store larger chunks of binary content. When a large amount of data has to be managed, structured queries (i.e., SPARQL queries) executed against RDF data are not always powerful enough. Accordingly, in the current SDArch prototype implementation I keep binary contents of semantic document units decoupled from their RDF representations. Binary contents are placed into appropriate binary data files and stored into the binary data repository. The binary data files are linked to RDF instances of the `sdm:DetaObject` class, and thus included into semantic documents. However, maintaining the links between binary contents and the document's RDF representation requires significant resources. Broken links can cause the lost of some information.

- **Versioning of semantic document units**. Semantic document units change and evolve through a number of versions over time. In order to verify the evolution path of semantic document units, the semantic document model provides the document unit version identifiers in addition to the document unit URIs. Moreover, the model provides appropriate structures for the formal representation of possible changes to semantic document units. Any change made to a document unit creates a new version of the document unit. While changes made to composite document units (i.e., reordering, adding or subtracting some of their constitutive atomic units) can be captured at the level of their RDF representations, changes made to atomic document units, which are also changes to their corresponding data objects, require replication of the underlying binary data files. In case of a document unit revision, when the new version of the document unit is meant to replace the previous one, the binary content of the previous document unit version can be deleted. In case of a document unit variant, which is meant to coexist with the previous document unit version, binary contents of the both versions must be present in the binary data repository. This could potentially lead to the overload of the storage resources.

- **Privacy and security of semantic document units**. Semantic documents are designed to be completely open resources, composed of uniquely identified document units that can be accessed and reused by everyone. Contrary to conventional documents, where reusing a document unit (i.e., copy and paste) means also loosing the document unit's proprietary data (unless it is explicitly cited in a new document), by reusing a semantic document unit (i.e. by linking the document unit's RDF node to the RDF representation of a new document) the document unit's proprietary data remains preserved. However, the actual semantic document model does not provide mechanisms for securing (restricting) access to document units nor mechanisms for protecting document units from unauthorized changing of their contents. Solutions for these issues should be in tune with 'privacy' and 'security' principles that are being applied to the Web of linked data.

Regarding the semantic document architecture, I first discuss possible enhancements of the storage of shared semantic documents, then I outline open issues concerning the

actual architecture's services, and at the end, I envision some new services and tools that would further improve the effectiveness and efficiency of desktop users in carrying out their tasks.

- **Decentralized RDF storage**. RDF representations of all semantic documents that exist on a local desktop are stored in the same RDF repository. Therefore, desktop semantic documents are a part of the unified desktop RDF space (i.e., graph). Sharing semantic documents means publishing RDF document representations to the shared RDF space (repository), while document binary contents remain stored in the local binary data repository. The actual SDArch RDF repository that is used for shared semantic documents is designed as a centralized repository. Although local SDArch RDF repositories expose remotely accessible SPARQL end-points, decentralized storage of shared semantic documents requires a more comprehensive solution. A peer-to-peer infrastructure seams as a promising solution to this problem.

- **Enhancing Knowledge Extraction and Conceptualization**. The knowledge extraction and conceptualization is a fundamental process for most of the semantic document management processes (i.e., semantic annotation, indexing, linking, search and navigation). Accordingly, its enhancement would have a positive impact on the whole proposed architecture. Some open issues that I plan to work on in the future include: knowledge extraction and conceptualization from document units of image, audio, and video data types, automatic selection of domain ontology, and weighting ontological relations (i.e., determining the semantic distance between two concepts connected by a given ontological relation) in selected domain ontologies.

- **Enhancing semantic document search**. The semantic layer, which SDM integrates inside the semantic document representation, is composed of conceptualized semantics (i.e., weighted ontological concepts) that are linked to document units as their semantic annotations and explicate semantic links established between semantically related document units. The semantic document search currently employs only the conceptualized semantics, while the semantic links are employed in the semantic navigation process. It can be investigated whether the semantic document search can benefit from the semantic links as well.

- **Automatic assembly of semantic documents**. Semantic documents opened a way for a number of 'intelligent' software agents that would exploit document units' conceptualized semantics as well as semantic links between the document units to act on behalf of a user by performing some goal-oriented tasks. One scenario, which has been challenging for me since the beginning of my research on semantic documents, is the automatic assembly of semantic documents by reusing existing semantic document units. This scenario requires some new services that will be activated by a user specifying a topic of interest, and then by taking into account user profile information (i.e., a range of formally specified user preferences), document units' conceptualized

semantics, and the semantic links, they should automatically assemble a semantic document on the specified topic. I plan to investigate on these services and potentially integrate them into the SDArch middleware.

- **A full-function semantic document editor**. My strategy for the development of the actual SDArch user interface was to extend some of the existing document application suites (e.g., MS Office) by adding new tools. In this way, I allowed users, still working in familiar environment, to take advantage of the new document architecture and services. In the future work, I plan to develop a standalone SDArch application suite, which would include the semantic document editor and the semantic document browser with an integrated support for the semantic document search. While the semantic document browser should mostly rely on the existing semantic document browser that is currently integrated into MS Office, the semantic document editor will be a completely new application. As described in the thesis, the actual semantic document authoring scenario is based on the transformation of existing conventional documents into semantic documents. Among other functionalities, the semantic document editor should enable the semantic document authoring completely from scratch and provide users with a possibility to manually annotate and link semantic document units.

With respect to the modeling of the SDArch social network and the design of the SDArch social network management service, I highlight the following research challenge.

- **Integration with popular, existing SNSs**. The SDArch social network was designed so that it leverages semantics in management of both network users and their shared document units. By conforming to the Semantic Web principles in identifying and describing the network users and shared document units, we provided a foundation for semantic integration of the SDArch social network with some of the existing, popular social networks such as those managed by Facebook, MySpace, Tweeter, and LinkedIn social networking services. However, the integration of the SDArch social network with some of these social networks was not addressed in the thesis and represents a research challenge for the future.

In addition to the above listed open issues and research challenges motivated by this thesis, the ultimate task in the future would be a long-term usability evaluation of the proposed SDArch services and tools by empaling them in a real-world case study.

Appendix A

SDArch ontologies - Specification

A.1 SDM Ontology - the Core Part

A.1.1 Classes Description

DocumentUnit

Superclasses	rdfs:Resource
Subclasses	sdm:AtomicDocumentUnit, sdm:CompositeDocmentUnit
In domain of:	sdm:hasIdentifier, sdm:hasVersionIdentifier, sdm:unitType, sdm:isPartOf, sdm:hasAnnotation
In range of:	sdm:hasPart
Description	A uniquely identified unit of document content, which can be annotated by different kinds of annotation data and linked to any other uniquely identified resource on the Semantic Web.

AtomicDocumentUnit

Superclasses	sdm:DocumentUnit
In domain of:	sdm:hasDataObject
In range of:	sdm:addedAtonmicUnit, sdm:subtructedAtomicUnit
Description	A document unit which contains only one data object.

DataObject

Superclasses	rdfs:Resource
Subclasses	sdm:DiscreteDataObject, sdm:ContinuousDataObject
In domain of:	sdm:contentStream
In range of:	sdm:hasDataObject, sdm:oldDataObject, sdm:newDataObject
Description	A unit of row digital content, which can not be disaggregated into smaller units.

DiscreteDataObject

Superclasses	sdm:DataObject
Subclasses	sdm:TextFragment, sdm:Graphic
Description	A unit of digital content which is not a function of a continuous argument (time).

TextFragment

Superclasses	sdm:DiscreteDataObject
Description	A unit of digital content containing a plain text (an arbitrary number of sentences or words).

Graphic

Superclasses	sdm:DiscreteDataObject
Subclasses	sdm:Photograph, sdm:Drawing, sdm:Graph, sdm:Diagram, sdm:Chart, sdm:Symbol
Description	A unit of digital content containing a visual representation of some phenomena.

Photograph

Superclasses	sdm:Graphic
Description	A unit of digital content containing a digital image created by light falling on a light-sensitive surface, such as a CCD or a CMOS chip.

Drawing

Superclasses	sdm:Graphic
Description	A unit of digital content a digitalized drawing (visual art that involves marking a two-dimensional medium).

Graph

Superclasses	sdm:Graphic
Description	A unit of digital content containing a graphical representation of a function, in the form of a curve on a cartesian plane.

Diagram

Superclasses	sdm:Graphic
Description	A unit of digital content containing a two-dimensional geometric symbolic representation of information according to some visualization technique.

Chart

Superclasses	sdm:Graphic
Description	A unit of digital content containing a graphical representation of data, in which the data is represented by symbols, such as bars in a bar chart, lines in a line chart, or slices in a pie chart.

