Proceedings of the International Workshop on Web Information Systems Modeling

Thiran, Philippe; Frasincar, Flavius; Houben, Geert-Jan

Publication date:
2006

Document Version
Early version, also known as pre-print

Link to publication

Citation for pulished version (HARVARD):
Thiran, P, Frasincar, F & Houben, G-J 2006, *Proceedings of the International Workshop on Web Information Systems Modeling: WISM 2006*..

Flavius Frasincar
Geert-Jan Houben
Philippe Thiran (Eds.)

WISM 2006
Web Information Systems Modeling

Workshop at

CAiSE 2006

The 18th Conference on Advanced Information Systems Engineering
Luxembourg, Grand-Duchy of Luxembourg, 5-9 June, 2006

WISM 2006 Workshop Organization

Program committee

Djamal Benslimane	France
Flavius Frasincar	The Netherlands
Martin Gaedke	Germany
Jaime Gomez	Spain
Geert-Jan Houben	The Netherlands
Philippe Thiran	Belgium
Lorna Uden	United Kingdom
Jean Vanderdonckt	Belgium

Workshop organizers

Flavius Frasincar	The Netherlands
Geert-Jan Houben	Belgium & The Netherlands
Philippe Thiran	Belgium
Peter Barna	The Netherlands

Preface

Information systems that the use the Web metaphor for the interface with the user, the so-called Web Information Systems (WIS), are the most popular information systems nowadays. The spectrum of WIS is very large, it ranges from simple information systems that merely allow the user to browse information (e.g., train departures or arrivals) to complex information systems that allow complex forms of interaction of the user with the system (e.g., buying train tickets and planning travels). Realizing the business potential of the Web there is a great need to build WIS or to migrate traditional information systems to the Web.

As WIS mature, there is increasing demand that these systems should satisfy complex requirements. Due to their complexity WIS benefit significantly from a model-driven approach such that one has a good understanding of the system, the communication among stakeholders is better supported, and also the system to be built is appropriately documented. As for Software Engineering, WIS modeling is done from different points of view and also at different levels of abstraction, so that only the needed information at a certain moment in time is represented. Differently than Software Engineering, engineering WIS needs to consider the Web pecularities (e.g., the hyperlink concept, a broad spectrum of users, etc.)

The Web Information Systems Modeling (WISM) workshop aims to emphasize the current modeling techniques that apply to WIS. This is the third edition of this international workshop after two successful editions organized in Riga and Sydney. As there are a growing number of applications that show the potential of the Semantic Web, we have decided in this third edition of the workshop to focus on how Semantic Web technologies can help in the process of WIS modeling.

In order to exploit the Semantic Web technologies one needs to provide the right metadata that comes with the information sources. As there are many existing data sources without metadata, a lot of research nowadays concentrates on the (semi-automatic) extraction of this metadata. Once having at its disposal the data source metadata, a WIS can provide querying facilities in languages similar to natural language.

As WIS are used in different contexts and by different users, it is important that one models the adaptation aspects of these system. A common approach is to represent the user information in a user profile and define rules that will use this profile in order to personalize the information presented by WIS. There are two types of personalization: static, done prior to user browsing, and a dynamic, done during user browsing. Next to adaptation these systems should also support the information access by the mobile worker.

Nowadays there is an increasing amount of multimedia being created and stored in databases. WIS can use this new source of information in order to improve the quality and appeal of the presented material. The multimedia aspects pose new challenges to the WIS data management. For example the metadata extraction from images and videos is quite different from the metadata extraction from plain texts. Also the querying paradigms used for querying multimedia are quite different from the traditional query languages.

We do hope that the above issues raised the readers' interest in the articles that we gathered in these proceedings and we wish them a pleasant reading through some of the topics of the fascinating world of WIS modeling. Finally, we would like to thank authors for their contribution to this field and make the organization of WISM2006 possible.

Flavius Frasincar
Geert-Jan Houben
Philippe Tihran
(June, 2006)

Table of Contents

An Infrastructure for Building Semantic Web Portals

Yuangui Lei, Vanessa Lopez, and Enrico Motta

Knowledge Media Institute (KMi), The Open University, Milton Keynes,
{y.lei, v.lopez, e.motta}@open.ac.uk

Abstract. In this paper, we present our KMi semantic web portal infrastructure, which supports two important tasks of semantic web portals, namely metadata extraction and data querying. Central to our infrastructure are three components: i) an automated metadata extraction tool, *ASDI*, which supports the extraction of high quality metadata from heterogeneous sources, ii) an ontology-driven question answering tool, *AquaLog*, which makes use of the domain specific ontology and the semantic metadata extracted by ASDI to answers questions in natural language format, and iii) a semantic search engine, which enhances traditional text-based searching by making use of the underlying ontologies and the extracted metadata. A semantic web portal application has been built, which illustrates the usage of this infrastructure.

1 Introduction

The Semantic Web [1] is the vision of the next generation of the World Wide Web, in which information is given well-defined meaning and thus becomes understandable and consumable not only for humans, but for computers as well. A number of semantic web portals have been developed through which semantic mark-ups can be gathered, stored and accessed in certain communities. Examples include MindSwap[1], OntoWeb[2], Knowledge Web[3], CS AKTive Space[4], Flink[5] and MuseumFinland[6]. These semantic web portals extend the concept of web portals by adding semantics to the contents or services, so that they can be seen as applications of the semantic web delivered in relatively small user communities.

One important task of these semantic web portals is to offer both end users and applications a seamless access to the knowledge contained in the underlying heterogeneous sources. As such, it is important to ensure that i) high quality knowledge (in terms of semantic metadata, i.e., data represented in terms of

[1] http://www.mindswap.org/
[2] http://www.ontoweb.org/
[3] http://knowledgeweb.semanticweb.org/
[4] http://triplestore.aktors.org/demo/AKTiveSpace/
[5] http://flink.semanticweb.org/index.jsp
[6] http://museosuomi.cs.helsinki.fi/

domain specific ontologies) is extracted from heterogeneous sources in an automated manner, and ii) comprehensive querying facilities are provided, enabling knowledge to be accessed as easily as possible.

Our overview of current semantic web portals reveals that they offer limited support for metadata extraction and data querying. In contrast with these, the system we present here, the KMi semantic web portal infrastructure, focuses on these issues. Central to our infrastructure are three components: i) *ASDI*, an automated semantic data acquisition tool, which supports the acquisition of high quality metadata from heterogeneous sources, ii) *AquaLog* [10], a portable ontology-driven question answering tool, which exploits available semantic markups to answer questions in natural language format, and iii) *a semantic search engine*, which enhances keyword searching by making use of the underlying domain knowledge (i.e. the ontologies and the extracted metadata).

The rest of the paper is organized as follows. We begin in section 2 by investigating how current semantic web portals approach the issues of metadata extraction and data querying. We then present an overview of the KMi semantic web portal infrastructure in section 3. Thereafter, we explain the core components of the infrastructure in sections 4, 5, and 6. In section 7, we present the application of our infrastructure and describe the experimental evaluations carried out in this application. Finally, in sections 8, we conclude our paper with a discussion of our results, the limitations and future work.

2 State of the art

In this section, we investigate how current semantic web portals address the two important issues described above namely metadata extraction and data querying. As we focus only on these two issues, other facilities (e.g., the generation of user interfaces, the support for ontology management) will be left out. We survey a representative sample of portals and tools without performing an exhaustive study of this research strand.

MindSwap, OntoWeb, and Knowledge Web are examples of research projects based semantic web portals. They rely on back-end semantic data repositories to describe the relevant data. Some portals (e.g. MindSwap and OntoWeb) provide tools to support metadata extraction. Such tools are however not directly deployed in these portals. Regarding data querying, most portals offer ontology-based searching facilities, which augment traditional information retrieval techniques with ontologies to support the querying of semantic entities.

The CS AKTive Space [12], the winner of the 2003 Semantic Web Challenge competition, gathers data automatically on a continuous basis for the UK Computer Science domain. Quality control related issues such as the problem of duplicate entities are only weakly addressed (for example, by heuristics based methods or using manual input). The portal offers several filtering facilities (including geographical filtering) to support information access. Regarding data

querying, it provides a querying language based interface, which allows users to specify queries using the specified languages (e.g., SPARQL [7]).

MuseumFinland [7] is the first semantic web portal that aggregates heterogeneous museum collections. The underlying metadata is extracted from distributed databases by means of mapping database schemas to the shared museum ontologies. The portal provides a view-based multi-facet searching facility, which makes use of museum category hierarchies to filter information. It also provides a keyword searching facility, which attempts to match the keyword with the available categories and then uses the category matches to filter information.

Flink [11], winner of the 2004 Semantic Web Challenge competition, extracts and aggregates online social networks in the research community of the semantic web. The data is aggregated in an RDF(S) repository and a set of domain-specific inference rules are used to ensure its quality. In particular, identity reasoning is performed to determine if different resources refer to the same individual (i.e., co-relation). FLink makes use of a range of visualization techniques to support information browsing. Data querying is however not supported.

In summary, most portals mentioned above support metadata extraction. While they do provide useful support in their specific problem domain, *their support for quality control is relatively weak*. Even though some co-relation and disambiguation mechanisms have been exploited (e.g., in CS AKTive Space and Flink), quality control has not been fully addressed. For example, the problems of duplicate or erroneous entities have not been addressed in any of the approaches mentioned above. Such problems may significantly decrease the quality of the acquired semantic data.

Regarding data querying, some portals (e.g. CS AKTive Space, MuseumFinland, and FLink) provide comprehensive support for information clustering and filtering. Some (e.g. MindSwap, MuseumFinland) provide ontology-based searching facilities. While these facilities do provide support for users to access the underlying information, *they typically suffer from the problem of knowledge overhead*, which is requiring users be equipped with extensive knowledge of the underlying ontologies or the specified query language in order to be able to i) formulate queries by means of filling out forms with ontology jargons (e.g., in OntoWeb) or specifying queries (e.g., in CS AKTive Space), ii) to understand the querying result (e.g., in MindSwap), or iii) to reach the data they are interested in (e.g., in MuseumFinland, FLink). Users are not able to pose questions in their own terms.

Another issue associated with data querying is that *the keyword-based searching support is relatively weak*. The benefits of the availability of semantic markups have not yet been fully exploited. Indeed, with a partial exception of MuseumFinland (which matches keywords against museum categories), no efforts have been made to try to understand the meaning of the queries. Furthermore, the techniques used are primarily focused on the enabling of search for semantic entities (e.g., MindSwap, OntoWeb). The underlying semantic relations of metadata have not been used to support the finding of other relevant information. For example,

[7] http://www.w3.org/TR/rdf-sparql-query/

when searching "phd student", we often get a list of instances of the class *phd student*. We may however be more interested in the news stories, the academic projects, the publications that phd students are involved in, rather than the instances themselves. The search engines will fail if such information is not directly defined in the class. This is because the complex relations of the semantic metadata have not been exploited.

3 An overview of the KMi semantic web portal infrastructure

Figure 1 shows the five layered architecture of our KMi semantic web portal infrastructure, which contains a source data layer, an extraction layer, a semantic data layer, a semantic service layer, and a presentation layer.

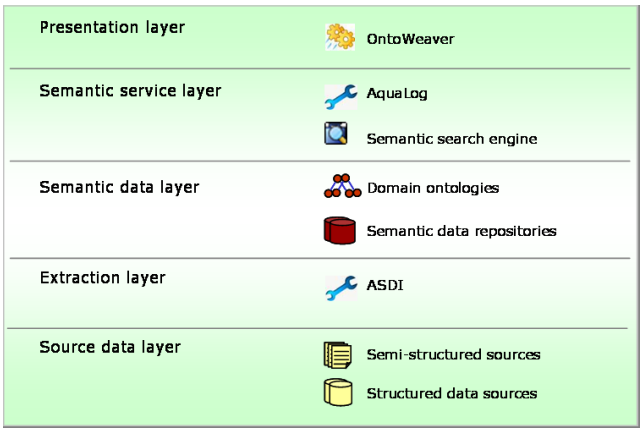


Fig. 1. An overview of the KMi semantic web portal infrastructure.

The source data layer comprises the collection of all available data sources such as semi-structured textual documents (e.g. web pages) or structured data in the form of XML feeds, databases, and knowledge bases.

The extraction layer is responsible for the extraction of high quality semantic data from the source data layer. The core component is the metadata acquisition tool ASDI. As will be described in section 4, ASDI provides several means to ensure the quality of the extracted data.

The semantic data layer contains a number of domain ontologies and semantic data repositories which store metadata extracted from the source data layer by the underlying extraction layer.

The semantic service layer provides semantic services over the semantic data repositories for the target semantic web portals. Data querying is seen as one of the important services, which enables knowledge to be easily accessed.

Central to this layer are two components. One is AquaLog, which takes questions in natural language format and an ontology as input and presents precise answers to users. As will be explained in section 5, AquaLog makes use of natural language processing technologies and the semantic representation of the extracted knowledge to achieve the task of question answering. The other component is the semantic search engine, which enhances the performance of traditional keyword search by augmenting current semantic search techniques with complex semantic relations extracted from the underlying heterogeneous sources. This component will be described in section 6.

The presentation layer supports the generation of user interfaces for semantic web portals. The KMi semantic web portal infrastructure relies on OntoWeaver-S [9] to support i) the aggregation of data from the semantic service layer and the semantic data layer and ii) the generation of dynamic web pages. OntoWeaver-S offers high level support for the design of data aggregation templates, which describe how to retrieve data and organize data content. As the generation of user interfaces is out of the scope of this paper, we will not go through the details of OntoWeaver-S.

4 The extraction of high quality metadata

To ensure high quality, we identify three generic tasks that are related to metadata extraction and which should be supported in semantic web portals. They include i) extracting information in an automatic and adaptive manner, so that on one hand the process can be easily repeated periodically in order to keep the knowledge updated and on the other hand different meanings of a given term can be captured in different context; ii) ensuring that the derived metadata is free of common errors; and iii) updating the semantic metadata as new information becomes available.

In ASDI we provide support for all these quality insurance related tasks. Figure 2 shows its architecture. ASDI relies on an *automatic and adaptive information extraction tool*, which marks-up textual sources, *a semantic transformation engine*, which converts data from source representations into the specified domain ontology according to the transformation instructions specified in *a mapping ontology*, and *a verification engine*, which checks the quality of the previously generated semantic data entries. These components will be explained in the following subsections.

4.1 Information extraction

To address the issue of *adaptive information extraction*, we use ESpotter [13], a named entity recognition (NER) system that provides an adaptive service. ESpotter accepts the URL of a textual document as input and produces a list of the named entities mentioned in that text. The adaptability is realized by means of domain ontologies and a repository of lexicon entries. For example, in

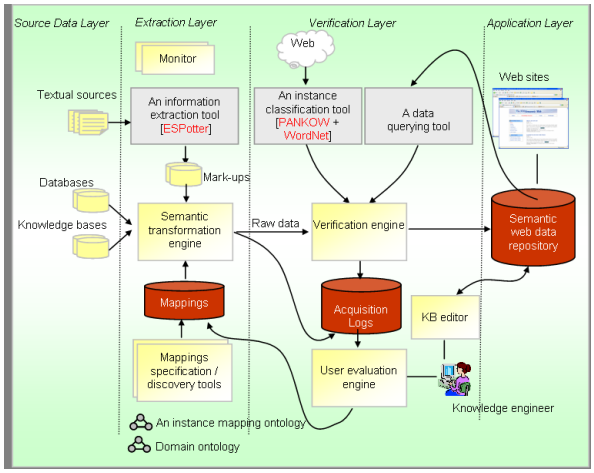


Fig. 2. The architecture of the metadata extraction tool ASDI.

the context of the KMi domain, ESpotter is able to mark the term "Magpie" as a project, while in other domains it marks it as a bird.

For the purpose of converting the extracted data to the specified domain ontology (i.e., the ontology that should be used by the final applications), an instance mapping ontology (see details in [8]) has been developed, which supports i) the generation of rich semantic relations along with semantic data entries, and ii) the specification of domain specific knowledge (i.e. lexicons). The lexicons are later used by the verification process. A semantic transformation engine is prototyped, which accepts structured sources and transformation instructions as input and produces semantic data entries.

To ensure that the acquired data stays *up to date*, a set of monitoring services detect and capture changes made in the underlying data sources and initiate the whole extraction process again. This ensures a sustainable and maintenance-free operation of the overall architecture.

4.2 Information verification

The goal of the verification engine is to check that each entity has been extracted correctly by the extraction components.The verification process consists of three increasingly complex steps as depicted in Figure 3. These steps employ several semantic web tools and a set of resources to complete their tasks.

Step1: Checking the internal lexicon library. The lexicon library maintains domain specific lexicons (e.g., abbreviations) and records the mappings between strings and instance names. One lexicon mapping example in the KMi semantic web portal is that the string "ou" corresponds to the instance *the-open-university* entity that has been defined in one of the domain specific ontologies. The verification engine will consider any appearances of this abbreviation as referring to the corresponding entity.

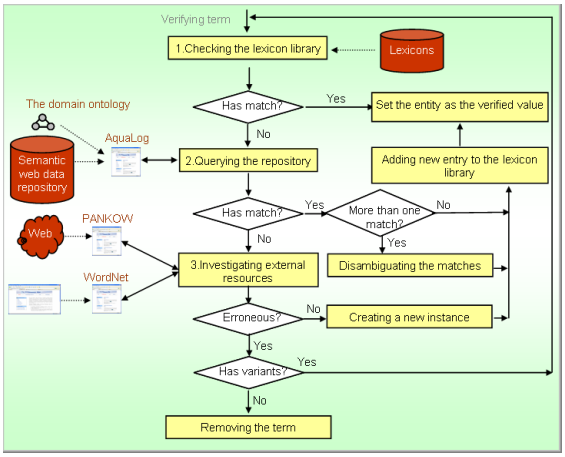


Fig. 3. The overall algorithm of the data verification engine.

The lexicon library is initialized by lexicons specified through the mapping instruction and expands as the verification process goes on. By using the lexicon library, the verification engine is able to i) exploit domain specific lexicons to avoid domain specific noisy data and ii) avoid repeating the verification of the same entity thus making the process more efficient.

Step2: Querying the semantic web data repository. This step uses an ontology-based data querying engine, to query the already acquired semantic web data (which is assumed to be correct, i.e. trusted) and to solve obvious typos and minor errors in the data. This step contains a disambiguation mechanism, whose role is to dereference ambiguous entities (e.g., whether the term "star wars" refers to the Lucas' movie or President Reagan's military programme).

The data querying engine employs a number of string matching algorithms to deal with obvious typos and minor errors in the data. For example, in a news story a student called *Dnyanesh Rajapathak* is mentioned. The student name is however misspelled as it should be *Dnyanesh Rajpathak*. While the student name is successfully marked up and integrated, the misspelling problem is carried into the portal as well. With support from the data querying engine, this problem is corrected by the verification engine. It queries the knowledge base for all entities of type *Student* and discovers that the difference between the name of the verified instance (i.e., *Dnyanesh Rajapathak*) and that of one of the students (i.e, *Dnyanesh Rajpathak*) is minimal (they only differ by one letter). Therefore, the engine returns the correct name of the student as a result of the verification. Note that this mechanism has its downfall when similarly named entities denote different real life objects.

If there is a single match, the verification process ends. However, when more matches exist, contextual information is exploited to address the ambiguities. The verification engine exploits the semantic relations between other entities appearing in the same piece of text (e.g. the news story) and the matches as the

contextual information. For example, when verifying the person entity *Victoria*, two matches are found: *Victoria-Uren* and *Victoria-Wilson*. To decide which one is the appropriate match, the verification engine looks up other entities referenced in the same story and checks whether they have any relation with any of the matches in the knowledge base. In this example, the *AKT* project is mentioned in the same story, and the match *Victoria-Uren* has a relation (i.e., *has-project-member*) with the project. Hence, the appropriate match is more likely to be *Victoria-Uren* than *Victoria-Wilson*.

Step3: Investigating external resources. If the second step fails, external resources such as the Web are investigated to identify whether the entity is erroneous, which should be removed, or correct but new to the system. For this purpose, an instance classification tool is developed, which makes use of PANKOW [2] and WordNet [4], to determine the appropriate classification of the verified entity. Now let us explain the mechanism by using the process of verifying the entity IBM as an example.

Step 3.1. The PANKOW service is used to classify the string *IBM*. PANKOW employs an unsupervised, pattern-based approach on Web data to categorize the string and produces a set of possible classifications along with ranking values. If PANKOW cannot get any result, the term is treated as erroneous but still can be partially correct. Thus, its variants are investigated one by one until classifications can be drawn. For example, the variants of the term "BBC news" are the term "BBC" and the term "news". If PANKOW returns any results, the classifications with the highest ranking are picked up. In this example, the term "company" has the highest ranking.

Step 3.2. Next the algorithm uses WordNet to compare the similarity between the type of the verified entity as proposed (i.e., "organization") and an alternative type for the entity as returned by PANKOW (i.e.,"company"). The algorithm here only checks whether they are synonyms. If they are (which is the case of the example), it is concluded that the verified entity is classified correctly. Thus, a new instance (*IBM* of type *Organization*) needs to be created and added to the repository. Otherwise, other major concepts of the domain ontology are compared to the Web-endorsed type (i.e.,"company") in an effort to find a proper classification for the entity in the domain ontology. If such classification is found, it is concluded that the verified entity was wrongly classified. Otherwise, it can be safely concluded that the verified entity is erroneous.

5 Question answering as data querying

In the context of semantic web portals, for a querying facility to be really useful, users have to be able to pose questions in their own terms, without having to know about the vocabulary or structure of the ontology or having to master a special query language. Building the AquaLog system has allowed us to address this problem.

AquaLog exploits the power of ontologies as a model of knowledge and the availability of up-to-date semantic markups extracted by ASDI to give precise,

focused answers rather than retrieving possible documents or pre-written paragraph of text. In particular, these semantic markups facilitate queries where multiple pieces of information (that may come from different sources) need to be inferred and combined together. For instance, when we ask a query such as "what is the homepage of Peter who has an interest on the Semantic Web?", we get the precise answer. Behind the scene, AquaLog is not only able to correctly understand the question but also competent to disambiguate multiple matches of the term *Peter* and give back the correct match by consulting the ontology and the available metadata.

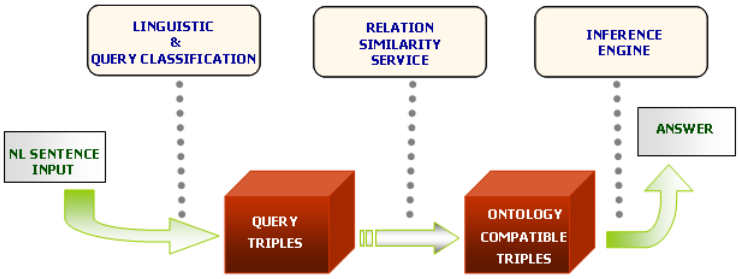


Fig. 4. An overview of the AquaLog architecture.

An important feature of AquaLog is its portability with respect to the ontologies. Figure 4 shows an overview of the AquaLog architecture. It relies on *a linguistic component*, which parses the natural language (NL) queries into a set of linguistic triples, *a relation similarity service* (RSS), which makes use of the underlying ontology and the available metadata to interpret the linguistic triples as ontological triples, and *an answer engine*, which infers answers from the derived ontological triples. To illustrate the question answering mechanism, we use the question *"what are the planet news in KMi related to akt?"* as an example. The process takes the following steps:

Step1: Parsing a NL query into linguistic querying triples. The linguistic component uses the GATE [3] infrastructure to annotate the input query and parses the query into a set of linguistic querying triples. At this stage the analysis is domain independent. It is completely based upon linguistic technologies. The example described above is parsed into two linguistic triples *(which is, planet news, KMi)* and *(which is, related to, akt)*.

