An Extensible Framework for Efficient Document Management Using RDF and OWL

Erica Meena
Laboratory LORIA
Vandoeuvre-les-Nancy
France
meena@loria.fr

Ashwani Kumar
M.I.T
Cambridge, MA
USA
ashwani@mit.edu

Laurent Romary
Laboratory LORIA
Vandoeuvre-les-Nancy
France
romary@loria.fr

Abstract

In this paper, we describe an integrated approach towards dealing with various semantic and structural issues associated with document management. We provide motivations for using XML, RDF and OWL in building a seamless architecture to serve not only as a document exchange service but also to enable higher level services such as annotations, metadata access and querying. The key idea is to manifest differential treatments for the actual document structure, semantic content of the document and ontological document organization. The deployment of this architecture in the PROTEUS project[1] provides an industrial setting for evaluation and further specification.

1. Introduction

Digital documents are ubiquitously used to encode, preserve as well as exchange useful information in order to accomplish information sharing across the community. As the growth in volumes of digital data is exponential, it is necessary to adopt a principled way of managing these documents. Besides, due to the distributed nature of information, it is also imperative to take into account the geographical and enterprise-level barriers for uniform data access and retrieval.

The ITEA (Information Technology for European Advancement) project, Proteus[2] has similar objectives. Proteus is a collaborative initiative of French, German and Belgium companies, universities and research institutes aimed at developing a European generic software platform usable for implementation of web-based e-maintenance centers. It implements a generic architecture for integrated document management using enabling technologies such as XML, RDF and OWL. Most of the existing document management systems ([1], [2]) limit themselves in the scope of application or document formats or simply neglect any structure-based analysis. However, considering our requirements, it is obvious that only a multi-layered functional architecture can cover various issues related to distributed document management such as localized vs global structural constraints, conceptual definition of documents, reasoning-based discovery etc.

Indeed, evolving technologies such as XML (eXtensible Markup Language), RDF (Resource Description Framework) and OWL (Web Ontology Language) provide us with rich set of application frameworks that if applied intelligently, can help a great deal in solving these problems. XML ([3]) is primarily designed for low-level structural descriptions. It provides a tree of structured nodes, which can be efficiently used to describe documents and check their models using DTDs (Document Type Definitions) or XML Schemas. Besides, XML enables easy human readability as well as efficient machine interpretability. However, there are issues if we only deal with the structural aspect. If one wants to pick some semantic information from a document, there is no straightforward way other than to constrain it by an schema or make an application hand-programmed to recognize certain document-specific semantics. Furthermore, if the schema changes over time, it could typically introduce new intermediate elements. This might have the consequences of invalidating certain queries and creating incoherencies in the semantic data-model of the document.

RDF (Resource Description Framework) and OWL (Web Ontology Language) build upon the XML syntax to describe the actual semantics of a document and provide useful reasoning and inference mechanisms. RDF ([4]) specifies graphs of nodes, which are connected by directed arcs representing relational predicates such as URIs (Uniform Resource Identifiers) and encode the

[1] This material is based upon **work supported** by the ITEA (Information Technology for European Advancement) programme under Grant 01011 (2002).
[2] http://www.proteus-iteaproject.com/

conceptual model of the real world. Unlike XML, an RDF schema is a simple vocabulary language. The parse of the semantic graph results in a set of triples, which mimic predicate-argument conceptual structures. OWL can be used on top of these semantic structures to do logical reasoning and discover relations that are not explicit and obvious.

In the following sections we discuss how we use these technologies to enable a generic document management system. Firstly, in Section 2 we describe the document management and the Proteus architecture followed by discussion on Annotations in Section 3. Section 4 provides brief account of the model theoretic access mechanisms enabled by OWL followed by description of data categories in Section 5.

2. Document Management Architecture

Without differentiating at the level of content, layout and formats, we treat documents as information resources. These information resources can potentially be distributed across various document repositories called e-Doc servers. Figure 2.1 demonstrates a simplified distributed document management system. The architecture shows how three different document repositories could co-exist functionally along with the Annotea enabled annotation framework ([5]). These servers implement procedural mechanisms for query access and retrieval of documents. Besides, these documents can be annotated and the annotations reside on an independent server known as the annotation server, which also serves as a document server. Principally, annotations can be viewed as information resources, which are described in RDF.

