Semantic Document Model to Enhance Data and Knowledge Interoperability

Saša Nešić

Abstract In order to enable document data and knowledge to be efficiently shared and reused across application, enterprise, and community boundaries, desktop documents should be completely open and queryable resources, whose data and knowledge are represented in a form understandable to both humans and machines. At the same time, these are the requirements that desktop documents need to satisfy in order to contribute to the visions of the Semantic Web. With the aim of achieving this goal, we have developed the Semantic Document Model (SDM), which turns desktop documents into *Semantic Documents* as uniquely identified and semantically annotated composite resources, that can be instantiated into human-readable (HR) and machine-processable (MP) forms. In this paper, we present the SDM along with an RDF and ontology-based solution for the MP document instance. Moreover, on top of the proposed model, we have built the Semantic Document Management System (SDMS), which provides a set of services that exploit the model. As an application example that takes advantage of SDMS services, we have extended MS Office with a set of tools that enables users to transform MS Office documents (e.g., MS Word and MS PowerPoint) into Semantic Documents, and to search local and distant semantic document repositories for document content units (CUs) over Semantic Web protocols.

1 Introduction

The Semantic Web aims at providing an environment in which both humans and software agents can unambiguously determine the meaning of resources and make better use of them [2]. Moreover, data and knowledge stored within a resource should be easily accessible across application, enterprise, and community bound-

Saša Nešić
Faculty of Informatics, University of Lugano, Via G. Buffi 13, Lugano, Switzerland, e-mail: sasa.nesic@lu.unisi.ch

aries. In traditional desktop architectures, applications are isolated islands with their own data, which are unaware of related and relevant data in other applications [10]. Heterogeneous, application-specific document formats, which keep data into schema specific elements, do not allow data interoperability between the applications. In a similar way, there is no standardized architecture for interoperation and data exchange between the desktops of different users. In order to achieve data interoperability on local desktops and its seamless integration with other resources of the Semantic Web, the first step is the organization of local desktops as complete RDF and ontology-based environments. This brings us to the notion of the Semantic Desktop [3] - the driving paradigm for desktop computing in the area of the Semantic Web. In other words, local desktops should become the Semantic Web for a single user.

Digital desktop documents (e.g., Word, PDF and PowerPoint) hold a significant part of the data and knowledge stored on local desktops and hence they play an important role in the vision of the Semantic Desktop and the Semantic Web. However, in order to fully participate in this vision traditional desktop documents need to be adapted first. The large variety of application-specific document formats hampers the interoperability of document data. Document data is kept into schema-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) [16] and Microsoft Open Office XML (OOXML) [5], which opened a way towards easier document transformation and data exchange. Yet, the development of export/import functions is a difficult and costly process, as it requires detailed knowledge of both the input and output formats.

The variety of application-specific document formats is not only issue that impacts the interoperability of document data. In order to be discovered and then reused, document data needs to be semantically annotated. So far, several document annotation frameworks [23] that apply one of the two annotation storage models, the Semantic Web model and Document-Centric (word processor) model, there have been developed. In the first model, annotations are stored separately from the source document. An advantage of this model is that no changes to a document are required. However, this model is rarely applied to desktop documents because an efficient solution of the maintenance of links between document content and annotations still does not exist. The Document-Centric model stores annotations inside the internal document representation and has been used as the dominant annotation model for desktop documents (e.g., Word, PDF, PowerPoint) [4, 6, 22], mainly because it overcomes the problem of keeping annotations and documents consistent. However, storing annotations inside a document usually requires the extension of the document format schema, which is not always possible. The other problem of the document annotation is the lack of appropriate schema elements for the annotations of document CUs of different levels of granularity (e.g., sections, paragraphs, images, tables). The majority of existing document schemas provides elements for document annotations at the level of a whole document, while document CUs of lower granularity remain unannotated. This is primarily because document schemas

do not define document CUs as uniquely identified and addressable entities that can hold their own annotations. Accordingly, the discoverability and reusability of document CUs is significantly decreased.

Moreover, the existing desktop documents are suited primarily for humans, so that knowledge modeled within them is not represented in a form that allows intelligent software agents to discover and use it. Document annotation with ontologies, known as Semantic Document approach [6], is as an attempt to conceptualize document knowledge. However, we believe that the term 'Semantic Document' should not denote only documents annotated with ontologies, but rather a new category of documents that can fully contribute in the environment envisioned by the Semantic Web. In order to be a part of the Semantic Web resources, digital desktop documents need to be adapted first. This adaptation will lead to a new generation of documents that we call semantic documents. We have identified following four principles, which can be considered as the basis of semantic documents:

1. Document content should be completely queryable, with addressable elements (i.e., CUs) of different granularity;
2. A whole document and all its CUs should be uniquely identified with URIs (Unique Resource Identifiers);
3. A document as a whole, as well as document CUs should be annotated with substantial sets of metadata;
4. Human-understandable knowledge that is modeled in document CUs should be also represented in a form it can be processed by machines (i.e., software agents).

In accordance to these principles we have developed Semantic Document Model (SDM), which turns digital documents into semantic documents. The model takes the existing digital documents as human readable (HR) instances of the semantic document and integrates it with the newly created machine processable (MP) instance. We have taken existing digital documents as the HR instances in order to let users continue to use well-established document formats and because of the development of new document authoring systems, which is an expensive investment that is not likely to happen.

This paper is organized as follows. In the next Section, we discuss the document evolution starting from paper-based documents then digital documents to the envisioned semantic documents. In Section 3, we first present the SDM, then propose an RDF and ontology-based solution for the MP document instance, and conclude the section with the explanation of how semantic documents are stored and organized. In Section 4, we first explain the notion of the Social Semantic Desktop (SSD) paradigm and outline the architecture of the NEPOMUK SSD [10] as a real example of the SSD. Then we explain the Semantic Document Management System (SDMS) that we have developed on the top of the SDM, and how it can be integrated into the NEPOMUK SSD platform. In Section 5, in order to illustrate semantic documents in real use, we present some application examples (i.e., MS Office add-ins) that take advantage of the SDMS services. In Sections 6, we identify some shortcomings of the presented work and continue with related works in Section 7. Discussion of future work and final remarks conclude the paper.

2 From Paper-Based and Digital to Semantic Documents

A document is a bounded physical representation of a body of information designed to convey information. Documents play a key role in the construction of social reality [1] and therefore play a part in accounts of every important aspect of human society and culture. The form of documents has been changing over time following the development of human society. One of the key changes happened with the introduction of 'Digital Era', which led to the main classification of documents into the paper-based and digital documents. Despite many differences between these two forms of documents, the purpose of documents remained unchanged. Both the paper-based and digital documents serve as a medium for the information sharing between humans. The next form of documents, which is inspired by the notions of the Semantic Web and the Semantic Desktop, aims at enabling document data and knowledge to be shared and understood not only by humans but also by machines. In the rest of the section, we discuss the main features of the three forms of documents: paper-based, digital and semantic.

The principle differences between paper-based 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. Paper-based documents are created and managed completely manually or by using some mechanical devices (e.g., a typewriter), and communicated among people in the same manner as any other mobile physical object. Throughout history, there have been used different materials such as clay tables, parchment and papyrus as the physical medium for the storage of paper-based documents. Nowadays, paper is the most common medium for this form of documents and hence the name paper-based documents. In contrast to paper-based documents, digital documents are stored on digital mediums such as hard disk drives (HDDs), CD-ROMs, external hard drives and DVDs. Digital documents 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 digital documents are document publishing on the Web and sending documents by e-mails.

Besides these principal differences, digital documents differ in many other ways from paper-based documents. Prominent examples of digital documents such as word processor documents, slide presentations, spreadsheets and PDF documents are structured documents, which have a visual layer separate from their logical structure. The logical structure of digital documents enables direct access to particular document parts, thus making the granularity in digital documents smaller than in paper documents. By using named anchors, document parts can be linked to external resources (e.g., other documents, Web pages and people). Moreover, readers can add extra information at a particular point of document without making changes to document content. This extra information is known as an annotation and may help other readers to better understand document content. Logical document structure, combined with possible annotations, opened the way for structuring and acquisition of document knowledge, which can turn a digital document into a widespread knowledge model. Digital documents have some drawbacks as well. They are less

stable in time (i.e., their content can change at any point in time) than paper documents, and can be cited only if they are managed by trustworthy sources.