Symbol

Superclasses	sdm:Graphic
Description	A unit of digital content containing a particular mark that represents something else by association, resemblance, or convention.

ContinuousDataObject

Superclasses	sdm:DataObject
Subclasses	sdm:Audio, sdm:Video, sdm:Animation
Description	A unit of digital content which is a function of a continuous argument (time).

Audio

Superclasses	sdm:ContinuousDataObject
Description	A unit of digital content containing audio data.

Video

Superclasses	sdm:ContinuousDataObject
Description	A unit of digital content containing video data.

Animation

Superclasses	sdm:ContinuousDataObject
Description	A unit of digital content containing an animation - the rapid display of a sequence of images of 2-D or 3-D artwork or model positions in order to create an illusion of movement.

CompositeDocumentUnit

Superclasses	sdm:DocumentUnit
Subclasses	sdm:Document, sdm:Header, sdm:Footer, sdm:Chapter, sdm:Section, sdm:Paragraph, sdm:Title, sdm:Illustration, sdm:Table, sdm:TableHead, sdm:TableRow, sdm:TableColumn, sdm:TableCell, sdm:List, sdm:ListItem, sdm:Comment, sdm:Caption, sdm:Footnote, sdm:Endnote, sdm:Slide, sdm:SlideItem
In domain of:	sdm:addedAtonmicUnit, sdm:subtructedAtomicUnit
In range of:	–
Description	A unit of document content composed of a number of atomic document units, ordered in a specific order.

Document

Superclasses	sdm:CompositeDocumentUnit
In range of:	sdm:reusedIn
Description	A composite document unit which represents the whole document.

Header

Superclasses	sdm:CompositeDocumentUnit
Description	A composite document unit containing block of text, which appears at the top of each page of a human-readable instance of the document.

Footer

Superclasses	sdm:CompositeDocumentUnit
Description	A composite document unit containing block of text, which appears at the foot of each page of a human-readable instance of the document.

Chapter

Superclasses	sdm:CompositeDocumentUnit
Description	A composite document unit, which represents one of the main divisions of a document.

Section

Superclasses	sdm:CompositeDocumentUnit
Description	A composite document unit containing a portion of a document with a specific title.

Paragraph

Superclasses	sdm:CompositeDocumentUnit
Description	A composite document unit containing a self-contained unit of a discourse composed of one or more sentences (i.e., text fragments).

Title

Superclasses	sdm:CompositeDocumentUnit
Description	A document unit containing a name or a very short description of the following document portion.

Illustation

Superclasses	sdm:CompositeDocumentUnit
Description	A composite document unit containing graphics and text, created to elucidate or decorate some portions of a document.

Table

Superclasses	sdm:CompositeDocumentUnit
Description	A document unit containing a set of data (e.g., facts or figures) systematically arranged in columns and rows.

TableCell

Superclasses	sdm:CompositeDocumentUnit
Description	A document unit containing a grouping of data within a table, which belongs to the intersection of a column and a row.

TableColumn

Superclasses	sdm:CompositeDocumentUnit
Description	A document unit containing a vertical arrangement of a number of table cells.

TableRow

Superclasses	sdm:CompositeDocumentUnit
Description	A document unit containing a horizontal arrangement of a number of table cells.

TableHead

Superclasses	sdm:CompositeDocumentUnit
Description	A document unit containing a table name or a short table description, placed at the top of the table.

List

Superclasses	sdm:CompositeDocumentUnit
Description	A composite document unit containing an ordered collection of list items.

ListItem

Superclasses	sdm:CompositeDocumentUnit
Description	A document unit containing a grouping of data that belongs to the list.

Comment

Superclasses	sdm:CompositeDocumentUnit
Description	A document unit containing a written remark related to a document portion.

Caption

Superclasses	sdm:CompositeDocumentUnit
Description	A document unit containing a block of text that appears below the table, the illustration and the other graphics. Caption can be consisted of few words or several sentences.

Footnote

Superclasses	sdm:CompositeDocumentUnit
Description	A document unit containing a note of text placed at the bottom of a page in a document. Semantic documents do not have pages, but their human-readable instances do have. Footnotes are parts of the document units in which their marks should appear, when the human-readable instance is created.

Endnote

Superclasses	sdm:CompositeDocumentUnit
Description	A document unit containing a note of text placed at the end of a document.

Slide

Superclasses	sdm:CompositeDocumentUnit
Description	A document unit containing a number of slide items. In a human-readable instance of a semantic document, a slide content is presented as an individual page.

SlideItem

Superclasses	sdm:CompositeDocumentUnit
Description	A document unit containing a grouping of data that belongs to the slide.

A.1.2 Properties Description

hasIdentfier

Type	rdf:Property, rdfs:Resource
Domain	sdm:DataObject, sdm:DocumentUnit
Range	xsd:string
Superproperties	dc:identfier
Subproperties	–
Description	String used to uniquely identify the resource (i.e., data object and document unit).

hasVersionIdentfier

Type	rdf:Property, rdfs:Resource
Domain	sdm:DocumentUnit
Range	xsd:string
Description	String used to identify a version of a document unit.

isPartOf

Type	rdf:Property, rdfs:Resource
Domain	sdm:AtomicDocumentUnit
Range	sdm:CompositeDocumentUnit
Inverse property	sdm:hasPart
Description	Property used to express containment relationships between atomic and composite document units.

hasPart

Type	rdf:Property, rdfs:Resource
Domain	sdm:CompositeDocumentUnit
Range	sdm:AtomicDocumentUnit, sdm:CompositeDocumentUnit
Inverse property	sdm:isPartOf
Description	Property used to express containment relationships between composite and atomic document units.

hasDataObject

Type	rdf:Property, rdfs:Resource
Domain	sdm:AtomicDocumentUnit
Range	sdm:DataObject
Description	Property used to express containment relationships between atomic document units and document objects.

order

Type	rdf:Property, rdfs:Resource
Domain	sdm:CompositeDocumentUnit
Range	rdf:list
Description	Property used to express the order of the atomic document units within the composite document unit.

contentStream

Type	rdf:Property, rdfs:Resource
Domain	sdm:DataObject
Range	xsd:base64Binary, xsd:string
Description	Property used to represent a binary content of the data object. In case of small amount of binary content, the content can be coded as 'base64' (xsd:base64Binary), otherwise the value of the property is a string containing URL of the content.

unitType

Type	rdf:Property, rdfs:Resource
Domain	sdm:DocumentUnit
Range	xsd:string
Description	Property used to distinguish between document units of the 'Document' type from the other (atomic and composite) document unit types.

A.2 SDM Ontology - the Annotation Part

A.2.1 Classes Description

Annotation

Superclasses	rdfs:Resource
Subclasses	sdm:MetadataAnnotation, sdm: SemanticAnnotation, sdm:SocialContextAnnotation
In domain of:	sdm:annotationType
In range of:	sdm:hasAnnotation
Description	Superclass that represents the annotation of a document unit.

MetadataAnnotation

Superclasses	sdm:Annotation
In domain of:	sdm:creator, sdm:created, sdm:principleDocumentFormat, sdm:byteSize, sdm:lastModified, sdm:mimeType, sdm:generator, sdm:copyrigth
In range of:	–
Description	Class that represents the meta-data annotation of a document unit.

SemanticAnnotation

Superclasses	sdm:Annotation
In domain of:	sdm:ontologicalConcept, sdm:conceptWeight
In range of:	–
Description	Class that represents the semantic (ontological) annotation of a document unit.

SocialContextAnnotation

Superclasses	sdm:Annotation
In domain of:	sdm:userAction
In range of:	–
Description	Class that represents the social-context annotation of a document unit.

Action

Superclasses	rdfs:Resource
Subclasses	sdm:Browse, sdm:Reuse, sdm:Modification
In domain of:	sdm:actionPerformed, sdm:actionPerformer
In range of:	sdm:userAction
Description	Class that represents the user action that is performed to a document unit.

Browse

Superclasses	sdm:Action
In domain of:	sdm:browser
In range of:	–
Description	Class that represents the user action of browsing the document unit's data.

Reuse

Superclasses	sdm:Action
In domain of:	sdm:reusedIn
In range of:	–
Description	Class that represents the user action of reusing the document unit into a new document.

Modification

Superclasses	sdm:Action
In domain of:	sdm:unitChange, sdm:modifier
In range of:	–
Description	Class that represents the user action of the modification of a document unit.

