An Ontology based document management

Jan Hreno[1] and Robert Kende[1]

1 Abstract

In this article an approach to the problem of associations of documents with a knowledge base is demonstrated in a real world application. It is based on combination of annotating documents with concepts from a knowledge base and grouping documents together into clusters. Our knowledge base is an ontology provided by a dedicated ontology server.

2 Introduction

WWW is slightly becoming the most important communication medium in a last time. There are many reasons for this, but the fact is that most people access information on Internet using web services. Usually, WWW provides one-way communication from publisher to user. In this case we meet a problem of huge amount of unstructured information when it is not easy to find relevant document. This is well known problem for which many techniques are being developing like intelligent search engines or ambitious Semantic Web initiative.

However, WWW can be also successfully used in two-way communication between two sides. Such a communication involves discussion, polling, chat, predefined reports, questionnaires, query systems etc., and of course, the classical publishing. Here the problem of too much information arises again, but new requirement appears in addition. We don't only want to be lost in available information space but also want from the system to control our communication, make advises, select or notify the right agent (usually person) on the other side, so that the communication was efficient. The need of user friendly and intelligent communication environment is very important point if we want people to regularly visit our site or even to be able to use it.

Webocrat is a web based system supporting direct participation of citizens in democratic processes, which is being developed within Webocracy project. The project partners are University of Technology in Košice, Slovakia, University of Wolverhampton, UK, University of Essen, Germany, JUVIER s.r.o, Slovakia, CITEC Engineering Oy Ab, Finland, City Ward Tahanovce, Slovakia, City Ward Furca, Slovakia, Wolverhampton Metropolitan Borough Council, UK.

From the point of view of functionality of the system it is possible to break down the system into several parts and/or modules (Mach et al 2001). They can be represented in a layered sandwich-like structure which is depicted in Figure 1.

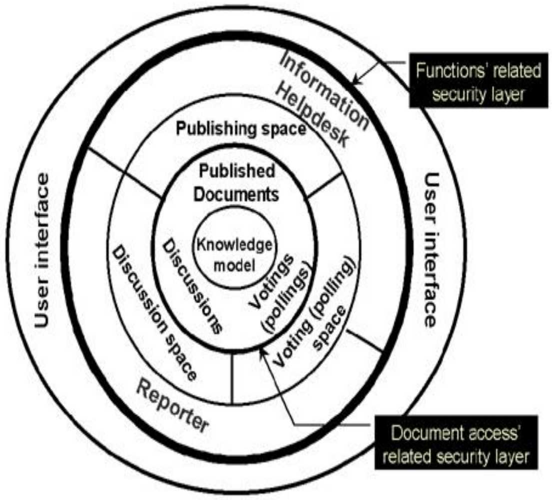

Figure 1 System structure from the point of the system's functionality

The central part of this structure is occupied by a knowledge model (KM) module. This system component contains one or more ontological domain models providing a conceptual model of a domain. The purpose of this component is to index all information stored in the system in order to describe the context of this information (in terms of domain specific concepts). The central position symbolises that the knowledge model is the core (heart) of the system – all parts of the system use this module in order to deal with information stored in the system (both for organising this information and accessing it).

Information stored within the system has the form of documents of different types. Since three main document types are expected to be processed by the system, a document space can be divided into three subspaces – publishing space, discussion space, and opinion polling space. These areas contain published documents expected to be read by users, users' contributions to discussions on different topics of interest, and records of users' opinions about different issues, respectively.

Documents stored in these three document subspaces can be inter-connected with hyper-textual links – they can contain links to other documents – to documents stored in the same subspace, to documents located in another subspace, and to documents from outside of the system. Thus, documents within the system are organised using net-like structure. Moreover, documents located in these subspaces should contain links to elements of a domain model.

Since each document subspace expects different way of manipulating with documents, three system's modules are dedicated to them. Web content management (WCM) module offers means to manage the publishing space. It

1 Technical University of Kosice, Dept of Cybernetics and Artificial Intelligence

enables to prepare documents in order to be published (e.g. to link them to elements of a domain model), to publish them, and to access them after they are published. Discussion space is managed by discussion forum (DF) module. The module enables users to contribute to discussions they are interested in and/or to read contributions submitted by other users. Opinion polling room (OPR) module represents a tool for performing opinion polling on different topics. Users can express their opinions in the form of polling – selecting those alternatives they prefer.