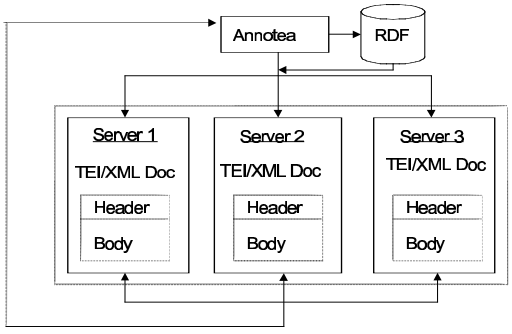


Figure 2.1: Simplified view of the distributed document server architecture

The e-Doc server consists of several functional layers that inter-communicate and holistically, serve the cumulative purpose of document management. These layers though distinct at the level of data flow and individual processing of information, afford functionalities that are exploited by the e-Doc server.

Figure 2.2 shows various such layers of the e-Doc server. On the foundation level, it is assumed that every document on the e-Doc server adheres to a single syntax i.e. XML, which represents the top most layer in the architecture. The second layer depicts the access points that are broadly categorized along various dimensions such as metadata, conceptual/ontology system and terminology. A detailed description of the access points will be carried out in the Section 4. The e-Doc server is assumed to be flexible enough to handle all possible ontology formats/standards whether it is a native XML document or a text or a picture/video data coming from some streaming applications. This forms the third important layer of the e-Doc server. The bottom layer represents Annotations [6], which adheres to the RDF [4] syntax. This layer forms an integral part of the e-Doc server as it enables annotation capability and RDF-describable semantics to the actively retrieved document or existing documents in the server [7]. Besides, RDF also provides the opportunity to utilize annotations as access points for the documents.

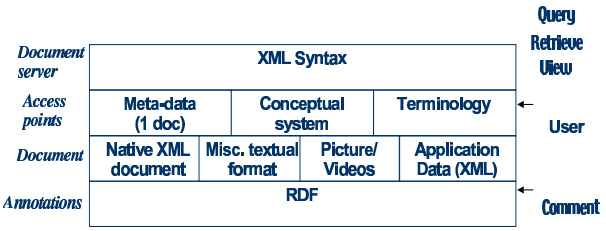


Figure 2.2: General Organization of the e-Doc Server

As can be seen from the Figure 2.2, a user interacts with the server through a client interface by launching his queries. The architecture provides the user ample flexibility in utilizing different levels of descriptions for retrieving documents by providing variety of access points. In the following sections, we describe each of these access layers in more detail.

3. Annotations: Specified as RDF Model

Annotations form the most abstract layer within the e-Doc architecture. They can be broadly defined as comments, notes, explanations, or other types of external remarks that can be attached to either a document or a sub portion of a document. As annotations are considered external, it is possible to annotate a document as a whole or in part without actually editing its content or structure. Conceptually, annotations can be considered

as metadata, as they give additional information about an existing piece of data. Annotations can have many distinguishing properties, which can be broadly classified as:-

Physical location:- An annotation can be stored locally or on one or more annotation servers;
Scope:- An annotation can be associated with a document as a whole or to a sub-portion of a document.
Annotation type:- Annotations can have various functional types such as, Comment , Remark , Query e.t.c.

Due to this abstract nature and multiplicity of functional types, a formal treatment of annotations is often unwieldy. Therefore, it is desired to have a semantically driven structural representation for annotations, which we describe below.

Annotation Semantics

Annotations are stored in one or multiple annotation servers. These servers endorse exchange protocols as specified by Annotea [5]. Essentially, the Annotation Server can be regarded as a general purpose RDF store, with additional mechanisms for optimized queries and access. This RDF store is built on top of a general SQL store. Annotations are stored in a generic RDF database accessible through an Apache HTTP server (see Figure 3.1). All communication between a client and an annotation server uses the standard HTTP methods such as POST or GET.