A.2.2 Properties Description

hasAnnotation

Type	rdf:Property, rdfs:Resource
Domain	sdm:DocumentUnit
Range	sdm:Annotation
Description	Property that relates the annotation to the document unit.

annotationType

Type	rdf:Property, rdfs:Resource
Domain	sdm:Annotation
Range	xsd:string
Description	Property that defines the annotation type (i.e., metadata, semantic and social-context annotations).

ontologicalConcept

Type	rdf:Property, rdfs:Resource
Domain	sdm:semanticAnnotation
Range	rdfs:resource
Description	Property that link an ontological concept to the semantic annotation of a document unit.

conceptWeight

Type	rdf:Property, rdfs:Resource
Domain	sdm:SemanticAnnotation
Range	xsd:double
Description	Property that holds a value that determines the relevance of the semantic annotation (ontological concept) to the document unit it annotates.

userAction

Type	rdf:Property, rdfs:Resource
Domain	sdm:SocialContextAnnotation
Range	sdm:Action
Description	Property that relates the instance of the action class to the social-context annotation.

actionPerformed

Type	rdf:Property, rdfs:Resource
Domain	sdm:Action
Range	xsd:dateTime
Description	Property that holds the date and time of the user action, performed to the document unit.

actionPerformer

Type	rdf:Property, rdfs:Resource
Domain	sdm:Action
Range	umo:User
Description	Property that holds the information about the user who performed the action to the document unit. The user is represented as an instance of the umo:User class of the user model ontology (umo).

browser

Type	rdf:Property, rdfs:Resource
Domain	sdm:Browse
Range	xsd:string
Description	Software used for browsing the document unit's data.

reusedIn

Type	rdf:Property, rdfs:Resource
Domain	sdm:Reuse
Range	sdm:Document
Description	Property that holds a reference to the document in which the document unit has been reused.

unitChange

Type	rdf:Property, rdfs:Resource
Domain	sdm:Modification
Range	sdm:DocumentUnitChange
Description	Property that holds a reference to an instance of the sdm:DocumentUnitChange class of the change-tracking vocabulary.

modifier

Type	rdf:Property, rdfs:Resource
Domain	sdm:Modification
Range	xsd:string
Description	Software used for modifying the document unit.

creator

Type	rdf:Property, rdfs:Resource
Domain	sdm:MetadataAnnotation
Range	umo:User
Superproperties	dc:creator
Description	Person who created the document unit.

created

Type	rdf:Property, rdfs:Resource
Domain	sdm:MetadataAnnotation
Range	xsd:dateTime
Subproperties	dc:date
Description	Data and time when the document unit has been created.

principleDocumentFormat

Type	rdf:Property, rdfs:Resource
Domain	sdm:MetadataAnnotation
Range	xsd:string
Description	Document format in which the human-readable representation of the document unit is implemented.

byteSize

Type	rdf:Property, rdfs:Resource
Domain	sdm:MetadataAnnotation
Range	xsd:integer
Description	Size of the document unit.

lastModified

Type	rdf:Property, rdfs:Resource
Domain	sdm:MetadataAnnotation
Range	xsd:dateTime
Superproperties	dc:date
Description	Date and time of the last modification of the document unit.

copyright

Type	rdf:Property, rdfs:Resource
Domain	sdm:MetadataAnnotation
Range	xsd:string
Superproperties	dcterms:accessRights
Description	Document unit copyright.

mimeType

Type	rdf:Property, rdfs:Resource
Domain	sdm:MetadataAnnotation
Range	xsd:string
Description	Content type of the document unit's content.

generator

Type	rdf:Property, rdfs:Resource
Domain	sdm:MetadataAnnotation
Range	xsd:string
Description	Software that is used to generate the document unit.

A.3 SDM Ontology - the Semantic-Linking Part

A.3.1 Classes Description

SemanticLink

Superclasses	rdfs:resource
In domain of:	sdm:linkedUnitOne, sdm:linkedUnitTwo, sdm:linkingConcept, sdm:linkStrenght
In range of:	–
Description	Class that represents a semantic relationship between two document units, either they belong to the same or different documents.

A.3.2 Properties Description

linkedUnitOne

Type	rdf:Property, rdfs:Resource
Domain	sdm:SmenaticLink
Range	sdm:DocumentUnit
Description	Property that holds a reference to the first of the two document units linked by the semantic link.

linkedUnitTwo

Type	rdf:Property, rdfs:Resource
Domain	sdm:SemanticLink
Range	sdm:DocumentUnit
Description	Property that holds a reference to the second of the two document units linked by the semantic link.

linkingConcept

Type	rdf:Property, rdfs:Resource
Domain	sdm:SemanticLink
Range	rdfs:resource
Description	Property that holds a reference to the ontological concept, which conceptualizes shared semantics by the linked document units.

linkStrength

Type	rdf:Property, rdfs:Resource
Domain	sdm:SemanticLink
Range	xsd:double
Description	Property that holds a numerical value, which determines semantic relatedness between the document units linked by the semantic link.

A.4 SDM Ontology - the Change-Tracking Part

A.4.1 Classes Description

DocumentUnitChange

Superclasses	rdfs:resource
Subclasses	sdm:AtomicUnitChange, sdm:CompositeUnitChange
In domain of:	sdm:relatedTo
In range of:	–
Description	Superclass that describes possible changes to document units.

AtomicUnitChange

Superclasses	sdm:DocumentUnitChange
In domain of:	sdm:oldDataObject, sdm:newDataObject
In range of:	sdm:hasAtomicUnitChange
Description	Class describing possible changes to atomic document units.

CompositeUnitChange

Superclasses	sdm:DocumentUnitChange
In domain of:	sdm:hasAtomicUnitChange, sdm:addedAtomicUnit, sdm:subtructedAtomicUnit, sdm:oldUnitsOrder, sdm:newUnitsOrder
In range of:	–
Description	Class describing possible changes to composite document units.

A.4.2 Properties Description

relatedTo

Type	rdf:Property, rdfs:Resource
Domain	sdm:DocumentUnitChange
Range	sdm:DocumentUnit
Description	Property used to relate instances of sdm:DocumentUnitChange class to the document units they belong to.

oldDataObject

Type	rdf:Property, rdfs:Resource
Domain	sdm:AtomicUnitChange
Range	sdm:DataObject
Description	Property that holds a reference to the old data object of the modified, atomic document unit.

newDataObject

Type	rdf:Property, rdfs:Resource
Domain	sdm:AtomicUnitChange
Range	sdm:DataObject
Description	Property that holds a reference to the new data object of the modified, atomic document unit.

hasAtomicUnitChange

Type	rdf:Property, rdfs:Resource
Domain	sdm:CompositeUnitChange
Range	sdm:AtomicUnitChange
Description	Property that holds an instance of the sdm:AtomicUnitChange class, which is a part of the instance of the sdm:CompositeUnitChange class.

addedAtonmicUnit

Type	rdf:Property, rdfs:Resource
Domain	sdm:CompositeUnitChange
Range	sdm:AtomicDocumentUnit
Description	Property that holds a reference to an atomic document unit, which is added to the composite document unit.

subtractedAtomicUnit

Type	rdf:Property, rdfs:Resource
Domain	sdm:CompositeUnitChange
Range	sdm:AtomicDocumentUnit
Description	Property that holds a reference to an atomic document unit, which is subtracted from to the composite document unit.

oldUnitsOrder

Type	rdf:Property, rdfs:Resource
Domain	sdm:CompositeUnitChange
Range	rdf:list
Description	Property that holds the rdf:list, defining the old order of the atomic document units within the modified, composite document unit.

newUnitsOrder

Type	rdf:Property, rdfs:Resource
Domain	sdm:CompositeUnitChange
Range	rdf:list
Description	Property that holds the rdf:list, defining the new order of the atomic document units within the modified, composite document unit.

A.5 User-Model Ontology

A.5.1 Classes Description

User

Superclasses	foaf:Person
In domain of:	umo:isExpertIn, umo:isInterestedIn, umo:isComunityExpertIn, umo:hasOpenId, umo:hasPreference, sno:publishedBy
In range of:	–
Description	Class that represents a user of SDArch.

Preference

In domain of:	umo:hasID, umo:hasLabel, umo:hasImportance, umo:hasNumValue, umo:hasEnumValue
In range of:	umo:hasPreference
Description	Class introduced to specify the userÕs preferences regarding the choice of document units to be reused.