Step2: Interpreting linguistic querying triples as ontology-compliant triples. In this step, the relation similarity service (RSS) disambiguates and interprets the linguistic querying triples, by making use of i) string metric algorithms, ii) lexical resources such as WordNet, and iii) the domain specific ontology. It produces ontology-compliant triples as results.

For the first linguistic triple (i.e. (which is, planet news, KMi)) derived from the example query, the RSS component matches the term *planet news* to the concept *kmi-planet-news-item* in the available metadata repository. It also identifies the term *KMi* as the instance *Knowledge-Media-Institute-at-the-Open-University* which is actually an instance of the concept *research-institute*.

Now, the problem becomes finding a relation which links the class *kmi-planet-news-item* to the class *research-institute* or viceversa. In order to find a relation all the subclasses of the non-ground generic term, kmi-planet-news-item, need to be considered. Thanks to ontology inference all the relations of the superclasses of the subject concept of the relation (e.g. "kmi-planet-news-item" for direct relations or "research institute" for inverse relations) are also its relations. The relations found include "owned-by", "has-author", and "mentions-organization". The interpreted ontology triples thus include *(kmi-planet-news-item owned-by research-institute)*, *(kmi-planet-news-item has-author research-institute)* and *(kmi-planet-news-item mentions-organization research-institute)*. Likewise, the second linguistic triple (i.e. *(which is, related to, akt)*) is linked to the first triple through the non-ground term *kmi-planet-news-item* and processed as ontology triples (*kmi-planet-news-item, mentions-project, akt)* and *(kmi-planet-news-item, has-publications, akt)*

Since the universe of discourse is determined by the particular ontology used, there will be a number of discrepancies between the NL questions and the set of relations recognized in the ontology. External resources like WordNet help in mapping unknown terms by giving a set of synonyms. However, in quite a few cases, such as in the two triples of our example, such resources are not enough to disambiguate between possible ontology relations due to the user or ontology specific jargon. To overcome this problem, AquaLog includes a learning mechanism, which ensures that, for a given ontology and a specific user community, its performance improves over time, as the users can easily give feedback and allow AquaLog to learn novel associations between the relations used by users, which are expressed in natural language, and the internal structure of the ontology.

Step3: Inferring answers from ontology-compliant triples. The answer engine looks through the available metadata to find data entries which satisfy the derived ontology triples and produces answers accordingly. As shown in figure 5, the answer of our query example is a list of instances of the class *kmi-planet-news-item* that mentions the project *akt*, in the context of the KMi semantic web portal, an application of our infrastructure, which will be explained in section 7. The user can navigate through each news entry extracted by the ASDI tool. Figure 5 also illustrates the derived linguistic triples and ontology triples.

6 Semantic search

Keyword search is often an appealing facility, as it provides a simple way of querying information. The role of *semantic* search is to make use of the underlying ontologies and metadata available in semantic web portals to provide

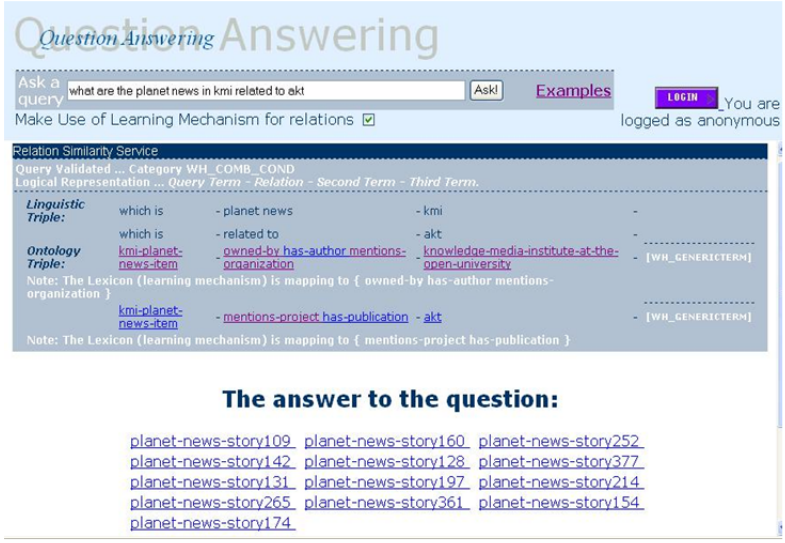


Fig. 5. An AquaLog running example.

better performance for keyword searching. In particular, the semantic search facility developed in our infrastructure extends current semantic search technologies[5, 6] (which are primarily centered around enabling search for semantic entities) by augmenting the search mechanisms with complex semantic relations of data resources which have been made available either by manual annotation or automatic/semi-automatic extraction in the context of semantic web portals.

To illustrate the semantic searching facility, we use the searching of news stories about phd students as an example. With traditional keyword searching technologies, we often get news entries in which the string "phd students" appears. Those entries which mention the names of the involved phd students but do not use the term "phd students" directly will be missed out. Such news however are often the ones that we are really interested in. In the context of semantic web portals where semantics of the domain knowledge are available, the semantic meaning of the keyword (which is a general concept in the example of phd students) can be figured out. Furthermore, the underlying semantic relations of metadata can be exploited to support the retrieving of news entries which are closely related to the semantic meaning of the keyword. Thus, the search performance can be significantly improved by expanding the query with instances and relations.

The search engine accepts a keyword and a search subject (e.g., news, projects, publications) as input and gives back search results which are ranked according to their relations to the keyword. The search subject is often domain dependent and can be predefined in specific problem domain. In customized portals, the search subjects can be extracted from user profiles. In the example described

above, the search subject is news stories. The search engine follows four major steps to achieve its task:

Step1: Making sense of the keyword. From the semantic meaning point of view, the keyword may mean i) general concepts (e.g., the keyword "phd students" which matches the concept *phd-student*), ii) instances entities (e.g., the keyword "enrico" which matches the instance *enrico-motta*, or iii) literals of certain instances (e.g., the keyword "chief scientist" which matches the values of the instance *marc-eisenstadt*). The task of this step is to find out into which category the keyword falls, by employing string matching mechanisms to match the keyword against ontology concepts, instances, and literals (i.e., non semantic entities, e.g., string values) step by step. The output of this step is the entity matches and the weight of the string similarity.

Step2: Getting related instances. When the keyword falls into the first category described above, the related instances are the instances of the matched concepts. In the second category case, the related instances are the matched instances. In the final category case, the related instances are the instances which have value as the matched literals. For example, when querying for the keyword "chief scientist", the matched instance is *marc-eisenstadt* whose value of the property *job-title* matches the keyword.

Step3: Getting search results. This step is to find those instances of the specified subject (e.g. news stories) which have explicit or (close) implicit relations with the matched instances. By explicit relations we mean the instances that are directly associated together in explicitly specified triples. For example, for instances i_1 and i_2, their explicit relations are specified in triples (i_1 p i_2) or (i_2 p i_1), where the entity p represents relations between them.

Implicit relations are the ones which can be derived from the explicit triples. Mediators exist in between, which bridge the relations. The more mediators exist in between, the weaker the derived relation is. For the sake of simplicity, we only consider the closest implicit relations, in which there is only one mediator in between. For instances i_1 and i_2, such implicit relations can be be represented as the following triples : i) (i_1 p_1 m), (m p_2 i_2); ii) (i_2 p_2 m), (m p_1 i_1); iii) (i_1 p_1 m), (i_2 p_2 m); and iv) (m p_1 i_1), (m p_2 i_2). In these relations, the semantic entity m acts as the mediated instance; the predicates p_1 and p_2 act as the mediated relations.

Step4: Ranking results. In this step, the search results are ranked according to their closeness to the keyword. We take into account three factors, including *string similarity*, *domain context*, and *semantic relation closeness*. The domain context weight applies to non-exact matches, which helps deciding the closeness of the instance matches to the keyword from the specific domain point of view. For example, with the keyword "enric", is the user more likely to mean the person *enrico-motta* than the project *enrich*? The domain context weight of a matched instance m_x is calculated as $\frac{Pm_x}{\sum_{i=1}^{n} Pm_i}$, where Pm_x denotes the count of the matched instance m_x serving as value of other instances in the metadata repository; and m_1, m_2, ..., and m_n represent all of the non-extact matches.

The semantic relation closeness describes how close the semantic relations are between a search result and the matched instances. The way of calculating it is to count all the relations a search result has with all of the matched instances. For this purpose, we give the explicit relations the weight 1.0, and the derived ones 0.5.

For the sake of experiments, we give each of the three factors described above (namely string similiary, domain context and semantic relation closeness) the same confidence. The confidence however can be easily changed to reflect the importance of certain factors. We applied the semantic search facility in the application of the KMi semantic web portal, which will be presented in the following section.

7 The KMi semantic web portal: an application

We have designed and tested our infrastructure in the context of building a semantic Web portal for our lab, the Knowledge Media Institute (KMi) at the UK's Open University, which provides integrated access to various aspects of the academic life of our lab[8].

7.1 Metadata extraction

The KMi semantic web portal has been built and running for several months generating and maintaining semantic data from the underlying sources in an automated way. The relevant data is spread in several different data sources such as departmental databases, knowledge bases and HTML pages. In particular, KMi has an electronic newsletter[9], which now contains an archive of several hundreds of news items, describing events of significance to the KMi members. Furthermore, related events are continuously reported in the newsletter and added to the news archive.

An experimental evaluation has been carried out to assess the performance of the metadata extraction in the context of the KMi semantic web portal, by comparing the automatically extracted data to the manual annotations in terms of *recall, precision* and *f-measure*, where recall is the proportion of all possible correct annotations that were found by the system with respect to the ones that can be in principle extracted from the source text, precision is the proportion of the extracted annotations that were found to be correct, and f-measure assesses the overall performance by treating recall and precision equally. The task is to recognize entities (including *people*, *projects* and *organizations*) mentioned in the randomly chosen news stories.

To illustrate the important role of the verification engine, the performance of ESpotter (which is the information extraction tool used in ASDI) is introduced in the comparison, which shows the quality of the extracted data before and after

[8] http://semanticweb.kmi.open.ac.uk
[9] http://kmi.open.ac.uk/news

Table 1. An experimental evaluation of the metadata extraction.

Type	People	Organizations	Projects	Total
Recall				
ESpotter	**0.815**	**0.783**	**0.761**	**0.798**
ASDI	**0.771**	**0.783**	**0.761**	**0.775**
Precision				
ESpotter	**0.873**	**0.667**	**1**	**0.787**
ASDI	**0.860**	**0.946**	**1**	**0.911**
F-measure				
ESpotter	**0.843**	**0.72**	**0.864**	**0.792**
ASDI	**0.813**	**0.856**	**0.864**	**0.837**

the verification process. Table 1 shows the evaluation results. Although the recall rate is slightly lower than Espotter (this is because PANKOW sometimes gives back empty classifications thus resulting in losing some correct values), the precision rate has been improved. The f-measure values show that ASDI performs better than ESpotter in terms of the overall performance. This means that the quality of the extracted data is improved by our verification engine.

7.2 Question answering

To assess the performance of the question answering facility, we carried out an initial study in the context of the KMi semantic web portal. We collected 69 different questions. Among them, 40 have been handled correctly. 19 more can be handled correctly if re-formatted by end user. This was a pretty good result, considering that no linguistic restrictions were imposed on the questions (please note that we have asked users not to ask questions which required temporal reasoning, as the underlying ontology does not cover it).

Among the failures, linguistic handling accounts for 16 out of the 20 failures, (23.18% of the queries). This is because the natural language processing component is unable to classify the query and generate appropriate intermediate representations (e.g., the question "what are the KMi publications in the area of the semantic web"). Such questions however can usually be easily reformulated by the end users and thus to be answered correctly. For example, changing the question mentioned earlier as "what are the KMi publications in the semantic web area", avoiding the use of nominal compounds (e.g, terms that are a combination of two classes as "akt researchers"). Another important observation is that some failures (10 out of 69) are caused by the lack of appropriate services over the ontology, e.g. the queries about "top researchers". Finally, the limitation of the coverage of the available metadata also plays an important role in the failures. For example, when dealing with the question "who funds the Magpie project", as there is no such information available in the metadata repository, the answer engine fails.

7.3 Semantic search

Although the semantic search engine is still in its infancy as the ranking algorithm has not yet been fully investigated, it produces encouraging results. Figure 6 shows the results of the search example which has been described in section 6. Behind the scene, the news stories are annotated by the metadata extraction component ASDI and thus being associated with semantic mark-ups. The metadata of phd students are extracted from the departmental databases.

As shown in the figure, the news entry which mentions most phd students appears in the top, as it gets the biggest relation weight. The news entries that mention other semantic entities (e.g., the news story ranked as the second mentions the project AKT) with which many phd students have relations also get good rankings. This is because in the experiment the ranking algorithm gives implicit relations half the weight of the explicit ones. This needs further investigation. The semantic search mechanism is however profound, which makes use of the available semantic relations between different resources to bring forward most relevant information.

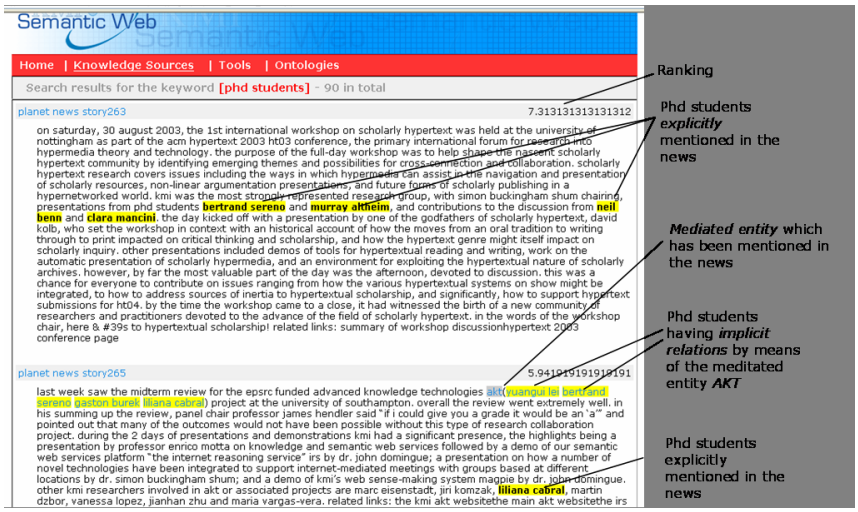


Fig. 6. A screenshot of the semantic search results of the example *"searching news about phd students"*.

8 Discussions

The core observation that underlies this paper is that, in the case of semantic web portals that offer both end users and applications an integrated access to information in specific user communities, it is crucial to ensure that i) high

quality metadata are acquired and combined from several data sources, and ii) simple but effective querying facilities are provided thus enabling knowledge to be accessed as easily as possible. By quality here we mean that the semantic data contains no duplicates, no errors and that the semantic descriptions correctly reflect the nature of the described entities. By simple but effective querying we mean that users are able to pose questions in their own terms and get precise results.

Our survey of a set of semantic web portals shows that on the one hand little or no attention is paid to ensure the quality of the extracted data, and on the other hand the support for data querying is limited. In contrast with these efforts, our semantic web portal infrastructure focuses on ensuring the quality of the extracted metadata and the facilities for data querying.

Our evaluation of the quality verification module shows that it improved the performance of the bare extraction layer. The metadata extraction component ASDI outperforms ESpotter by achieving 91% precision and 77% recall. In the context of semantic web portals, precision is more important than recall - erroneous results annoy user more than missing information. We plan to improve the recall rate by introducing additional information extraction engines to work in parallel with ESpotter. Such a redundancy is expected to substantially improve recall.

Our initial study of the question answering tool AquaLog shows that it provides reasonably good performance when allowing users to ask questions in their own terms. We are currently working on the failures learned from the study. We plan to use more than one ontology so that limitations in one ontology can be overcome by other ones.

The semantic search facility developed within the framework of our infrastructure takes advantage of semantic representation of information to facilitate keyword searching. It produces encouraging results. We plan to further investigate i) a more fine grained ranking algorithm which gives appropriate consideration for all the factors affecting the search results, and ii) the effect of implicit relations on search results.

We are, however, aware of *a number of limitations* associated with this semantic web portal infrastructure. For example, the manual specification of mappings in the process of setting up the metadata extraction component makes the approach heavy to launch. We are currently addressing this issue by investigating the use of automatic or semi-automatic mapping algorithms. A semi-automatic mapping would allow our tool to be portable across several different application domains.

Another limitation of the infrastructure is the support for *trust* management. As anyone can contribute to semantic data, trust is an important issue, which needs to be addressed. We plan to study this issue in the near future, by looking at i) how trust factors can be associated with the metadata of semantic web portals, ii) how they can be combined together when one piece of markup comes from different sources, and iii) what roles the processing tools play from the trust point of view.

Acknowledgements

We wish to thank Dr. Marta Sabou and Dr. Victoria Uren for their valuable comments on earlier drafts of this paper. This work was funded by the Advanced Knowledge Technologies Interdisciplinary Research Collaboration (IRC), the Knowledge Sharing and Reuse across Media (X-Media) project, and the OpenKnowledge project. AKT is sponsored by the UK Engineering and Physical Sciences Research Council under grant number GR/N15764/01. X-Media and OpenKnowledge are sponsored by the European Commission as part of the Information Society Technologies (IST) programme under EC Grant IST-FP6-26978 and IST-FP6-027253.

References

1. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34 – 43, May 2001.
2. P. Cimiano, S. Handschuh, and S. Staab. Towards the Self-Annotating Web. In S. Feldman, M. Uretsky, M. Najork, and C. Wills, editors, *Proceedings of the 13th International World Wide Web Conference*, pages 462 – 471, 2004.
3. H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. Gate: A framework and graphical development environment for robust nlp tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*, Philadelphia, 2002.
4. C. Fellbaum. *WORDNET: An Electronic Lexical Database*. MIT Press, 1998.
5. R. Guha, R. McCool, and E. Miller. Semantic search. In *Proceedings of WWW2003*, 2003.
6. J. Heflin and J. Hendler. Searching the web with shoe. In *Proceedings of the AAAI Workshop on AI for Web Search*, pages 35 – 40. AAAI Press, 2000.
7. E. Hyvonen, E. Makela, M. Salminen, A. Valo, K. Viljanen, S. Saarela, M. Junnila, and S. Kettula. MuseumFinland – Finnish Museums on the Semantic Web. *Journal of Web Semantics*, 3(2), 2005.
8. Y. Lei. An Instance Mapping Ontology for the Semantic Web. In *Proceedings of the Third International Conference on Knowledge Capture*, Banff, Canada, 2005.
9. Y. Lei, E. Motta, and J. Domingue. Ontoweaver-s: Supporting the design of knowledge portals. In *Proceedings of the 14th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2004)*. Springer, October 2004.
10. V. Lopez, M. Pasin, and E. Motta. AquaLog: An Ontology-portable Question Answering System for the Semantic Web. In *Proceedings of ESWC*, 2005.
11. P. Mika. Flink: Semantic Web Technology for the Extraction and Analysis of Social Networks. *Journal of Web Semantics*, 3(2), 2005.
12. M.C. Schraefel, N.R. Shadbolt, N. Gibbins, H. Glaser, and S. Harris. CS AKTive Space: Representing Computer Science in the Semantic Web. In *Proceedings of the 13th International World Wide Web Conference*, 2004.
13. J. Zhu, V. Uren, and E. Motta. ESpotter: Adaptive Named Entity Recognition for Web Browsing. In *Proceedings of the Professional Knowledge Management Conference*, 2004.

Towards an Architecture of Ontological Components for the Semantic Web

Nesrine Ben Mustapha[1], Marie-Aude Aufaure[2], and Hajer Baazhaoui-Zghal[1]

[1] Riadi Lab., ENSI Campus Universitaire de la Manouba, 2010 Tunis, Tunisie
{nesrine.benmustapha, hajer.baazaouizghal} @riadi.rnu.tn

[2] Supelec, Computer Science Department, Plateau du Moulon,
91 192 Gif sur Yvette, France
Marie-Aude.Aufaure@Supelec.fr

Abstract. This paper presents an architecture of ontological components for the Semantic Web. Many methods and methodologies can be found in the literature. Generally, they are dedicated to particular data types like text, semi-structured data, relational data, etc. Our work deals with web pages. We first study the state of the art of methodologies defined to learn ontologies from texts. Then, our architecture of ontological components for the Semantic web is defined, in order to improve knowledge discovery from the web. At least, we detail the incremental construction of the domain ontology component and we present the general conceptual model associated to it.

1 Introduction

The volume of available information on the web is growing exponentially. Consequently, integration of heterogeneous data sources and information retrieval become more and more complex. Adding a semantic dimension to web pages is a response to this problem and is known as the semantic web [1]. Ontologies can be seen as a fundamental part of the semantic web. They can be defined as an explicit, formal specification of a shared conceptualization [2]. Indeed, their use allows facilitating web information retrieval, domain knowledge sharing as well as knowledge integration. Meanwhile, building ontology manually is a long and tedious task. Many approaches for learning ontologies can be found in the literature. Section 2 synthesizes such ontology learning methodologies. We present our architecture of ontological components for the semantic web, integrated in a customizable ontology building environment, in section 3. At least, we conclude and give some perspectives for this work.

2 Ontology building methodologies

Methodologies for ontology building can be classified according to the use or not of a-priori knowledge (such as thesaurus, existing ontologies, etc.) and to learning meth-

ods. The first ones were dedicated to enterprise ontology development [3] [4] and manually built. Then, methodologies for building ontologies from scratch were developed. They do not use a-priori knowledge. An example of such a methodology is OntoKnowledge [5] which proposes a set of generic techniques, methods and principles for each process (feasibility study, initialization, refinement, evaluation and maintenance). Some research work is dedicated to collaborative ontology building such as CO4 [6] and (KA)2 [7]. Another research area deals with ontology reengineering [8]. Learning methodologies can be distinguished according to their input data type: texts, dictionaries [9], knowledge bases [10], relational [11], [12] and semi-structured data [13], [14], [15].

In the following section, we focus on the general dimensions implied in ontology learning. Section 2.3 deals with ontology learning from texts, including web pages.

2.1 General dimensions implied in ontology learning

The existing methods can be distinguished according to the following criteria: learning sources, the type of ontology to build, techniques used to extract concepts, relationships and axioms, and existing tools. The most recent methodologies generally use a priori knowledge such as thesaurus, minimal ontology, other existing ontologies, etc. Each one proposes different techniques to extract concepts and relationships, but not axioms. These axioms can represent constraints but also inferential domain knowledge. As for instance extraction, we can find techniques based on first order logic [32], on Bayesian learning [33], etc. We have to capitalize the results obtained by the different methods and to characterize existing techniques, their properties and how we can combine them. The objective of this section is to synthesize the characteristics of these methods in order to expose our problematic and to argue our choices.

Learning ontologies is a process requiring at least the following development stages:

- Knowledge sources preparation (textual corpus, collection of web documents), eventually using a priori knowledge (ontology with a high-level abstraction, taxonomy, thesaurus, etc.),
- Data sources preprocessing,
- Concepts and relationships learning,
- Ontology evaluation and validation (generally done by experts)..