Digital documents render a significant part of the knowledge base stored on local desktops and as such they are important potential resources for the Semantic Desktop and the Semantic Web. However, digital documents in a current form do not meet the requirements demanded of the Semantic Web resources. The Semantic Web resources have to be uniquely identified resources with easily accessible content and annotated with machine processable meta-level descriptions. The content of current digital documents is usually locked into specific schema elements and is not always addressable and accessible from the outside of the documents. Moreover, knowledge modeled within documents is represented in a HR form and cannot be discovered and processed by machines. The transformation of digital documents into the Semantic Web resources leads to the semantic documents as a new form of documents. Several existing technologies, in particular ontologies and RDF, can be taken as basis for semantic documents. Annotating digital documents with ontologies [23] has been the most common strategy to adapt digital documents to the Semantic Web. However, in our opinion the ultimate goal of semantic documents is not merely to provide annotations for documents, but to integrate two representations of the same knowledge: human readable (HR) and machine processable (MP) and to create platform/tool independent, unified view of document data.

Semantic documents will even more differ from digital documents than digital documents differ from paper documents. If we consider a paper document as a hard copy of a digital document, then a digital document can be seen as human readable form of a semantic document. Therefore, the generation of semantic documents can be equated with the generation of the MP document form and its integration with the existing digital documents (i.e., HR document instances).

3 Semantic Documents

In accordance to four principles that we stated in Section 1, we give the following definition of semantic documents:

> A Semantic Document is a uniquely identified and semantically annotated composite resource. It is built of smaller resources (CUs), which can be either composite or atomic and which are also uniquely identified and semantically annotated. Each document CU is characterized by its content (data) and knowledge, which are represented in a form understandable to both humans and machines. CUs can be put in different kinds of relationships with other, uniquely identified resources (e.g., other CUs, Web Pages, people, institutions, etc.). Hierarchical and navigational relationships among CUs are used to define document logical structure.

Based on the given definition, semantic documents are resources, which exist independently of their concrete implementation. The two categories of possible users (i.e., humans and machines) determine the two possible forms of the semantic document implementation: human-readable (HR) and machine-processable (MP). Both forms are persistent, with the difference that there is only one MP document instance and that can be zero or several HR instances. The same document CUs in the MP and HR instances are identified with the same URIs, establishing in that way the links between the two document instances. These links allow users to take advantage of new features enabled by the introduction of the MP document instance. Therefore, the MP instance plays the key role in our vision of the semantic documents.

The MP document instance is universal, platform independent and completely queryable. It enables the annotation of the semantic document and its CUs with different kinds of annotations and let the HR instances to be intact by the annotations. Over the links between the MP and HR instances users can query the MP instance to get the annotations. Moreover, the MP instance holds conceptualized document knowledge, thus enabling users to deploy some software agents in discovering documents CUs based on their knowledge rather than performing full-text search. The MP instance also provides mechanisms for versioning of CUs and formal representation of changes made to CUs over time. In addition, universal and platform independent MP instance, which can be rendered into different platform/tool specific HR instances, serves as a transformation bridge between platform/tool specific document formats.

3.1 Semantic Document Model (SDM)

We have developed the SDM in accordance to the given definition and being partially inspired by the Abstract Compound Content Model (ACCM) [14] and the Abstract Learning Object Content Model (ALOCoM) [9], which both have roots in the IBM's Darwin Information Architecture (DITA) [18]. The ACCM model defines CUs of different levels of granularity as well as the content aggregation architecture that organizes CUs for content deliverables. From the prospective of the ACCM, digital documents can be regarded as instances of the content aggregation architecture in which document parts are considered as CUs. The ALOCoM has served as the basis for supporting learning content personalization as well as learning content authoring. Unlike ALOCoM, which is an abstract content model for learning content, the ACCM is applicable to any kind of digital content.

We have chosen ontologies to formally describe the SDM because they are promising solution to both: modeling document logical structure and document knowledge representation. Ontologies provide a number of useful features for intelligent systems, knowledge representation in general and for the knowledge engineering process. Although the major purpose of ontologies is knowledge sharing and knowledge reuse by applications [20], in the last decade, ontologies have also emerged as one of the most popular modeling approaches to taxonomies, classifica-

tions and other structures used in information systems (IS). The ontological foundation of the SDM consists of the document ontology, annotation ontology and change ontology. We have developed the all three ontologies as OWL ontologies by using Protg ontology editor and they are available at [24].

3.1.1 Document ontology

The document ontology formally describes the SDM by providing definitions of possible types of document CUs as well as architectural elements that organize CUs into document logical structure. Fig. 1 gives the graphical representation of the document ontology in a form of a Resource Description Framework (RDF) graph. The ontology contains two groups of concepts: content elements and architectural elements.

The first group, that is content elements, contains concepts that define document CUs as uniquely identified resources, which hold peaces of document content and can be extracted form document context and reused in other documents. The main concept in this group is *ContentUnit (CU)* concept, which has two sub-concepts: *ContentFragment (CF)* and *ContentObject (CO)*. The *CF* represents *CUs* in their most basic form (i.e., raw digital resources) and can be further specialized into *DiscreteCF* (e.g., *Graphic* and *TextFragment*) and *ContinuousCF* (e.g., *Audio*, *Video*, and *Simulation*). The *CO* represents *CUs* (e.g., *Paragraph*, *Table* and *Slide*), which aggregate *CFs* and other *COs* by using the *hasPart* property and add navigation among them by the *hasOrdering* property.

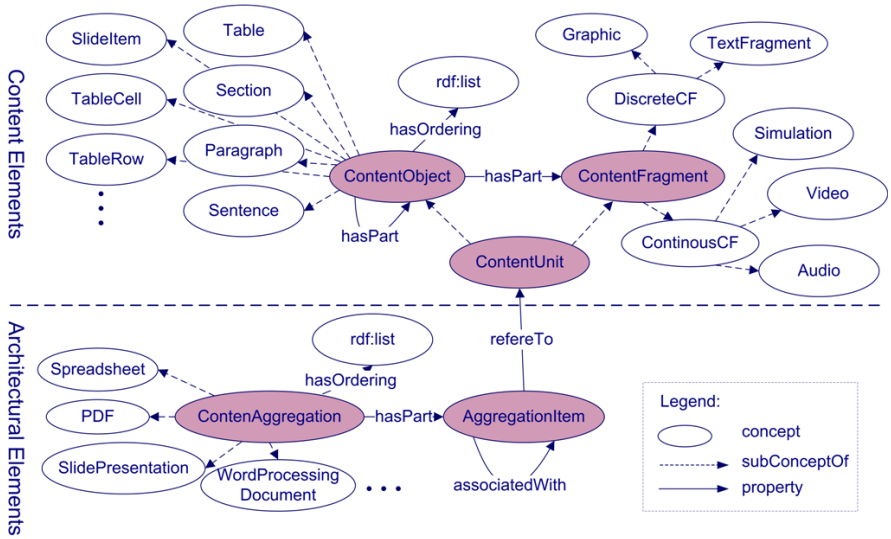

Fig. 1 Document Ontology

The second group, that is, architectural elements contains concepts that define elements of the document logical structure. Two core concepts are *Aggregation-Item (AI)* and *ContentAggregation (CA)*. The *AI* holds a reference to an instance of the *CU* concept via a *refersTo* property and represents the appearance of the *CU* within a document. The *CA* defines logical structure of a document by establishing relationships among the *AIs*. Besides aggregational relationships, which are expressed with a *hasPart* property, the *ContentAggregation* defines navigational and associative relationships via the *hasOrdering* and *associateWith* properties. The aggregational and navigational relationships enable sequencing and structuring of the document content in a form of the tree structure. On the other hand, the associative relationships enable links among *AIs* based on the given criteria (e.g., this can be used for modeling hyperlinks inside a document).