A.5.2 Properties Description

isInterestedIn

Type	rdf:Property, rdfs:Resource
Domain	umo:User
Range	sno:Topic
Description	This property holds a reference to a topic that the user is interested in.

isComunityExpertIn

Type	rdf:Property, rdfs:Resource
Domain	umo:User
Range	sno:Topic
Description	This property holds a reference to a topic that is determined to be a part of the userÕs expertise based on the amount of his document contents which has been reused by other members from the same SDArch social network.

isExpertIn

Type	rdf:Property, rdfs:Resource
Domain	umo:User
Range	sno:Topic
Description	This property holds a reference to a topic that the user self-asserts that he is an expert in.

isMemeberOf

Type	rdf:Property, rdfs:Resource
Domain	umo:User
Range	sno:Group
Subproperties	
Description	This property holds a reference to a user group within the SDArch social network that the user is member of.

hasOpenId

Type	rdf:Property, rdfs:Resource
Domain	umo:User
Range	xsd:string
Description	This property holds an OpenId that is used to uniquely identifies the SDArch user.

hasPreference

Type	rdf:Property, rdfs:Resource
Domain	umo:User
Range	umo:Preference
Description	This preference holds an instance of the umo:Preference class.

hasID

Type	rdf:Property, rdfs:Resource
Domain	umo:Preference
Range	xsd:string
Description	This property holds the identifier of the user preference (i.e., the instance of the umo:Preference class).

hasLabel

Type	rdf:Property, rdfs:Resource
Domain	umo:Preference
Range	xsd:string
Description	This property holds the label of the user preference (i.e., the instance of the umo:Preference class).

hasImportance

Type	rdf:Property, rdfs:Resource
Domain	umo:Preference
Range	xsd:double
Description	This property holds the value that determines the importance of the user preference (i.e., the instance of the umo:Preference class) for the user.

hasNumValue

Type	rdf:Property, rdfs:Resource
Domain	umo:Preference
Range	xsd:double
Description	This property holds the preferenceÕs numerical value.

hasEnumValue

Type	rdf:Property, rdfs:Resource
Domain	umo:Preference
Range	rdf:List
Description	This property holds the preferenceÕs enumerated value.

A.6 Social Network Ontology

A.6.1 Classes Description

Group

Superclasses	
In domain of:	sno:hasForum, sno:hasTopic, sno:createBy, sno:hasMember, sno:hasDocument
In range of:	umo:isMemberOf
Description	This class models a group of SDArch users who are organized around a given topic of interest.

Topic

Superclasses	
In domain of:	
In range of:	sno:hasTopic, umo:isExpertIn, umo:isInterestedIn, umo:isComunityExpertIn
Description	The groupÕs topic of interest is specified by an instance of this class.

Forum

Superclasses	
In domain of:	sno:hasPost
In range of:	sno:hasForum
Description	Class that specifies an on line-user forum of a user group.

Post

Superclasses	
In domain of:	
In range of:	sno:postedBy, sno:hasPost
Description	Class that specifies a user post on the group's forum.

A.6.2 Properties Description

hasTopic

Type	rdf:Property, rdfs:Resource
Domain	sno:Group
Range	sno:Topic
Description	This property holds a reference to a group's topic (i.e., an instance of the sno:Topic class).

createdBy

Type	rdf:Property, rdfs:Resource
Domain	sno:Group
Range	umo:User
Description	This property holds a reference to a user who created the user group.

hasMember

Type	rdf:Property, rdfs:Resource
Domain	sno:Group
Range	umo:User
Description	This property holds a reference to a user who is a member of the user group.

hasForum

Type	rdf:Property, rdfs:Resource
Domain	sno:Group
Range	sno:Forum
Description	This property holds a reference to a forum (i.e., an instance of sno:Forum class) of the user group.

hasPost

Type	rdf:Property, rdfs:Resource
Domain	sno:Forum
Range	sno:Post
Description	This property holds a reference to a post (i.e., an instance of the sno:Post class) of the group's forum.

postedBy

Type	rdf:Property, rdfs:Resource
Domain	umo:User
Range	sno:Post
Description	This property holds a reference to a user who wrote a post on the group's forum.

hasDocument

Type	rdf:Property, rdfs:Resource
Domain	sno:Group
Range	sdm:Document
Description	This property holds a reference to a document that belongs to the user group.

publishedBy

Type	rdf:Property, rdfs:Resource
Domain	umo:User
Range	sdm:Document
Description	This property holds a reference to a user who published the document to the user group's document repository.

Appendix B

Evaluation Resources

B.1 Summary of the Formative Evaluation

The formative evaluation was conducted by the alpha release of the SDArch prototype, whose development was finished at the beginning of 2009. That prototype release was feature complete. It deferred from the beta release, which was used in the main usability evaluation study, only in design of some user interface elements of the SDarch tools, which I redesigned according to the user feedback obtained in the formative evaluation. Moreover, a couple of failures in the implementation of the SDArch services were discovered during the formative evaluation and they were resolved for the beta release.

Besides identifying missing functionalities of the user interface and discovering possible failures in the prototype implementation, the formative evaluation was intended to reveal deficiencies in the evaluation design and to assess the feasibility of the proposed evaluation procedure. The same as the summative evaluation, the formative evaluation considered the user *effectiveness*, *efficiency* and *satisfaction* in using the SDArch services and tools.

A total of six participants, from two universities, took a part in the study. A motivational scenario of a usability test that was conducted in the formative evaluation was the same as in the summative evaluation, that is, the authoring of the course material. However, the usability test used in the formative evaluation involved only one use case that corresponds to some extent to the third use case of the summative evaluation's usability test (Section 7.2.6). The intended, resulting PowerPoint presentation of the use case was supposed to contain seven slides, entitled as: 1) *Introduction*, 2) *Role of Design Patterns*, 3) *Design Patterns Definition*, 4) *Design Patterns Classification*, 5) *Pattern Example 1*, 6) *Pattern Example 2*, and 7) *Conclusions*. The creation of each slide was considered as one task of the usability test. The evaluation document collection consisted of 50 documents related to the software design patterns. The documents were available in their original formats (i.e., Word andPowerPoint) and as semantic documents stored in the shared semantic document repository.

The evaluation session involved of the usability test and the follow-up questionnaire. The participants were asked to perform the tasks of the usability test two times, in two continuous but unlimited time sessions, once using the system with conventional system and once using the SDArch system. Moreover, the participants were divided in two control groups using the systems in the opposite order. The screen activities of the participants, were recorded by a screen recorder. All participants performed the tasks on our PC-server using remote desktop connection software.

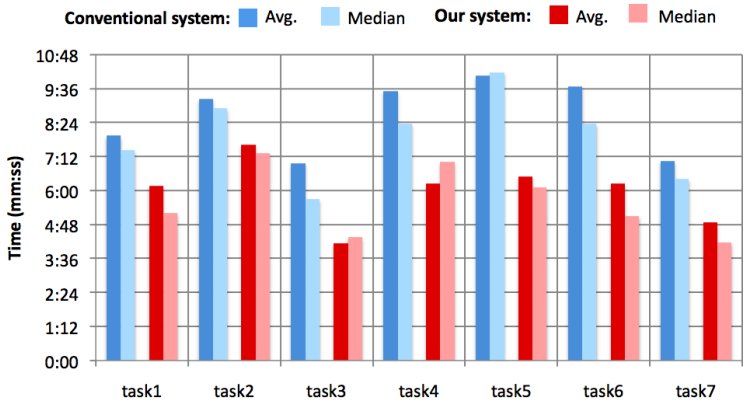


Figure B.1. Avgerage and median task execution times

With respect to the user effectiveness, I considered how many and which tasks the participants completed successfully. Results showed that all the participants successfully completed all seven tasks using both systems. It was mostly because of unlimited time sessions and the genuine motivation of the participants. With respect to the user efficiency, I considered the execution times, the number of mouse clicks and the number of window switches during the tasks' executions. Figure B.1 presents the average and median execution times, for each task for both systems. As can be seen from the figure, with respect to task execution times, the SDArch system outperformed from the conventional system for all the tasks.

Table B.1 shows the relative performance of the participants when using the SDArch system with respect to the use of the conventional system. For example, a value of 87.1% indicates that participants using the SDArch system needed 87.1% of the mouse clicks that participants using the conventional system needed. All values from the table are less than 100%, which means that the performance of the participants regarding both number of mouse clicks and window switches are better when using the SDArch system. The measured execution times, number of mouse clicks and number of window switches indicated that the efficiency of the participants when using the SDArch system was better than their efficiency when using the conventional system, with respect to these applied metrics.