The ontology is built according to the following dimensions:

- Input type (data sources, a priori knowledge existence or not, …),
- Tasks involved for preprocessing : simple text linguistic analysis, document classification, text labeling using lexico-syntactic patterns, disambiguating, etc.,
- Learned elements : concepts, relationships, axioms, instances, thematic roles,
- Learning methods characteristics: supervised or not, classification, clustering, rules, linguistic, hybrid,
- Automation level: manual, semi-automatic, automatic, cooperative,

- Characteristics of the ontology to build: structure, representation language, coverage,
- Usage of the ontology and users' needs [16].

2.2 Learning ontologies from texts

The proposed approaches can be classified according to the technique used, namely linguistic, lexico-syntactic patterns extraction, clustering or classification, and hybrid ones. The input data is constituted by linguistic resources like a list of terms and relationships. The huge volume of these data, their quality and relevance has to be taken into account by filtering methods. These methods can be guided by an expert (knowledge acquisition from texts) or automatic (text mining, learning).

Linguistic-based techniques include lexical, syntactic and semantic texts analysis. The objective is to extract a conceptual model of a domain. We can quote two main methodologies defined by [16] and [17]. The methodology defined by [18] intends to extract knowledge from technical documents. Two hypothesis are given by the authors: the first one states that the ontology designer has a good knowledge of the application domain and can determine the relevant terms, while the second one states that he also have a precise idea of the ontology usage. This methodology analyses a corpus with tools appropriate for natural language automatic processing and linguistic techniques and extracts terms and relationships. The normalization step concerns the mapping between natural language and a formal language. The semantic interpretation of the texts is managed by usage and expertise. Semantic relationships are obtained from lexical relationships, and the concepts hierarchy is built using the semantic relationships. The formalization step automatically translates the ontology into a given format like RDF, OWL, etc.

In these linguistic approaches, lexico-syntactic patterns are manually defined by linguists. Some research work has been proposed to automatically extract lexico-syntactic patterns. [19] starts from an existing ontology and extract a set of pairs of concepts linked by relationships, in order to learn hyponymy relationships and produce lexico-syntactic patterns. These ones are used to discover other relationships, based on the learned patterns, between the concepts of the existing ontology. This approach is used to extend an existing lexical ontology. [20] proposes to combine the previous approach with contextual signatures to improve the classification of new concepts. The KAT system (Knowledge acquisition from Texts) [21] includes four steps: learning new concepts, classification, learning relationships and ontology evaluation. Concept classification consists in analyzing words that appears in the expression associated to a candidate concept: [word, seed concept], where "word" can be a noun or an adjective. This classification states that the concept [word, seed concept] subsumes the seed concept that is equivalent to add a hyponymy relationship.

These techniques, based on lexico-syntatic patterns learning, lead to good results for learning hyponymy relationships. In the meantime, some problems appear like terms polysemy or errors produced that are dependant from the corpus. The use of classification techniques like hierarchical or conceptual clustering is a way to solve

these problems. The methodology proposed by [22] consists in classifying documents into collections related to words sense, using a labeled corpus and Wordnet. Then for each collection, the relative frequencies are extracted and compared to the other collections. Topic signatures are computed and compared to discover shared words. This methodology is dedicated to enrich concepts of existing ontologies by analyzing web texts. Other methods [23] [24] combine linguistic techniques and clustering to build or extend an ontology.

Some research work is done to study the distribution of words in texts to improve concepts clustering by the way of new similarity measures. DOODLE II, an extension of DOODLE [25], is an environment for the rapid development of domain ontologies. It is based on the analysis of lexical co-occurrences and the construction of a multidimensional space of words [26]. This approach extracts taxonomic relationships using Wordnet and non taxonomic relationships learning by searching association rules and extracting pairs of similar concepts using the words multidimensional space.

2.3 Web-based ontology learning

Our main objective is to define an approach to build ontologies for the semantic web. This kind of ontology, closely linked to the web usage, has to integrate the dynamic aspects of the web. In this section, we present some approaches defined specifically for the web.

Many propositions have been done to enrich an existing ontology using web documents [22][27]. However, these approaches are not specifically dedicated to web knowledge extraction.

The approach proposed by [28] attempts to reduce the terminological and conceptual confusion between members of a virtual community. Concepts and relationships are learned from a set of web sites using the Ontolearn tool. The main steps are: the terminology extraction from web sites and web documents data warehouse, the semantic interpretation of terms and the identification of taxonomic relationships.

Some approaches transform html pages into hierarchical semantic structured encoded in XML, taking into account html regularities [29].

Finally, we can also point out some approaches only dedicated to ontology construction from web pages without using any a priori knowledge.

The approach described in [30] is based on the following steps: (1) extract some keywords representative of the domain, (2) find a collection of web sites related to the previous keywords (using for example Google), (3) exhaustive analysis of each web site, (4) the analyzer searches the initial keywords in a web site and finds the preceding and following words; these words are candidates to be a concept, (5) for each selected concept, a statistical analysis is performed based on the number of occurrences of this word in the web sites and at last, (6) for each concept extracted using a window around the initial keyword, a new keyword is defined and the algorithm recursively iterates.

In [31], a method is proposed to extract domain ontology from web sites without using a priori knowledge. This approach takes benefit from the web pages structure and defines a contextual hierarchy. The data preprocessing is an important step to

define the more relevant terms to classify. Weights are associated to the terms according to their position in this conceptual hierarchy. Then, these terms are automatically classified and concepts are extracted.

3 Ontological components for the Semantic Web

Starting from the state of the art in ontology learning, we propose a hybrid approach to build domain ontology; our objective is to increase the capability of this ontology to specify and extract web knowledge in order to contribute to the semantic web. Analyzing the web content is a difficult task relative to relevance, redundancies and incoherencies of web structures and information. Moreover, semantic similarity measures highly depends on the quality of data, and the complexity of algorithms such as conceptual clustering increase with the volume of data. For these reasons, proposing an approach to build automatically an ontology still remains utopian.

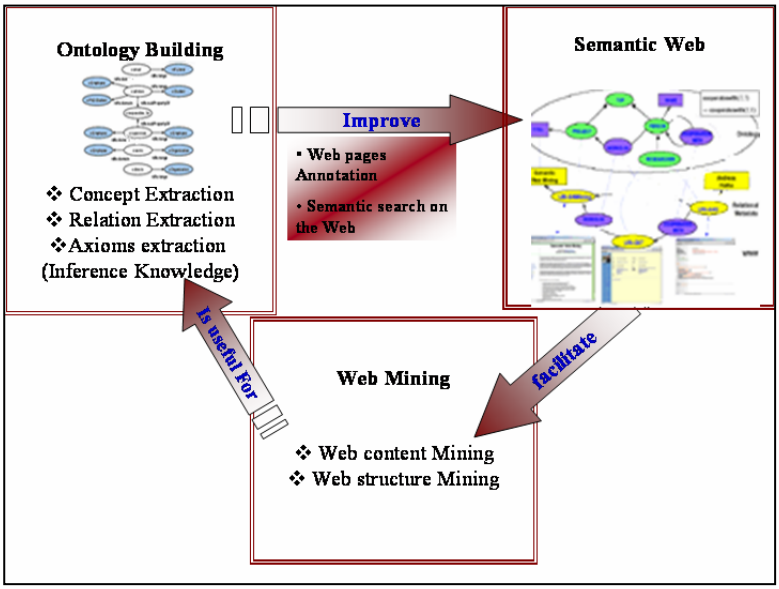


Fig. 1. The cyclic relation between web mining, semantic web and ontology (extracted from [34]

Our approach is based on the cyclic relation between web mining, semantic web and ontology building as stated in [34] and resumed in figure 1. Our proposal is based on the following statements: (1) satisfy the fact that the ontology is useful to specify and extract knowledge from the web, (2) link the semantic content within the web documents structure, and (3) combine linguistic and learning techniques taking into account the scalability and the evolution of the ontology. Our ontology is produced using web mining techniques. We mainly focus on web content and web structure

mining. Building this ontology leads us to solve two main problems. The first one is relative to the heterogeneity of web documents structure while the second one is more technical and concerns technical choices to extract concepts, relationships and axioms as well as the selection of learning sources and the scalability. We propose an architecture of ontological components to represent the domain knowledge, the web sites structure and a set of services. These ontological components (figure 3) are integrated into a customizable ontology building environment (figure 2).

3.1. Architecture

Learning ontologies from web sites is a complex task because web pages can contain more images, hypertext and frames than text. Learning concepts is a task that needs texts able to explicitly specify the properties of a particular domain. A positive point in the context of learning ontologies from web pages is that web sites structure can be exploited by web mining techniques.
Starting from the state of the art (section 2.2), we can say that no learning method to extract concepts and relationships is better (in most cases, the ontology evaluation is manually done). For these reason, we propose a customizable ontology building environment as depicted in figure 2. The customization takes into consideration the general dimensions defined in section 2.1.

In this environment, we propose a set of interdependent ontologies to build a Web knowledge base on a particular domain. These ontologies are related to the content, structure and services semantics. Such environment is composed by the following modules: (1) Learning data sources module, (2) Ontological components enrichment module, (3) Linguistic module, (4), Ontological components editor. Theses modules uses resources such as: data warehouse in XML format, linguistic resources (Wordnet, general ontologies, thesaurus, patterns collection), web knowledge bases, constituted by a set of web documents, their structure and associated services, as depicted in figure 3. We distinguish two types of actors in the environment, namely, software actors as the miner agents and human actors as the linguist expert, the domain experts and the system adminstrators. The functionnalities provided by this system are inspired from the web discovery processus, starting from the data sources pretreatment to the knowledge discovery, including the datawarehouse building. Besides, this environment intends to relate domain ontologies, ontologies of services and web structure ontologies in order to build and enrich web knowledge bases of the domain.

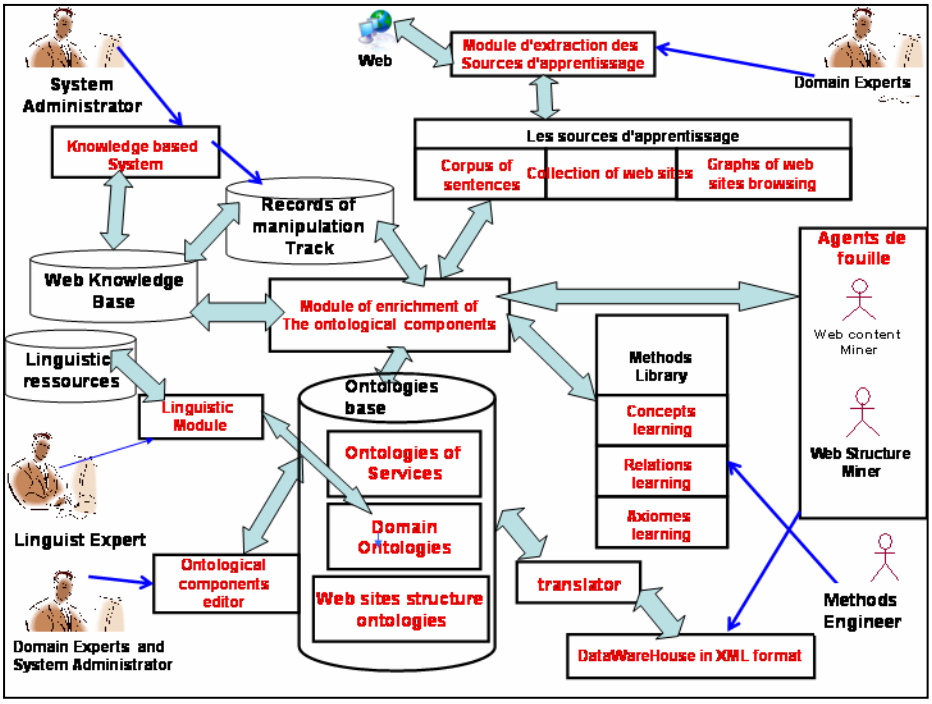


Fig. 2. Customizable ontology building environment

We distinguish three ontologies, namely a generic ontology of web sites structures, a domain ontology and a service ontology.

The generic ontology of web sites structure contains a set of concepts and relationships allowing a common structure description of HTML, XML and DTD web pages. This ontology enables to learn axioms that specify the semantic of web documents patterns. The main objective is to ease structure web mining knowing that the results can help to populate the domain ontology. The ontology of Web sites structure is useful to improve the extraction of the concepts and the relevant relations of the domain by studying semantics of the various markup elements of the languages HTML or XML. We can find in different sites the same contents but presented by different manners which indicates the degree of importance granted to some information according to the context. The second perspective is to translate a semantic relation (e.g. “part-of” relationship, etc.) into a set of adequate markup elements (e.g. a relationship between a concept and instances can be represented by a list, a markup “Small” before a word can express that this word is less important than the previous one). Mining the structure allows us to extract some regularity from which we can define an axiom. Indeed, the axioms of the structure ontology help in elaborating the mapping of HTML markups to the semantic relations inferred. Such an ontology may also adjust the gap existing between the physical and logical structure of a document Web.

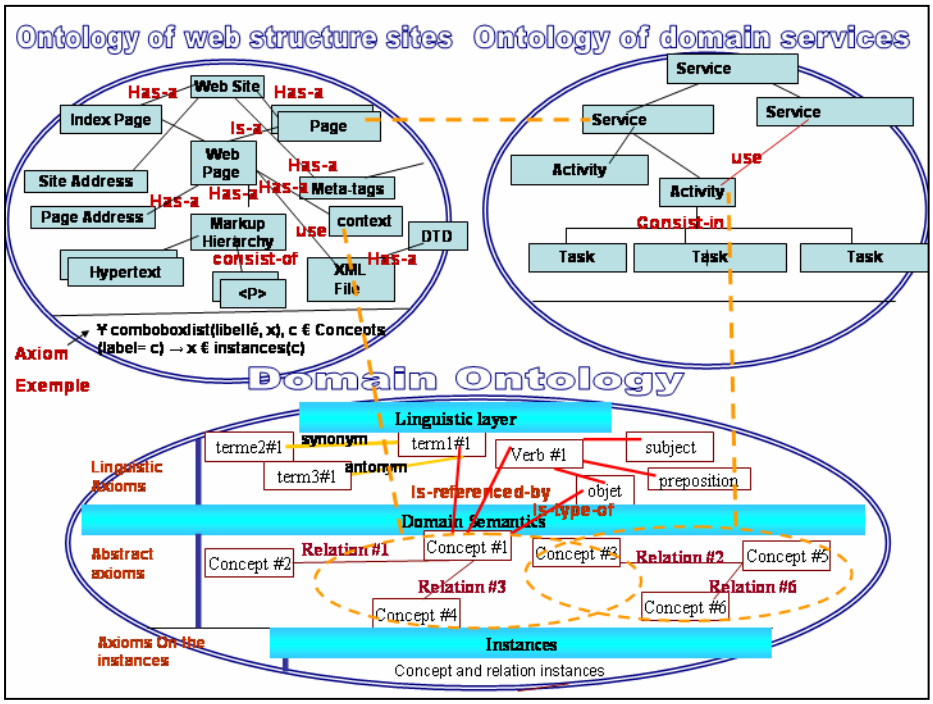


Fig. 3. Architecture of Ontological Components for the Semantic web

The domain ontology is divided into three layers according to their level of abstraction. The first level is a lexical one: this layer specifies high-level lexical knowledge which can help discovering lexico-syntactic patterns. The lexical knowledge covers the concepts, relationships and general axioms of the central layer of our domain ontology. The last layer is more operational and, additionally to concepts and relationships instances, also contains a set of axioms specifying domain knowledge. These axioms are incrementally enhanced by web content and web structure mining.

The ontology of services is defined starting from the concept of task ontology [35]. In our web context, we speak of web services instead of tasks. This ontology specifies the domain services and will be useful to map web knowledge into a set of interdependent services. This is built from the central layer of our domain ontology in order to constitute a set of domain services, and web pages constitute the instances of these services. This ontology is a macroscopic view of the domain and is hierarchically structured: the upper level is the root service while the leaves are elementary tasks for which a triplet "concept-relation-concept" belonging to the domain ontology is associated. These three ontological components are interdependent where the axioms included in an ontology are used to enhance another ontology component. As an example, if we consider the axiom defined on the structure ontology in figure 3, we can say that if the label of a "combobox" is a concept of the domain ontology then all

the elements of this structure are instances of this concept. This axiom is used to populate the ontology domain. Meanwhile, these ontologies differ from their use. The domain ontology is used to specify the domain knowledge. The service ontology specifies the common services that can be solicited by web users and can be attached to several ontologies defined on subparts of the domain. As we said previously, the axioms of the structure ontology are used to extract instances of the domain ontology.

The structure of our domain ontology is more complex than usual domain ontologies. This particularity is oultlined in figure 4.

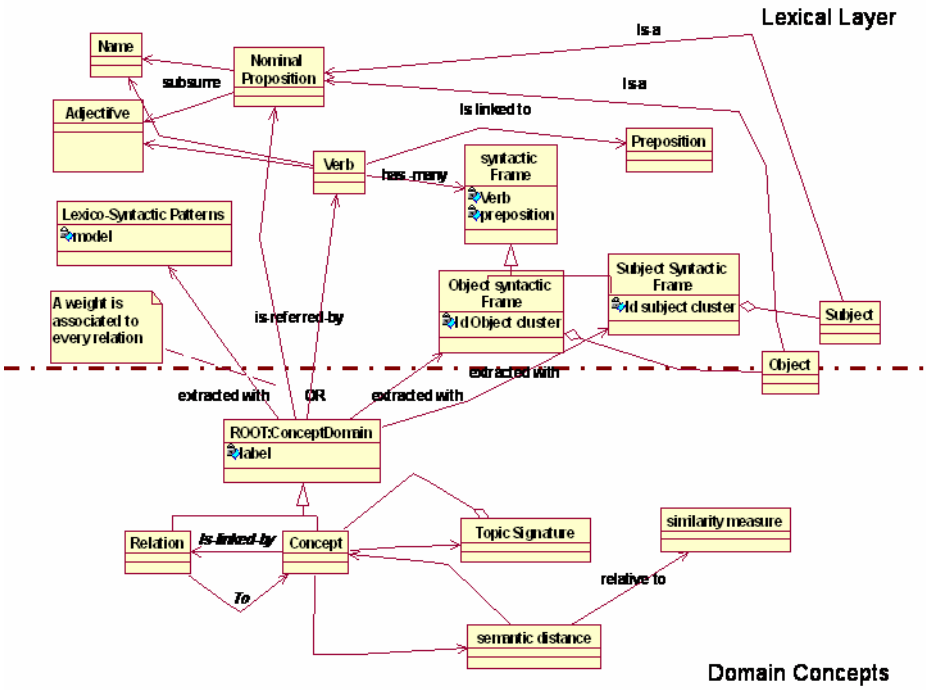


Fig. 4. Domain ontology abstract conceptual model

This figure shows the abstract concepts from which the learned concepts and relationships will be inserted. Confidence weight according to the learning technique used, are associated to concepts and relationships in this ontology. This abstract model is divided into two levels. The linguistic level specifies how a concept was extracted using linguistic techniques, namely the lexicosyntactic patterns [19, 20] or the syntactic frame learning techniques [24]. The domain concepts and relationships are derived from the root of the ontology. They are referenced by verbs or nominal groups. Concepts are extracted by learned lexicosyntaxiques patterns. Such concepts can also be a frequent object or a subject of a syntactic frame. A syntactic frame is a triplet (< subject > < verb > < Object >). The extracted concepts will be weighted by the frequency of the learned patterns in the corpus. Relationships are extracted if a syntactic frame is frequent between two concepts having a close semantic distance. The similarity measures are extensible. We distinguish several semantic distances

according to the chosen measure. We quote for example the cosine between two vectors of concepts in the word space and the measure of the closest neighbors between two concepts in the graph of the concepts. As an example, a concept can be learned using a lexico-syntactic pattern. In this case, the weight associated to this concept is relative to the appearance frequency of the concept satisfying the pattern. These weights are updated at each step dedicated to the ontology enhancement. The semantic distance in the conceptual model is related to the similarity measure between two concepts. This measure is computed from a multidimensional space of words.

3.2. Building the domain ontology

In this section, we focus on the domain ontology extraction. Our strategy (figure 5) is based on three steps. The first one is the initialization step. The second one is an incremental learning process based on linguistic and statistic techniques. The last one is a learning step based on web structure mining. We now define them.

The initialization is based on the following steps:

- The design and manual building of a minimal ontology related to the domain; this construction is based on concepts and relationships of Wordnet,
- Composition of concepts and relationships learning sources:
    - Web search of documents related to our domain using the concepts defined in the minimal ontology as requests,
    - Classification of these web documents,
    - Composition of a textual corpus containing a set of phrases in which we can find at least one concept of the minimal domain ontology,
    - Composition of a corpus of HTML and XML documents indexed by their URL.

Each iteration of the second stage includes two steps. The first one (Procedure A) is defined by the following tasks:

- Enrichment of the ontology with new concepts extracted from semi-structured data found in the web pages (XML, DTD, tables),
- Construction of a word space [36] based on the concepts of the minimal domain ontology,
- Lexico-syntactic patterns learning based on the method defined in [20] (by combining lexico-syntactic patterns and topic signatures); these patterns are related to non taxonomic relationships between the concepts of the minimal ontology,
- Lexico-syntactic patterns learning to extract synonymy, hyponymy and part-of relationships (lexical layer of the domain ontology),
- Similarity matrix building: this matrix allows computing the similarity between pairs of concepts found in the multidimensional space word.

The second step (Procedure B) consists in:

- Update the textual corpus and the web documents collection by searching them according to the concepts defined in the minimal ontology,
- New concepts and non taxonomic relationships extraction by the application of lexico-syntactic patterns,

- Attribution of a weight for each extracted relationship relative to the frequency of the relationships that apply the lexico-syntactic pattern,
- Update the minimal ontology.

Each iteration can be validated by the domain expert. This process is incremental: we repeat the procedures A and B until we do not want to integrate new data.

The last stage consists in an enrichment of the structure ontology and an extraction of structure patterns for each relationship of the domain ontology. The objective is to ease instances extraction using the tagged structure of web pages.

The implementation of this strategy is still in progress.

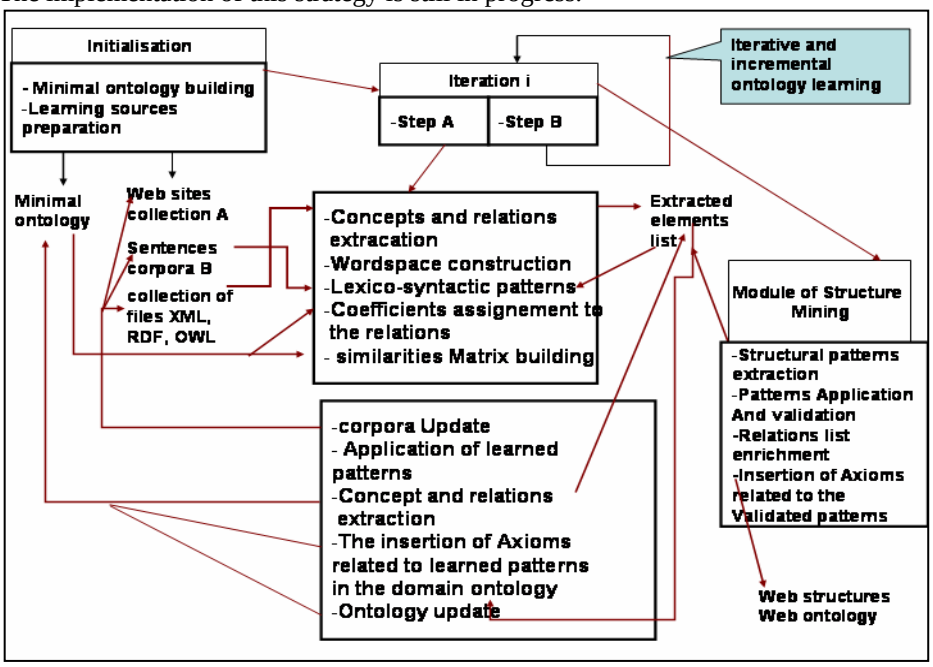


Fig. 5. Learning domain ontology strategy

We have realized a little case study to identify the main characteristics for learning ontology from web sites. From this case study, we have concluded that structure mining techniques are useful for the extraction of non taxonomic and sub-part of relationships as well as for the construction of services ontology. Building a words space is also useful to compute the similarity between concepts but highly depends on web sites corpus. The techniques used are dependant from the learning sources. For this reason, defining a flexible environment should help satisfying the personalization of ontology learning.