3.1.2 Annotation Ontology

One of the main objectives of our semantic document model is to enable software agents to easily discover, access, and reuse document CUs of different levels of granularity without affecting the document as a whole. However, prior to the access and reuse, document CUs have to be discovered which demands their semantic annotation. In order to enable infrastructure for semantic annotation of document CUs, we have developed the annotation ontology.

Our intention with the annotation ontology was not to cover all possible kinds of annotations, but to provide common interface (e.g., classes and properties) for adding annotations to the document CUs. Considering document CUs as constitutive blocks of document context on the one hand, and isolated pieces of content on the other hand, we have identified two types of possible annotations: 1) annotations that belong to CUs independently of the document (i.e., *context-free* annotations) and 2) annotations that belong to CUs when they are parts of the document (i.e., *context-dependent* annotations).

The annotation ontology relates the context-free annotations to instances of the content elements defined by the document ontology. For this purpose, the ontology introduces the *hasAnnotation* property and the *ContentUnitAnnotation* concept, which act as a metadata binding to a CU. We have identified three categories of context-free annotations: a) standardized metadata, b) usage metadata, and c) subject-specific metadata.

Standardized metadata is described by internationally recognized vocabularies like Dublin Core[1] (DC) [6] or IEEE Learning Object Metadata[2] (LOM) [14], which are designed to describe any kind of resources, that is, anything that has identity. We have chosen a subset of this metadata which is meaningful for document CUs: `dc:creator`, `dcterms:created`, `dc:format`, `dc:language`, `dc:title` and `dc:description` referring to the author(s), creation date, me-

[1] Dublin Core Metadata Initiative: http://dublincore.org/

[2] IEEE standard for learning object metadata: http://ltsc.ieee.org/wg12

dia type, language(s), title and short description respectively and incorporated it in the annotation ontology.

Usage metadata tracks information about the usage of document CUs in different contexts by different users. Since one of our goals is to enable users to share their documents by interacting (e.g., visiting, modifying and reusing) directly with the document CUs, we have extended the annotation ontology with a set of properties and concepts to capture this interaction (Fig. 2). There are three new concepts: *Modification*, *Reuse*, and *Visit*; and three new properties: *numOfVisits*, *numOfModifications* and *numOfReuses*. All the three introduced concepts are characterized by the time of the interaction and person who is involved in it. The *Modification* and *Reuse* concepts also track information about deployed applications during the interaction. Every time the user interacts with the document CU, the interaction metadata is added to the CU. This metadata is primarily used to determine how some document CUs correspond to the users preferences (e.g., if the user prefers recently modified CUs or CUs reused many times, etc.) and has an important role in the ranking algorithm for document CUs that we have presented in [13].

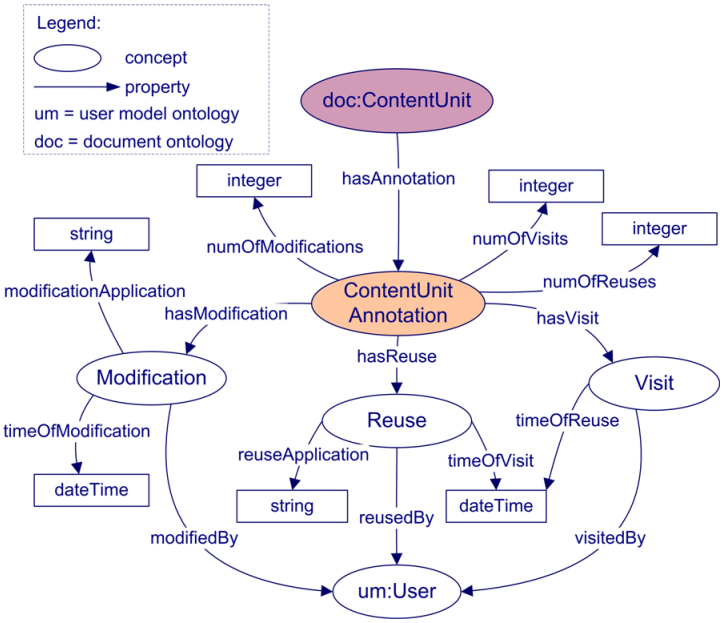

Fig. 2 Annotation ontology - usage metadata

Subject-specific metadata of the CU is ontological metadata [20], which conceptualize the same subjects as those described by the CU. The annotation ontology uses the `dc:subject` property to add the subject-specific metadata to the CU. This metadata is actually a set of ontological concepts, which can be regarded as the con-

ceptualization of knowledge/phenomena modeled by the CU. Since one of our goals with the SDM is to represent document knowledge not only in human readable but also in machine-processable form, this metadata plays an important role in our approach. It allows intelligent software agents to manipulate with conceptualized document knowledge. On the other hand, it also enables humans to use software agents in locating document CUs based on the knowledge modeled within them. There are two approaches to obtaining this metadata, that is, to finding and relating appropriate ontological concepts to document CUs. One is the manual annotation based on the domain experts knowledge and the other is automatic annotation based on the information extraction (IE) and natural language processing (NLP) techniques.

The context-dependent annotations characterize the CUs only when they are parts of the document context. The annotation ontology relates these annotations to the instances of the *AggregationItem* concept defined by the document ontology, which refer to the CUs and represent their appearance within the document. For this purpose, we have introduced the *ItemAnnotation* concept that serves as a metadata binding around the AggregationItem. Currently, we use the context-dependent annotations for adding rhetorical and cognitive descriptions [8, 9] to CUs. To achieve this we extended the annotation ontology with two new concepts: *RhetoricalElement* (e.g., *Abstract*, *Overview* and *Introduction*) and *CognitiveElement* (e.g., *Definition* and *Procedure*). These annotations enable users to include rhetorical and cognitive aspects when they search for document CUs.

3.1.3 Change Ontology

Change ontology (Fig. 3) tracks possible changes to document CUs and document logical structure in accordance to their definitions by the document ontology. The change ontology has three main concepts: *CFChange*, *COChange*, and *CAChange*.

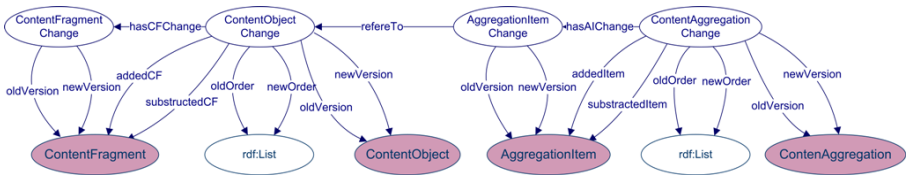

Fig. 3 Change ontology

The *CFChange* is a change made to a CF, which creates a new version of the CF. Since the CF is an atomic CU that can not be disaggregated into smaller units, a *CFChange* can only be determined by comparing the old and new versions of the CF. In order to keep track of this kind of changes, the presence of both the old and new versions of the CF is necessary. Thereby, we defined the *oldVersion* and *newVersion* properties, to link the old and new versions of the CF to the instance

of the *CFChange* concept. By deleting the old version, we lose all the information about the changes that translate this version into the new version.

The *COChange* is a change made to a CO, which create a new version of the CO. Based on the CO definition, we have identified the following possible types of the CO modifications: 1) addition of CFs; 2) subtraction of CFs; 3) reordering of CFs; and 4) changes to CFs which are part of the CO. The first two are modeled by the *addedCF* and subtractedCF properties which link added and subtracted CFs to COChange. The third is modeled by the *newOrder* and *oldOrder* properties, which link instances of the *rdf:List* concept to the *CFChange*. The instances of the rdf:list keep the new and old order of CFs in the CO. If the modification to the CO is a result of changes to CFs, which are part of the CO, than instances of the *CFChange* are linked to the *COChange* by the *hasCFChange* property. In contrast to the *CFChange*, the *COChange* does not necessarily need to keep link to the old version of the CO. In the case of losing the old version of the CO, it can be rebuilt based on the new version and captured changes by the *COChange*.