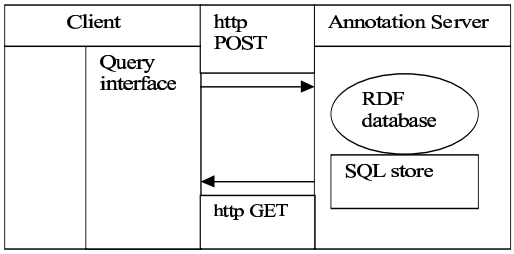


Figure 3.1: Access to the Annotation server

Annotations have metadata associated with them, which is modeled according to an RDF schema and encode information such as date of creation of the annotation, name of the author, the annotation type (e.g. comment, query, correction) the URI [8] of the annotated document, and an Xpointer [9] that specifies what part of the document was annotated, and the URI to the body of the annotation which is assumed to be an XHTML [10] document (Figure 3.2).

```
<rdf:RDF
xmlns:NS0='http://www.w3.org/2000/10
/annotation-ns#'
 >
<dc:creator>Ashwani</dc:creator>
<rdf:type
rdf:resource='http://www.w3.org/2000
/10/annotation-ns#Annotation'/>
<NS1:origin rdf:nodeID='A0'/>
<NS0:created>2004-05-
24T01:11Z</NS0:created>
<NS0:annotates
rdf:resource='http://docB4.teiSpec.o
rg'/>    <rdf:type
rdf:resource='http://www.w3.org/2000
/10/annotationType#Comment'/>
<NS0:body rdf:resource='Please re-
view this document.'/>
<dc:title>review</dc:title>
<dc:date>2004-05-24T01:11Z</dc:date>
</rdf:Description>  <rdf:Description
rdf:nodeID='A1'>
 .
</rdf:Description>
</rdf:RDF>
```

Figure 3.2: An abridged Annotation in RDF

Xpointers are used to point to the Annotated portions within the document, while Xlinks [11] are used to setup a link between the Document and it's annotation.

Annotation Operations

The user makes a selection of the text to be annotated and provides the annotation along with other details such as author name, date of creation, type of annotation, URI of the annotated document etc. The annotations are published using standard HTTP POST method. To do this the client generates an RDF description of the annotation that includes the metadata and the body and sends it to the server. The annotation server receives the data and assigns a URI to the annotation i.e. the body, while metadata is identified by the URI of the Document.

For annotation retrieval, the client queries the annotation server via the HTTP GET method, requesting the annotation metadata by means of the document's URI. The annotation server replies with an RDF-specified list of the annotation metadata. For each list of annotations that the client receives, it parses the metadata of each annotation, resolves the Xpointer of the annotation, and if successful, highlights the annotated text. If the user clicks on the highlighted text, the browser uses an HTTP GET method to fetch the body of the annotation from the URI specified in the metadata.

The following are the broad categories of the annotation functions implemented by the annotation server:

Annotate a document as a whole.

Annotate a portion of a document.
Query to access all the annotations for a particular document.
Query to access type specific or any of the metadata property specific annotations, which serve as query parameters for all the annotated documents.

4. Model Based Access: Using OWL

As described in the previous section, the RDF layer provides an enhanced mechanism for querying and accessing a document. However, to enable full-fledged management of documents, it is imperative to incorporate some reasoning-based abstract semantics such as OWL (Web Ontology Language) over a cluster of documents. OWL provides formal mechanisms for describing ontology of documents. By doing so, the architecture can provide flexible access points as well as logical inference mechanisms, which are necessary while performing metadata queries.

Access points play an important role by providing flexibility and intuitiveness in access mechanisms to the user. Figure 4.1 depicts a very basic characterization of the access points. As it is illustrated in the figure, a specific access point is needed to direct a query to attain certain desired result set. Within the Proteus framework, the e-doc architecture provides a model driven specification of access points such as metadata-based, ontological, or terminological model. The model driven approach has strong significance in the sense that every access point is associated by certain abstract information structure so that it provides transparency to the queries, which remain independent from actual implementation and data formats (e.g. XML DTD). Even though these models are independent, they are flexible enough to interact among themselves. For example, results of queries on one model can act as a reference for another model. The references may be transformed into document excerpts by requests made synchronously at the query stage or asynchronously when the user wants to visualize the information.