4 Conclusion and perspectives

The methodologies for building ontologies described in this paper are fairly complementary. Indeed, methodologies defined to build ontologies from scratch are oriented towards ontological engineering and ontology life-cycle and are based on information systems development methodologies. Learning methodologies try to give a response to the time-consuming manual ontology building task. Learning techniques can be either numeric or symbolic. They have been exploited to semi-automate some foundation tasks such as concept hierarchy building, taxonomic relationships extraction, non taxonomic relationships learning, etc. All these research works constitute a methodological toolbox which can be used to semi-automate ontology construction. We have to take into account previous experiences and to solve cited problems. Let us now outline our contributions in the field of ontology learning and building. Firstly, we conceived an ontological architecture based on a semantic triplet, namely, semantics of the contents, the structure and the services of a domain. So, we take into account both the content and the structure, and a set of services are based upon the domain ontology concepts. The second point is that all the ontologies defined in our architecture contain the weightings of their concepts, relationships and axioms according to their sources. These ontologies could be handled by the inference engine. Concepts and the relationships can be represented by facts and predicates. Axioms are the inference rules allowing the insertion of new facts and predicates.

At least, lexico-syntactic patterns will be stored in the linguistic layer of the domain ontology, so that such patterns will allow the specification of the axioms. These ones define the existing types of relationships, in order to identify such relationships in the web pages. In, our case, axioms are the basis of the incremental enrichment of the ontology. Moreover, the data sources are incrementally updated to satisfy a good semantic coverage of the ontology.

Our perspective is to use semantic web mining techniques and to restructure web pages in order to implement an adaptive web based on the semantic structure, content and services. Our framework presented in this paper is based on an ontological components architecture integrated into a customizable ontology construction environment. We first focus on the automation of the domain ontology construction. Then, we will implement the other ontological components. The final goal is to build a knowledge web base on a specific domain. From our case study, we can conclude that we must take into account various learning sources (like on-line dictionaries) and structure regularities in web sites to go further in the implementation.

References

1. Berners-Lee, T, Hendler, J, Lassila, O.: The Semantic Web, Scientific American (2001)
2. Gruber, T.: Toward principles for the design of ontologies used for knowledge sharing. International Journal of Human-Computer Studies, special issue on Formal Ontology in Conceptual Analysis and Knowledge Representation. Eds, Guarino, N. & Poli , R. (1993)

3. Grüninger, M., Fox M.S.: Methodology for the design and evaluation of ontologies. IJCAI'95 Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, Canada (1995)

4. Uschold, M., King, M.: Towards a Methodology for Building Ontologies. IJCAI'95 Workshop on Basic Ontological Issues in Knowledge Sharing.,Ed. D. Skuce (1995) 6.1-6.10.

5. Staab, S., Schnurr, H.-P., Studer, R., Sure, Y.: Knowledge Processes and Ontologies. In: IEEE Intelligent Systems 16(1), January/Febrary 2001, Special Issue on Knowledge Management

6. Euzenat, J.: Building consensual knowledge bases: context and architecture, Proceedings of 2nd international conference on *building and sharing very large-scale knowledge bases*, Enschede, IOS press, Amsterdam (1995)

7. Decker, S., Erdmann, M., Fensel, D. Studer, R.: Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. In: Semantic Issues in Multimedia Systems, Proceedings of DS-8. Kluwer Academic Publisher, Boston, 351-369 (1999)

8. Gòmez-Pérez, A., Rojas, M.D.: Ontological Reengineering and Reuse. European Workshop on Knowledge Acquisition, Modeling and Management (EKAW), Lecture Notes in Artificial Intelligence LNAI 1621 Springer-Verlag, 139-156, eds., Fensel D. & Studer R. (1999)

9. Jannink, J.: Thesaurus Entry Extraction from an On-line Dictionary. In: Proceedings of Fusion '99, Sunnyvale CA (1999)

10. Suryanto, H., Compton, P.: Discovery of Ontologies from Knowledge Bases. Proceedings of the First International Conference on Knowledge Capture, The Association for Computing Machinery, New York, USA, pp171-178 (2001)

11. Rubin D.L., Hewett, M., Oliver, D.E., Klein, T.E, Altman, R.B.: Automatic data acquisition into ontologies from pharmacogenetics relational data sources using declarative object definitions and XML. In: Proceedings of the Pacific Symposium on Biology, Lihue, HI (2002)

12. Stojanovic, L., Stojanovic, N., Volz, R.: Migrating data-intensive Web Sites into the Semantic Web. Proceedings of the 17th ACM symposium on applied computing (SAC), ACM Press, (2002)

13. Deitel,, A., C. Faron,, C., Dieng, R.: Learning ontologies from RDF annotations. In: Proceedings of the IJCAI Workshop in Ontology Learning, Seattle (2001)

14. Papatheodrou, C., Vassiliou, A., Simon, B.: Discovery of Ontologies for Learning Resources Using Word-based Clustering, ED MEDIA 2002, Copyright by AACE, Reprinted, Denver, USA (2002)

15. Volz, R., Oberle, D., Staab, S., Studer, R.: OntoLiFT Prototype. IST Project 2001-33052 WonderWeb Deliverable 11 2003)

16. Aussenac-Gilles, N., Biébow, B, Szulman, S.: Revisiting Ontology Design: A Methodology Based on Corpus Analysis. In: 12th International Conference in Knowledge Engineering and Knowledge Management (EKAW). Juan-Les-Pins, France (2002)

17. Nobécourt, J.: A method to build formal ontologies from text. In: EKAW-2000 Workshop on ontologies and text, Juan-Les-Pins, France (2000)

18. Aussenac-Gilles, N., Biébow, B., Szulman, S.: Corpus Analysis For Conceptual Modelling. Workshop on Ontologies and Text, Knowledge Engineering and Knowledge Management, 12th International Conference EKAW, Juan-les-pins, France (2000)

19. Hearst, M.A.: Automated Discovery of WordNet Relations. In WordNet: In C Fellbaum ed. "*Wordnet An Electronic Lexical Database*". MIT Press, Cambridge, MA, 132-152 (1998)

20. Alfonseca, E., Manandhar, S.: Improving an Ontology Refinement Method with Hyponymy Patterns. Language Resources and Evaluation (LREC-), Las Palmas, Spain (2002)

21. Moldovan, D. I., Girju, R.: Domain-Specic Knowledge Acquisition and Classication using WordNet, In: proceeding of FLAIRS2000 Conference, Orlando (2000)

22. Agirre, E., Ansa, O., Hovy, E., Martinez, D.: Enriching very large ontologies using the WWW. In: Proceedings of the Workshop on Ontology Construction of the European Conference of AI (2000)

23. Khan, L., Luo, F.: Ontology Construction for Information Selection. In: Proc. of 14th IEEE International Conference on Tools with Artificial Intelligence, Washington DC,122-127 (2002)

24. Faure, D., Nédellec, C., Rouveirol, C.: Acquisition of Semantic Knowledge using Machine learning methods: The System ASIUM. Technical Report ICS-TR-88-16, LRI Paris-Sud University, January (1998)

25. Sekiuchi, R., Aoki, C., Kurematsu, M., Yamaguchi, T.: DODDLE: A Domain Ontology Rapid Development Environment. PRICAI98 (1998)

26. Hearst, M., Schütze, H.: Customizing a Lexicon to Better Suit a Computational Task. Proc. of the Workshop on Extracting Lexical Knowledge (1993)

27. Faatz, A., Steinmetz, R.: Ontology enrichment with texts from the WWW. Semantic Web Mining 2nd Workshop at ECML/PKDD-2002, Helsinki, Finland (2002)

28. Navigli, R., Velardi, P.: Learning Domain Ontologies from Document Warehouses and Dedicated Web Sites. Computational Linguistics, MIT press (2004)

29. Davulcu, H., Vadrevu, S., Nagarajan, S.: OntoMiner: Bootstrapping and Populating Ontologies from Domain Specific Websites. Proceedings of the First International Workshop on Semantic Web and Databases (SWDB 2003), Berlin (2003)

30. Sanchez, D., Moreno, A.: Automatic generation of taxonomies from the WWW. In: proceedings of the 5th International Conference on Practical Aspects of Knowledge Management (PAKM 2004). LNAI, Vol. 3336. Vienna, Austria (2004)

31. Karoui, L., Aufaure, M.-A., Bennacer, N.: Ontology Discovery from Web Pages : Application to Tourism. Workshop on Knowledge Discovery and Ontologies (KDO), co-located with ECML/PKDD, Pisa, Italy, Sept. 2004, pp. 115-120 (2004)

32. Junker, M., Sintek, M., Rinck , M.: Learning for Text Categorization and Information Extraction with ILP. In: J. Cussens (eds.), Proceedings of the 1st Workshop on Learning Language in Logic, Bled, Slovenia, 84-93 (1999)

33. Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., Slattery, S.: Learning to construct knowledge bases from the World Wide Web. Artificial Intelligence, (2000)

34. Berendt, B., Hotho, A., Stumme, G.: Towards semantic web mining. In: International Semantic Web Conference, volume 2342 of Lecture Notes in Computer Science, Springer, 264–278 (2002)

35. Gomez-Perez, A., Fernandez-Lopez, M., Corcho, O.: Ontological Engineering. Springer (2003)

36. Yamaguchi, T.: Acquiring Conceptual Relationships from Domain-Specific Texts. Proceedings of the Second Workshop on Ontology Learning OL'2001 Seattle, USA, August 4 (2001)

37. Maedche, A., Volz, R.: The Text-To-Onto Ontology Extraction and Maintenance Environment. Proceedings of the ICDM Workshop on integrating data mining and knowledge management, San Jose, California, USA (2001)

38. Maedche, A., Staab, S.: Ontology Learning for the Semantic Web. IEEE Intelligent Systems,Special Issue on the Semantic Web, (2001)

VisAVis: An Approach to an Intermediate Layer between Ontologies and Relational Database Contents

Nikolaos Konstantinou, Dimitrios-Emmanuel Spanos, Michael Chalas, Emmanuel Solidakis and Nikolas Mitrou

nkons@cn.ntua.gr, el00057@mail.ntua.gr, el00114@mail.ntua.gr, esolid@telecom.ntua.gr, mitrou@softlab.ntua.gr
National Technical University of Athens, Division of Communications, Electronics and Information Systems, Heroon Polytechniou 9, 15773 Athens, Greece

Abstract. This paper introduces an approach to mapping relational database contents to ontologies. The current effort is motivated by the need of including into the Semantic Web volumes of web data not satisfied by current search engines. A graphical tool is developed in order to ease the mapping procedure and export enhanced ontologies linked to database entities. Moreover, queries using semantic web query languages can be imposed to the database through its connection to an ontology. Using Protégé, we were able to map ontology instances into relational databases and retrieve results by semantic web query languages. The key idea is that, instead of storing instances along with the ontology terminology, we can keep them stored in a database and maintain a link to the dataset. Thus, on one hand we achieve smaller size ontologies and on the other hand we can exploit database contents harmonizing the semantic web concept with current wide-spread techniques and popular applications.

1 Introduction

It is a well known fact that the rapid evolution of the Internet has brought up significant changes to information management. The web of knowledge changed the way people share, access and retrieve data. The existing data that lie on the web offer a significant source of information for almost anything.

Search engines have been evolved and research has been conducted in order to exploit the web resources. Since Internet became a public common currency and mostly referred to as the World Wide Web, its content growth made the use of search engines a gateway to information. Without Google, Yahoo! or Altavista the web as it is today would be useless. Terabytes of data in millions of web pages are impossible to be searched without the use of special Information Retrieval (IR) techniques offered by the current search engine implementations.

Nevertheless, still much work needs to be done so as to render all this information retrievable. It is well known that there is a large quantity of existing data on the web stored using relational database technology. This information is often referred to as the Deep Web [7] as opposed to the Surface Web comprising all static web pages.

Deep Web pages don't exist until they are generated dynamically in response to a direct request. As a consequence, traditional search engines cannot retrieve their content and the only manageable way of adding semantics to them is attacking directly its source: the database.

The creator of the web, Tim Berners-Lee proposed the idea of the 'Semantic web' to overcome the handicaps referenced. As stated in [1], the Semantic Web is about bringing "structure to the meaningful content of Web pages, creating an environment where software agents, roaming from page to page, can readily carry out sophisticated tasks for users". The Semantic Web will not replace the Web as it is known today. Instead, it will be an addition, an upgrade of the existing content in an efficient way that will lead to its integration into a fully exploitable world-wide source of knowledge. This process will consume much effort. Researchers have developed Semantic Web tools to facilitate this effort.

The key role to this effort is played by ontologies. Their use aims at bridging and integrating multiple and heterogeneous digital content on a semantic level, which is exactly the point of the idea. Ontologies provide conceptual domain models, which are understandable to both human beings and machines as a shared conceptualization of a specific domain that is given [6]. With the use of ontologies, content is made suitable for machine consumption, contrary to the content found on the web today, which is primarily intended for human consumption.

Nevertheless, ontologies suffer from a scalability problem. Keeping large amounts of data in a single file is a practice with low efficiency. Ontologies should rather describe content than contain it. The application developed and presented in this paper introduces a solution to this problem. Current data is and should be maintained separately while ontologies can be developed in order to enhance its meaning and usefulness.

The paper is organized as follows: Section 2 provides the definitions of the aspects discussed in this paper. Section 3 details the mapping architecture, the mapping process and the integrity checks performed to assure consistent results. In Section 4 we present our case study and compare it to related efforts in Section 5. We conclude and discuss about future work in Section 6.

2 Definitions

This section provides a background definition of our work. This will make clearer every aspect of the proposed idea.

2.1 Description Logics Ontologies

Formally, an ontology can be defined as a model of a knowledge base (KB). The main difference between an ontology and a KB is that the latter offers reasoning as an addition to the model. Thus, it is not possible to extract implicit knowledge from the ontology without the use of reasoning procedures. A KB created using Description Logics comprises the two following components [16].

The first part contains all the concept descriptions, the intentional knowledge and is called Terminological Box (TBox). The TBox introduces the terminology, the vocabulary of an application domain. It can be used to assign names to complex descriptions of concepts and roles. The classification of concepts is done in the TBox determining subconcept/superconcept relationships between the named concepts. It is then possible to form the subsumption hierarchy.

The second part contains the real data, the extensional knowledge and is called Assertional Box (ABox). The ABox contains assertions about named individuals in terms of the vocabulary defined in the TBox.

2.2 Relational Models

According to [4], a database is a collection of relations with distinct relation names. The relation consists of a relation schema and a relation instance. The relation schema consists of the schemas for the relations in the database. The relation instance contains the relation instances, whose contents are sets of tuples, also called records. Instances can also be though of as tables, in which each tuple is a row. All rows have the same number of fields. Fields' domains are essentially the type of each field, in programming language terms (i.e. string, boolean, integer etc).

In order to provide access to their contents, databases support many query languages, but SQL is the practical choice. SQL is powerful and provides various means of manipulating relational databases. The inputs and outputs of SQL queries are Relations. Queries are evaluated using instances of input relations (tables) and produce instances of output relations. SQL is said to be relationally complete – since it is a superset of relational algebra – meaning that every query that can be posed to a relational algebra can be expressed in SQL. In other words, with the use of SQL queries, there is no combination of subsets of tuples in a database that cannot be retrieved.

2.3 Mapping Relational Database Contents to Ontologies

In a simplified point of view, we can compare the schema of a relational database to the TBox of a KB and the instance to the actual data to the ABox. We could claim that databases are similar to Knowledge Bases because of the fact that they both are used to maintain models and data of some domain of discourse (UofD) [17]. There is, though, a great difference, besides the fact that databases manipulate large and persistent models of relatively simple data while knowledge bases contain fewer but more complex data. Knowledge bases can also provide answers about the model that have not been explicitly stated to it. So, mapping a database to a KB is enhancing the database's ability to provide implicit knowledge by the use of the terminology described in the TBox.

In our work, we succeeded in mapping the relational database contents to the TBox of the ontology. The description language chosen is OWL-DL [2, 5], based on standard predicate calculus. The ontology produced does not contain an ABox, only a reference to the dataset in the database. The dataset can be any data combination, not just table tuples. A visualization of the mentioned work is shown in the figure below.

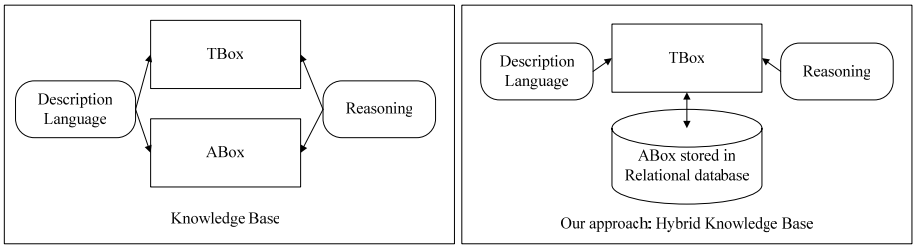


Fig. 1. Main Idea. The Assertion Box of the Knowledge Base is kept separately in a relational database. The resulting hybrid ontology will contain references to database datasets that comprise the class individuals

What has been accomplished in the current work is to reference database tuples through the ontology TBox. Common approaches describe mapping mechanisms between ontology classes and database tables. Our enhancement from the scope of the Semantic Web lies on the fact that the mappings are capable of corresponding class individuals (alt. instances) to any possible dataset combination. From the view of database theory, we enriched the database schema structure to include description logics and answer queries on a semantic level.

2.4 Issues and Limitations Rising from the Mapping Process

There are significant differences between the knowledge bases and databases. Among the differences is the 'open-world' semantics in opposition to the 'closed-world' semantics that hold respectively for each one.

The relational database schema abides by the so called 'closed world assumption'. The system's awareness of the world is restricted to the facts that have been explicitly told to it. Everything that has not been stated as true fact is false. In a 'closed world', a null value about a subject's property denotes the non-existence i.e. a null value in the 'isCapital' field of a table 'Cities' claims that the city is not a capital. The database answers with certainty because, according to the closed world assumption, what is not currently known to be true is false. Thus, a query like 'select cities that are capitals' will not return a city with a null value at a supposed boolean isCapital field.

On the other hand, a query on a KB can return three types of answers that are true, false and 'cannot tell'. The 'open world assumption' states that lack of knowledge does not imply falsity. So a question in an appropriate schema 'Is Athens a capital city?' will return 'cannot tell' if the schema is not informed while a database schema would clearly state that 'no'.

Class properties were not mapped to relations between database tables. The reason is that regarding the ontology semantics, the class hierarchy is fully different from the property hierarchy. Foreign key constraints involving relations, though, can be mapped to Object Properties linking OWL classes. This approach however, would not produce more powerful results; it would rather complicate reasoning checks. So, in the work that is presented here, concept integrity checks are being held only during the mapping process without being an extra burden to the resulting ontology.

3 Mapping Architecture and Process

In this chapter we clarify the architecture of the technique through which the target goal is reached. We describe the mapping process, analyze the integrity constraints and explain how the user can retrieve database data through the intermediate mapping ontology.

3.1 Mapping Process

Our proposed mapping method consists of four steps. First, we capture data from the database. The data can be any combination of datasets. More particularly, the ability is given to select the desired subset of records belonging to various tables. In step two, we select a class of the ontology. Step three is the most important as all consistency checks are being performed in this step along with evaluation, validation and refinement of the mapping. In step four takes place the modification of the resulting ontology and the additional information is added to it. These four steps can be repeated as many times as required. By saving the mapping in a new ontology, additional information will be kept such as the database type and name, the initial ontology name and when the mapping is last modified.

The final result is the initial ontology enhanced to include references to datasets. These references will be under the form of class properties in the ontology, all assigned as value a string containing the actual SQL query that returns the dataset. Each query posed to the ontology will check the existence of the mapping property. In the case that it does not exist, the results are no affected. Otherwise, the results will be database entries instead of class individuals.

3.2 Validation of the Mapping Process – Consistency checks

During the mapping process, tests are run to insure the concept's consistency. These tests require enough computational time and are therefore computed once, responding with a positive or negative answer, informing the user in each case. Every time a new mapping is defined by the user, the following requirements must be met.

First of all, two disjoint classes cannot have mappings to the database that contain common records or subsets of records. Even in the case of finding a single datum in common, the mapping will be rejected. We note that 'common data' will as well be considered data that belong to different fields of two tables connected with a foreign key relationship. Disjoint classes cannot be mapped to records of these fields because a foreign key connects the two tables semantically. The classes' disjointment does not allow such connections. This constraint reassures that the datasets will be as well disjoint. In addition, the same check is run respectively for the class' sub- and superclasses.

Moreover, a check is performed to reassure that subclass hierarchy is maintained in the mapped data. More specifically, if a class A is subclass of a class B and the two classes are mapped to record sets R(A) and R(B) respectively, then R(A) must be a

subset of R(B) as well. This check must hold recursively for every mapping statement. Special care is taken of table fields that have foreign key relationships. Subclasses can be mapped to field entries outside the domain of the superclass mapping as long as these fields are referred to by foreign keys. In conclusion, for a mapping to be accepted, the set of data for each class must be a subset of the class' superclasses.

The checks are performed after each mapping command. The checks were intentionally not strict as the system was designed keeping in mind the final user: the domain expert who will produce the mapping result. Thus, only the necessary consistency checks were implemented balancing the user's freedom with the concept's integrity.

3.3 Semantic Query Execution

Our implementation includes the functionality of imposing queries by terms of semantic web query languages. Fig. 3 illustrates how the application handles user's queries.

First, users' queries are parsed and checked syntactically. The query would instantly return with the desired results but here is where the mapping intervenes. Instead of returning the class resources, for each class we check if there exists the mapping property. If the related property is found, the query is redirected to the database. The results are then formatted and served to the user.

The achievement lies in the fact that we added an extra layer of interoperability, between the application GUI and the actual data. This hybrid approach leaves the ontology and the actual data intact, only serving as a semantics addition to current systems. The generalization of the current concept to current web technologies is about to offer a semantic layer that will add the desired intelligence without modifying legacy systems.