Besides changes to document CUs, the change ontology also captures possible changes to document logical structure. The logical structure of a document can be changed either by adding, subtracting, or reordering references (i.e., *AggregationItems*) to the document CUs. The ontology defines the *CAChange* concept and the *addedItem*, *subtractedItem*, *oldOrder*, and *newOrder* properties to model changes to document logical structure.

3.2 The MP and HR instances of semantic documents

Semantic documents can be instantiated into HR and MP forms. Both forms are persistent with the difference that exists only one MP instance and zero or several HR instances. Actually, a new HR instance can be rendered from the MP instance at any time when humans want to browse or edit the semantic document. Changes to a HR instance are not necessarily changes to the semantic document. Only the author and a set of authorized users have rights to change the semantic document by incorporating changes from HR instances. The MP and HR instances are stored separately without restricting each other, but well linked in order to ensure consistency and synchronous evolution of data and knowledge modeled by the semantic document. Fig. 4 gives the illustration of the semantic document by showing the couple of its MP and HR instances and the application range of the document, annotation and domain ontologies within the document. From the human point of view the advantage of this coupling is that users can continue to work with documents as before, but now they can also use different services provided by software agents, which are capable to process the MP document instance. For example, the ontology-based software agents can be used to locate and retrieve document CUs, which model the desired knowledge (i.e., CUs to which are related ontological concepts that conceptualize the desired knowledge). From the machine point of view the advantage of the coupling, that is, the advantage of the existence of the MP document instance

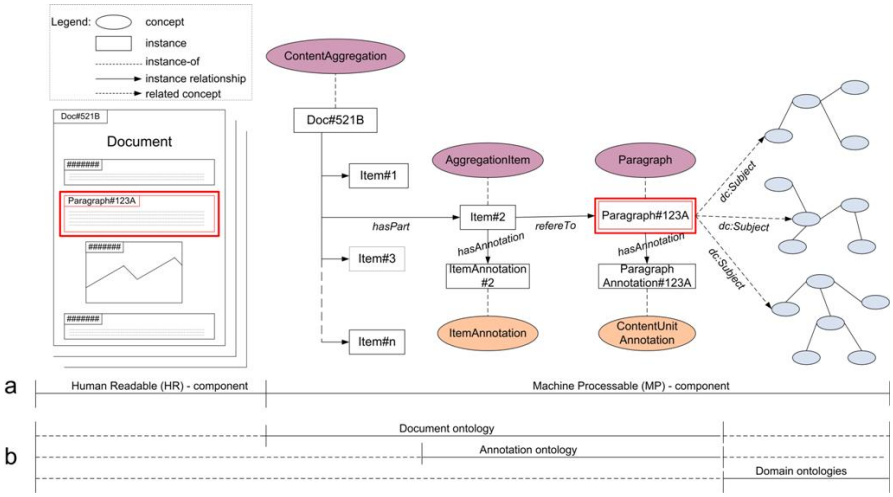

Fig. 4 The illustration of the semantic document: a) The two instances of the semantic document HR and MP; b) The range of the document, annotation, and domain ontologies

is that intelligent software agents can 'understand' and use document knowledge in reasoning and answering to some domain-specific questions.

We have identified following principles, which act as the basis of the MP and HR document instances and define the correlation between them:

- The use of existing document formats as a HR instance;
- Universal, platform-independent and queryable MP instance;
- Bidirectional links between the MP and HR instances;
- Semantic annotations stored only into the MP instance;

First, the use of existing, well-established document formats, as a HR document instance is preferred because the development of new formats and tools for their management requires expensive investment and it is not likely to happen. Therefore, the success of the SDM demands the use of existing document formats as the HR instances of the semantic document. Accordingly, new services introduced by the model should be implemented through extensions to the existing document management environments. Currently, there are numerous document formats, many of which have format schemas that are very strict and difficult to extend. This actually means that the possibility of storing some meta-level descriptions into internal document representation is limited to the ability of a formats schema to be extended. In our approach, we do not face this problem, since we store the MP document instance separately from the HR instances. Changes to documents (e.g., PDF and MS Office documents) are minimized: specifically, the only necessary change is embedding CU URIs and CU version IDs (VIDs), which are used to uniquely identify a CU. The majority of existing document formats has some support for hidden bookmarks or simple types of annotation (e.g., PDF annotation element for PDF documents and

custom XML markup and hidden bookmarks for MS Office documents) and we take advantage of this for embedding the CU URIs and VIDs.

Second, the MP document instance should be universal and platform-independent, so that it does not have to be rebuilt each time a new technology comes along. Also, it has to be completely open and queryable to allow easy access and retrieval of the document CUs and their semantics. Responding to these requirements, we have chosen the Semantic Web technologies, in particular ontologies and RDF, as the basis of the MP instance. The MP instance is an RDF graph, whose nodes are instances of CU concepts defined by the document ontology (e.g., *Paragraph*, *Table*, *Image*, and *Slide*). To these nodes are linked concepts from the domain ontologies, which conceptualize the same phenomena as those described by the document CUs, that is, subject-specific metadata in accordance to the annotation ontology. The other two types of annotations defined by the annotation ontology are linked to the RDF nodes as well. The RDF nodes that are instances of the CUs of the CF type (e.g., *TextFragment*, *Image*, *Audio*, and *Video*) should also hold CU binary content (data). However, current RDF repositories are not meant to store large chunks of binary data so that the content from CUs is placed into binary data stores and linked to RDF nodes. In addition, the relationships between the nodes that are defined by the properties from the document ontology, model the document logical structure. Traditionally, a document structure is considered as a tree-like structure, although some document formats (e.g., MS Office) support hyperlinks between non parent-child document CUs. In accordance to our definition of semantic documents, a document logical structure is a graph structure, which enables not only parent-child navigation but also other navigation paths through the document. Therefore, the RDF vocabulary, which is defined primarily to describe resources as interlinked graph nodes, is a promising solution to modeling document logical structure.

Third, as mentioned before, the MP instance of the document has an RDF node for each document CU. Each node is identified with the URI and Version Identifier (VID). While rendering the HR instance from the MP instance, copies of these identifiers are embedded in the HR instances, thus forming the link between the same CUs of the two instances. Via these links, humans can obtain the additional information of the CUs, which is store only into the MP instance. On the other hand, the links between the HR and MP instances also enable users to edit the document by editing its HR instances. By using appropriate services, the authorized users can incorporate changes they made to the HR instances into MP document instance.

Fourth, semantic annotations of the semantic documents are stored only into the MP instance, thus enabling the semantic annotation model to be unconstrained by the specifics of different document format schemas. The majority of existing desktop document annotation models [6, 22, 23] store annotations inside the internal document representation. Such annotation storage model has been used mainly because it overcomes the problem of keeping annotations and documents consistent. However, storing annotations inside a document usually demands the extension of 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. In our approach, we simplify the annotation process by adding annotations to RDF

nodes of the MP instance instead of embedding them into the internal document representation of the HR instances. The semantic document annotation becomes independent of the HR document format. Through links between the MP and HR instances the user can access annotations of each document CU while browsing its content. In the same way, the user can edit existing or add new annotations to the semantic document.

3.3 Storage and Organization of Semantic Documents

After the transformation of digital documents into semantic documents, the HR instance (i.e., a digital document with embedded CUs' URIs) can stay on the same location in the file system as it was before the transformation or can be moved somewhere else. The embedded document's URI is unaffected by the location of the document in the file system. Moreover, by making a copy of the document or by changing the document's name, the document remains the same resource. If the user changes content of the already transformed digital document, system, which manages semantic documents, should initiate process of updating the MP instance.