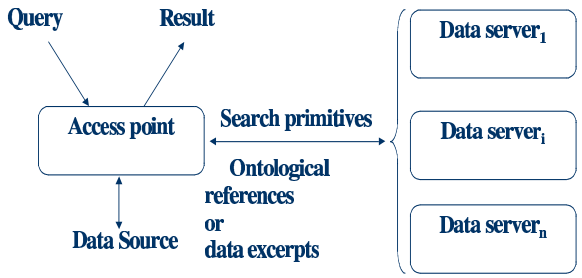


Figure 4.1 Characterization of Access Points

Terminological Access:

Terminology can be defined as the description of specialized vocabulary of an application domain. As it contains a nomenclature of technical terms, it is capable of providing a conceptual view of the domain. Terminology can be either monolingual or multilingual by nature. Monolinguality specifies a one to one relation between a term and a concept or a term to its equivalences or a term to the related documents, while multilinguality specifies relation between term to certain target terms or term to certain target documents.

The Following is a simplified Proteus terminology example:

```
<struct type= TE >[Terminological
Entry]
 <feat type= definition
 xml:lang= fr > Dispositif
 permettant d'imprimer un
 d□placement lin□aire ou angulaire □
 un □l□ment mobile. </feat>
<struct type= LS >[Language Section]
<feat type= language >fr</feat>
<struct type= TS >[Term Section]
<feat type= term >v□rin</feat>
<struct type= TCS >[Term Component
Section]
    <feat type= partOfSpeech >
          noun
    </feat>
 ..
</struct>
```

Description of terminological model

A general terminological model contains a Terminological entry section, a Language section and a Term section.

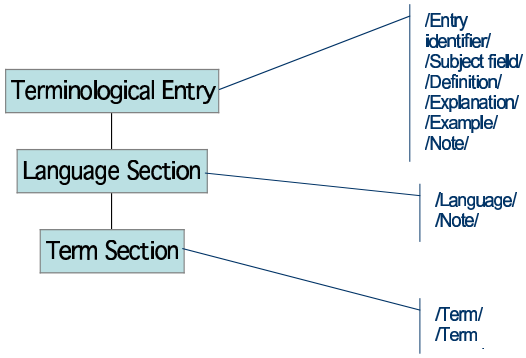


Figure 4.2 Simplified terminological model

Figure 4.2 describes a simplified terminology model - the terminological section contains entries such as identifier, subject field, definition, and explanations etc., where as the other sections such as the language and the term sections contain details regarding the language

used and the term status respectively. This can also be seen within the sample Proteus terminology described above.

Terminological access is significant in cases where the user is aware of the specific term and needs to make a search within the related domain to access certain documents of his interest. For example, an operator of a firm might be willing to retrieve all the maintenance documents related to the term Pump .Thanks to the terminological access point, the operator needs nothing but just the term to launch his query and retrieve the desired document. The above-mentioned scenario is depicted in Figure 4.3

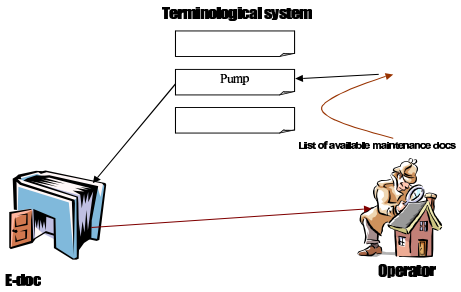


Figure 4.3: A Sample Terminological Access

Terminological access plays a dual role. On one hand it acts as a data source providing support for finding mono or multilingual equivalences or linguistic descriptions. On the other hand, it provides access for on-line documents. When seen as a data source, it can also provide indexing support for manual indexing and can perform semi-automated indexing:

> Graphic files (drawings, pictures, video, scanned texts etc): manual indexing
>
> Text files: semi-automatic indexing; suggestion of descriptors to be confirmed by a human expert
>
> Data, e.g. from monitoring: automatic indexing with metadata.

Terminological model serves as a gateway to the Ontology-based Conceptual model of the domain (Figure 4.4). Use of a technical term as a query parameter relates to set of relevant concepts, which can further be used to retrieve the desired set of documents.