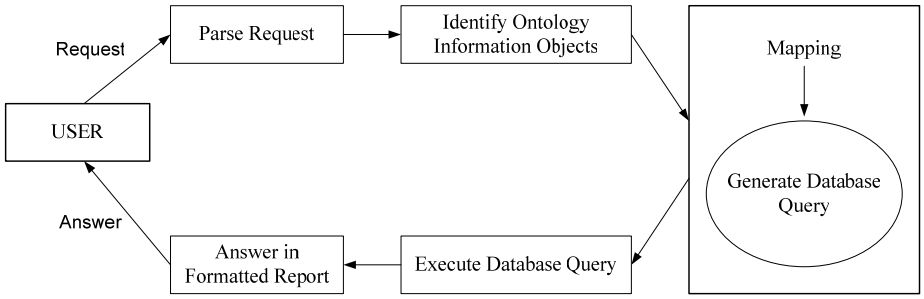


Fig. 3. After the generation of the mapping ontology, user's queries are processed invisibly by the database mapping mechanism

3.4 Implementation-Specific Information

Our implementation was built as a plug-in on top of the Protégé [3] ontology editor. Protégé was chosen among other open-source ontology editors like SWOOP [18],

SMORE [19], OilEd [20] or Ontolingua [21] because of its extensible structure and documentation offered about writing a plug-in. Protégé is java-based and supports ontologies authored in OWL.

For a database backend, the VisAVis tool supports connections via JDBC to the MySQL and PostgreSQL database management systems.

As far as it concerns the Semantic Web query languages, several recommendations have been considered. The issue is that query languages about OWL ontologies are still in their infancy [22]. We would therefore have to select between the SPARQL and RQL families of query languages for RDF. The practical choice is RDQL [15] as it is implemented and fully supported in the Jena [12] framework upon which Protégé is built. RDQL has recently been submitted to the W3C for standardization (October 2003).

4 The 'VisAVis' Use Case Scenario

In this section, we give a practical example of the mapping method described above. For our example's purpose, we will use an OWL-DL ontology that can be found at [3] and includes some terms concerning tourism, such as lodgings, activities and destinations. Our aim is to map the classes of the ontology to data that refer to the same concept as the ontology classes. That is why, in this example, we are using as well a database containing contextual data.

We have implemented the mapping method described in the previous section, as a Protégé tab plug-in, colorfully named as VisAVisTab and shown in Fig. 4. The first step is to open the ontology that contains the classes of interest. Then, we connect with the corresponding database that contains the real data. At that point, we can see both the ontology and the database hierarchy and we are able to select a set of data from the database using the application's graphical user interface. In fact, the application constructs, in the background, an SQL query that, when executed, returns the desired dataset. This SQL query, and not the data set itself, will be correlated with a certain ontology class selected by the user.

To demonstrate VisAVis in action, we are using a database called 'travel' that contains among others the tables 'cities', 'hotels', 'museums' and 'activities'. The table 'activities' contains information about activities offered to tourists, including its description and its type (surfing, hiking etc.). The activities' types are stored in the 'activity_type_id' column, which is foreign key to the 'id' column of the 'activities_types' table.

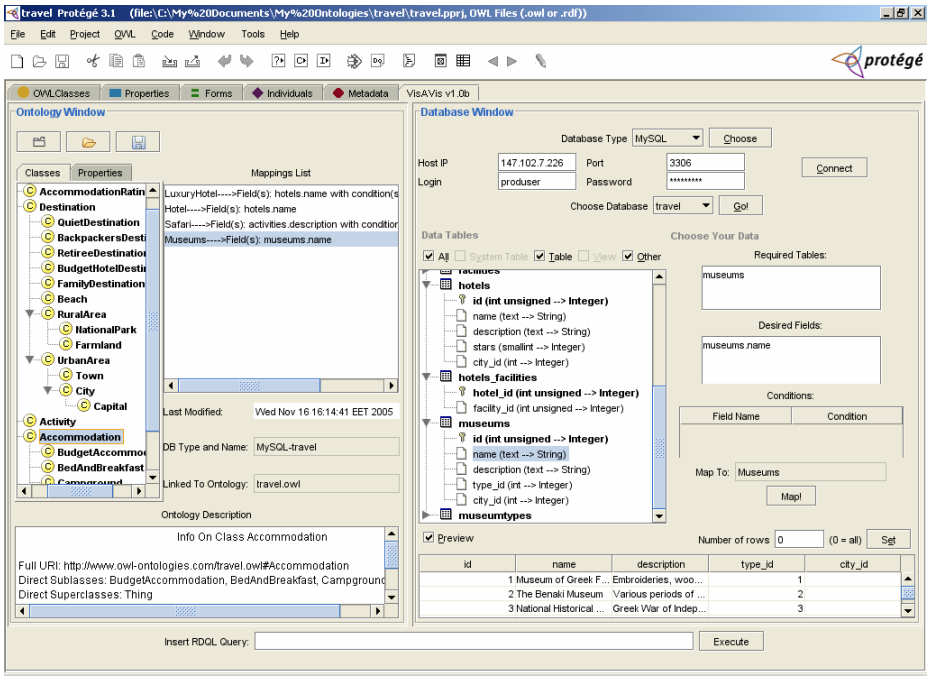


Fig. 4. The VisAVis Protégé Tab. The left panel contains the class hierarchy, the mapped classes list and information about the selected class as well as for the ontology itself. The right panel is dedicated to the database. The user can select multiple tables and fields, declare possible mapping conditions and confirm mappings. At the bottom of the screen, RDQL queries can be executed

A common approach would be to map the class 'Activity' of the OWL ontology to the columns 'id', 'description', 'activity_type_id' from the 'activities' table and to the 'id', 'name', 'description' columns of the 'activities_types' table. The data selection process is easily carried out by dragging the columns needed from the database hierarchy and dropping them into the 'Desired Fields' field. In order to map the class 'Hiking', which is a subclass of the 'Activity' class, we should add a condition that would enable us to select only the activities of the certain type. Likewise, the classes 'Surfing' and 'Safari', which are also subclasses of the 'Activity' class, could be mapped to the appropriate data that would satisfy the necessary conditions. The conditions can be applied to a certain set of columns by dragging the necessary column, dropping it to the 'Conditions' field and then, filling in the rest of the constraint. When the data selection has finished, the user can complete the mapping process by pressing the 'Map!' button.

It is worth mentioning that when columns from two or more tables are selected, our application includes in the SQL query an additional condition representing the referential integrity constraint. In other words, foreign keys are taken into account whether they are user-specified or not and added to the final SQL query. This happens even in the case where two or more tables have two or more distinct relations

between them. For instance, for the following example a mapping concerning the class ‘Activity’ is the following:

```
  Hiking -> Field(s): activities.description with condi-
tion(s): (activities_types.name=’Hiking’)
```

The corresponding SQL query, which is automatically generated by our SQL builder, is shown below in a snapshot of the enhanced ontology:

```
<owl:Class rdf:about="#Hiking">
    <queryString>SELECT activities.description FROM
    activities, activities_types WHERE
    (activities.activity_type_id=activities_types.id)
    AND (activities_types.name = "Hiking")
    </queryString>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Sports"/>
    </rdfs:subClassOf>
</owl:Class>
```

As we notice in the above sample, the mapping process has as a result the introduction of a datatype property to the class. The property is arbitrarily named ‘queryString’ and is given as domain the class’ domain and as value the corresponding SQL query. Thus, the initial OWL ontology is enhanced with additional properties that represent the mappings accomplished. Following the same course of actions, we can map the entire ontology to sets of data from the database.

We should also note that during the mapping process, consistency checks are executed in order to check whether mappings conform to the ontology rules or not as described in Section 3.2. For instance, if we tried to map to the class ‘Activity’ some data from the ‘hotels’ table, the mapping validator would reject this mapping as inconsistent with the ontology rules, since classes ‘Activity’ and ‘Accomodation’ are disjoint.

Finally, in order to test the whole mapping process, our application enables the execution of a semantic query, which will return actual data from the database, instead of ontology elements. As it has already been mentioned, a similar approach could be followed in a sophisticated Semantic Web search engine, that would first execute a semantic query and then, using mappings between concepts and sets of actual data or documents, it would return these sets of data or documents.

5 Related Work

The Semantic Web is the new WWW infrastructure that will enable machine process of the web content and seems to be a solution for many drawbacks of the current Web. The semantic information addition to the current web will create a new information layer on top of the current technologies. Ontology to database mapping is an active research topic to this direction based on the observation that the web content is

dynamic and most of it originates from databases. Several approaches have been presented aiming at semantic integration proposing techniques for database-to-ontology correspondence. Related work can be divided in two fields, database-to-ontology migration and database-to-ontology collaboration. Our approach belongs to the latter; hence similar work will be given reference in the current chapter.

The Ontomat Application Framework introduced in [9] was one of the first prototypes for database and semantics integration. It was a front-end tool built upon a component-based architecture. Ontomat Reverse [10] is part of the framework and offers the means to semi-automatically realize mappings between ontology concepts and databases through JDBC connections. Nevertheless, it only supports RDF graphs and mappings between database tables on the server and ontology classes on the client.

D2RMap was first presented in [11] and provides means to declaratively state ontology-to-database mappings. D2R is based on XML syntax and constitutes a language provided to assign ontology concepts to database sets. The result includes SQL commands directly to the mapping rules and is capable of performing flexible mappings. The mapping procedure in D2R is similar to ours. Results are extracted in RDF, N3, RDF or Jena [12] models.

R_2O [14] is another declarative language and serves the same purpose as it is obvious in the full title, Relational 2 Ontology. The R_2O language is fully declarative and extensible while expressive enough to describe the semantics of the mappings. In general, like D2R, it allows the definition of explicit correspondences between components of two models.

The Unicorn workbench [13] was developed aiming at the semantic integration which is as well the concept of our work. Unicorn's architecture is two-layered meaning that the external assets layer is separated from the model-information layer.

6 Conclusions and Future Work

We have developed a novel, integrated and semi-automated approach for mapping and reusing large sets of data into the Semantic Web. Using the tool described above, it is possible to create a mapping between any relational database and any ontology. For the mapping to have a meaning, however, it is essential that the concepts described are at some point similar. For example, it has no meaning mapping a database containing a company's customers to an ontology describing plants. The logical aspect is left to the judgment of the user.

The current version implements a rather small subset of the SQL language. The mapping to the actual data is in fact accomplished using 'select from where' functions. We are currently working on enriching the ontology result to include more functions such as 'group by' and 'order by'. Functions such as max(), count() and avg() are not supported either, in the current version. Support needs to be added for several RDBMS such as Oracle or Microsoft SQL Server to cover a significant percentage of today's choices. Yet, these topics are only implementation-specific and of no special research interest. They are however needed if the aim is to provide the Protégé plug-in the tool's corresponding plug-in section.

An important open issue in the current implementation is also the fact that queries only return results from the database. The user should also be informed about class individuals, not only database records. The current development is not purposed to distinct between ontologies and databases; it is about offering a collaboration framework and an intermediate layer for database exploitation by Semantic Web applications.

The resulting enhanced ontology is subject to easy programmatic access. In addition, once the mapping result is created, our tool will only be needed in case of change of the database schema. Practice has proven that database schemas are not frequently changing. Our intention is to use such ontologies combined with software agents in order to create an intelligent environment where data search and retrieval will implement the Semantic Web concept: users will be able to search among large volumes of semantically annotated data.

References

1. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web - A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. Scientific American, 2001.
2. I. Horrocks, P. Patel-Schneider, F. van Harmelen: From SHIQ and RDF to OWL: the making of a Web Ontology Language.
3. The Protégé Ontology Editor and Knowledge Base Framework, http://protege.stanford.edu/.
4. Ramakrishnan and Gehrke: Database Management Systems 3rd Edition, McGraw Hill 2003, Chapter 3: The Relational Model, pages 57-99.
5. OWL Web Ontology Language Overview, http://www.w3.org/TR/owl-features/.
6. Gruber, 1993: Toward principles for the design of ontologies used for knowledge sharing.
7. Bergman MK. *The Deep Web: Surfacing hidden value*. White paper. Sept 2001.
8. Codd, E. F. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387
9. Erol Bozsak et al: KAON - Towards a large scale Semantic Web. *E-Commerce and Web Technologies, Third International Conference, EC-Web 2002, Aix-en-Provence, France, September 2-6, 2002, Proceedings*, volume 2455 of Lecture Notes in Computer Science, pp. 304-313. Springer, 2002.
10. *R. Volz et al: Unveiling the hidden bride*: deep annotation for mapping and migrating legacy data to the Semantic Web, *Web Semantics: Science, Services and Agents on the World Wide Web 1 (2004) 187–206.*
11. Christian Bizer: D2R MAP – A Database to RDF Mapping Language, *The twelfth international World Wide Web Conference, WWW2003, Budapest, Hungary*.
12. Jena: A Semantic Web Framework for Java, http://jena.sourceforge.net/.
13. Jos de Bruijn: Semantic Integration of Disparate Data Sources in the COG Project, in *Proceedings of the International Conference on Enterprise Information Systems (ICEIS2004)*, Porto, Portugal, 2004.
14. Barrasa J, Corcho O, Gómez-Pérez: R2O, an Extensible and Semantically Based Database-to-Ontology Mapping Language. A. *Second Workshop on Semantic Web and Databases (SWDB2004)*, Toronto, Canada. August 2004.
15. A. Seaborne, RDQL – RDF Data Query Language, part of the Jena RDF Toolkit, HPLabs Semantic Web activity, http://hpl.hp.com/semweb/, RDQL grammar: http://www.hpl.hp.com/semweb/rdql-grammar.html, Online only.

16. Franz Baader, Werner Nutt: Basic Description Logics. In the Description Logic Handbook, edited by F. Baader, D. Calvanese, D.L. McGuiness, D. Nardi, P.F. Patel-Schneider, Cambridge University Press, 2002, pages 47-100.
17. A. Borgida, M. Lenzerini, R. Rosati. Description Logics for Databases. In the Description Logic Handbook, edited by F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider, Cambridge University Press, 2002, pages 472-494.
18. Aditya Kalyanpur, Bijan Parsia, James Hendler: A Tool for Working with Web Ontologies, In *Proceedings of the International Journal on Semantic Web and Information Systems, Vol.1, No.1, Jan-Mar 2005*.
19. Aditya Kalyanpur, Bijan Parsia, James Hendler, and Jennifer Golbeck. SMORE - semantic markup, ontology, and RDF editor.
20. Sean Bechhofer, Ian Horrocks, Carole Goble, Robert Stevens. OilEd: a Reason-able Ontology Editor for the Semantic Web. In *DL2001, 14th International Workshop on Description Logics, Stanford, USA, August 2001*.
21. A. Farquhar, R. Fikes, & J. Rice. The Ontolingua Server: A Tool for Collaborative Ontology Construction. Knowledge Systems, AI Laboratory, 1996.
22. James Bailey, François Bry, Tim Furche, and Sebastian Schaffert: Web and Semantic Web Query Languages: A Survey, In: *Reasoning Web, First International Summer School 2005*, Norbert Eisinger and Jan Małuszynski, editors. Springer-Verlag, LNCS 3564, 2005.

The Virtual Query Language for Information Retrieval in the SemanticLIFE Framework*

Hanh Huu Hoang and A Min Tjoa

Institute of Software Technology and Interactive Systems
Vienna University of Technology
Favoritenstraße 9-11/188
A-1040 Vienna, Austria
+43 1 58801 18861
{hanh, tjoa}@ifs.tuwien.ac.at

Abstract. Creating an innovative mediator environment such as graphical user interfaces, lighter weight languages to help the users in the challenging task of making unambiguous requests is crucial in semantic web applications. The query language in the Virtual Query System (VQS) of the SemanticLIFE framework is designed for this purpose. The proposed query language namely Virtual Query Language (VQL) is in a further effort of reducing complexity in query making from the user-side, as well as, simplifying the communication for components of the SemanticLIFE system with the VQS, and increasing the portability of the system.

1 Motivation

Making unambiguous queries in the Semantic Web applications is a challenging task for users. The Virtual Query System (VQS) [7] of the SemanticLIFE framework [1] is an attempt to overcome this challenge. The VQS is a *front-end* approach for user-oriented information retrieval.

The issue is in the user-side, where users are required to make queries for their information of interest. The RDF query languages are powerful but they seem to be used for back-end querying mechanisms. To users, who are inexperienced with them, these query languages are too complicated to use. Additionally, the communication of components of the SemanticLIFE system with the VQS requires the facility to transfer their requests without knowledge of the RDF query language. Another important point is the portability of the system, i.e. if the system is bound to specific RDF query language, we could have problems when shifting to another one.

In the effort of coming over the above tackles, we, in this paper, present a new query language namely *Virtual Query Language* (VQL). The language is used by the VQS for information querying in the SemanticLIFE system. The

* This work has been generously supported by ASEA-UNINET and the Austrian National Bank within the framework of the project Application of Semantic-Web-Concepts for Business Intelligence Information Systems - Project No. 11284.

VQL is a much lighter weight language than RDF query languages; but it offers interesting features to complete the tasks of information querying in the Semantic Web applications.

The rest of this paper is organized as follows: firstly, similar approaches is briefly presented in Section 2. Next, details of the VQL are pointed out in Section 3. Section 4 introduces VQL query operators; and the issues of mapping a VQL query to the respective RDF query are then described in Section 5. Finally, the paper is concluded with a sketch of the future work.

2 Related Work

There are two main approaches to reduce the difficulty in creating queries from user-side in Semantic Web applications. The first trend is going to design the friendly and interactive query user interfaces to guide users in making the queries. The high-profiled examples for this trend are GRQL [2] and SEWASIE [4].

GRQL - Graphical RQL - relies on the full power of the RDF/S data model for constructing on the fly queries expressed in RQL [8]. More precisely, a user can navigate graphically through the individual RDF/S class and property definitions and generate transparently the RQL path expressions required to access the resources of interest. These expressions capture accurately the meaning of its navigation steps through the class (or property) subsumption and/or associations. Additionally, users can enrich the generated queries with filtering conditions on the attributes of the currently visited class while they can easily specify the resource's class(es) appearing in the query result.

Another graphical query generation interface, SEWASIE, is described in [4]. Here, the user is given some pre-prepared domain-specific patterns to choose from as a starting point, which he can then extend and customize. The refinements to the query can either be additional property constraints to the classes or a replacement of another compatible class in the pattern such as a sub or superclass. This is performed through a clickable graphic visualization of the ontology neighbourhood of the currently selected class.

The second approach of reducing complexity is the effort in creating much lighter query languages than expressive RDF query languages. Following this trend, the approach in [6] and another one known as GetData Query interface [10] are hight-rate examples.

[6] describes an a simple expressive constraint language for Semantic Web applications. At the core of this framework is a well-established semantic data model with an associated expressive constraint language. The framework defines a 'Constraint Interchange Format' in form of RDF for the language, allowing each constraint to be defined as a resource in its own right.

Meanwhile, the approach of GetData Query interface [10] of TAP[1] expresses the need of a much lighter weight interface for constructing complex queries. The reason is that the current query languages for RDF, DAML, and more generally

[1] TAP Infrastructure, http://tap.stanford.edu/.

for semi-structured data provide very expressive mechanisms that are aimed at making it easy to express complex queries. The idea of GetData is to design a simple query interface which enables to network accessible data presented as directed labeled graph. This approach provides a system which is very easy to build, support both type of users, data providers and data consumers.

Our approach, VQL, continues this trend to design an effective and lighter weight query language to assist the users make queries in simple manner and simplify the communication between components of the SemanticLIFE system with the query module - the VQS.

3 The Virtual Query Language - VQL

3.1 The Goals of the VQL

A number query languages have been developed for the Semantic Web data such as data in form of RDF (RQL [8], RDQL [11], SPARQL [9], and iTQL [12]). Why do we need yet another query language?

These query languages all provide very expressive mechanisms that are aimed at making it easy to express complex queries. Unfortunately, with such expressive query languages, it is not easy to construct queries to normal users as well as to ask abstract information. What we need is a much lighter weight query language that is easier to use. A simple lightweight query system would be complementary to more complete query languages mentioned above. VQL is intended to be a simple query language which supports "semantic" manner from users' queries. In the context of the VQS and SemanticLIFE system, we can see the aims of VQL as follows:

- VQL helps clients making queries without knowledge of RDF query languages. The user just gives basic parameters of needed information in VQL queries, and would receive the expecting results.

- VQL assists users in navigating the system via semantic links or associations provided in the powerful query operators based on ontologies.

- VQL simplifies the communication between Query module and other parts. Since the components asking for information do not need to issues the RDF query statement, which is uneasy task for them. As well as this feature keeps the SemanticLIFE's components more independent.

- VQL enables the portability of the system. Actually, the SemanticLIFE and VQS choose a specific RDF query language for the its back-end database. However, in the future, they probably could be shifted to another query language, so that this change does not effect other parts of the systems, especially the interface of the system database.

3.2 The Syntax of VQL

Query Document Syntax. Generally, a VQL query is a document having four parts: parameters, data sources, constraints, and query results format as the schema depicted in Fig. 1.

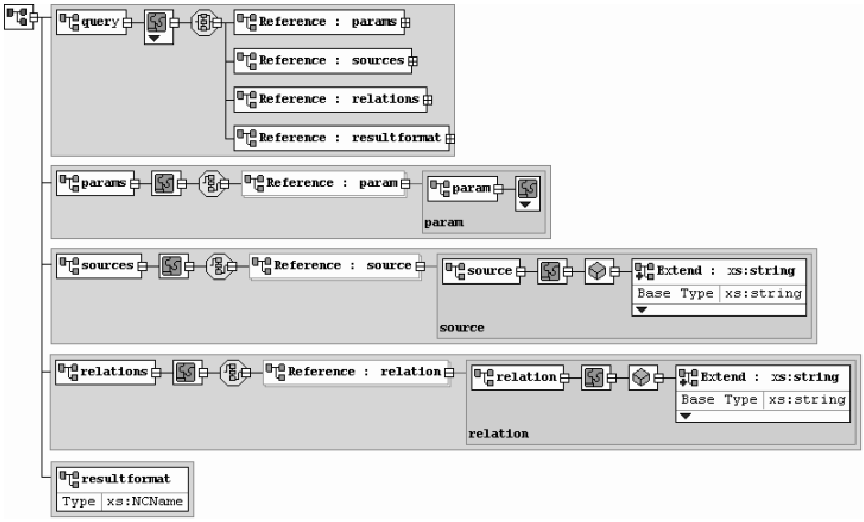


Fig. 1. The schema for general VQL queries

The first part contains parameters of specifying the information of interest. A parameter consists of a variable name, the criteria value, and additional attributes for sorting or eliminating unneeded information from the results. The second part is used for specifying the sources where the information will be referred to extract from. Obviously, the information need defined in the first part must be related to the sources specified in this part.

Thirdly, the constraints of the query are defined in the third part of the document. Here the relations between sources, parameters are combined using the VQL operators. Finally, the format of query results is identified in the fourth part. VQL supports four query results formats that are XML, text, RDF graph, and serialized objects of query result sets. This provides flexibility for clients to process the query results.

XML-based Format. A standard format for information exchange, a easy-to-use and familiar-to-clients format, a widely accepted standard, and a flexible and open format are the requirements for the VQL query document. We have considered some alternatives and decided to choose XML as the format for VQL query syntax. A XML-based VQL query is structured as follow (Fig. 2):

- The top level or the body of query is `<query>` element. Here, the type of the query must be specified in `type` attribute. The reserved terms used for this attribute are `"data"`, `"schema"`, `"itql1"` and `"itql2"` for different VQL query types (see section 3.4). For example, `<query type="data">` is a VQL data query.