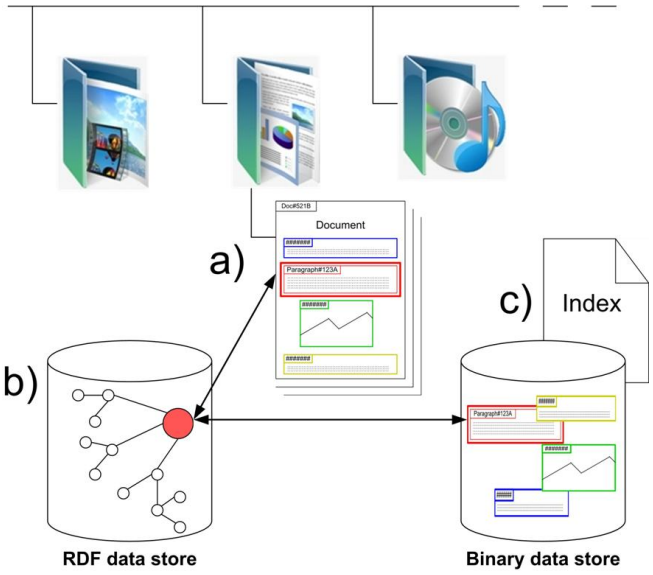

Fig. 5 Semantic Document Store: a) Digital documents with embedded CUs' URIs (HR instances); b) RDF data store (MP document instances); c) Binary data store (CUs binary content)

The MP instance is stored in an RDF repository. Conceptually each RDF node can store string data in an `xsd:string` datatyped literal and binary data in an `xsd:base64Binary` datatyped literal. However, current RDF repositories are

not meant to store larger chunk of binary data [11]. When a large amount of data has to be managed, queries in structured query languages such as SPARQL [19] are not always powerful enough. The MP instance does not approach such problem since it is not meant to store the CU's content. The content stays in the source digital document and can be accessed via links made by embedded URIs. For example, if logical structure of the digital document is described by XML, access to content of the document CUs can be realized by using XPointer[3] references. However, in order to speed up the access to the content while utilizing the MP instance, we extract the content of each CU and store it in a binary store during the transformation process. Content from textual CUs is placed into plain text files, while content from media CUs is placed in appropriate media files. Names of these binary files encrypt the URIs of document CUs and establish the link between the CUs and their binary content. Files stored in the binary store are only accessible through the system's services. The binary content is indexed and enables full-text search as supplement to structured queries over RDF data. Fig. 5 illustrates stores of semantic document components (i.e., HR and MP instances) and relations between them.

4 Social Semantic Desktop (SSD)

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 [3] a vision of how these different thrusts will evolve and produce the SSD The main goal of the SSD is to transform the conventional desktop into a seamless, networked working environment, by loosing the borders between individual applications and the physical workspace of different users [21]. The SSD adopts some of the ideas of the Semantic Web. The aims of allowing data to more easily be shared could be considered as a subset of those of the Semantic Web, but extended to a user's local computer, rather then just files stored on the Internet.

Formal ontologies that capture a shared conceptualization of desktop data, and RDF as a common data representation format, can be used to unify all forms of data and allow data to be accessed in a format independent way. Our aim in building the semantic document model (SDM) on these principles is to enable data from digital desktop documents to take part in the vision of the SSD. Since the SSD is regarded as a building block of the Semantic Web [3], by including semantic documents into the SSD they will also become resources of the Semantic Web. In order to test the developed model and to explore its advantages and drawbacks in a real use, we chose the NEPOMUK SSD [10] platform. In the rest of this section we first outline the architecture of the NEPOMUK SSD platform and then discuss the integration of semantic documents into the platform.

[3] XPointer Framework: http://www.w3.org/TR/xptr-framework/

4.1 Architecture of the NEPOMUK SSD

The NEPOMUK SSD is made up of the user's individual desktops, which are organized in a peer-to-peer (P2P) fashion. The NEPOMUK architecture is organized in three layers: Network Communication Layer, NEPOMUK Semantic Middleware and Presentation Layer.

The role of the Network Communication Layer is to enable the communication between peers (i.e., networked desktops). This layer provides: 1) the Event Based System for distribution of the events between NEPOMUK peers, 2) the Messaging System for routing messages and 3) the Peer-To-Peer File Sharing System which enables the shared information space.

On the top of the Network Communication Layer is the NEPOMUK Semantic Middleware. The role of this layer is to provide core services of the NEPOMUK SSD, to enable the inter-service communication and to establish the infrastructure for possible platform extension with new services. In order to operate within the platform, services first need to be register at the Service Registry. Based on the operating system (e.g., MS Windows, Mac OS, and Linux), different communication techniques such as SOAP over HTTP, OSGI[4] [21], or D-Bus[5] [3] can be used for interaction between the services. The core services are divided into two subsets. The first subset contains services that are more specific in terms of purpose such as *Data Wrapping*, *Context Elicitation*, *Mapping and Alignment*, and *Text Analytics*. The second subset contains *Data Services*, which are more general and usually called by services from the first subset. *Data Services* are also important for our work, and the services that we have developed and added to the NEPOMUK SSD take advantage of them. The *Data Services* control the insertion, modification, and deletion of resources in the NEPOMUK SSD. A resource can be any digital or non-digital entity that is identified with a URI and described by RDF descriptions. For local queries and offline work, the RDF descriptions and resource binary data are stored by the *Data Storage* services in the NEPOMUK RDF data store and NEPOMUK binary data store respectively. If a user wants to share a resource with other users, the RDF descriptions of the resource need to be uploaded to the distributed index of the peer-to-peer file sharing system (i.e., distributed RDF data store). The *Data Search* services can either issue a local search in the local store or a distributed search in the underlying peer-to-peer system or both.

The top layer of the architecture is the Presentation Layer, which provides a user interface to the services provided by the NEPOMUK Semantic Middleware. The aim of the layer is not to build completely new applications and systems for managing different types of resources that can be stored on the NEPOMUK SSD, but to extend existing popular applications, such as Office applications, Email clients and Web browsers so that they can take advantage of the middleware services.

---

[4] OSGi Alliance: http://www.osgi.org/

[5] D-Bus: http://www.freedesktop.org/wiki/Software/dbus

4.2 Semantic Document Management System (SDMS)

In order to integrate semantic documents into NEPOMUK SSD we have developed the SDMS as a set of services that can be integrated into NEPOMUK Middleware. The SDMS enables the transformation of digital documents into semantic documents and storage of MP instances into NEPOMUK RDF data store. Moreover it provides support for different functionalities, which are enabled by the introduced SDM. The SDMS consists of the following services: *Transformation*, *Annotation*, *Indexing*, *Change-Tracing*, *Ranking*, and *Terms-Mapping*.

The Transformation service scans the structure of a digital document to be transformed, recognizes document CUs and generates the MP document instance as defined in the SDM. For each CU (e.g., paragraph, image, and table), the MP instance contains instances of appropriate concepts from the document ontology (e.g., *Paragraph*, *Image*, and *Table*). The service generates a URI for each of those instances and embeds them into the source digital document as hidden-bookmarks. Moreover, the service extracts all textual, audio and video data from the CUs and stores them into the NEPOMUK binary data store. For each identified CU, the *Transformation* service also calls the *Annotation* service to semantically annotate the CU. At the end of the transformation the MP component is generated and the NEPOMUK *Data Storage* service stores it either into local NEPOMUK RDF data store if the user does not want to share the document or into distributed RDF data store.