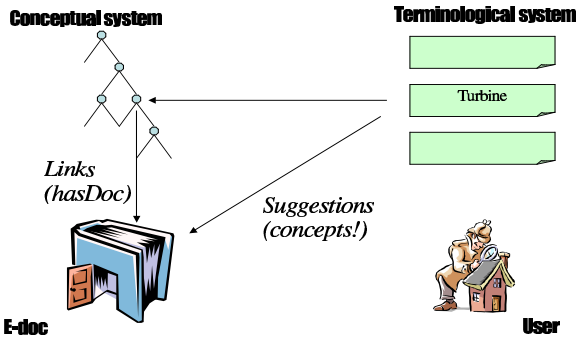


Figure 4.4 interaction of terminological model with other models

Meta-Data Access

Metadata can be loosely defined as dataabout data .. Specifically, metadata encodes certain attributive information about the data, in our case documents, which can be used to access data. Within this platform the meta-model can be seen as a meta-tree of nodes in which every node refers to certain precise set of information descriptors. For example, Dublin Core descriptors such as title, author, date, publisher, etc can potentially be represented as nodes in the description trees.

Meta-model Description

This Meta-model is discussed keeping the specific Dublin Core [12] model in mind. Meta model consists of three basic components, a Resource, an Element, and its value.

> Resource the object being described.
>
> Element a characteristic or property of the Resource.
>
> Value the literal value corresponding to the Element.

Figure 4. 5 shows a simplified view of the Dublin core reference model, within which the Element Qualifiers are nothing but additional attributes that further specify the relationship of the element to the resource. On the other hand, the value qualifiers can be described as additional attributes that further specify the relationship of the value to the element.

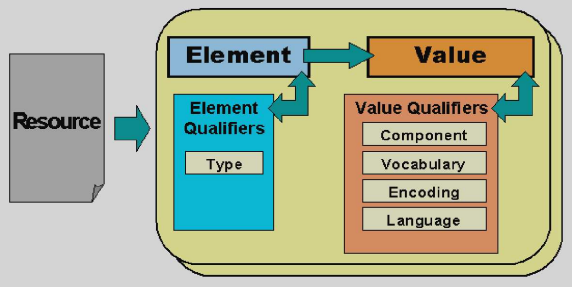


Figure 4.5: Simplified view of the Dublin Core reference model.

For Example:

Element = Creator
Component = Firstname Value = Ashwani
Component = Lastname Value = Kumar
Component = Email Value = ashwani@kumar.com
Element = Contributor
Value = fn:Erica Meena; org:DSTC
Type = Illustrator
Encoding = vCard
Resource = http://www.loria.fr/projets/proteus/RDU/NOTE-PIR 20040304.html

Access of documents by means of metadata is a very important as well as a practical usage, as the user can directly retrieve a well defined piece of information, under the condition that he knows a small number of facts about the information: e.g. the authors name, the date, the reference number or the date of a previous maintenance. This corresponds to a typical situation within the Proteus framework (see Figure 4.6). Metadata access, in other way, can be seen as an advanced index functionality, which can update itself and grow automatically in the same form as the amount of stored information grows.
For example:
While sorting documents by date or type, the date, time, source or author information can always be automatically collected. However, in case of a new maintenance document, advanced metadata can be collected by asking a human to enter it into the system.

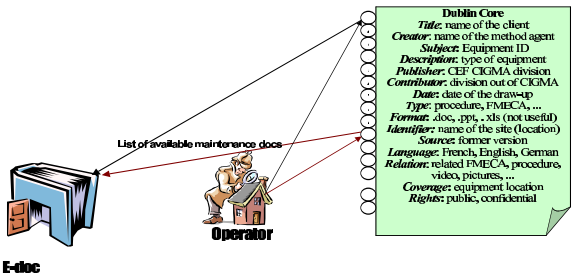


Figure 4.6: Document access via metadata

Metadata model can be seen as an enhanced search mechanism. A sequence of access points i.e. terminology followed by metadata, when launched can help in refining the search along an attribute dimension.