- The second level contains required sub-elements. Depending on the type of a VQL query the elements are used respectively: for the data query, elements of

```
<query type="data">
 <params>
  <param show="1" name="s1:messageTimeStamp">2005-11-01</param>
  <param show="0" name="s2:messageTimeStamp">2005-11-31</param>
 </params>
 <sources>
  <source name="fileupload">FileUpload</source>
  <source name="browsingsession">BrowsingSession</source>
 </sources>
 <relations>
  <relation id="1" param="s1" source="">dt:gt</relation>
  <relation id="2" param="s2" source="">dt:lt</relation>
 </relations>
 <resultformat>xml</resultformat>
</query>
```

Fig. 2. An XML-based VQL Query Example

`<params>`, `<sources>`, and `<relations>` are used once for each. These elements have their children specified in the third level. Fig. 2 is an example. For the schema query or the iTQL type 1 query, we use only one `<statement>` element which contains a RDF query statement (Fig. 3). For the iTQL2 query, elements of `<select>`, `<from>`, `<where>`, `<orderby>`, `<limit>` and `<offset>` are used in the same way, where last three are optional as described in Fig. 4. Moreover at this level, we must identify the query results format in the `<resultformat>` element by a term among `"xml"`, `"text"`, `"rdf"`, or `"object"`.

- The third level elements are only applied for the VQL data queries. The elements are children of `<params>`, `<sources>`, and `<relations>`; and the tags consequently are `<param>`, `<source>`, and `<relation>` with their own attributes.

- `<param>` *element*: each parameter is identified by this element with required attributes: `show` and `name`. While `show` is set to 1 or 0 that means the result of this parameter is shown in the result sets or not; `name` has two parts: a *variable* and the $meta - information$ which are put together with `":"`, i.e. `variable:metainfo`. Besides, this element has two optional attributes known as `order="1"/"0"` and `exclude="string"`. The `order` attribute is used for sorting, while `exclude` is for excluding some information from query results. The element is enclosed with an optional value for filtering.

- `<source>` *element*: names of concerned sources for users' requests will be put here. This element has only one attribute `name` which is an internal name of data sources. This internal name is assigned automatically by the system.

- `<relation>` *element*: contains a constraint of the VQL query. The required attributes of each constraint are: `id=` *"number"*, `param=` *"variable"*, `source=` *"source-name"*, and optional `or=` *"id"*. The `id` is assigned a number, *variable* in the related `<param>` is used in this `param` attribute. Attribute `source` is identified with $source - name$ or left empty in the case of only one data source specified; and `or` is assigned by `id` of another `<relation>` in order to make an OR expression. The operator for the constraint is put as value of the `<relation>` element.

3.3 Operators and Expression in VQL

Logic Operators. VQL's logic operators consist of AND, OR and NOT. While NOT is defined in each `<param>` element by using the `exclude` attribute, AND and OR operators are identified in `<relation>` items. OR is defined by the `"or"` attribute, e.g. `or="2"` means that the current constraint will be combined with the constraint number "2" using logic OR operator. AND is implied in a relation without `"or"` attribute.

Comparison Operators. The comparison operators are used in `<relation>` part of VQL queries. These operators are presented as follow:

equal An equal comparison in form of triples which is used by leaving the `<relation>` item empty.

gt The operator means `GREATER-THAN` which is used for comparing values of basic data types such as String, numbers.

lt Similarly to the previous one, the operator means `LESS-THAN`.

dt:gt This operator is used for comparing the date/time value `AFTER` a point of time. Value format of this type confronts XML Schema (XSD) data types.

dt:lt Similarly to the `dt:lt` operator; but it means for the date/time value `BEFORE` a point of time.

str:match This is the pattern matching operator for strings. This comparison operator uses common pattern expression.

ft:match This is the powerful operator relying the full-text index applied in SemanticLIFE's metastore. It is used for searching the full-text data such as content of a file or email attachments, or stored WWW pages.

Expressions. In order to formulate the expressions for the query criteria, VQL uses `"or"` attribute in `<relation>` elements as boolean OR operator to combine these constraints first, and then it uses boolean AND to combine into the final expression. The sequence of `<relation>` elements are important in combining expressions.

3.4 The VQL Query Types

Data Query Type. VQL data query type is commonly used for information querying. A query of this type consists of the four parts as discussed above. In order to inform VQL parser to process the query as a VQL data query, we identify `"data"` term in the query: `<query type="data">`.

An example for this query type is shown in Fig. 2: the query retrieves messages' time-stamp of files uploaded and browsed web pages in the SemanticLIFE repository in November 2005.

From this query type, we obtain deductive queries for special operations in semantically information retrieval. These operations help users easily get information of interest and obey the principle of VQL design: *"ask less, get more."* The deductive queries, so-called VQL query operators, are detailed in Section 4.

Schema Query Type. The syntax of this query type consists of two parts: the first one is the `<statement>` element containing a RDF query statement; and the second part is used to set the query results format. Necessarily, `"schema"` must be identified in the query body. A schema query example is illustrated in Fig. 3.

```
<query type="schema">
 <statement>
   select $s
   from &amp;lt;rmi://192.168.168.174/OntologyModel&amp;gt;
   where $s &amp;lt;rdfs:subClassOf&amp;gt;
   &amp;lt;slifeont:FileUploadData&amp;gt; ;
 </statement>
<resultformat>text</resultformat>
</query>
```

Fig. 3. An example of VQL SCHEMA query

However, the question is that how does the user create RDF query statements? Actually, the schema or ontology queries are offered to clients in form query templates or programmatic VQL API.

iTQL Query Types. Similarly, with 'iTQL query types' RDF query statements are wrapped in VQL query documents. The name of 'iTQL query type' comes from the RDF query language used in the system. Since the SemanticLIFE framework uses Kowari [12] as its back-end, so that iTQL RDF query language is used for query statements. We also distinguish two ways of embedding the RDF statements: firstly, a whole statement is embedded; while their parts is embedded separately in the second type.

```
<query type="itql2">
 <select>$s $p $o</select>
 <from>
   rmi://192.168.168.174/slife#BaseModel
 </from>
 <where>$s $p $o</where>
 <resultformat>text</resultformat>
</query>
```

Fig. 4. An example of a VQL iTQL query type 2.

The format of the first iTQL query type, so-called VQL iTQL query type 1, is similar to the schema query depicted Fig. 3, where the term `"itql1"` is used instead of `"schema"` in the `<query>` tag. Meanwhile, in VQL iTQL query type 2, each parts of RDF query statement, such as `select` and `from`, are put in respective query parts. With the iTQL's `SELECT` statement, its parts are: `select`, `from`, `where`, `orderby`, `limit`, and `offset`.

Fig. 4 is an example of VQL iTQL query of type 2, in which `"itql2"` is specified in `<query>` tag. As depicted in the figure, the expressions of clauses of the RDF query statement are filled in respective elements of the VQL query.

3.5 Query Results Format.

In order to increase the flexibility in query results processing, VQL provides four query results formats that are XML format, text format, RDF graph, and serialized query results. In a VQL query, we identify the query results format in the `<resultformat>` element at the second level.

Query Results XML Format: `<resultformat>xml</resultformat>`

VQL query results XML format is similar to a W3C's format for SPARQL query results presented in [3]. This XML format has two main parts: the first one is the list of the query's parameters and needed additional information. The second part is the list of found instances.

Query Results Text Format: `<resultformat>text</resultformat>`

The text format of VQL query results is structured as following: the variable and its value of each item are then paired in form of `VAR_NAME = VALUE`. These pairs are connected by semicolons, and one row is for each items.

Query Results RDF Format: `<resultformat>rdf</resultformat>`

The RDF-graph format of query results is designed for semantic web client applications, here they prefer semantic enriched data. The query results will be transformed to RDF graphs before returning them to the clients.

Serialized Query Results Object: `<resultformat>object</resultformat>`

Concerning with the inside components communication in the SemanticLIFE system, components sometimes would like to use query results object without format transformation. The VQL serializes the results and send the serialized objects back to the asking component.

4 Query Operators of VQL

In this section, we present deductive queries for special operations in making complex queries in simpler manner which helps users *"ask less, get more."* This is the principle of building user-centered applications [5].

4.1 GetInstances Operator

GetInstances operator is the common form of VQL data queries. The operator retrieves appropriate information according the criteria described in parameters, sources, constraints of the query. The properties with `show` attribute set to `"1"` will be involved in the query results.

Fig. 2 is an example of *GetInstances* operator. As depicted, the query is about retrieving the message's time-stamp from 01/11/2005 to 30/11/2005 of uploaded files and browsed WWW pages. The operators in the constraint part, `"dt:gt"` and `"dt:lt"`, are combined using boolean `'AND'`.

4.2 GetInstanceMetadata Operator

This query operator assists the user easily retrieve all metadata properties and their values of resulting instances. This query operator is very useful when the

user does not care or not know exactly what properties of data instances are. The case could happen when he/she makes request; or he/she would like to get all metadata of these data items by the simplest way.

In order to make a *GetInstanceMetadata* operator, we must put one parameter in the query document with the reserved string `"METADATA"`. The other parameters could be used for the criteria of the query to filter the query results. The rest of the query document is similar to a normal data query. Fig. 5 describes a example of this operator.

```
<query type="data">
 <params>
  <param show="1" name="p0:messageTimeStamp">2005-11-01T00:00:00</param>
  <param show="1" name="p1:METADATA"/>
 </params>
 <sources>
  <source name="fileupload">FileUpload</source>
 </sources>
 <relations>
  <relation id="1" param="p0" source="">dt:gt</relation>
 </relations>
 <resultformat>xml</resultformat>
</query>
```

Fig. 5. VQL GetInstanceMetadata Query Operator.

Here, the query is about getting the *metadata* and their values of uploaded files which are sent from 01/11/2005, as well as the timestamps of these files.

4.3 GetRelatedData Operator

In semantic web applications, particularly in the SemanticLIFE system, finding relevant or associated information plays an important role. When we make a query to search for a specific piece of information, we also would like to see associated information to what we found. For example, when we are looking for an email message with a given email address, we also want to see the linked data to this email such as the contact having this email address, appointments of the person in the email, web pages browsed by this person. Obviously, this operator shows the VQL power and obeys the principle of *"ask less, get more"* for users.

In order to make a request of this operator, we must identify a parameter containing the a reserved word `"RELATED-WITH"` in the query document. The `<sources>` element is used to limit a range of data sources in searching associated information as presented in following figure (Fig. 6).

In this example, the query asks for instances of `Email` data source containing a specific email address, e.g. `hta@gmx.at`; and from found messages, the related information in the appreciate data sources, which are identified in `<sources>` of the query, will be located as well.

```
<query type="data">
 <params>
  <param show="0" name="p1:emailTo">hta@gmx.at</param>
  <param show="1" name="p2:RELATED-WITH"/>
 </params>
 <sources>
  <source name="email">Email</source>
  <source name="contact">Contact</source>
 </sources>
 <relations>
  <relation id="1" param="p1" source="email"/>
 </relations>
 <resultformat>xml</resultformat>
</query>
```

Fig. 6. VQL GetRelatedData Query Operator.

4.4 GetLinks Operator

This query operator operates by using the system's ontology and RDF graph pattern traversal. The operator aims at finding out the associations/links between instances and objects. For instance, we are querying for a set of instances of emails, contacts and appointments, and normally, we receive these data separately. However, what we are expecting here is that the links between instances (as well as objects) provided additionally. The links are probably properties of email addresses, name of the persons, locations, and so on.

GetLinks operator helps us to fulfill this expectation. This operator is similar to *GetInstanceMetadata* in the way of exploiting the metastore. While *GetInstanceMetadata* operator tries to get related instances based on analysis of a given link or information and the ontologies, *GetLinks* extracts the associations in instances or objects. In order to make a *GetLinks* operator, the reserved word `"SLINKS"` (*semantic links*) must be identified in one `<param>` element.

```
<query type="data">
 <params>
  <param show="1" name="p1:emailTo">hta@gmx.at</param>
  <param show="1" name="p2:conName">Hoang Thieu Anh</param>
  <param show="1" name="p3:SLINKS"/>
 </params>
 <sources>
  <source name="email">Email</source>
  <source name="contact">Contact</source>
  <source name="calendar">Calendar</source>
 </sources>
 <relations>
  <relation id="1" param="p1" source="email‟ or="2"/>
  <relation id="2" param="p2" source="contact"/>
 </relations>
 <resultformat>xml</resultformat>
</query>
```

Fig. 7. VQL GetLinks Query Operator.

We distinguish the detecting links between instances and objects as following: if sources are specified in the query document without any parameters except

"SLINKS", then the links will be detected between objects. Otherwise, if some parameters are shown, the links are implied for instances' associations. An example of *GetLinks* query operator is described in Fig. 7. The query will return the associations between instances having the given receiver's email and the contact name in three data sources Email, Contact, and Calendar.

By providing these operators, VQL offers a powerful feature of navigating the system by browsing data source by data source, instances by instances based on found semantic associations.

4.5 GetFileContent Operator

The SemanticLIFE system covers a large range of data sources, from personal data such as contacts, appointments, emails to files stored in his/her computer, e.g. the office documents, PDF files, media files. Therefore, a query operator to get the contents of these files is very necessary.

```
<query type="data">
 <params>
  <param show="0" name="p0:filePath">
   c:/slifedata/uploadedfiles/2006/01/15/CFP_WISM_06.pdf
  </param>
  <param show="1" name="p1:CONTENT"/>
 </params>
 <sources>
  <source name="fileupload">FileUpload</source>
 </sources>
 <relations/>
 <resultformat>xml</resultformat>
</query>
```

Fig. 8. VQL GetFileContent Query Operator.

Normally, to carry out this task, we must define two parameters in the query document, the first one is the file path got from the previous query; and the second one is defined with reserved word "CONTENT". In the <source> element of the query, a data source is identified as a reference; and the constraint part is often left empty. The query is described in Fig. 8 is an example of *GetFileContent* operator, where a content of the file having name "CFP_WISM_06.pdf" with its full path will be extracted from the metastore.

5 VQL Parser: VQL - RDF Query Languages Mapping

The VQL parser translates the VQL query to correspondent RDF query statements; accesses the metastore to get results, then transforms them to the appreciate format and sends the processed results back to the user. In this section, we present the first stage of a query process in the VQS [7] using VQL that is the mechanism to map VQL queries to RDF queries. The mappings consist of three phases: expressions mapping, syntax mapping; and semantic mapping.

5.1 Expression Mapping

Expressions in VQL queries are likely to be mathematical ones, while the "expressions" in RDF queries are in form of the triples. Therefore, an accurate expression mapping from normal expressions in VQL queries to triple expressions in RDF queries is crucial. VQL carries out this task as following steps:

1. Forming mathematical expressions from elements of a VQL query.
2. Representing the final aggregated expression into an expression tree.
3. From the expression tree, we traverse and formulate triple expressions for RDF query with referring to ontologies and related sources in the VQL query.

Example: Looking back at Fig. 2 as an example, an expression will be formed from described query's parts as follows (here $msgTS$ is used instead of $messageTimeStamp$ for shorter representation):

$$(msgTS \geq \#01/11/2005\#) \cap (msgTS \leq \#30/11/2005\#)$$

From this expression, we can create the expression tree as depicted in Fig. 9.

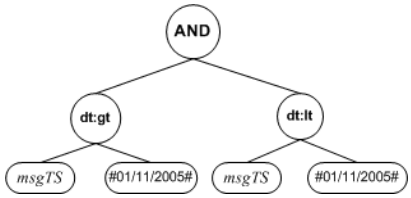


Fig. 9. The expression tree for the Example 4.

Using this tree, we generate the triple expressions for the RDF query by traversing the tree. The generated triple expressions are described below in iTQL RDF query language:

```
(<slife:msgTS> <tucana:after> '2005-11-01T00:00:00Z') and
(<slife:msgTS> <tucana:before> '2005-11-30T00:00:00Z')
```

where, `"slife"` is the namespace for the ontology and data schema of SemanticLIFE metastore; and `<tucana:after>` and `<tucana:before>` are comparison operators of iTQL.

Furthermore, the data sources are taken into account during mapping expressions. The specified data sources in the VQL query document (in `sources` part and `<relation>` elements) are used for either identifying the expression applied on them or generating correspondent queries for them. Hence from a VQL query, more than one RDF query are probably generated.

5.2 Syntax Mapping

Syntax mapping takes care the issue of translating from VQL query syntax to a RDF query language syntax. VQL queries are actually interpreted as `SELECT` statements of RDF query languages. The iTQL's `SELECT` statement contains three required clauses `select`, `from`, `where`, and optional clauses such as `orderby`, and `exclude`. The syntax mapping will parse VQL query's parts to the clauses of RDF `SELECT` statement(s).

First of all, the `select` clause of the RDF query will be filled by `<params>` parts of the VQL query. The parameters set to `"1"` will be put as variables of the `select` clause, and unshown parameters (set to `"0"`) are used for forming the criteria only. Additionally, some extra variables will be added in the clause to get the message's URI and the name of data sources. Secondly, the `from` clause will be generated automatically by VQL parser. For the time being, all data sources are stored in one huge metastore with a unique network address. And this network address plugged access protocol will be placed in `from` clause. Thirdly, generated triple expressions will be used for the `where` clause. As discussed, the process of generating the triple expression combines all three parts of the VQL query - `params`, `sources`, and `relations` - along a further analysis by adding more expressions to clarify the criteria.

Last but not least, we have two optional attributes, `order` and `exclude`, for each parameter. If the `order` is set to `"1"`, then the variable will be added into `orderby` clause. And if an `exclude` is specified, an expression in form of `exclude($param $op $exclstr)` will be added into the `where` clause.

5.3 Semantic Mapping

The semantic mapping takes part in both mapping tasks above to resolve semantic ambiguity problems. This is the vital and decisive mapping task of the VQL. The semantic mapping is going to solve the following concerns during query generating process from the user's initial VQL query:

- Disambiguating query items
- Resolving semantic conflicts

Disambiguating query items. The inaccuracy of a query is mainly due to the ambiguities inside itself. Coping with ambiguous items in the VQL query made by a user is a decisive step in parsing it later on. Here the ambiguity could be in terminological manner, i.e. requested data properties are not clear. For example, a `"Name"` property in a query is ambiguous because the query parser can not identify which "name" will be extracted, contact name or name of email sender/receiver.

Clarifying these properties could be done by using data source specified in the query and the ontologies of the system. Based on the data sources, the appreciate properties in the ontology will be detected and used instead of ambiguous items. For instance, concerning the `"Name"` property is described above, this mapping

task must rely on the related source described either in source or constraints elements, e.g. `Contact`; after on, based on ontologies, appreciate properties will be located such as `contactFirstName`, `contactLastName`.

Resolving semantic conflicts. In a further disambiguation users' queries, especially when the user asks for information using an ambiguous entity over many data sources. The issue is that how to identify user's "intended" properties for which data sources. For example, `"Name"` property is constrained with `Email` and `Contact`. If the `"Name"` properties is constrained with a source identified in a `<relation>` element, then the issue is similar to the discussion above. Otherwise, the query has to:

- get all related properties in system ontology based on specified data sources. In this case, there are probably more than one query generated; or
- suggest the most related properties from ontology based on a *semantic similarity* of properties in the same query. For instance, if other properties are major requesting for `Contact` items, so that the "names" of `Contact` object would be suggested.

The semantic mapping is applied for all types of the VQL query. Generally, all user-entered queries should be check for implied semantic problems as well as the syntax of them.

6 Conclusion and Future Work

In this paper we have presented the Virtual Query Language, a design of a query language aiming at a significant complexity reduction in formulating semantic meaningful queries.

As the next steps of VQL, the issues of improving by considering RDF as the alternative format for VQL, and building a graphical interface on top of this language will be discussed consequently in the details. We see the advantages of the RDF as standard for the Semantic Web applications, so that coding the VQL in RDF as 'RDF forms' would be considered. As well as, a powerful interface for VQL in context of the VQS will be taken into account.

References

1. M. Ahmed, H. H. Hoang, S. Karim, S. Khusro, M. Lanzenberger, K. Latif, E. Michlmayr, K. Mustofa, T. H. Nguyen, A. Rauber, A. Schatten, T. M. Nguyen, and A. M. Tjoa. Semanticlife - a framework for managing information of a human lifetime. In *Proceedings of the 6th International Conference on Information Integration and Web-based Applications and Services*, September 2004.
2. N. Athanasis, V. Christophides, and D. Kotzinos. Generating on the fly queries for the semantic web: The ics-forth graphical rql interface (grql). In *Proceedings of the 3rd International Semantic Web Conference*, pages 486–501, 2004.

3. D. Beckett. Sparql query results xml format. Technical report, W3C Working Draft, August 2005.
4. T. Catarci, P. Dongilli, T. D. Mascio, E. Franconi, G. Santucci, and S. Tessaris. An ontology based visual tool for query formulation support. In *Proceedings of the 16th European Conference on Artificial Intelligence*, pages 308–312, 2004.
5. E. Garcia and M.-A. Sicilia. User interface tactics in ontology-based information seeking. *PsychNology Journal*, 1(3):242–255, 2003.
6. P. Gray, K. Hui, and A. Preece. An expressive constraint language for semantic web applications. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, 2001.
7. H. H. Hoang, A. M. Tjoa, and T. M. Nguyen. Ontology-based virtual query system for the semanticlife digital memory project. In *Proceedings of the 4th International Conference on Computer Sciences*, February 2006.
8. G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, and M. Scholl. Rql - a declarative query language for rdf. In *Proceedings of the Eleventh International World Wide Web Conference*, pages 592–603. ACM Press, August 2002.
9. E. Prud'hommeaux and A. Seaborne. Sparql query language for rdf. W3C Working Draft, November 2005.
10. R. M. R. Guha. Tap: a semantic web platform. *International Journal on Computer and Telecommunications Networking*, 42(5):557–577, August 2003.
11. A. Seaborne. Rdql - a query language for rdf. Member submission, W3C, 2004.
12. D. Wood, P. Gearon, and T. Adams. Kowari: A platform for semantic web storage and analysis. In *Proceedings of the 14th International WWW Conference*, 2005.

An Efficient Implementation of a Rule-based Adaptive Web Information System

Davide Valeriano, Roberto De Virgilio,
Riccardo Torlone, and Daniele Di Federico

Dipartimento di Informatica e Automazione
Università degli studi Roma Tre
{valeriano,devirgilio,torlone,difederico}@dia.uniroma3.it

Abstract. Mobile devices provide a variety of ways to access information resources available on the Web and a high level of adaptability to different aspects (e.g., device capabilities, network QoS, user preferences, location, an so on) is strongly required in this scenario. In this paper we propose a technique for the efficient adaptation of Web Information Systems. The approach relies on special *rules* that allows us to specify, in a declarative way, how to generate, in an automatic way, the adaptation specifications that satisfy the requirements of a given context. Rules are classified in a set of clusters and a matching relation between clusters and context requirements is defined. The technique of rule evaluation and clustering guarantee that different contexts and orthogonal requirements of adaptation, possibly not fixed in advance, can be efficiently organized and taken into account in the adaptation process.

1 Introduction

Modern Web Information Systems (WIS) are challenged to generate (automatically) a hypermedia presentation built from information that is gathered from different, possibly heterogeneous, sources that are distributed over the Web. Nowadays, given the spread of mobile devices, a novel and fundamental requirement of these systems is the ability to adapt and personalize content delivery according to the *context* of the client. An important point that needs to be taken into account is that this scenario is very dynamic: some aspects of the context can dynamically change. For instance new users with new preferences can be enabled to access the information system, possibly unpredicted types of access devices can be used, alternative network connections can be made available, further aspects of the context (like the location or others environmental factors) need to be incorporated. Also, the technology used to implement the adaptation can be changed and this should have a limited impact on the overall application. It follows that the adaptation process should guarantee a high level of flexibility in terms of responsiveness to rapidly changing requirements of adaptation and suitable actions to be undertaken to meet these requirements.