The Annotation service performs semantic annotation of the CUs with annotations defined in the annotation ontology. It relates them to the RDF nodes within the MP instance. The annotation process is fully automated. Values of standardized metadata are derived from the documents metadata or generated based on the available formatting information. Some metadata is just literally copied from the documents metadata like `dc:creator`, `dcterms:created`, `dc:format`and `dc:language`, referring to the author(s), creation date, media type and language(s) respectively. A value of a dc:title element is generated based on the formatting information. For example, a text fragment with a font style (e.g., *title* or *heading1*) is used as a value for the `dc:title` element of all successive CUs up to the next formatted text fragment. A value of a `dc:description`element is generated out of the values of previously explained elements using the following text pattern: "A content unit of `dc:format` media type with a title `dc:title` authored by `dc:creator`; creation date `dcterms:created`" [9]. The usage metadata comes from capturing interaction between the users and CUs over time. Always when the users interact with the CUs the annotation service generates the usage metadata and relates it to RDF nodes that represent the CUs. Moreover, the service generates the subject-specific metadata by performing the ontology-based information extraction from the CUs. The service first queries a set of specified domain ontologies to find labels of their concepts and then for each found label it generates a set of synonyms (the synonyms are obtained through a lexical ontology such as WordNet). After that, for each CU the service checks if the CU contains some of the labels or their synonyms and if so, relates the labeled concepts to the

CU. Although, in the current implementation of the annotation service we are primarily focused on the automatic CU annotation, we want to stress that the semantic annotation model that we propose does not make any difference between manually and automatically generated annotations. In the near future we plan to enhance the annotation module with support for the manual CU annotation. Moreover, document CUs become sharable resources which can be annotated with unstructured vocabularies (e.g., collaborative tags) as well.

The Indexing service does text indexing of all textual data from the document CUs. The data is indexed after the transformation service places it into binary files. The SDMS maintains a single index, which is updated always when a new document is transformed. The index stores inverse mappings for pairs (CU's URI, CU's text). Text indexing is included in order to supplement structured queries (e.g., SPARQL, RQL and RDQL) on RDF data with full-text search. For the index implementation we use the Apache Lucene[6] [1] IR Library.

The Change-Tracing service inspects the HR document instance and if there are some changes to document CUs creates a change-log as an instance of the change ontology. For changed CUs, the service creates a new VID and relates it to the ontological instance of the CU in the MP document instance. The VID is also added to the CU's hidden-bookmark within the HR instance and along with the CU's URI uniquely identifies the CU.

The Terms-Mapping service maps a set of terms with a set of domain concepts. The service queries the domain ontology for concepts whose labels contain some of the specified terms or their synonyms. Ideally, there is only one ontology for each domain. In reality, however, we are faced with many ontologies of partially overlapping domains (e.g., FOAF, SIOC and hCard for the description of the Web users). The NEPOMUK Semantic Middleware provides the Mapping service that can be used to find related or equivalent concepts from different ontologies. The Terms Mapping service takes advantage of the Mapping service in resolving potential redundancy within the found set of domain concepts. The Terms Mapping service is mostly used as a part of the ontology-based search for document CUs, which are annotated with ontological concepts.

The Ranking service ranks the document CUs within a set of CUs that is retrieved by the Data Search service. The applied ranking algorithm is based on the user's preferences regarding CUs, such as number of CU's versions and occurrences in different documents, which are specified as a part of the user's profile, and weighting schemas, which we have developed for each preference [13]. By applying the weighting schemas, the service first calculates the weight of each CU and than according to CUs weights, ranks them within the retrieved set.

[6] Apache Lucene IR Library: http://lucene.apache.org/

5 Application Examples

Many desktop applications are possible sources of resources that could be managed by the NEPOMUK SSD. By integrating the SDMS services into the NEPOMUK Semantic Middleware we extend the set of possible resources with digital desktop documents, which are first transformed into semantic documents and then stored on the NEPOMUK SSD. Currently, the SDMS supports the transformation of only MS Office documents but in the near future we plan to add support for other common desktop document as well. In order to provide a user interface to the SDMS services for MS Office users, we have developed two MS Office add-ins: Transformer Add-In and Authoring Recommender Add-In [15]. Transformer Add-In enables users to transform MS Office documents into semantic documents, and store them on the NEPOMUK SSD. Authoring Recommender Add-In enables the users to search local and distributed semantic document stores for desired document CUs and incorporate them in their documents. We now briefly describe the main features of these two add-ins from the perspective of the processes in which they participate. Further information, snapshots and demos can be found on the SDMS Web page [24].

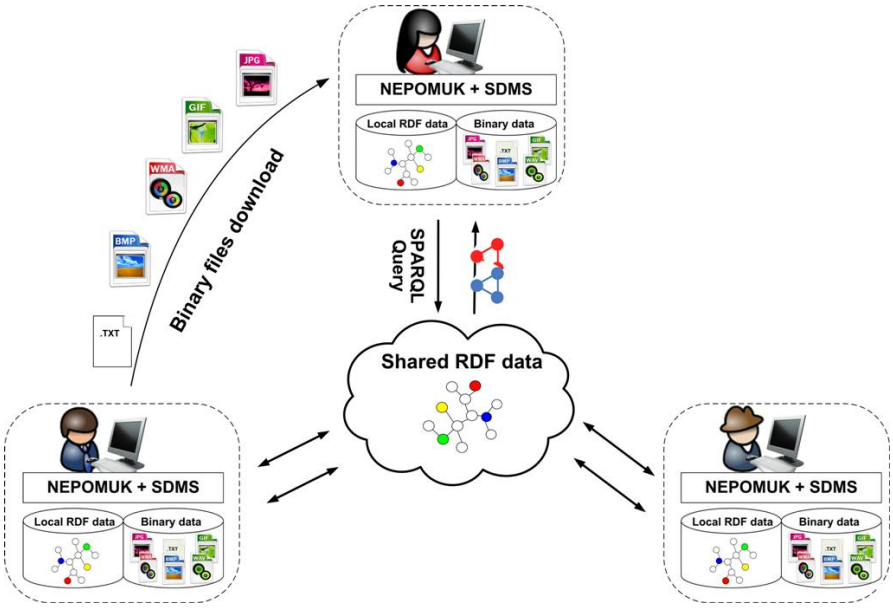

Fig. 6 Sharing document CUs between peers in the Social Network using the SDMS

The transformation process is almost fully automated and the user workload is minimized. The GUI of the Transformer Add-In is simple to use and follows the main design principles of the MS Office GUI. Prior to the transformation, the user can select domain ontologies that describe the tentative topic of the document to be transformed. If the ontology repository, which is a part of the NEPOMUK RDF data

store, does not contain appropriate domain ontology, the user can add the new ontology to the repository and then select it. After the selection of the ontologies the user starts the transformation. During the transformation, the add-in utilizes four SDMS services: *Transformation*, *Annotation*, *Indexing*, and *Change-Tracing*. At the end of the successful transformation, the MP instance of the document is generated and stored in the local NEPOMUK RDF data store or delegated to the distributed store. The HR instance (i.e., MS Word or MS PowerPoint) is extended with embedded CUs' URIs and document CUs' binary data (e.g. text, images, audios) is extracted, indexed and stored in the NEPOMUK binary data store. To store the MP instance as well as binary data the add-in calls the *Data Storage* services.

By integrating semantic documents in the NEPOMUK SSD, semantic documents become part of a collaborative environment, which enables sharing and exchanging of document CUs across social and organizational relationships (see Fig. 6). In order to enable users to search and retrieve document CUs form local or distributed stores while working in MS Office applications we have developed the Authoring Recommender Add-In. Through the GUI of the add-in (see Fig. 7), the user can specify following information: 1) a set of ontologies that conceptualize the domain of interest (see Fig. 7a), 2) a set of tentative terms, and 3) a CU media type (e.g. text, image, audio and video). The add-in then calls the *Data Search* service, which searches the repository(ies) of semantic documents for document CUs by combining the ontology-based and content/text-based search. For the ontology-based search, the add-in first calls the *Terms-Mapping* service to translate the set of specified terms into ontological concepts. The set of ontological concepts along with the specified CU media type are then combined and internally transformed into a query in the SPARQL query language [19]. The *Data Search* service executes the query over the RDF data stores and retrieves the URIs of found CUs. For the full-text search the add-in composes query out of the specified terms and calls the *Indexing* service, which delegates the query to the system's index. The result of the full-text search is also a set of CU URIs. The results of both the ontology-based and the full-text search can be combined in different ways. Since, we prefer to search document CUs based on the phenomena they describe rather than simple keyword matching, the full-text search is just a secondary option, which is used only if the ontology-based search does not return any results.