Regarding the user satisfaction, I considered the two factors: a system usefulness

Task	Rel. performance for (mouse cliks)		Rel. performance for (window switches)	
	Average	Median	Average	Median
1	87.1%	86.9%	21.5%	33.3%
2	79.6%	86.4%	35.8%	26.6%
3	75.3%	68.8%	24.0%	29.4%
4	78.9%	83.8%	28.4%	27.7%
5	51.1%	53.3%	25.6%	21.0%
6	57.1%	56.6%	28.1%	26.3%
7	80.4%	86.2%	21.6%	28.5%

Table B.1. Relative user performance when using the SDArch system with respect to the conventional system

and a system ease-of-use. The follow-up questionnaire contained the 9 positive statements:

S1: Using SemanticDoc enables me to accomplish tasks more quickly;
S2: Using SemanticDoc increases my productivity;
S3: Using SemanticDoc improves the quality of the work I do;
S4: Using SemanticDoc makes it easier to do my work;
S5: Overall, I find SemanticDoc useful in my work;
S6: Learning to operate SemanticDoc is easy for me;
S7: I find it easy to get SemanticDoc to do what I want it to do;
S8: Interaction with SemanticDoc is clear and understandable;
S9: Overall, I find SemanticDoc easy to use;

The first 5 statements from S1 to S5 were to gather a subjective user feedback on the system usefulness. The following four statements from S6 to S9 were to gather to gather a subjective user feedback on the ease-of-use of the system. The participants were supposed to rate each of the statements using 5-level Likert scale, starting from 1 (strongly disagree) to 5 (strongly agree). As Table B.2 shows, statements S5 and S9, which actually expressed the overall satisfaction regarding the usefulness and ease-of-use respectively, were the two highest-rated statements with an average rating of 4.8 out of 5. The other statements were also rated as highly positive, with average ratings range from 4.1 to 4.7. In addition to the questionnaire, I did a short, verbal interview with each participant aiming to gather their suggestions of how to further improve the user interface of the SDArch tools.

The formative evaluation, first of all, proved the feasibility of the usability test and the applicability of the chosen evaluation metrics (i.e., task execution times, number

Statement	S1	S2	S3	S4	S5	S6	S7	S8	S9
Average Rating	4.7	4.3	4.1	4.7	4.8	4.7	4.3	4.6	4.8
Median Rating	5	4	4	5	5	4	4	4	5

Table B.2. User satisfaction feedback

of mouse clicks, number of window switches). Then, it showed that the size of the evaluation document collection (50 documents) was sufficient for the kind of use case implemented in the usability test. Regarding the design of the usability test, I realized that there was some redundancy in the tasks (i.e., the presentation slides to be created). Accordingly, for the summative evaluation, the number of presentation slides to be created within the usability test was reduced to 3. Moreover, instead of considering the creation of a whole slide as one task, in the summative evaluation the tasks were tuned to the creation of the slide items (Section 7.2.6, Use Case 3). Finally, based on the feedback gathered through the interviews, I decided to redesign some of the user interface elements of the document recommender (Section 6.4.5) and the semantic document browser (Section 6.4.6) tools.

B.2 Entrance Questionnaire

1. Identify yourself as a:

☐ Master student

☐ PhD student

☐ Post-doc

☐ Professor

☐ Other (please specify) _____

2. How long have you been enrolled in the above selected role?

_____

3. Age Group:

☐ 18 - 24

☐ 25 - 34

☐ 35 - 44

☐ 45 - 55

4. Gender:

☐ Female

☐ Male

5. How often do you use MS Office (Word and PowerPoint)?

☐ Daily

☐ Weekly

☐ Monthly

☐ Occasionally

☐ Never

6. How many years of experience do you have in using MS Office or any other similar software (e.g. OpenOffice)?

☐ I have never used this kind of software

☐ less than 3 years

☐ 3 - 6 years

☐ 7 - 10 years

☐ more than 10 years

7. What purpose do you usually use MS Office for?

☐ Preparation of course materials

☐ Preparation of project documentations

☐ Preparation of scientific articles

☐ Preparation of business reports

☐ Other (please specify) _____

8. How familiar are you with Semantic Web technologies (e.g., Ontologies, OWL, RDF and SPARQL)?

- ☐ Never heard about them
- ☐ Have heard, but never tried to use them
- ☐ Have tried, but never really used them
- ☐ Have used and know well them
- ☐ Have developed some of them

9. What purpose do you usually use Semantic Web technologies for?

- ☐ Personal information management
- ☐ Search of a local desktop's (computer's) resources
- ☐ Annotation of a local desktop's (computer's) resources
- ☐ Annotation of the Web resources
- ☐ I do not use Semantic Web technologies
- ☐ Other (please specify) _____

B.3 The Usability Test's Use Cases: Step-by-Step Instructions

Here is the guideline for the usability test's execution:

- Entrance Questionnaire (9 questions)
- Use Case 1 - see the use case's description below
  - Questionnaire A
- Use Case 2 - see the use case's description below
  - Questionnaire B
- Use Case 3 Ð see the use case's description below
  - Questionnaire C
- Questionnaire D

B.3.1 Use Case 1: Setting-Up the User Profile and the Social Network

Objectives: using the SemanticDoc tools that enable users to manage their user profile and social networking data.

Instructions:

- Start MS Word (shortcut is available on the desktop);
- Go to 'SemanticDoc' menu tab;
- Log-In to the system (Account and Profile → Log in/out);
- Start the User Profile tool (Account and Profile → My Profile);
- Set the following profile data:
  - Personal Info
  - Social Network
  - Content preferences
- Close the User Profile tool;
- Start the Social Networking tool (Social Networking → Groups);
- Browse the details of the 'Design Patterns' group;
- Join the 'Design Patterns' group;
- Close the Social Networking tool;
- Leave MS Word active, minimize the RDC window, and complete the questionnaire related to task1 (Questionnaire A);

B.3.2 Use Case 2: Authoring and Publishing Semantic Documents

Objectives: using the SemanticDoc tools that enable users to transform MS Office documents (e.g., Word and PowerPoint) into semantic documents.

Instructions:

- Maximize the RDC windows;
- Open the existing word document called 'TransformationExample.docx' (Document Transformation → Open → É and then navigate to the document Ð document is saved on the desktop);

- Start the Transformation and Publishing tool (Document Transformation → Transform and Publish);
- Select the location where you want to store the transformed document (i.e., local or shared semantic document repository);
- Select the document's topic (in this case it should be 'Design Patterns');
- Choose an annotation ontology that will be used during the transformation (in this case it should be the 'Design Patterns' ontology); while choosing the ontology, browse the ontology details;
- Start the transform process;
- When the transformation is done, close the Transformation and Publishing tool and then close MS Word too;
- Minimize the RDC window;
- Complete the questionnaire related to task2 (Questionnaire B)

B.3.3 Use Case 3: Searching and Navigating across Semantic Documents

Objectives: using the SemanticDoc tools that enable semantic search and navigation across semantic documents.

- This task is about creating a short MS PowerPoint presentation (3 slides) by reusing data from a given document collection. The document collection is available in two forms: 1) as a collection of Word documents stored in the 'Evaluation Documents' folder on the desktop, and 2) as a collection of semantic documents stored in the shared semantic documents repository under the 'Design Patterns' topic.
- The task should be executed two times, once by using our SemanticDoc tools and once by using only conventional MS PowerPoint functionalities. The order of the executions will be specified prior to the evaluation session.
- Participant's activities during the taskÕs execution will be recorded (for both executions) by using Camtasia screen recorder. Instructions of how to start recording, stop recording and save the reordered material, will be provided to the participant prior to the evaluation session.

Instructions for executing this task by using the SemanticDoc tools:

- Open a PowerPoint document called 'SemanticDocPresentation.pptx', which is saved on the desktop and then start the recording;
- Check what missing information/data you should put on the slides (slide 2, 3, and 4);
- Start the Recommender tool (Document Authoring → Recommender);
- Search for the information/data units (choose 'shared' document repository, select 'Design Patterns' as a topic and check a 'suggest' keyword checkbox); Try both the content based and the semantic search;
- For a selected document unit from the search results open the detailed view (by clicking on the 'info' link), and then also try to navigate across semantically related document units by browsing units that are annotated by the same ontological concepts as the selected initial data unit;
- When you make a decision, add a chosen document unit to the slides;
- Once you have added all missing data/information on the slides or you decide to quit the task, please save the actual document (do not make another copy of it), stop the recording and save the recoded material. The recorded material should be saved at the default location offered by the recording software and named as 'NameSurname-1'.
- Close MS PowerPoint application and minimize the RDC window;
- Complete the questionnaire related to task3 (Questionnaire C);

Instructions for executing this task without using the SemanticDoc tools:

- Open the PowerPoint document called 'ConventionalPresentation.pptx', which is saved on the desktop and start recording;
- Check what missing information/data you should put on the slides (slide 2, 3, and 4);
- Explore documents from the 'Evaluation Documents' folder, which is saved on the desktop, to find solutions for the missing information/data and add them to the slides;

- Once you have added all missing data/information to the slides or you decide to quite the task please save the actual document (do not make another copy of it), stop the recording and save the recorded material. The recorded material should be saved at the default location offered by the recording software and named as 'NameSurname-2'.
- Close MS PowerPoint and minimize the RDC window;

Bibliography

[1] K. Anyanwu, A. Maduko, and A. P. Sheth. SemRank: ranking complex relationship search results on the semantic web. In *Proceedings of the 14th International World Wide Web Conference, WWW*, pages 117–127, 2005.