In order to navigate among information stored in the system in an easy and effective way, one more layer has been added to the system. This layer is focused on retrieving relevant information from the system in various ways. It is represented by two modules, each enabling easy access to the stored information in a different way. Citizens' information helpdesk (CIH) module is dedicated to search. It represents a search engine based on the indexing of stored documents. Its purpose is to find all those documents which match user's requirements expressed in the form of a query.

The other module performing information retrieval is the Reporter (REP) module. This module is dedicated to providing information of two types. The first type represents information in an aggregated form. It enables to define and generate different reports concerning information stored in the system. The other type is focused on providing particular documents – but unlike the CIH module it is oriented on off-line mode of operation. It monitors content of the document space on behalf of the user and if information the user may be interested in appears in the system, it sends an alert to him/her.

The upper layer of the presented functional structure of the system is represented by a user interface. It integrates functionality of all the modules accessible to a particular user into one coherent portal to the system and provides access to all functions of the system in a uniform way.

3 Using domain model in Webocrat

3.1 Annotation

To give a system some kind of intelligence, it must know a meaning of the document - its semantics. Standard HTML pages contain almost unstructured information that is understandable only by humans, not by computer. There is no way to tell the computer that this article is about cars unless it contains word *car* explicitly or semantic analysis is applied. The solution is to *annotate* the document. This means that explicit information about its meaning is attached to it whether manually or automatically. Thus, the system can extract relevant information from every annotated document and use it in some intelligent task like searching. Semantic Web initiative is based on this method. It gives proposals and suggestions for annotating HTML pages, using special meta-tags and XML. There is an implicit (tacit) information about document in those tags, which is not visible to end-user, it is only used by system. In knowledge engineering this information is called meta-knowledge. There are many ways how to store meta-knowledge, it doesn't need to be in meta-tags (it is not technically possible with MS Word documents), but it can be stored in special files or databases. Based on meta-knowledge one can perform intelligent retrieval, which gives more relevant results than pure full-text search.

Meta-knowledge can be of two types:

1. **List of keywords or description in natural language.** Document is enriched with some kind of thesaurus here. Full-text search is performed also with this part giving more precise results.
2. **Link to a concept** in predefined vocabulary. This method assumes that there exists some vocabulary of terms or concepts used in the area of our interest. More about this in the next section.

In our work, we concentrated our effort to annotate electronic document (in our case any document published in WCM system) by linking it together with other relevant documents to relevant concepts from the Knowledge base (in our case ontology). It is based on grouping together relevant documents and concepts from the ontology. Such a group of documents and concepts we call Association. Every association has its name, description, and some other attributes needed later for the document retrieval. Basic idea can be seen on Figure 2.

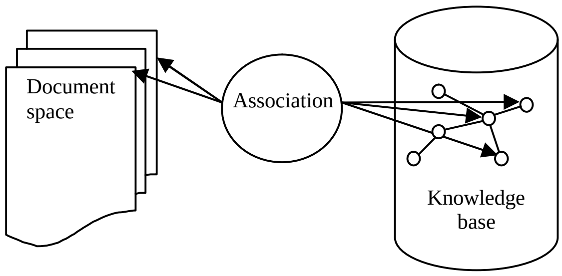

Figure 2 Basic idea of the associations

3.2 Domain model

In the previous section there was mentioned the word vocabulary. In the simplest case it is just a list of terms, where each term has its own description – thesaurus. Such a structure is not satisfactory for our purposes, because it doesn't reflect relations among the terms. What we want is the model of the real world or its part. The part of the world we are interested in is called *domain* and its model is called *domain model*. Domain model is based on *conceptualisation*. A conceptualisation is an abstract, simplified view of the world that we wish to represent for some purpose. It consists of concepts that represent the objects of our interest in a real world and relationships that hold them. To formally represent domain model we use *ontology*. Ontology is an explicit specification of a conceptualisation [1].