Once the search is completed, the add-in calls the *Ranking* service, which ranks the retrieved set of CUs. Finally, based on the CU's URI, the *Data Storage* service determines if the CU's content is stored in the local binary data store or in the binary data store of the other NEPOMUK peers, and retrieves the content to the add-in. The add-in provides a preview of the retrieved CUs content as well as the CU's metadata (Fig. 7b). In the same preview, the user can find information about the evolution path of the CU. Once the user selects the CU to reuse, the add-in adds the CU to the current cursor position in the active document. Along with the addition of the CU to the document, the add-in also incorporates a hidden-bookmark with the CU's URI.

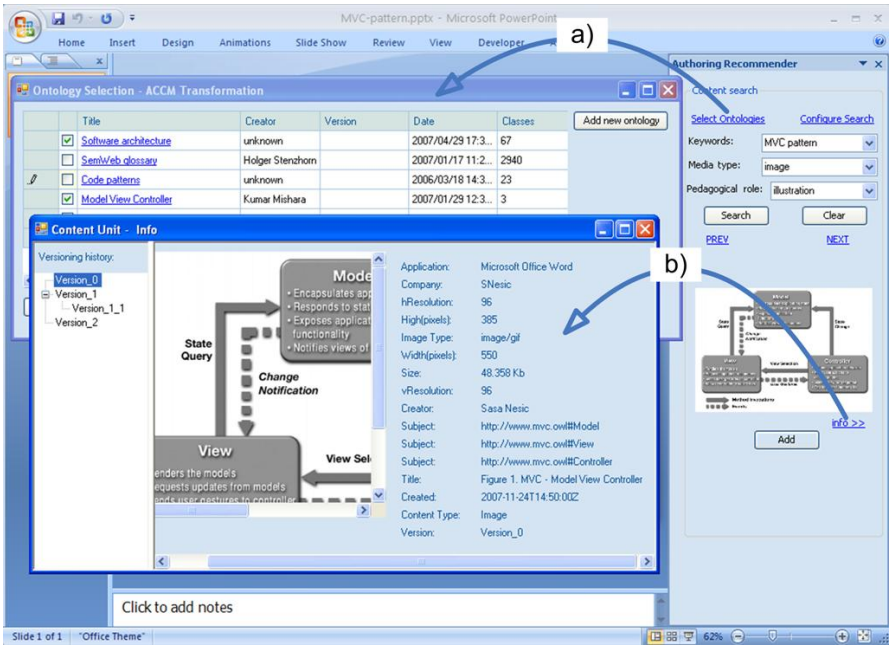

Fig. 7 Authoring Recommender add-in: a) ontology selection b) content unit preview

6 Discussion

The existing types of digital desktop documents (e.g., PDF, MS Office, and OpenOffice) offer a plenty of nice features, and many users have become deeply familiar with them. By using the existing types of documents as HR instances of the semantic documents, we do not put any additional burden on users and let them work with documents in the same manner as before. On the other hand, the semantic documents (i.e., the MP instances) bring new services, which can further improve document usability and make users' lives easier. Having the MP instance represented as an RDF graph and by using structured query languages such as SPARQL, RDQL and RQL, we can reach any part (CU) of the document without knowing anything about the document structure, which is not the case with the regular XML-based documents. By linking annotations to RDF nodes in the MP document instance, we solve the problem of insufficiency of appropriate schema elements for different kinds of annotations. Instead of extending schemas with new elements and attributes, which is very difficult because of the strict schema definitions, the new kinds of annotations can be easily added to the RDF nodes by defining the new properties in the annotation ontology. The links between the MP document instance and HR instances allow users to access and modify document data and annotations stored in the MP instance. Moreover, the RDF-based MP instance allows remote search of semantic documents over Semantic Web protocols [19]. The RDF nodes

are also envisaged to store binary data, but because of the low performance of currently available RDF stores when RDF nodes store large amount of binary data, we place binary data of document CUs into a binary data store. Storing document binary data outside of the MP instance has some advantages as well. Document binary data can be efficiently indexed and searched by using traditional IR techniques. In the search service that we have developed as a part of the SDMS, we combine structured SPARQL queries with full-text search of binary data.

A critical factor of the introduced SDM is the linkage between the HR and MP document instances. The way of the linkage that we propose in the SDM is possible if document format has the logical structure described via a markup language (e.g., XML). However, the use of existing digital documents as the HR instances was chosen purely for technical reasons since the development of completely new tools for processing the new document format is an expensive investment. However, we believe that over time the HR instances will be rendered from the MP instance at the time when humans want to see the document content and there will be no permanent copy of them. Of course, in this case the MP instance will have to keep some formatting information as well. Believing that the permanent copy of the HR instances in a form of today's documents will disappear, we consider the obvious problem of the linkage between the MP and HR instances to be temporary. The MP instance could serve as the only permanent form of the document, which will be rendered into different human readable forms on demand.

The other possible bottleneck of the SDM, which is a concern of the Semantic Web and Knowledge Representation areas as well, is the existence of many overlapping ontologies that conceptualize the same domain. By using different ontologies to annotate documents whose topics belong to the same domain, there is a lack of shared understanding of knowledge modeled within them. There are two solutions for this problem: 1) all people should use universally recognized, standardized domain ontologies, or 2) apply some ontology alignment techniques [7]. Ontology alignment is the process of determining correspondences between ontological concepts. Both solutions, the standardized ontologies and the ontology alignment, have comparative advantages and drawbacks.

Ontology evolution may also have the influence on the semantic documents modeled by the SDM. The semantics of some concept can change over time and become irrelevant for knowledge modeled within the document. Therefore, the semantic documents will need to be periodically checked in order to determine if some ontological concepts are no longer relevant.

In terms of the development of the SDMS we have faced several problems as well. For example, we apply some content analysis techniques to extract information from textual CUs and then use this information to find appropriate ontological concepts and link them to the CUs. However, for media CUs the existing content analysis techniques are more complicated and they are missing in the current implementation of the SDMS. Media CUs are annotated only with information that comes form context and usage analyses. Moreover, the full-text search, which we use as a secondary (optional) search is only possible for local document stores.

7 Related Work

So far, the term 'semantic documents' has mainly been used to describe approaches to combining documents and ontologies [6]. The majority of them are focused on document annotation with ontologies by linking ontological concepts to regions of text and graphics in the document. Some examples of annotation frameworks that apply this kind of annotations are: 1) PDFTab [6], an extension to the Protg ontology editor that allows developers to annotate PDF documents with OWL-based ontologies; 2) Semantic Word [22], which provides a GUI based tool to help analysts annotate MS Word documents with DAML ontologies; 3) SALT [8], an ontology-based authoring framework that allows authors to semantically annotate LATeX documents; and 4) ActiveDoc [12], which enables annotation of documents at three levels: ontology-based content annotation, free text statements and on-demand document enrichment. We have identified several general shortcomings that characterize the document annotation approach used in all above listed frameworks. Firstly, they all try to store annotations and their definitions (i.e., ontological concepts and properties) inside the document's internal representation, so that the annotation is strongly dependent on available schema elements that are designed to store additional information and on provided linkage mechanisms. Secondly, ontological concepts are related to regions of the document, which are usually delimited by schema-defined structural elements, or by their size and position within the document. These parts of documents are usually not uniquely identified and are hardly addressable. Different schemas have different definitions for the same type of structural elements, so that during the document transformation the links between the annotations and appropriate structural elements are mainly lost. Finally, in order to reach document annotations, knowledge on a document schema is necessary. In spite of these shortcomings, the document annotation with ontologies and the developed frameworks has improved discoverability of document content. However, unlike our approach, these approaches do nothing about document decomposition, CUs versioning, and tracking information about the CUs usage and changing. Their contribution in improving data interoperability across application boundaries is also minor. In order to access some data from one application-specific document within other applications, it is still necessary to apply export/import functions to transform document data into an appropriate application format.