[2] Apple Inc. OpenDoc Programmers' Guide. `http://developer.apple.com/documentation/mac/ODProgGuide`, 2006.

[3] S. Auer, S. Dietzold, and T. Riechert. OntoWiki - A Tool for Social, Semantic Collaboration. In *Proceedings of the 5th International Semantic Web Conference, ISWC*, pages 736–749, 2006.

[4] R. Baeza-Yates and B. Ribeiro-Neto. Modern Information Retrieval. *Adison Wesley, ACM Press*, 1999.

[5] A. Barr, P. Cohen, and E. Feigenbaum. *The Handbook of Artificial Intelligence*. William Kaufman, Los Altos, CA, 1981.

[6] C. Becker and C. Bizer. Exploring the geospatial semantic web with dbpedia mobile. *Journal of Web Semantics*, 7(4):278–286, 2009.

[7] P. L. Berger and T. Luckman. *The Social Construction of Reality: A Treatise its the Sociology of Knowledge*. Anchor Books, New York, 1966.

[8] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific America*, pages 29–37, May 2001.

[9] T. Berners-Lee, W. Hall, J. A. Hendler, K. O'Hara, N. Shadbolt, and D. J. Weitzner. A Framework for Web Science. *Fondation and Trends in Web Science*, 1(1):1–130, 2006.

[10] P. M. Berry, K. Myers, T. E. Uribe, and N. Yorke-smith. Task Management Under Change and Uncertainty. In *Proceedings of the Workshop on Constraint Solving under Change*, 2005.

[11] N. Bevan, C. Barnum, G. Cockton, J. Nielsen, J. M. Spool, and D. R. Wixon. The "magic number 5": is it enough for web testing? In *Proceedings of the CHI 2003 Conference on Human Factors in Computing Systems*, pages 698–699, 2003.

[12] C. Bizer, T. Heath, and T. Berners-Lee. Linked Data - The story so far. *Int. Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.

[13] C. Bizer, J. Volz, G. Kobilarov, and M. Gaedke. Silk - A Link Discovery Framework for the Web of Data. In *Proceedings of the Linked Data on the Web workshop at 18th WWW*, 2009.

[14] L. Blunschi and et al. A Dataspace Odyssey: The iMeMex Personal Dataspace Management System (Demo). In *Proceedings of the 3rd Conference on Innovative Data Systems Research*, pages 114–119, 2007.

[15] R. M. M. Braga, C. M. L. Werner, and M. Mattoso. Using Ontologies for Domain Information Retrieval. In *Proceedings of the DEXA Workshop*, pages 836–840, 2000.

[16] J. G. Breslin and S. Decker. The Future of Social Networks on the Internet: The Need for Semantics. *IEEE Internet Computing*, 11(6):86–90, 2007.

[17] M. K. Buckland. The centenary of 'Madame Documentation': Suzanne Briet. *Journal of the American Society for Information Science*, 42:586–588, 1995.

[18] M. K. Buckland. What is a "Digital Document"? *JASIS*, 48(9):804–809, 1997.

[19] L. Carr, T. Miles-Board, A. Woukeu, G. Wils, and W. Hall. The Case for Explicit Knowledge in Documents. In *Proceedings of the 2004 ACM Symposium on Document Engineering, DocEng*, pages 90–98, Milwaukee, Wisconsin, USA, 2004.

[20] M. Christopher, R. Prabhakar, and S. Hinrich. *Introduction to Information Retrieval*. Cambridge University Press, first edition, 2008.

[21] C. Clarke. A Resource List Management Tool for Undergraduate Students Based on Linked Open Data Principles. In *Proceedings of the 6th European Semantic Web Conference, ESWC*, pages 697–707, 2009.

[22] B. Collis and A. Strijker. Technology and Human Issues in Reusing Learning Objects. *Journal of Interactive Media in Education*, 4:1–25, 2004.

[23] Compound Document Format. `http://www.w3.org/TR/WICD/`, 2002.

[24] J. M. Cortina. What is coefficient alpha? An examination of theory and applications. *Journal of Applied Psychology*, 78(1):98–104, 1951.

[25] L. Cronbach. Coefficient alpha and the internal structure of tests. *Psychometrika*, 16(3):297–334, 1951.

[26] F. D. Davis. Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *MIS Quarterly*, 13(3):319–340, 1989.

[27] S. Decker and M. R. Frank. The Networked Semantic Desktop. In *Proceedings of the WWW Workshop on Application Design, Development and Implementation Issues in the Semantic Web*, 2004.

[28] Y. Q. Department, Y. Qiu, and H. P. Frei. Concept Based Query Expansion. In *Proceedings of the 16th annual int. ACM SIGIR conference on Research and Development in information Retrieval*, pages 160–169, 1993.

[29] T. H. Devenport. *Working Knowledge*. Harvard business school press, 2000.

[30] S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, K. S. McCurley, S. Rajagopalan, A. Tomkins, J. A. Tomlin, and J. Y. Zien. A case for automated large scale semantic annotation. *Journal of Web Semantics*, 1(1):115–132, 2003.

[31] L. Ding, R. Pan, T. W. Finin, A. Joshi, Y. Peng, and P. Kolari. Finding and Ranking Knowledge on the Semantic Web. In *Proceedings of the 4th International Semantic Web Conference, ISWC*, pages 156–170, 2005.

[32] X. Dong and A. Y. Halevy. A Platform for Personal Information Management and Integration. In *Proceedings of the 1st Biennial Conference on Innovative Data Systems Research, CIDR*, pages 119–130, 2005.

[33] D. Dori, D. Doermann, C. Shin, R. Haralic, I. Phillips, M. Buchman, and D. Ross. The Representation of Document Structure: a Generic Object-Process Analysis. *Handbook on Optical Character Recognition and Document Image Analysis, Word Scientific*, 1995.

[34] Doublin Core Metadata Initiative. `http://dublincore.org/`, 2004.

[35] Draft Standard for Learning Object Metadata. `http://ltsc.ieee.org/wg12/`, 2006.

[36] A. Duke, T. Glover, and J. Davies. Squirrel: An Advanced Semantic Search and Browse Facility. In *Proceedings of the 4th European Semantic Web Conference, ESWC*, pages 341–355, 2007.

[37] J. Durkin. *Expert Systems: Design and Development*. Macmillan, New York, 1994.

[38] E. Duval, E. Forte, K. Cardinaels, B. Verhoeven, R. V. Durm, K. Hendrikx, M. W. Forte, N. Ebel, M. Macowicz, K. Warkentyne, and F. Haenni. The Ariadne knowledge pool system. *Commun. ACM*, 44(5):72–78, 2001.

[39] Ecma International, Office Open XML. `http://openxmldeveloper.org/`, 2006.

[40] H. Eriksson. The semantic-document approach to combining documents and ontologies. *International Journal of Human-Computer Studies*, 65(7):624–639, 2007.

[41] J. Euzenat and P. Shvaiko. *Ontology Matching*. Springer-Verlag, 2007.

[42] E. Feigenbaum and P. McCorduck. *The fifth Generation (Reading)*. Addison Wesley, 1983.

[43] H. Fenderl, K. Fischer, and J. Kamper. The Open Document Architecture: From standardization to the market. *IBM System Journal*, 31(4), 1992.

[44] J. feng Song, W. M. Zhang, W. Xiao, G. hui Li, and Z. ning Xu. Ontology-Based Information Retrieval Model for the Semantic Web. In *Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service*, pages 152–155, 2005.

[45] D. Fensel, F. van Harmelen, and I. H. and. An Ontology Infrastructure for the Semantic Web. *IEEE Intelligent Systems*, 16(2):37–46, 2001.

[46] E. Friedman-Hill. *Jess in Action*. Manning, Greenwich, UK, 2003.

[47] R. Furita, V. Quint, and J. Andre. *Electronic Publishing - Origination, Dissemination, and Design*. John Wiley and Sons, April 1989.

[48] G. Gediga, K. Hamborg, and I. Duntsch. Evaluation of Software Systems. *Encyclopedia of Library and Information Science*, 72:166–192, 2002.

[49] J. Gemmell, G. Bell, R. Lueder, S. M. Drucker, and C. Wong. MyLifeBits: fulfilling the Memex vision. In *ACM Multimedia*, pages 235–238, 2002.