In the literature, some adaptation approaches rely on the definition of rules to generate suitable adaptation specifications [2, 3, 10]. In [5] we have presented

a general methodology for context adaptation, based on special *rules* that model the adaptation at an abstract level, inspired by the approaches cited above. The rules specify, in a declarative way, how to generate, in an automatic way, the adaptation specifications that satisfy the requirements of a given context. A relevant problem is to rapidly select and activate the rules that best fit the requirements of adaptation of the context (in particular when the context changes frequently). In this paper we present an efficient implementation to optimize the activation process of adaptation rules. To this aim, we introduce a clustering technique to classify the adaptation specifications.

The rest of the paper is organized as follows. Section 2 introduces the rule-based approach. In Section 3, we illustrate how to define the set of clusters and the process to efficiently activate a rule, using the clusters. In Section 4 we present a practical implementation and experimental results and finally, in Section 5 we draw some conclusions and sketch future works.

2 A Rule-based approach to model adaptation requirements

In this section we present special rules to specify, in a declarative way, how to generate, in an automatic way, adaptation specifications. The rules are based on two basic notions: the generic profile and the abstract configuration.

2.1 Generic profiles

A (*generic*) *profile* is a description of an autonomous aspect of the context in which the Web site is accessed (and that should influence the presentation of its contents). Examples of profiles are the user, the device, the location, and so on. A *dimension* is property that characterizes a profile. Each dimension can be conveniently described by means of a set of *attributes*. Attributes can be *simple* or *composite*. A *simple attribute* has a value associated with it, whereas a *complex attribute* has a set of (simple or complex) attributes associated with it. In this model, a context is a collection of profiles. Figure 1 reports a graphical example of profiles. In our model, different profiles over the same dimensions can

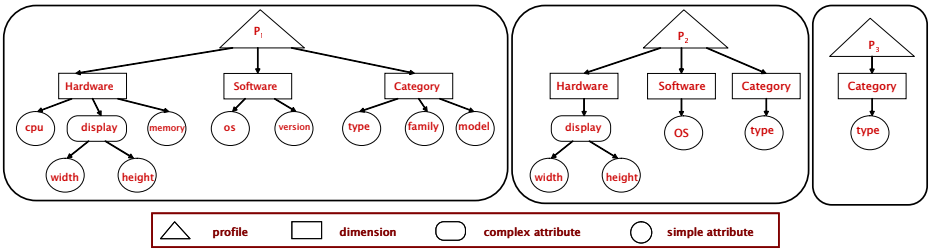


Fig. 1. An example of profiles

be compared making use of a subsumption relationship $\lhd$. Intuitively, given two profiles P_1 and P_2, if $P_1 \lhd P_2$ then P_2 is more detailed than P_1 since it includes the attributes of P_2 at the same or at coarser level of detail. More precisely, we first say that an attribute A *covers* another attribute B if either they are simple and $A = B$ or they are composite and for each sub-attribute A_i of A there is a sub-attribute B_j of B such that A_i covers B_j . The subsumption relationship is then defined as follows.

Definition 1 (Subsumption) *Given two profiles P_1 and P_2, we say that P_1 is subsumed by P_2, $P_1 \lhd P_2$, if for each dimension d of P_1 there is a dimension d' of P_2 such that for each attribute A of d there is an attribute A' of d' covered by A.*

As an example, given the profiles reported in Figure 1 we have that $P_2 \lhd P_1$ and $P_3 \lhd P_2$.

2.2 Abstract configurations

We introduce the notion of (abstract) configuration as a triple *C = (q, h, s)* where:

- q is a query over the underlying database expressed in relational calculus, a declarative and abstract language for relational databases ([1]);
- h is an abstract hypertext definition expressed in WebML ([4]), a conceptual model for Web application which allows us to describe the organization of Web pages in a tight and concise way, by abstracting their logical features;
- s is presentation specification expressed in terms of an original notion of logical style sheet, which is a set of tuples over a predefined collection of Web Object Types (WOTs) like text, image, video, form and so on; each WOT τ is associated with a set of presentation attributes: they identify possible styles (e.g. font, color, spacing, position) that can be specified for τ.

An example of configuration is reported below.

$q = \{T : x_1, S : x_2, D : x_3, C : x_4, N : x_5 \mid$
$\mathit{NewsItems}(N : x_0, T : x_1, S : x_2, D : x_3, G : x_6, J : x_7),$
$\mathit{Details}(NK : x_0, P : x_8, C : x_4), \mathit{Newspapers}(J : x_7, N : x_5, C : x_9), x_6 = \mathit{Sport}\}$

```
h =  IndexUnit NewsIndex
       ( source News Items; attributes Title, Date; orderby Date )
     DataUnit ContentData
       ( selector Item = CurrentItem; attributes Title, Date, Content)
     link NewsToDetails
       ( from NewsIndex to ContentData
         parameters CurrentItem = NK)
     link ContentToRestOfContent
       ( from ContentData to ContentData
         parameters CurrentItem = NK)
s =  Text( Font: Arial, ..., Border: 0pt)
     Link( Note: False, ..., Color: Blue)
     Image( Format: jpeg, ..., Alignment: left)
```

It is important to note that a configuration is indeed a logical notion that can be represented and implemented in several ways and with different syntaxes. This property guarantees the generality of the approach with respect to actual languages and tools used to implement the adaptive application. For instance, we can implement a configuration using SQL at the content level, XHTML at the navigation level and a set of CSS files at the presentation level.

2.3 Adaptation Rule

The matching relationship between configurations and profiles is represented by means of a novel notion of *adaptation rule*. An adaptation rule has the form $P_r : C_d \Rightarrow C_f$ where:

- P_r is a parametric profile, that is, a profile in which parameters can appear in place of values,
- C_d is a condition, made out of a conjunction of atoms of the form $A\theta B$ or $A\theta c$, where A and B are parameters occurring in P_r, c is a constant value, and θ is a comparison operator,
- C_f is a parametric configuration in which parameters occurring in P_r can appear in place of values.

The intuitive semantics of an adaptation rule is the following: if the client has a profile P_r and the condition C_d is verified then generate the configuration C_f.

An example of adaptation rule for a device profile with the hardware dimension H is the following:

$$H(\mathit{ImgCapable} : X, \mathit{ScreenSize} : Y, \mathit{ColorCapable} : Z) :$$
$$X = \mathit{false}, Y < 1500 \Rightarrow \{q, h, s\}$$

where X, Y, and Z are the parameters of H and $\{q, h, s\}$ is the following parametric configuration:

$$\begin{aligned} q = \; & \{ T : x_1, S : x_2, D : x_3, C : x_4 \mid \\ & \mathit{NewsItems}(N : x_0, T : x_1, S : x_2, D : x_3, G : x_5, \\ & J : x_6), \mathit{Details}(NK : x_0, P : x_7, C : x_4)\} \end{aligned}$$

```
h =  Page NewsPage ( units NewsIndex )
     Page ContentPage ( units ContentData )
     IndexUnit NewsIndex
       ( source News Items; attributes Title, Date;
         orderby Date )
     DataUnit ContentData
       ( selector Item = CurrentItem;
         attributes Title, Date, Content)
     link NewsToDetails
       ( from NewsIndex to ContentData
         parameters CurrentItem = NK)
s =  Text( Color: Black, Size: 8pt)
     Link( Style: Underline, Color: Black)
     Image( Size: Y × 0,5, Color: Z)
```

Note the use of the parameters Y and Z of H to resize the images and to eliminate/maintain colors.

A profile P *activates* a rule $P_r : C_d \rightarrow C_f$ if $P_r \lhd P$. Hence, a profile P can activate a rule for a profile that is more general than P. Now, let R be a rule $P_r : C_d \rightarrow C_f$ activated by a profile P and let C be the configuration obtained from C_f by substituting the parameters of P_r with the actual values occurring in P: we say that C *is generated* by P using R.

3 An optimization strategy

We assume to have at disposal an initial set of rules $\mathbf{S_R}$. In an adaptability scenario, context changes generate events that produce one or more profiles (instances). Each profile (instance) can activate a rule of $\mathbf{S_R}$. We should optimize the research to select and activate efficiently the most appropriate rules and generate the most suitable configuration. So we organize S_R classifying the rules in a set of clusters. Each cluster represents a *class* of adaptation that matches the requirements of a *class* of context.

3.1 A distance between profiles

We define a distance that allows to compare profiles. We take advantage of the tree structure of the generic profile. We get inspiration from the *Tree Edit Distance* [9], where the distance between two trees T_1 and T_2, that we indicate as $d_{ted}(T_1, T_2)$, depend on the sequence of operations (chosen in a limited predefined set) that *transforms* T_1 into T_2. We assign a fixed cost to each operation: the distance between the considered trees is the sum of essential costs for the transformation. For our case, let's consider the set of operations *edit*, *remove* and *add*, respectively to modify the content of a node, to delete a node and to create a new node. In our tree structures, we distinguish two principal types of node: dimension and attribute. We have operations to modify, remove or create dimensions and attributes and we indicate the cost of an operation with γ.

Definition 2 (S-derivation) *Given two trees T_1 and T_2, $S = s_1, \ldots, s_k$ the sequence of operations to transform T_1 in T_2, an S-derivation between T_1 and T_2 is a sequence $U_0, \ldots, U_k$ of trees resulting by applying the single operation, where $U_0 = T_1$ and $U_k = T_2$, and the cost $\gamma(S - derivation)$ is $\sum_{i=0}^{i=k} \gamma(s_i)$.*

So we can intuitively define the distance between two profiles P_1 and P_2, the minimum cost for a S-derivation between them.

Definition 3 (Distance between profiles) *Given two profiles P_1 and P_2, we define the distance between them as $d_{ted}(P_1, P_2) = min\{\gamma(S - derivation) \mid S\text{-}derivation\ from\ P_1\ to\ P_2\}$*

3.2 Definition of Clusters

Let's represent a cluster as a tree where the nodes are profiles. The *root* represents the minimum set of information that nodes of the tree share. Basically a client is identified by the profile sent to the system. Let's remember that a profile is principally described by dimensions, that characterize the main information, articulated in attributes. So, the root of a cluster is a profile characterized only by dimensions. It will contain the minimum information that several profiles has to include to be in the same cluster. Given a profile P, pruned by all attributes, we can build several beginner clusters, rooted with profiles generated from P, on the number and type of dimensions. However the designer can decide the detail level for the roots. For instance a designer can decide to articulate some dimensions of a root with attributes, to extend the level of detail of the minimum information to share.

We define a cluster as follows

Definition 4 (Cluster Tree) *A cluster CL is a couple* $\{N_{CL}, E_{CL}\}$ *where* N_{CL} *is the set of nodes (profiles) and* E_{CL} *is the set of edges* (n_i, n_j) *such that* $n_i \lhd n_j$

We can relate a node P_r of a cluster with the rule $P_r : C_d \rightarrow C_f$ in an Hash table using P_r as access key.

Given the set of roots $R_t = \{R_{t_1}, \ldots, R_{t_n}\}$ and the set of rules S_R, we initialize a set of clusters $CL = \{CL_1, \ldots, CL_n\}$ as follows:

- we set $CL_i = \{\{R_{t_i}\}, \emptyset\}$,
- $\forall\ (P_r : C_d \rightarrow C_f) \in S_R$, (i) we search a CL_i such that $R_{t_i} \lhd P_r$, and we navigate the cluster in depth until level k where doesn't exist any node subsumed by P_r, (ii) we search at level $k-1$ the node Nd such that $Nd \lhd P_r$ and if there are more nodes Nd such that $Nd \lhd P_r$ we choose the node with lowest distance, (iii) we add P_r to N_{CL_i} and (Nd, P_r) to E_{CL_i}, (iv) if there are one or more nodes Nd_i such that $P_r \lhd Nd_i$ then we delete any edge of type (Nd_j, Nd_i) in E_{CL_i} and add to it the edge (P_r, Nd_i),
- for each node of a cluster we add the relative rule to the Hash table.

3.3 The activation process

Built the set of clusters, now we can perform an optimization strategy to activate a rule in an efficient way. Given the set of rules S_R, the set of clusters $CL = \{CL_1, CL_2, ..., CL_n\}$, the Hash table H_t and a profile instance P_I of a profile schema P_S, we produce a set of configurations as follows:

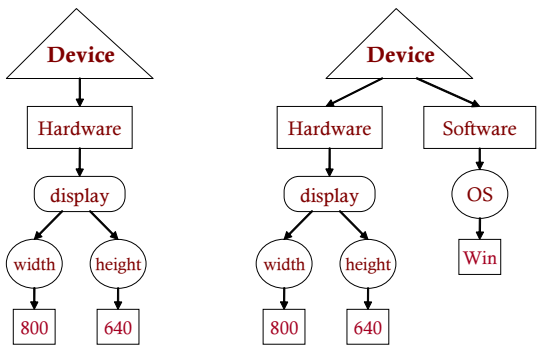
1. we initialize P' with P_S;
2. we search the cluster CL_i in CL, with a root R_{t_i} such that $R_{t_i} \lhd P'$ and that has the lowest distance from it;
3. (i) we navigate in depth the cluster until level k where any node, that is subsumed by P', doesn't exist; (ii) we search at level $k-1$ the node Nd such that $Nd \lhd P'$ and if more nodes are subsumed by P' we choose the one with

lowest distance; (iii) we extract the rule R_l, related in the H_t to Nd, and if the condition is satisfied, we activate R_l and instance the configuration with the value of P_I; (iv) we update P' as P' - Nd;

4. we iterate from step (2.) until P' is not empty and a rule that can be activated by P' exists;
5. if no rule can be activated in this way, the system will return a default configuration.

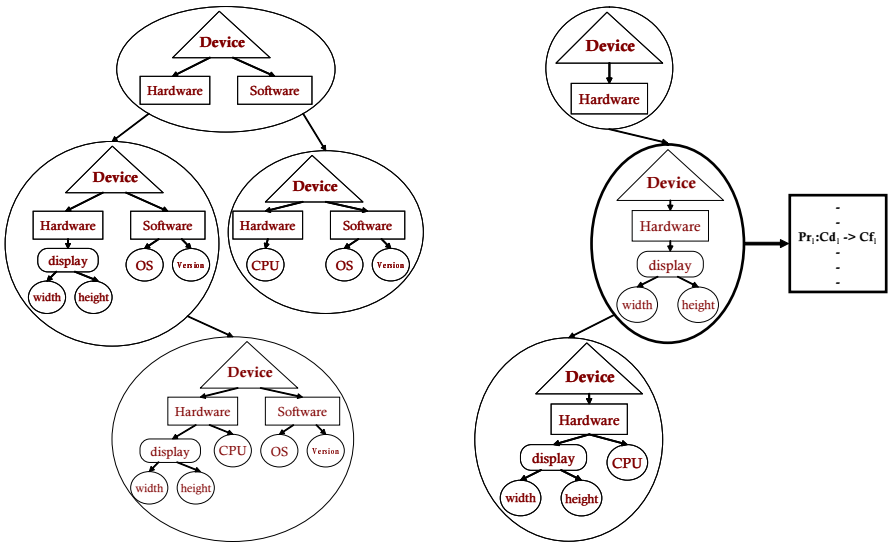
3.4 An example of interaction between profiles and clusters

Let's assume a client sending a profile P_I (see Figure. 2(a)), instance of a profile schema P to the system. Assume also that the system has already initialized its own set of clusters, composed by clusters C_1 and C_2 (see Figure 3). We easily see that the root of the cluster C_2 is the one which subsumes P and has the lowest distance from it. So we navigate that cluster and note that the only child N of the root it is equal to P; then we extract the rule R_l related in the H_t to N and if the condition is satisfied, we activate it and instance the configuration with the values of P_I. Subsequently, assume that a new profile P'_I (see Figure 2(b)), instance of a profile P', is sent by another client to the system. At the same way explained before, we see that the root of the cluster that subsumes P' and has the lowest distance from it, is cluster C_1 root. Navigating that cluster, we need to confront P' with root's two children. We easily see that no one of them is equal to or has a subsumption relation with P': being their parent a root, is not possible to activate its rule; so, as we explained before, the system will return a default configuration.



(a) Profile P_I (b) Profile P'_I

Fig. 2. Profiles sent to the system.



(a) Cluster C_1 (b) Cluster C_2

Fig. 3. Cluster Set

4 A practical implementation

4.1 Benefits of Clustering

We can easily see that using this clustering algorithm we can reduce the number of accesses to S_R. In our system, we have seen that a profile given by a client can activate a rule in the default set. So when a client sends a profile to the system it has to search for the most suitable rule in S_R. We can assume that the worst case is when the system has the greatest possible number of default rules, that is when the system has in S_R one rule for each possible profile that a client can send. Without using clusters we should execute a number of ordered accesses equal to the total number of possible profiles. It is easy to see that the number of possible profiles is the number of profiles obtained with every possible different combination of dimensions and attributes. Assuming to have a complete schema with n dimensions and m attributes for each dimension, we can see that the number of different combinations of dimensions is $\sum_{i=1}^{n} \binom{n}{i}$ while the number of different combinations of attributes for each single dimension is $\sum_{j=1}^{m} \binom{m}{j}$. Combining these two results and looking carefully at our structure, we see that the number of different profiles is equal to $\sum_{i=1}^{n} \left[\sum_{j=1}^{m} \binom{m}{j}\right]^{\binom{n}{i-1}} (n-i+1)$. If we use the clustering algorithm, in this case we will obtain a number of clusters equal to the number of different combinations of dimensions, seen above. To access the right clusters, the system has to access to all of their roots and then find the right way on the selected tree. So it can ignore a number of trees equal to $\sum_{i=1}^{n} \left[\binom{n}{i}\right] + 1$. We can see that in this situation the worst case is when the root of the selected

cluster has the maximum number n of dimensions, because it would be the tree with the maximum number of nodes. The remaining clusters will contain a number of profiles equal to $\sum_{i=1}^{n-1} \left[\sum_{j=1}^{m} \binom{m}{j}\right]^{\binom{n}{i-1}} (n-i+1)$. These profiles will be ignored by the system, which has only accessed the roots of their clusters, that are not profiles. So, counting also these accesses, we find that *at least* we obtain to save a number of accesses equal to $\sum_{i=1}^{n-1} \left[\sum_{j=1}^{m} \binom{m}{j}\right]^{\binom{n}{i-1}} (n-i+1) - \sum_{i=1}^{n} \binom{n}{i}$. This number doesn't take count of the fact that the system won't access all the nodes of the selected cluster, but only some of them, depending on its structure. Anyway it gives evidence that, using this clustering algorithm, it is possible to obtain a certain benefit in terms of number of accesses.

4.2 Experimental results

We have designed and developed a practical implementation to test the our approach. The system relies on an RDF database of contents and the selection of data is done by performing RDF queries expressed in SPARQL [8]. We use an RDF(S) representation for schema level and RDF for instance level. The implementation makes use of Jena [6], a Java framework for building RDF based applications. Each rule is implemented in terms of Inference Rule of Jena [7]. Using the Inference Engine of Jena, the system generates a configuration that infers the adaptation on RDF(S) documents. Finally the system returns several response as output, using XSLT transformations on RDF instances. We have tested our system to experiment the effectiveness and the efficiency of the approach illustrated in this paper in several practical cases. On the server side, we used an IBM computer xSeries 225, equipped with a Xeon2 2.8Ghz processor, a 4 GB RAM, and a 120 GB HDD SCSI. The Web site have been accessed by three different types of devices: a mid-range desktop, several PDAs with different capabilities, and some cellular phones. Figure 4 reports the average response time (vertical axis) for each device type (horizontal axis). The Figure shows promising times to produce the adapted response. The Desktop client presents an average time of 72 ms, the PDA client 97 ms and the Wap client 121 ms. The diagram compares the medium response time in absence and in presence of clusters. In particular the system, using clusters, capitalizes the approach for mobile devices, almost halving the response time. For instance the Wap client benefits from 121 ms to 59 ms.

5 Conclusions and future works

In this paper we have presented a novel rule-based approach supporting the automatic adaptation of content delivery in Web Information Systems. This problem arises in common scenarios where the information system is accessed, in different contexts, by a variety of mobile devices. The adaptation is achieved automatically by means of special rules that specify, in a declarative way, how a configuration can be generated to meet the requirements of adaptation of a

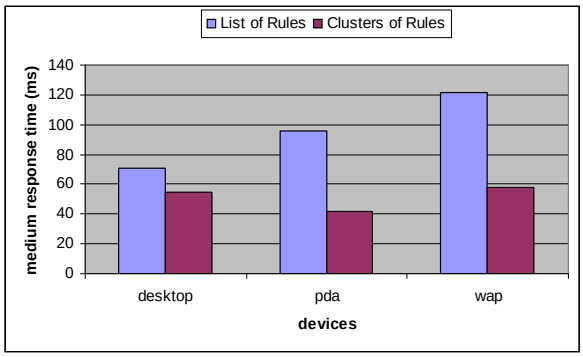


Fig. 4. Results of experiments

given profile. Finally the organization of rules in a set of clusters guarantees the responsiveness of the system in a dynamic scenario. Different contexts and orthogonal requirements of adaptation, possibly not fixed in advance, can be taken into account in this process. The results presented in this paper are subject of further conceptual and practical investigation. From a conceptual point of view, we are currently investigating the expressive power of rules and the generality of clusters. In particular we are improving the distance between profiles, using a comparison between common pathes. From a practical point of view we are improving the efficiency of the system and we are extending its functionality.

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases.* Addison-Wesley, 1995.
2. C. Bettini, D. Maggiorini, and D. Riboni. Distributed context monitoring for continuous mobile services. In *IFIP TC8 Working Conference on Mobile Information Systems (MOBIS)*, 2005.
3. S. Ceri, F. Daniel, V. Demaldé, and F. M. Facca. An approach to user-behavior-aware web applications. In *5th International Conference on Web Engineering (ICWE)*, pages 417–428, 2005.
4. S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai, and M. Matera. *Designing Data-Intensive Web Applications.* Morgan Kaufmann Publishers Inc., 2002.
5. R. DeVirgilio, R. Torlone, and G.-J. Houben. A rule-based approach to content delivery adaptation in web information systems. In *7th International Conference on Mobile Data Management (MDM'06) - to appear* -, 2006.
6. HP-Labs. Jena. http://jena.sourceforge.net/, 2004.
7. HP-Labs. Jena 2 inference support. http://jena.sourceforge.net/inference/, 2005.
8. Hp-Labs. Sparql, query language for rdf. www.w3.org/TR/rdf-sparql-query/, 2005.
9. C. Isert. The editing distance between trees. Technical report, Ferienakademie, for course 2: Bume: Algorithmik Und Kombinatorik, 1999.
10. H. Kiyomitsu, A. Takeuchi, and K. Tanaka. Activeweb: Xml-based active rules for web view derivations and access control. In *International workshop on Information technology for virtual enterprises (ITVE01)*, pages 31–39, 2001.

Modeling User Behaviour Aware WebSites with PRML

Irene Garrigós and Jaime Gómez

Universidad de Alicante, IWAD, Campus de San Vicente del Raspeig, Apartado 99
03080 Alicante, Spain
{igarrigos, jgomez}@dlsi.ua.es

Abstract. Adaptive websites usually change as effect of user navigational actions. Most current web engineering approaches (which consider personalization) allow to detect basic user browsing behaviour (e.g. user click on a link) but don't consider the definition of (complex) behaviour events or user behaviour pattern recognition. In this paper, we use a method independent language called PRML to specify behaviour aware websites. PRML evolved out the experience of OO-H and was designed to be a generic personalization specification method that can be reused for different web design approaches. PRML allows the personalization when a complex behaviour event is triggered (i.e. a sequence of links) and also allows the definition and recognition (at runtime) of user behaviour patterns.