Ontology Access

Ontology is a hierarchy of concepts or in other way a platform for describing the concepts used within a specific domain, maintenance in our case. Its independence with regard to specific model or format makes it interoperable. For example, one can have an ontology represented in a UML [13] class diagram whereas the same ontology can be represented in an XML schema. As already discussed, Ontology is complementary to terminology in terms of attribution of concepts to terms. Conceptually, it serves as an abstract structure, which can be populated by the interested parties and thus, can serve as a very important access point. An abridged example of an abstract Proteus OWL [14] ontology version can be seen in the figure 4.7

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:owl="http://www.w3.org/2002/07/ow
l#"
xmlns:gmoloc="http://www.proteus.com/GM
O/locations#">
   <owl:Ontology rdf:about="">
   <rdf:comment>The Engineering compo-
nent of the PIR</rdf:comment>
  </owl:Ontology>
    <owl:Class rdf:ID="Contract">
     <rdfs:subClassOf>
        <owl:Class rdf:ID="Document"/>
     </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Manager">
     <rdfs:subClassOf>
        <owl:Class rdf:ID="Actor"/>
      </rdfs:subClassOf>
  </owl:Class>

</rdf:RDF>
```

Figure 4.7: An example of Proteus ontology

Ontology model description

As per the requirements of the Proteus project, an ontology model comprises of a three-tiered structure. The three layers consist of General concepts (General Maintenance Ontology), Application Profiles, and the industrial contexts respectively. These layers are built up keeping in mind the interoperability with other external applications. As can be seen from the Figure 4.8 below, the general concept layer has the highest interoperability as it contains basic level concepts such as Actors, Documents, Location, Equipments etc. The

second layer (Application Profiles) consists of concepts, which are specific to a certain application, for instance pertinent to a train manufacturing company, or an aviation company. All the layers are bound to inherit concepts, but not necessarily all from the first layer (general concept), which in turn forms the parent layer of all other layers. The third layer (Industrial contexts) contains concepts very specific to an industry for instance, car manufacturing companies such as Ford, GM etc. Instances can be derived only from the last layer i.e. the Industrial contexts layer.

The model is open for external sources i.e. ontology from external sources can be merged within each layer, for example, SUMO [15], which is a higher-level ontology. It contains very general concepts, which can be used directly within our ontology.

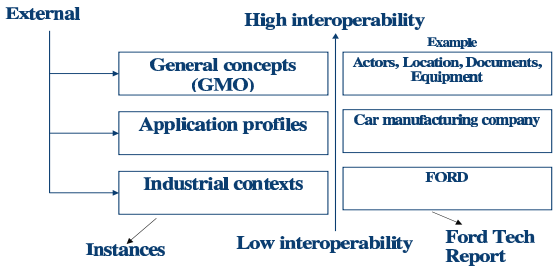


Figure 4.8: Proteus Ontology model

OWL-DL is used for specifying the ontological model as it provides the following advantages:

Basic support for describing classification hierarchies and simple constraint features. e.g. migration path for thesauri and other taxonomies;
Rich expressiveness;
Computational completeness and decidability;
Allows imports of OWL Lite simple description;
Allows consistency checks across description levels;
Existence of optimized inference platforms. E.g. Racer [16].

In a way, ontology access is a complementary approach to the terminology access, as terminology structure describes the global concept behind a thematic domain, but does not deliver a functional description of the domain. The ontology access exactly provides this functional description (as is usually needed in the maintenance domain). The concept remains global when referring to a generic class of entities and gets specific when describing a particular entity type. Apart from the normal functionality of this access point, it can be very important when combined with retrieval by natural language and by visual elements (hierarchy structured sets of pictures). In a way we can see ontology as an empty structure with user-defined class relationships, which can be filled with visual elements (photos, drawing, scheme) and then the referring terms.