As our approach does not only consider a problem of semantic annotation, but also document decomposition into reusable and uniquely identified document CUs, we want to compare our approach with some existing Composite Document Models/Frameworks. Accordingly, we found a comparison with OpenDoc [17] to be enlightening. OpenDoc envisages a document as being composed of material contributed from a variety of sources such as MacWrite, Adobe Photoshop and Adobe Illustrator. Each piece of material is rendered by calling on the appropriate application at the appropriate time. The main and crucial difference to our approach is that pieces of materials, which are used in composing the document, exist as parts of diverse application specific documents, while in our approach they are considered as resources that exist independently of the implementation form. 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.

In terms of applications, we find it interesting to compare our work to the ALOCoM framework [9, 24]. The goal of this framework is sharing and reusing of document CUs of different granularity based on their semantic annotation. The framework decomposes document content into CUs, enriches them with a set of extracted metadata and then stored them into a centralized repository. In this way the extracted document CUs are no longer a part of the document context and their further evolution has no any effect to the document. On the contrary, in our approach we transform complete documents into the semantic documents and document CUs remain parts of the document context. Also, ALOCoM CUs are not considered as unique resources, which can be included in different contexts. Instead, they are considered as annotated pieces of the document content, which can be copied and reused many times. Moreover, in our approach we store semantic documents on the local desktop, which is networked over the NEPOMUK SSD platform. This enables users to access and retrieve document CUs directly from the other users, instead of searching the centralized repository as it is the case with the ALOCoM.

8 Conclusions

In this paper, we have presented the Semantic Document Model (SDM), which defines semantic documents as composite resources with uniquely identified and semantically annotated CUs, whose data and knowledge can be represented in two forms: human readable (HR) and machine processable (MP). The full potential of the model comes from the MP document instance, which is unique and platform independent and can be accessed from any HR document instances. We have chosen the Semantic Web technologies, in particular ontologies and RDF, as the basis of the proposed MP instance. The proposed MP instance enables: unique CUs identification; flexible semantic annotation of document CUs with different types of annotations; conceptualized representation of document CUs knowledge; and capturing and formal representation of changes made to CUs over time. The user can query the MP instance over Semantic Web protocols and access and retrieve document CUs based on their semantics and conceptualized knowledge. The intelligent software agents can potentially use conceptualized knowledge from the MP instance to answer some domain questions. In accordance with the proposed model and MP instance, we built the SDMS for managing documents represented by the model (i.e., Semantic Documents). The services provided by the SDMS are currently integrated into MS Office (e.g., MS Word and PowerPoint) through the add-ins (i.e., Transformer add-in and Authoring Recommender add-in), which we developed. These add-ins enable office users to transform MS Office documents (i.e., Word and PowerPoint) into semantic documents and search local and distant repositories of semantic documents for document CUs by executing remote SPARQL queries. More importantly, this facilitates a collaboration of users by being able to seamlessly ex-

change their document CUs. To achieve this, we take advantage of some of the services provided by the NEPOMUK SSD platform.

In the future work, we plan to perform some studies to evaluate the proposed model and to figure out its impact on the document authoring in collaborative environments such as the NEPOMUK SSD. Moreover, we plan to work on capturing different aspects of the interaction between users and document CUs and to apply such observed results to further improve discoverability and retrieval of document CUs. Application support for other common documents, such as PDF and OpenOffice, is also something we plan to work on.

Acknowledgements Research reported in this paper has been partially financed by the European Commission in the NEPOMUK (IST-FP6-027705) project.

References

1. Berger, L.P., Luckmann, T.: The Social Construction of Reality: A Treatise its the Sociology of Knowledge. Anchor Books, pp. 51-55, 59-61 New York (1966)
2. Berners-Lee, T., Hendler, J. and Lassila, O.: The Semantic Web. Scientific Am., 2001, pp. 34-43.
3. Decker, S., Frank, M.: The social semantic desktop. WWW2004 Workshop Application Design, Development and Implementation Issues in the Semantic Web (2004)
4. Dill, S., Eiron, N., Gibson, D., Gruhl, D., Guha, R., Jhingran, A., Kanungo, T., McCurley, K.S., Rajagopalan, S., Tomkins, A., Tomlin, J.A., Zien, J.Y.: A case for automated large-scale semantic annotation. J. Web Semantics 1 (1), (2003)
5. Ecma International: Standard ECMA-376, Office Open XML File Formats, Dec. 2006.
6. Eriksson, H.: The semantic-document approach to combining documents and ontologies. Int'l Journal of Human-Computer Studies, 65(7), pp. 642-639. (2007)
7. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer-Verlag, Berlin (2007)
8. Groza, T., Handschuh, S., Moller, K. and Decker, S. SALT -Semantically Annotated LATEX for Scientific Publications. In: 4th European Semantic Web Conference, (2007)
9. Jovanović, J., Gašević, D., Devedžić, V.: Ontology-based Automatic Annotation of Learning Content. International Journal on Semantic Web and Information Systems. 2(2), pp. 91-119. (2006)
10. Handschuh, S., Groza, T., Moller, M., Grimnes, G., Sauermann L., Jazayeri, M., Mesnage, C., Reif, G., Gudjonsdottir, R.: The Nepomuk Project On the Way to the Social Semantic Desktop. In: I-Semantic 07', pp. 201-211 (2007)
11. Harth, A., Decker, S.: Optimized Index Structures for Querying RDF from the Web. 3rd Latin American Web Congress (2005)
12. Lanfranchi, V., Ciravegna1, F., Petrelli, D.: Semantic Web-based document: editing and browsing in AktiveDoc. In: 2nd European Semantic Web Conference, Heraklion, Greece (2005)
13. Nešić, S., Gašević, D., Jazayeri, M.: An Ontology-Based Framework for Authoring Assisted by Recommendation. In: 7th ICALT Conference, pp. 227-231. (2007)
14. Nešić, S., Jovanović, J., Gašević, D., Jazayeri, M.: Ontology Based Content Model for Scalable Content Reuse. In: 4th ACM K-CAP Conf.2007, pp. 195-196. (2007)
15. Nešić, S., Gašević, D., Jazayeri, M.: Extending MS Office for Sharing Document Content Units Over the Semantic Web. In: 8th International Conference on Web Engineering (2008).
16. OASIS Consortium: Open Document Format for Office Applications, Version 1.1 (2007)

17. OpenDoc Programmers' Guide, Addison Wesley Publishing Company, 1995. ISBN 0-202-47954-0.
18. Priestley, M. DITA XML: a reuse by reference architecturefor technical documentation. In: 19th International Conference on Computer Documentation, pp. 152-156. (2001)
19. Prudhommeaux, E., Seaborne, A.: SPARQL Query Language for RDF. http: //www.w3.org/TR/rdf- sparql- query/ (2007)
20. Stabb, S., Studer, R.: Handbook on Ontologies, Springer, Berlin, (2004)
21. Sintek, M., Elst, L., Scerri, S., Handschuh, S.: Distributed Knowledge Representation on the Social Semantic Desktop: Named Graphs, Views and Roles in NRL. In: 4th European Semantic Web Conference, pp. 594-608. Innsbruck (2007)
22. Tallis, M. SemanticWord processing for content authors. In: Knowledge Markup and Semantic Annotation Workshop at 2nd K-CAP conf., Sanibel, Florida USA, (2003)
23. Uren, V., Cimiano, P., Iria, J., Handschuh, S., Vargas-Vera, M., Motta, E., Ciravegna, F.: Semantic annotation for knowledge management: Requirements and a survey of the state of the art. J. Web Semantics: Science, Services and Agents on the World Wide Web, 4(1), pp. 14-28 (2006)
24. Verbert, K., Gašević, D., Jovanović, J., Duval, E.: Ontology-based Learning Content Repurposing: The ALOCoM Framework. Intl Journal on E-Learning, 5(1), pp. 67-74. (2006)