Domain model allows the system to perform reasoning and thus to find relevance of a document not only on lexical but also on semantic basis. An example of a part of an ontology is shown in Figure 3

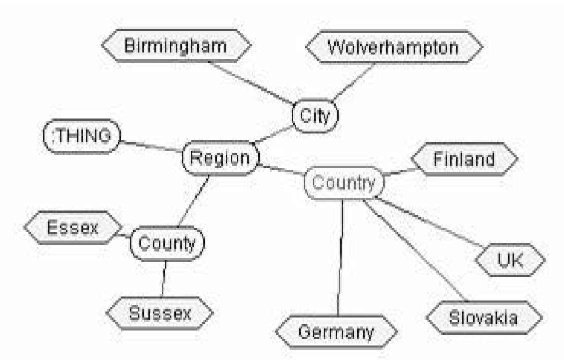

Figure 3 A part of sample ontology

4 Using domain model in Webocrat

The main idea behind whole Webocrat system is to treat documents of various types that are associated with a part of domain model – ontology. This way it is possible to annotate discussions, chats, reports, polling or ordinary WWW pages. By ordinary documents we mean all the documents that are published by local authority, such as news, announcements, reports and other documents that could be interesting for public. When they are published, they are annotated first, whether manually or semi-automatically. After that they are prepared for intelligent retrieval. When accessing information, user can make his query consisting of words for full-text search and of terms (concepts) used in ontology. By use of concepts in the query it is ensured that also its hidden meaning will be discovered. Formulation of such query also allows the user to define his personal profile of interest in terms of ontology. Personalised reports and newsletters can be then automatically generated and sent to user.

Described scenario assumes that the ontology covers all relevant parts of real life concerning to structure of public institutions, communal matters, ecology etc. Figure 3 shows sample ontology about institutions. (This is only testing example. Real life ontologies are being developed in the time of writing this paper).

So we showed how classical web content can be annotated for aforementioned one-way communication. But knowledge about the semantics of document can play also active role during communication. Discussions are typical examples in Webocrat. We consider the discussion as a thread of documents that are all annotated. In order to enable to retrieve discussion contributions according to their content, it is necessary to create links to elements of a domain model when creating new discussion. These elements will represent topics on which the discussion will be focused. Each contribution which will be added to this discussion later will be linked to the same elements from the domain model in an automatic way (contributions inherit links from their discussion).

In order to enable organising contributions within the discussion not only according to the date and time of submissions or authors of submissions, it is possible to complete the contribution with a set of links. These links can be of two types – links to elements of a domain model and links to other contributions from within the discussion. The former type of links enables to define the content in more detail (not only in the sense that the contribution is about exactly the same issues as the discussion as a whole) – this includes not only adding some more links to the set of links inherited from the discussion definition but reducing this inherited set as well. The latter type of links enable user to determine to which existing contribution(s) he/she responds. In addition, it is possible to enrich a contribution to some discussion with links to documents from inside or outside of the system, e.g. in case when the users (submitters) refers in their contributions to those documents.

In order to read particular contributions it is necessary to access them. User has several possibilities how to complete this task. First of all, he/she can choose from a list of all available discussions. Another alternative way is to use linking of contributions to elements of a domain model in order to create groups of contributions dealing with the same set of issues [2].

Using links to ontology, system can suggest the discussion on some topic when user reads document on that topic. Or when user contributes to some discussion, system can advise where to find more relevant information. It would be impossible without links to domain model. Even more, when user links his contribution to some concepts, overriding linkage of whole discussion, system can automatically find more relevant discussion, if existing, and suggest it. Similarly, if some contributions get more and more distant from topic of original thread, administrator can be notified to split discussion. The similarity of contributions is measured using distances of corresponding concepts in the ontology.

On this discussion example we showed how the domain model can enhance communication and how classical tools could be used more efficiently.