[50] Z. Gong, C. W. Cheang, and L. H. U. Multi-term Web Query Expansion Using WordNet. In *Proceedings of the 17th Database and Expert Systems Applications Conference, DEXA*, pages 379–388, 2006.

[51] S. Greenberg and W. Buxton. Usability evaluation considered harmful (some of the time). In *Proceedings of the CHI 2008 Conference on Human Factors in Computing Systems*, pages 111–120, 2008.

[52] M. L. Griss. *Software agents as next generation software components*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2001.

[53] W. G. Griswold, R. T. Boyer, S. W. Brown, and T. M. Truong. A Component Architecture for an Extensible, Highly Integrated Context-Aware Computing Infrastructure. In *Proceedings of the 25th International Conference on Software Engineering, ICSE*, pages 363–373, 2003.

[54] F. A. Grootjen and T. P. van der Weide. Conceptual Query Expansion. *Data Knowl. Eng.*, 56(2):174–193, 2006.

[55] T. Groza, S. Handschuh, K. Möller, G. Grimnes, L. Sauermann, E. Minack, C. Mesnage, M. Jazayeri, G. Reif, and R. Gudjonsdottir. The NEPOMUK Project - On the Way to the Social Semantic Desktop. In *Proceedings of the I-Semantics*, pages 187–211, 2007.

[56] T. Gruber. Collective knowledge systems: Where the Social Web meets the Semantic Web. *Journal of Web Semantics*, 6(1):4–13, 2008.

[57] R. V. Guha, R. McCool, and E. Miller. Semantic Search. In *Proceedings of the 12th International World Wide Web Conference, WWW*, pages 700–709, 2003.

[58] H. Haller. Mappingverfahren zur Wissensorganisation. Master's thesis, Institut AIFB, Berlin, Germany, 2002.

[59] W. Harrison. Eating Your Own Dog Food. *IEEE Software*, 23(3):5–7, 2006.

[60] T. Heath and E. Motta. Revyu: Linking Reviews and Ratings into the Web of Data. *Journal of Web Semantics*, 6(4):266–273, 2008.

[61] J. Heflin and J. Hendler. A portrait of the Semantic Web in Action. *IEEE Intelligent Systems*, 16(2):54–59, 2001.

[62] R. Hemayati, W. Meng, and C. Yu. Semantic-based grouping of search engine results using WordNet. In *Proceedings of the APWeb/WAIM'07*, pages 678–686, Berlin, Heidelberg, 2007. Springer-Verlag.

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

[64] J. Hendler and D. L. McGuinness. The DARPA Agent Markup Language. *IEEE Intelligent Systems*, 15(6):67–73, 2000.

[65] J. A. Hendler, N. Shadbolt, W. Hall, T. Berners-Lee, and D. J. Weitzner. Web science: an interdisciplinary approach to understanding the web. *Commun. ACM*, 51(7):60–69, 2008.

[66] D. Hofstadter. *Fluid Concepts And Creative Analogies: Computer Models Of The Fundamental Mechanisms Of Thought*. Basic Books/Harper Collins, New York, 1994.

[67] A. Hotho, S. Staab, and G. Stumme. Wordnet improves Text Document Clustering. In *Proceedings of the SIGIR 2003 Semantic Web Workshop*, pages 541–544, 2003.

[68] E. Hyvnen, E. Mkel, M. Salminen, A. Valo, K. Viljanen, M. Junnila, and S. Kettula. MuseumFinland - Finnish Museums on the Semantic Web. *Journal of Web Semantics*, 3:25, 2005.

[69] K. Järvelin and J. Kekäläinen. IR evaluation methods for retrieving highly relevant documents. In *Proceedings of the 13th annual int. ACM SIGIR conference on Research and Development in information Retrieval*, pages 41–48, 2000.

[70] J. Jovanovic, D. Gasevic, and V. Devedzic. Ontology-Based Automatic Annotation of Learning Content. *Int. J. Semantic Web Inf. Syst.*, 2(2):91–119, 2006.

[71] J. Jovanovic, D. Gasevic, C. A. Brooks, V. Devedzic, M. Hatala, T. Eap, and G. Richards. Using Semantic Web Technologies to Analyze Learning Content. *IEEE Internet Computing*, 11(5):45–53, 2007.

[72] D. R. Karger, K. Bakshi, D. Huynh, D. Quan, and V. Sinha. Haystack: A General-Purpose Information Management Tool for End Users Based on Semistructured Data. In *Proceedings of the 1st Conference on Innovative Data Systems Research*, pages 13–26, 2005.

[73] S. Kendal and M. Creen. *An Introduction to Knowledge Engineering*. Springer, 2007.

[74] A. Kidd. The marks are on the knowledge worker. In *Proceedings of the 1994 Conference on Human Factors in Computing Sustems, CHI*, pages 212–220, Boston, Massachusetts, USA, 1994. ACM.

[75] A. Kiryakov, B. Popov, I. Terziev, D. Manov, and D. Ognyanoff. Semantic Annotation, Indexing, and Retrieval. *Journal of Web Semantics*, 2(1):49–79, 2004.

[76] P. Kogut and W. Holmes. AeroDAML: applying information extraction to generate DAML annotations from web pages. In *Proceedings of the Workshop on Knowledge Markup and Semantic Annotation at 1st International Conference on Knowledge Capture, K-CAP*, Victoria B.C., Canada, 2001.

[77] V. Lanfranchi, F. Ciravegna, and D. Petrelli. Semantic Web-based Document: Editing and Browsing in AktiveDoc. In *Proceedings of The Semantic Web: Research and Applications, Second European Semantic Web Conference, ESWC*, pages 623–632, Heraklion, Crete, Greece, 2005. Springer.

[78] Y. Lei, V. S. Uren, and E. Motta. SemSearch: A Search Engine for the Semantic Web. In *Proceedings of the 15th International Conference on Knowledge Engineering and Knowledge Management, EKAW*, pages 238–245, 2006.

[79] F. Liu, C. Yu, and W. Meng. Personalized Web search by mapping user queries to categories. In *Proceedings of the 11th International Conference on Information and Knowledge Management*, pages 558–565. ACM Press, 2002.

[80] A. Maedche and S. Staab. Ontology learning for the Semantic Web. *IEEE Intelligent Systems*, 16(2):72–79, 2001.

[81] C. Mangold. A survey and classification of semantic search approaches. *International Journal of Metadata, Semantics and Ontologies*, 2(1):23–34, 2007.

[82] C. C. Marshall and F. M. S. III. Which Semantic Web? In *Proceedings of the 14th ACM Conference on Hypertext and Hypermedia*, pages 57–66, 2003.

[83] Microsoft SocialPC. `http://research.microsoft.com/en-us/projects/socialdesktop/`, 2006.

[84] E. L. Morse. Evaluation Methodologies for Information Management Systems. *D-Lib Magazine*, 8(9), 2002.

[85] J. Nelsen. *Usability Engineering*. Academic Press, 1993.

[86] S. Nešić. Semantic Document Model to Enhance Data and Knowledge Interoperability. *Annals of Information Systems, Springer*, 6:135–162, 2009.

[87] S. Nešić, D. Gašević, and M. Jazayeri. An Ontology-Based Framework for Authoring Assisted by Recommendation. In *Proceedings of the 7th IEEE International Conference on Advanced Learning Technologies, ICALT*, pages 227–231, Niagata, Japan, 2007.

[88] S. Nešić, D. Gašević, and M. Jazayeri. An Ontology-Based Framework for Author-Learning Content Interaction. In *Proceedings of the 6th International Conference on Web-based Education, WBE*, pages 359–364, Chamonix, France, 2007.

[89] S. Nešić, J. Jovanović, D. Gašević, and M. Jazayeri. Ontology-Based Content Model for Scalable Content Reuse. In *Proceedings of the 4th ACM SIGART International Conference on Knowledge Capture, K-CAP*, pages 195–196, Whistler, Canada, 2007.

[90] S. Nešić, D. Gašević, and M. Jazayeri. Semantic Document Management for Collaborative Learning Object Authoring. In *Proceedings of the 8th IEEE International Conference on Advanced Learning Technologies, ICALT*, pages 751–755, Santander, Spain, 2008.

[91] S. Nešić, D. Gašević, and M. Jazayeri. Extending MS Office for Sharing Document Content Units over the Semantic Web. In *Proceedings of the 8th International Conference on Web Engineering, ICWE*, pages 350–353, New York, USA, 2008.

[92] S. Nešić, M. Jazayeri, F. Lelli, and D. Gašević. Towards Efficient Document Content Sharing in Social Networks. In *Proceedings of the 2nd International Workshop on Social Software Engineering and Applications, co-located with ESEC/FSE*, pages 1–8, Amsterdam, The Netherlands, 2009.