For example Figure 4.9 depicts visual search of documents via ontology. In order to avoid complexity, only recommended terms are used to name the objects represented by the visual elements. Other terms can be left apart pointing to plain concepts (without visual concepts). The index of the metadata tool could be virtually integrated into the index administrated by the terminology tool. This enables a two-step-search, beginning with a word and then finding the actually searched item not by selecting a more specific term from the terminology tool, but by looking for a picture of the searched item in the functional concept. This index could also be virtually integrated into the index of the functional concept. Thus the user could situate the search results provided by the metadata tool within the functional structure of the maintained equipment (instead of getting designation, ID-Number, description and meta data only).

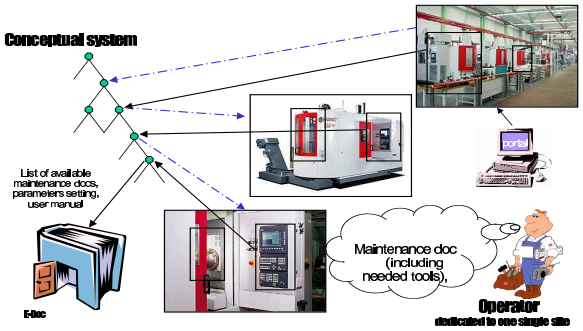


Figure 4.9: Visual search of documents via ontology

5. Data Category Specification

The various models (terminology, annotations, etc.) and functionalities (access primitives to an e-doc server) have to be defined in such a way that a similar piece of information (e.g. author, subject field, term, etc.) means the same thing from one place to another. Such a semantic definition of data categories (in the terminology of ISO committee TC 37) acts in complementary to an ontology such as the one we define in the Proteus system since it is intended to be a general purpose layer of descriptors that may be used in other environments than that of a specific project. Therefore, we adopted a similar methodology as that of the efforts within the ISO TC 37 committee to deploy a data category registry of all descriptors used in the project as reference semantic units described in accordance to ISO standard 11179

(metadata registries). Such a registry plays a double role:

It provides unique entry point (of formal public identifier) for any model that refers to it;
It gives a precise description of the data category by means of a definition and associated documentation (examples, application notes, etc.).

6. Conclusions

We have provided a brief account of how document structure and inherent semantics can be captured and processed efficiently by the emerging technologies such as XML, RDF and OWL. By doing so, we have brought innovations in correlating different levels of document management with respect to various services afforded by these technologies. The differential treatment of structure, content and organization provides ample flexibility and extensibility, which are the primary requirements for such a system.

References

[1]Lagoze C, Dienst - An Architecture for Distributed Document Libraries, Communications of the ACM, Vol. 38, No 4, April 1995. 12

[2]Satoshi Wakayama, Yasuki Ito, Toshihiko Fukuda and Kanji Kato, Distributed Object-Based Applications for Document Management, Hitachi Review Vol. 47 (1998), No.6

[3]Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Extensible Markup Language (XML) 1.0., eds.W3C Recommendation 10-February-1998.

[4]Swick Lassila, Resource Description Framework (RDF) Model and Syntax Specification., World Wide Web Consortium Recommendation, 1999. *http://www.w3.org/TR/REC-rdf-syntax/.*

[5]Jos□Kahan, Marja-Riitta Koivunen, Eric Prud'Hommeaux, and Ralph R. Swick, Annotea: An Open RDF Infrastructure for shared Web Annotations, in *Proc. of the WWW10 International Conference*, Hong Kong, May 2001.

[6]The W3C Collaborative Web Annotation Project ... or how to have fun while building an RDF infrastructure. *http://www.w3.org/2000/Talks/www9-annotations/Overview.html.*

[7]N. F. Noy, M. Sintek, S. Decker, M. Crubezy, R. W. Fergerson, & M. A. Musen. Creating Semantic Web Contents with Protege-2000. *IEEE Intelligent Systems* 16(2):60-71, 2001.

[8]T. Berners-Lee, R. Fielding, and L. Masinter, Uniform Resource Identifiers (URI): Generic Syntax, IETF Draft Standard August 1998 (RFC 2396).

[9]XML Pointer Language. *http://www.w3.orgwtr/xptr/*

[10]The Extensible HyperText Markup Language. *http://www.w3.org/TR/xhtml1/*

[11]XML Linking Language. *http://www.w3.org/TR/xlink/*

[12]Dublin Core Metadata Initiative. OCLC, Dublin Ohio. *http://purl.org/dc/* .

[13]Unified Modeling Language Home Page. *http://uml.org/* .

[14]Deborah L. McGuinness and Frank van Harmelen, OWL Web Ontology Language Overview, W3C Proposed Recommendation, 15 December 2003. *http://www.w3.org/TR/owl-features/.*

[15]Niles, I., and Pease, A. 2001. Towards a Standard Upper Ontology. In Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001), Chris Welty and Barry Smith, eds, Ogunquit, Maine, October 17-19, 2001.

[16]V.Haarslev and R. Moller. Description of the RACER system and its applications. In *DL2001 Workshop on Description Logics, Stanford, CA, 2001.*