5 Domain model requirements

Using experiences from other projects and related work with ontologies, we had specified some basic attributes, which we expect our ontology will have. They was as follows:

- some constant types are defined e.g. integer, float, string, date, currency
- basic objects are classes, instances, relations
- classes can be primitive (definition represents necessary but not sufficient conditions) or non-primitive (both sufficient and necessary)
- a class can be associated with a collection of slots
- slots with predefined semantics: documentation
- a collection of facets can be associated with a slot
- slot facets with predefined semantics (for classes only): value-type, can be constant type, constant expression (and, or, not), enumerated type, min-cardinality, max-cardinality, range, can be constant tuple or list of constant tuples, (not) same value as other slot has, subset-of-values as other slot has, documentation, default value, value
- an instance can inherit a collection of slots
- only one facet can be associated with a slot of an instance:
- value and default value of a slot can be constant or set of constants

- relations can be n-ary for n=1,2,3,...
- relations are defined on basic objects
- relations can have defined attributes: inverse-relation - which relation is an inverse to the one, disjoint, covered, equivalent, transitive, symmetric, functional
- predefined relations are: instance-of - between a class and an instance, semantics: inheritance of slots (values, facets), type-of - an inverse relation to instance-of, subclass-of - between two classes, semantics: inheritance of slots (values, facets), superclass-of - an inverse relation to subclass-of
- slot facet values are inherited but can be overwritten (new value must be more constraining than the old one)
- multiple inheritance (from more parents) is allowed
- special classes
    - THING - represents the root of the class hierarchy
        - every defined class is a subclass of THING,
        - every instance is an instance of THING
        - has slot "documentation" with value-type STRING
    - CLASS - class of all classes
    - INSTANCES - class of all instances

In current state of the project we needed to offer for our partners tool for creating and editing ontology. Because Knowledge Module task starts in our project in future, we had specified some other requirements for knowledge editor:

- it has to be flexible, to enable later modifications in knowledge model
- platform independence
- it should enable importing ontologies from other formats

Thus we dedicated to use some kind of Open Source knowledge editor programmed in JAVA instead of programming new one and to modify it for our purposes. Tool, which best fitted into mostly all of our requirement seemed to be Protégé 2000 from Stanford University. Other knowledge editors we have tested was OntoEdit, JOT, GEF, Apollo, SiLRI.

6 Using Protege 2000 for creating ontologies

Protégé-2000 is the latest component-based and platform-independent generation of the ontology editor. Two goals have driven the design and development of Protégé-2000:

1. achieving interoperability with other knowledge-representation systems, and
2. being an easy-to-use and configurable knowledge-acquisition tool.

The first goal is achieved by compatibility of the knowledge model of Protégé-2000 with OKBC (Open Knowledge Base Connectivity). As a result, Protégé-2000 users can import ontologies from other OKBC-compatible servers and export their ontologies to other OKBC knowledge servers. Protégé-2000 uses the freedom allowed by the OKBC specification to maintain the model of structured knowledge acquisition tools and to achieve the second design goal of being a usable and extensible tool.
Protégé fitted almost all of our requirements for the knowledge editor. The only one noticeable difference was in form, how relations are represented in Protégé. Because of freedom of the ontology specification in Protége knowledge model, relations are not defined as basic objects [3]. We discuss later in this article, how to solve this lack. Other modifications we did to Protégé were:

1. **Localisation** of Protégé into more languages (at this time it is localised into Slovak version)
2. **Adding ability to graphically view classes structure** (Figure 4). It will help the user easily browse ontology in a graphical view. The graph layout is computed automatically or can be changed by user.

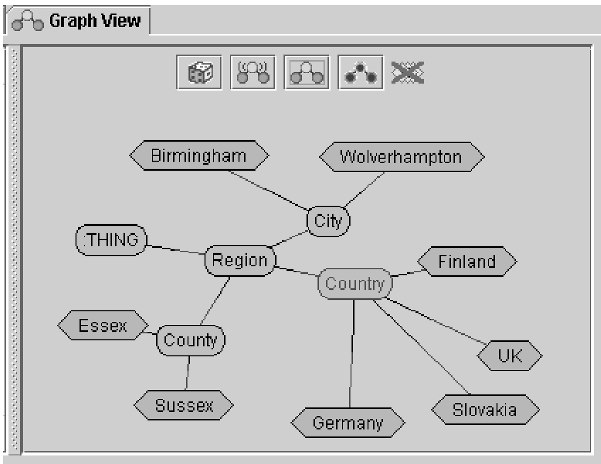

Figure 4 Graphview tab for Protégé 2000

7 Representing relations in Protégé

Because relations are not basic Protégé objects, we have to model them. In the discussion within Protégé community four possible solutions were proposed:

Option 1
We can use own slots. This is probably the easiest way to go, but it is also the most restrictive one. Here the relations are own slots on all subclasses of the class that first specified those slots. The values of the slots are classes that they are related to in one way or another.
Advantage:
- Very easy to model
- We already have all the interface and underlying structures in Protégé for this.

Problems:

- We can not add additional information, such as orientation, in particular, when the value of a slot is a list of classes and not a single class
-

Option 2 (extension of Option 1)
Use facets on own slots (own slots on own slots) to specify orientation and other additional properties
Problem:

- Too complicated: it is hard even to explain exactly how things are going to work.
-

Option 3
Use template slots. Since slots are first-class objects in Protégé (they are themselves frames) , it is easy to express attributes of relations such as reflexivity, transitivity, etc, as well as a hierarchy of relations (the same is true for Option 1).
Advantage:

- Can use advantages of inheritance more extensively.
- Own slots on classes are harder to explain and understand template slots are easier.

Problems:

- It is harder to express additional constraints on relations, such as orientation.
-

Option 4
Relations are themselves classes. We can go one step further and reify relations as classes themselves. Relations between particular classes are instances of these Relation classes
Advantages:

- Can easily encode meta-information on relations: Reflexive, Transitive, Inverse. All of these properties are own slots on a Relation class
- Relations can have additional slots, such as orientation, that get instantiated when we define relations between classes.

The first advantage also carries over to most of the earlier options with the exception that the additional information (relation attributes, hierarchy) would be on slots and not classes, which is often harder to understand and manipulate.
Problem:

- Specialized browsing that "jumps over" a level to view hierarchies of entities based on each relation will be needed (for example, view the part-of hierarchy).

All of these four options can be combined. Price for this is then loose of the uniform approach to describing properties of relations such as transitivity, inverses and so on.
Option 4 looks like the most suitable one, but it would be uncomfortable for user to define special class for any possible type of relation. Since real applications are not developed yet, we cannot predicate the number of relations needed.
We decided for option 3. The EXTENDED_SLOT class has been defined with new facets TRANSITIVE and DISJOINT. Other attributes can be easily added at any time. This EXTENDED_SLOT class is set to be default, so that every new slot that is created on any class is a subclass of EXTENDED_SLOT and thus it automatically contains required attributes TRANSITIVE and DISJOINT. Relation between two objects is modelled as a slot, where one class of relation contains that slot and second class is a value of that slot.
Protégé 2000 does not treat DISJOINT or TRANSITIVE facets in some special way. They are only used by reasoning mechanism which will be developed later and will not be a part of Protégé itself.

8 Acknowledgements

This work is done within the Webocracy project "Web Technologies Supporting Direct Participation in Democratic Processes", which is supported by European Commission DG INFSO under the IST program, contract no. IST-1999-20364, and within the VEGA project 1/8131/01 "Knowledge Technologies for Information Acquisition and Retrieval" of Scientific Grant Agency of Ministry of Education of the Slovak Republic.

The content of this publication is the sole responsibility of the authors, and in no way represents the view of the European Commission or its services.

9 References

[1] Gruber, T., R. (1993): A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2):199-220.
[2] Mach, M.; Dridi, F.; Furdik, K. (2001): Webocrat System Architecture and Functionality. *Webocracy report 2.4*.
[3] Noy, N., F.; Fergerson, R., W.; Musen, M., A. (2000): The knowledge model of Protégé-2000: combining interoperability and flexibility. *International Conference on Knowledge Engineering and Knowledge Management (EKAW '2000)*, Juan-les-Pins, France.
[4] Sabol, T.; Jackson, M.; Dridi, F.; Palola, I.; Novacek, E.; Cizmarik, T.; Thompson, P. (2001): Dissemination and Use Plan. *Webocracy report 15.2.1*.