[93] S. Nešić, F. Crestani, D. Gašević, and M. Jazayeri. Concept-Based Semantic Annotation, Indexing and Retrieval of Office-Like Document Units. In *Proceedings of the 9th International conference on Adaptivity, Personalization and Fusion of Heterogeneous Information, RIAO*, pages 234–237, Paris, France, 2010.

[94] S. Nešić, F. Crestani, D. Gašević, and M. Jazayeri. Search and Navigation in Semantically Integrated Document Collections. In *Proceedings of the 4th International Conference on Advances in Semantic Processing, SEMAPRO*, pages 55 – 60, Florence, Italy, 2010.

[95] S. Nešić, M. Jazayeri, and D. Gašević. Semantic Document Architecture for Desktop Data Integration and Management. In *Proceedings of the 22nd International Conference on Software Engineering and Knowledge Engineering, SEKE*, pages 73–78, San Francisco, USA, 2010.

[96] S. Nešić, M. Jazayeri, M. Landoni, and D. Gašević. Using Semantic Documents and Social Networking in Authoring of Course Material: An Empirical Study. In *Proceedings of the 10th IEEE International Conference on Advanced Learning Technologies, ICALT*, pages 666–670, Sousse, Tunisia, 2010.

[97] OASIS Consortium, Open Document Format for Office Application. `http://www.oasis-open.org/committees/office/`, 2006.

[98] Object Linking and Embedding. `http://www.microsoft.com/com/default.mspx`, 2006.

[99] OWL Web Ontology Language Guide. `http://www.w3.org/TR/owl-guide/`, 2006.

[100] R. Ozcan and Y. A. Aslandogan. Concept-Based Information Access. In *Proceedings of the International Conference on Information Technology: Coding and Computing, ITTC*, pages 794–799, 2005.

[101] T. A. Phelps and R. Wilensky. Multivalent Documents. *Communications of ACM*, 43(6):82–90, 2000.

[102] J. Pitkow, H. Schutze, T. Cass, R. Cooley, D. Tumbull, A. Edmonds, E. Adar, and T. Breuel. Personalized search. *Communications of ACM*, 45:50–55, 2002.

[103] D. Quan, D. Huynh, and D. R. Karger. Haystack: A Platform for Authoring End User Semantic Web Applications. In *Proceedings of the 2nd International Semantic Web Conference, ISWC*, pages 738–753, 2003.

[104] V. Quint and I. Vatton. Active Documents as a Paradigm for human-Computer Interaction. Submitted to the workshop "Research issues in the intersection between software engineering and human-computer interaction", Sorrento, Italy, 1994.

[105] J. Redish, R. G. Bias, R. Bailey, R. Molich, J. S. Dumas, and J. M. Spool. Usability in practice: formative usability evaluations - evolution and revolution. In *Proceedings of the CHI 2002 Conference on Human Factors in Computing Systems*, pages 885–890, 2002.

[106] Reference description of the DAML+OIL ontology markup language. `http://www.daml.org/2001/03/reference.html`, 2003.

[107] B. K. Reid. *Scribe: a document specification language and its compiler*. PhD thesis, Carnegie-Mellon University, Pittsburgh, USA, 1981.

[108] L. Rensis. A Technique for the Measurement of Attitudes. *Archives of Psychology*, 140:1–55, 1932.

[109] P. Resnik. Semantic similarity in a taxonomy: An information-based measure and its applications to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11(95-130), 1999.

[110] C. Rocha, D. Schwabe, and M. P. de Aragão. A hybrid approach for searching in the semantic web. In *Proceedings of the the 13th international conference on World Wide Web, WWW*, pages 374–383, 2004.

[111] RuleML - The rule Markup Initiative. `http://www.ruleml.org/`, 2005.

[112] T. Ruotsalo and E. Hyvönen. A Method for Determining Ontology-Based Semantic Relevance. In *Proceedings of the 18th Database and Expert Systems Applications Conference, DEXA*, pages 680–688, 2007.

[113] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. 2nd edition. Prentice-Hall, Englewood, NJ, 2002.

[114] G. Salton and C. Buckley. Term-Weighting Approaches in Automatic Text Retrieval. *Inf. Process. Manage.*, 24(5):513–523, 1988.

[115] L. Sauermann and D. Heim. Evaluating Long-Term Use of the Gnowsis Semantic Desktop for PIM. In *Proceedings of the 7th International Semantic Web Conference, ISWC*, pages 467–482, 2008.

[116] B. Schandl and B. Haslhofer. The Sile Model - A Semantic File System Infrastructure for the Desktop. In *Proceedings of the 6th European Semantic Web Conference, ESWC*, pages 51–65, 2009.

[117] W. R. Shadish, T. D. Cook, and D. T. Campbell. *Experimental and Quasi-Experimental Designs for Generalized Causal Inference*. Wadsworth Publishing, first edition, 2001.

[118] M. A. Sicilia. Metadata, semantics and ontology: providing meaning to information resources. *International Journal of Metadata, Semanatics and Ontologies*, 1 (1):83–86, 2006.

[119] T. D. Simone and D. Kazakov. Using WordNet Similarity and Antonymy Relations to Aid Document Retrieval. In *Recent Advances in Natural Language Processing, RANLP*, 2005.

[120] M. F. Smith. *Software Prototyping: Adoption, Practice, and Management*. Mcgraw Hill Book Co Ltd, 1991.

[121] S. Stabb and R. Studer. *Handbook on Ontologies*. Springer, Berlin, 2004.

[122] Standfor Encyclopedia of Philosophy. `http://plato.stanford.edu/entries/cognitive-science/`, 1989.

[123] SWRL: A Semantic Web Rule Language Combining OWL and RuleML. `http://www.w3.org/Submission/SWRL`, 2005.

[124] M. Tallis. Semantic Word Processing for Content Authors. In *Proceedings of the Knowledge Markup and Semantic Annotation Workshop at the 2nd International Conference on Knowledge Capture*, Sanibel, Florida, USA, 2003.

[125] P. L. Thomas and D. F. Brailsford. Enhancing Composite Digital Documents Using XML-based Standoff Markup. In *Proceedings of the 2005 ACM Symposium on Document Engineering, DocEng*, pages 177–186, Bristol, United Kingdom, 2005. ACM.

[126] J. Tuominen, M. Frosterus, and E. Hyvönen. ONKI SKOS Server for Publishing and Utilizing SKOS Vocabularies and Ontologies as Services. In *Proceedings of the 6th European Semantic Web Conference, ESWC*, pages 768–780, 2009.

[127] V. Uren, P. Cimiano, J. Iria, S. Handschuh, M. Vergas-Vera, E. Motta, and F. Ciravegna. Semantic annotation for knowledge management: Requirements and survey of the state of the art. *Journal of Web Semantics*, 4(1):14–28, 2006.

[128] D. Vallet, M. Fernández, and P. Castells. An Ontology-Based Information Retrieval Model. In *Proceedings of the 2nd European Semantic Web Conference, ESWC*, pages 455–470, 2005.

[129] V. Vianu. Rule-based languages. *Annals of Mathematics and Artificial Intelligence*, 19(1-2):215–259, 1997.

[130] M. Volkel. From Documents to Knowledge Models. In *Proceedings of the ProKW workchop, Konferenz Professionelles Wissensmanagement*, Postdam, Germany, 2007.

[131] E. M. Voorhees. Query expansion using lexical-semantic relations. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '94*, pages 61–69, New York, NY, USA, 1994. Springer-Verlag New York, Inc. ISBN 0-387-19889-X.

[132] D. J. Weitzner. Beyond Secrecy: New Privacy Protection Strategies for Open Information Spaces. *IEEE Internet Computing*, 11(5):94–96, 2007.

[133] D. J. Weitzner, H. Abelson, T. Berners-Lee, J. Feigenbaum, J. A. Hendler, and G. J. Sussman. Information Accountability. *Commun. ACM*, 51(6):82–87, 2008.

[134] C. A. Welty. *An integrated representation for software development and discovery*. Rensselaer Polytechnic Institute, Troy, NY, USA, 1996.

[135] S. Whittaker, L. Terveen, and B. A. Nardi. Let's stop pushing the envelope and start addressing it: a reference task agenda for HCI. *Human-Computer Interaction*, 15(2):75–106, 2000. ISSN 0737-0024.

[136] D. W. Zimmerman. Teachers Corner: A Note on Interpretation of the Paired-Samples t-Test. *Journal of Educational and Behavioural Statistics*, 22(3):349–360, 1997.