1. Introduction

In the last decade, the number and complexity of websites and the amount of information they offer is rapidly growing. We face a new, wide spectrum of web applications that leads to new challenging requirements like the need for continuous evolution. Moreover websites typically serve large and heterogeneous audience what can lead to maintenance and usability problems [11].

Introduction of web design methods and methodologies [5] [6] [8] [9] [10] [12] have provided some solutions for designers (design support, help in determining consistent structuring, easier maintenance) and for visitors (better tailored content, easier navigation). In order to better tailor the site to one particular user, or a group of users, some methods provide personalization support. Usually web applications are adapted as effect of user browsing behaviour. Most current web design methods limit the detection of browsing behaviour to simple user navigational actions (i.e. start of a session, activation of a link…). Sometimes this leads to too simple personalization strategies definition.

To tackle the problems described above we provide a method-independent personalization specification language called PRML (Personalization Rule Modelling Language) [7] to build a (reusable) Personalization Model. This language allows explicit modelling of personalization policies and their effective deployment and reuse for different web design methods. PRML considers four

adaptivity dimensions: *the user characteristics* (e.g. user's age, user's hobbies, user's vision level…), *the user context* which includes different types of context like *device context* (e.g. PDA, PC, WAP), *time context* (i.e. date and local time of the connection), *network context* (e.g. latency, speed, bandwidth and *location context* (i.e. ubiquity of the user), the third dimension are *the user requirements* (specifying the needs and goals of the user) and the fourth is *the user browsing behaviour*.

If we focus on the *browsing behaviour* dimension we can see that the definition of complex navigational actions (i.e. sequence of link activation) is supported. Moreover it supports the definition and detection of user behaviour patterns. The consequence is that the complexity of the personalization strategies to define (based on user browsing behaviour) increases. The detection of user navigational patterns (defined for a very common user behaviour) at runtime implies some problems like how to track the user behaviour and how to recognize certain behaviour pattern. This is explained in section 4 of the paper.

Most web design methods [2] [5] [10] [14] and adaptive reference models have support for adaptivity considering (only) basic user browsing behaviour actions. The adaptive web engineering approach UWE [10] has some basic support for personalization based on user behaviour. The AHAM [14] reference model, which is mainly used for teaching applications, describes behaviour being tracked and used to update the user model. WSDM [5] supports adaptivity (i.e. not for a specific user) tracking the user browsing behaviour. WebML [6] is an exception supporting the definition of a personalization strategy based on complex behaviour user actions. It uses the WBM formalism based on a timed-state transition automaton to define the behaviour user actions that should be performed to trigger a personalization action. However, none of these approaches consider the detection (at runtime) of behaviour patterns.

The paper is structured as follows. In the next section, a case study defined in the context of the OO-H method is presented. The paper continues describing in section 3 the PRML fundamentals. How PRML is extended to support (complex) browsing behaviour and to detec behaviour patterns is presented in section 4. We continue in section 5, explaining the OO-H execution architecture supporting PRML rules. Finally, section 6 sketches some conclusions and further work.

2. Case Study: Blog of a University Lab

We are going to explain our approach in the context of the OO-H (Object Oriented Hypermedia) method [8] in which the PRML language was born. This is a user-driven approach based on the object oriented paradigm and partially based on standards (UML, OCL, XML…). OO-H allows the development of web-based interfaces and their connection with pre-existing application modules. In OO-H four views are defined: *the domain view¸ the navigational view, the presentation view* and *the personalization view.* Each of these views has associated a model (or

set of models). We are going to show the OO-H views by means of a case study excepting the presentation view that is not considered in this work.

The system to model is an online blog of a university lab in which (registered) users can basically post and consult messages (classified by categories) and set comments on them. In the left part of Figure 1 we can see the *domain model* (from the OO-H domain view) of the system. The right part of the Figure 1 shows the *user model* which will be explained further.

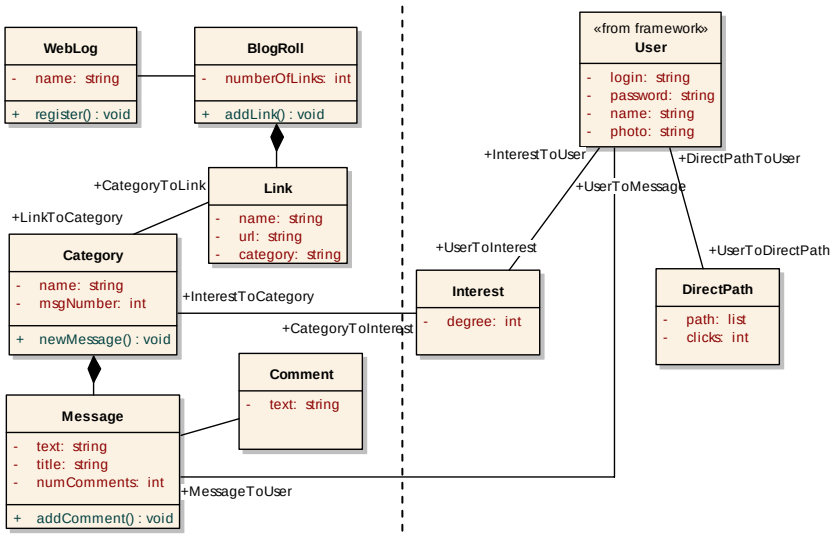


Figure 1. OO-H Domain and User Model

In the OO-H navigational view the navigation structure is defined by means of the *Navigation Abstract Diagram (NAD*). This diagram enriches the domain view with navigation and interaction features. In figure 2 we can see the (zero level of the) *navigational model* of the online blog system. The starting point (*entry point* element) is the home page where the user has to register. Registered users access to a page which has a collection of links (represented by an inverted triangle) in which there is a set of links to other related blogs (blogroll), there is also a set of categories and a set of messages. The links we can see in Figure 2 (ViewBlogRoll, ViewCategories and CheckMessages) are automatic links represented by a discontinuous line (i.e. the user does not need to click on them) and they show the information in origin (we can see that from the white arrow head). In Figure 2 we cannot see the whole NAD (for simplification reasons), we have two navigational targets which group modeling elements that collaborate to solve some functional requirements. A navigational target (NT) is represented by a UML package symbol. For a more extensive description of the NAD diagram see [8].

In Figure 3 we can see the details of the categories navigational target. In the menu page the user will find a set of categories. This set is an indexed list, so the user can click in one of the categories names to view the messages attached to it (the *Message* class). Moreover the messages can have attached comments. There are

also two method invocation links that allow to *add a new comment* and *add a new message* to a certain category.

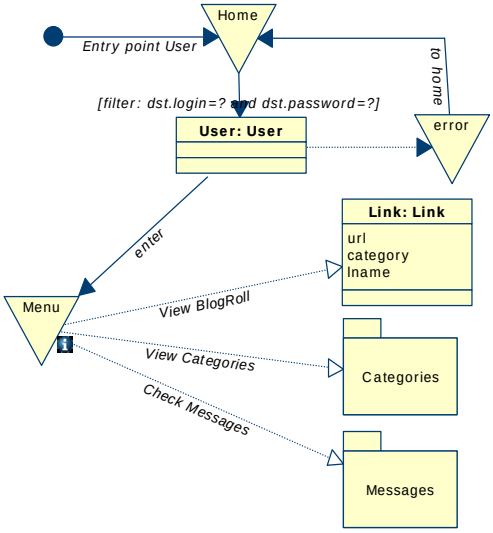


Figure 2. (Level 0 of the) NAD for the blog case study

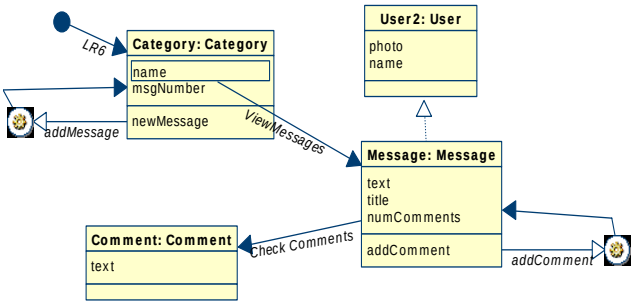


Figure 3. Categories NT

Due to space limitations, the details of the messages navigational target cannot be presented. It shows the whole set of messages with their corresponding data.
Finally, the *personalization view* is supported by a personalization framework that is part of the OO-H approach. This framework can be instantiated by the web designer and connected to any OO-H based website to which provide personalization support. It is divided in two parts: the *User model* and the *Personalization model.* They are explained next.

2.1 Personalization Support

The *user model* (UM) specifies the data (and its structure) needed for personalization (including user data). While a UM is conceptually a separate model, is used in a similar way as a *domain model* (DM): it is typically associated

with DM and can be represented as part of the DM. The UM of the system can be seen in the right part of Figure 1. To build this model we have to take into account the personalization requirements of the system to determine the (updatable) information needed to personalize. In the case study we consider the following (personalization) requirement:

- *Users will see the categories (for messages) sorted by (the user) interest on them.*

To cover this requirement in the updatable data space defined by the UM of the example we store the user interest degree on a category. For this purpose, in the UM we have the *Interest* class. To determine the user interest we use several strategies as we will see in section 4.

To store and update the information in the UM we need some mechanism. Also to specify the personalization strategies based on that information. For this purpose we need to define the *personalization model.* As we already explained, it is specified by a (method-independent) specification language which was born in the context of the OO-H approach. This language is called PRML (Personalization Rules Modeling Language) and its fundamentals are explained in next section.

3. PRML Fundamentals

PRML [7] is a modelling language for personalization that evolved out of the experience from the OO-H approach [6] [8] and was designed to be a generic personalization specification method. In PRML two conformance levels[1] are defined depending on the richness of the personalization specifications supported:

- In the first one of these levels, called basic (or zero) level, the (personalization) actions that can be specified have been defined by abstracting the basic operations supported by known web design methods [5] [6] [9] [10] [12]. The purpose of its definition is being able to specify a (universal) reusable personalization policy. This level is a subset of the advanced level.
- The second of the PRML conformance levels, called advanced level, comprise all the operations supported by PRML allowing the specification of richer personalization policies than the basic level (limiting though the reusability).

Moreover, the definition of these two levels allows to (easily) compare the level of personalization supported by the different methodologies because we have a unique way of specifying it (for all the methods). A methodology can have full or partial conformance with one or two of the levels, allowing then a better comparison.

Rules that can be defined using this language are ECA [4] rules. This decision is natural and conforms to initial requirements on a personalization model, where actions are taken as reactions on concrete events caused by users. The effectiveness of the concept of ECA rules for user-adaptation is proven by different personalization frameworks and tools (e.g. AHA! [14]). The PRML has its own language defining conditions and actions. An important part of the language is a

[1] This idea has been inspired by the SQL conformance levels.

mechanism specifying access to data attributes by means of path expressions that can be exported to different query languages for different data modeling techniques (relational, object-oriented, or semantic web modeling languages). This decision targets the requirement of independence on concrete web and data modeling methods. This is an important design requirement because it is obvious that the language should not contain any constructs dependent on concrete web or data modeling methods for being generic.

For satisfying a personalization requirement we often have to define how to update the knowledge about user (acquisition rule). This information is stored in the UM data space. Furthermore, we have to define the effects this personalization causes to the presented content and navigation structure by means of personalization rules (we do not consider presentation features in this work). These rules use the information specified by the UM to describe adaptation actions. A PRML rule is formed by an event and the body of the rule containing a condition (optional) and an action to be performed. The basic structure of a rule defined with this language is the following:

When *event* **do**
    If *condition* **then** *action* **endIf**
endWhen

When an event is triggered if the condition (optional) is satisfied an action is performed. PRML considers the following event types to trigger a rule:

- *Navigation event:* It is caused by the activation of a navigational link. Links have information about an instance or a set of instances they are associated with and are going to be shown in the activated node (i.e. the page resulting from the navigation). When the link is activated the rule/s attached to this event is/are triggered passing to the rule/s this information as a parameter[2] for personalizing the (adaptive) active node based on this parameter. We need the parameter value before the node resulting from navigation is generated (in order to use this value in the rules and build the adaptive web page). The parameter passed to the rule can be a simple parameter (when the data on the webpage is the instance of a concept of the DM) expressed in PRML as follows: "**When** *Navigation.activelink(NM.activenode parametername)* **do"** or a complex parameter (when the data of the webpage is a set of instances of a concept of the DM) expressed in PRML as: "**When** *Navigation.activelink(NM.activenode* parametername)* **do"**.
- *LoadElement event:* It is associated with the instantiation of a node. The difference with the *navigation event* is that its activation is independent of the link that caused (useful when a set of links have the same target node and we want to personalize during the activation of any of them). In the case of loading a node a parameter is passed containing the information of the root concept (class) of the node loaded. In the case of loading a collection of links the parameter passed would be the set of links.

[2] All the parameters have the prefix NM to indicate that they come from the Navigation Model.

- *Start event* It is associated with the entrance of the user in the website (i.e. the start of the browsing session). In PRML this event is expressed as "**When** *SessionStart* **do**".
- *End event:* It is associated with the exit of the user from the website (i.e. expiration of the browsing session). In PRML this event is expressed as "**When** *SessionEnd* **do**".

The actions we can perform depend on the PRML conformance level supported by the methodology. OO-H supports (fully) both PRML levels. We are not going into detail into the operations supported because we focus here on the behaviour definition. In next section we explain how we have extended PRML for modelling (complex) behaviour aware websites.

4. Extending PRML for Modeling Behaviour Aware Websites

PRML already supports personalization on basis of (simple) user's behaviour (i.e. user's click on a link), however, as already mentioned, we think is important of being able to support the detection of more complex actions of the user. We explain how to do so in subsection 4.1. Moreover PRML supports the definition and (runtime) recognition of (predefined) navigational patterns, being able to personalize in basis of this information. When doing so, we have to face some problems like how do we track the user behaviour and how do we recognize certain behaviour pattern. We are going to explain this subsection 4.2.

5.1 Support for Complex Browsing Behaviour

To support not so trivial browsing behaviour of the user (i.e. user clicks on a link) we need to consider more complex events, like a sequence of clicks on several links in a specific order, etc. For this purpose we have extended PRML events adding the following composite events[3]:

- *NavigationSet event:* It is caused by the activation of a concrete set of navigational links. We can express that all the events should be triggered using the operator "," or express that at least one of the specified events has to be triggered using the "||" operator. We can also state that the events should be triggered in the order they are specified using the "&" operator. In PRML is expressed as: **"When** NavigationSet*[link1(NM.activenode par1),…linkn(NM.activenode parn)]* **do"**. If we want to consider that a certain link has to be visited a concrete number of times and/or a certain number of seconds we can specify it besides the link ID in this way: "**When** NavigationSet[*link1(NM.activenode par1,numClicks,numSec), … linkn(NM.activenode parn, numClicks,numSec)]* **do"**.

[3] Note that the parameters of each of the events are the same as described in section 3

- *LoadSet event:* It is associated with the instantiation of a set of nodes. In PRML is expressed as: "**When** LoadSet*[node1(NM.activenode par1), …noden(NM.activenode parn)]* **do"**. To specify the user has to be browsing a node a certain number of seconds for the event to be triggered and/or the number of times the node has been loaded by the user we express it in PRML as follows: "**When** LoadSet *[node1(NM.activenode par1,numLoads,numSec),… noden(NM.activenode parn,numLoads,numSec)]* **do"**. We use the same set of operators than in the *NavigationSet* event (",", "&", "||") to express order between events and if it is mandatory that all the events should be triggered or not.

Moreover combinations of these two types of events (and also combinations with simple events) can be done in the same rule using the operators "," to express that all the events should be triggered, "&" to express all the events should be triggered in a specific order and "||" to express at least one of the specified events has to be triggered (e.g. "**When** (LoadSet[node1*(NM.activenode par1)* &node2*(NM.activenode par2)*] || NavigationSet[link1*(NM.activenode par1)* ||link2*(NM.activenode par2)*])**do**).

Simple events defined in previous section are also extended in this way:

- *Navigation event*, we can specify the number of seconds the user was browsing in the node resulting from the navigation, and the number of times the link has been activated, in the same way as the *NavigationSet* event.
- *LoadElement event*, we can specify the number of times the node (element) has been loaded by the user and the number of seconds that the user was browsing on it.

We are going to show the definition of (complex) user behaviour actions by means of examples in the case study. For this purpose, let's consider now the (already specified) personalization requirement:

- *Users will see the categories (for messages) sorted by (the user) interest on them. Sort the categories by the ones in which the user has most interest on.*

Two things are required here, first is to detect which are the categories in which the user is most interested on, and second would be sorting the categories by this value.

To detect the interest of the user in the categories we need to track the user behaviour and it is needed to define when we consider that the user has interest on a certain category. This can be defined in several ways, we show here three possible alternatives:

1. *The user has interest on a category when s/he has checked one message of that category.*

In this case, we are going to update the interest degree on a category when the user accesses it. To store/update the value of the *Interest.degree* attribute from the UM we need the following acquisition rule:

```
When Navigation.ViewMessages(NM.Category cat) do
   Foreach i in (UM.User.uToInterest) do
```

```
        If(cat.name=i.ItoCategory.name) then
                SetContent(i.degree,i.degree+10)
        endIf
    endForeach
endWhen
```

This rule is triggered when the user activates the link *ViewMessage*s (see Figure 3). It updates the degree of interest in the category of the consulted message using the *SetContent* statement, in which we specify the attribute to be modified, and the value or formula that calculates the new value. This rule compares each of the instances of the class interest with the consulted instance (to properly update the value). As explained before, in some cases we need to access the data from the navigation prior to trigger the event for using it in rules. For this purpose the rule event carries a parameter containing the visited instance of the Category class (from the NM). This parameter has the prefix "NM". The rest of the data of this rule have the "UM" prefix, indicating that information is stored in the UM data space.

2. *The user has interest on a category when s/he has checked at least 3 messages of that category and has been navigating at least 1 minute on each of the messages pages.*

In this case, we are going to update the interest degree on a category when the user accesses 3 of the messages attached to that category and stays at least 1 minute checking each of the three messages. In this case to store/update the value of the *Interest.degree* attribute from the UM we need the following acquisition rule:

```
When NavigationSet[ViewMessages(NM.Category cat, 3,60)]do
    Foreach i in (UM.User.uToInterest) do
        If(cat.name=i.ItoCategory.name) then
                SetContent(i.degree,i.degree+10)
        endIf
    endForeach
endWhen
```

This rule is triggered when the user activates the link *ViewMessage*s three times and stays 60 seconds at least in each of the message visited. It updates the degree of interest in the category of the consulted message using the *SetContent* statement, in the same way as the previous rule.

3. *The user has interest on a category when s/he has checked at least 3 messages of that category and has been navigating at least 1 minute on that page or if the user posted a comment on one of these messages.*

A new alternative is added for triggering the acquisition rule to update the interest degree on a category: the rule will be also triggered when the user adds a comment on a message visited. In this case we need the following acquisition rule:

```
When NavigationSet[ViewMessages(NM.Category cat ,3,60) ||
AddComment(NM.Message msg)] do
    Foreach i in (UM.User.uToInterest) do
        If(cat.name=i.ItoCategory.name) then
```

```
                SetContent(i.degree,i.degree+10)
        endIf
   endForeach
endWhen
```

The event of this rule gets more complex, this rule is triggered when the user activates the link *ViewMessage*s three times and navigates over the messages at least 60 seconds or when the user clicks on the *AddComment* link operation to add a new comment on a visited message. Once it is triggered it updates the degree of interest in the category of the consulted message in the same way as previous rules.

Once the needed information is stored (*user interest degree*) we need to sort the categories (basing it on the UM data). For this purpose we need a personalization rule:

```
When LoadElement.Category(NM.Category* categories) do
        SortLinks categories orderBy ASC UM.User.uToInterest.degree
endWhen
```

The rule is triggered when the node *Category* is loaded (through any link). It sorts the categories ordered by their user interest (stored in the previously presented rule) in an ascending way. This rule has no condition so when the node is loaded, categories titles are sorted. Note that here we do not specify a loop for sorting the books since *SortLinks* runs over all the instances of the categories set passed as a parameter.

4.2 Support for Behaviour Patterns

Different behavioural patterns are possible in the browsing behaviour of the visitors. In order to better accommodate the users, we can analyze their behavioural patterns, and adapt the site accordingly if we recognize a certain pattern (over a significant amount of time). The definition of such patterns makes easier the tracking of complex user browsing behaviour.
To support behaviour patterns definition and recognition in PRML we add a new type of rules, called *behavioural rules.* These rules track the user browsing behaviour and detect (at runtime) navigational patterns (defined also with behavioural rules). Moreover when a pattern is detected the proper action is performed. The main difference with the rest of PRML rules (i.e. acquisition and personalization rules) is the parameters passed in the events. In this kind of rules we pass (as a parameter) the navigational path that the user is following. We need to define (for each defined pattern) what we consider a navigational path. We show next an example for a browsing behaviour patterns defined in [3].

Direct path pattern:

Intent: provide direct navigation access to information relevant for the particular user, instead of forcing the user to follow each time the same set of navigation tracks.

Solution: adapt the navigation access point of the user if a direct path from that access point to a certain (other) node is detected in a certain

percentage of session (for example, 80%) or in a significant number of times. In that case, a link to the relevant information is added.
Consequence: reduces amount of clicks to relevant information (for the particular user)

When a user enters in the system we start tracking his/her behaviour, keeping the navigational paths that s/he follows. We (only) consider a (finished) navigational path (for this browsing pattern) a path that starts in the page where the user entered the application and finishes in the page in which s/he stays at least 2 minutes. We can see a representation of the navigational path for this pattern in the statechart of figure 4.

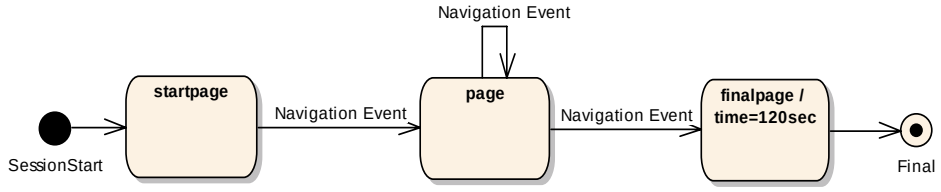


Figure 4: Navigational Path for the Direct path pattern

The navigational paths are stored in the UM, each of them having an identifier. A behavioural pattern can be defined as a certain sequence of navigation actions. We define the following PRML rules to detect the direct path pattern:

```
When SessionStart do
SetContent(UM.UserToDirectPath.path, startpage)
endWhen
```

This rule is triggered when the user enters the website and it initializes the navigational path of the user with the starting page (i.e. the page in which the user starts the session on the website). Next we need to build this navigational path meanwhile the user is browsing the web application. For this purpose we have the following *behavioural* rule:

```
When Navigation.Link(NM.CurrentPath path) do
Foreach a in (UM.UserToDirectPath) do
If (a.path=path) then
        SetContent(a.path, a.path &link)
endIf
endWhen
```

This behavioural rule is triggered when the user activates any link (represented by the *Link* id). As a parameter (as already stated) we have the current path of the user in the website. The path stored in the user model is updated, adding the new browsed link. What is left now is to detect when the path has been “finished”. As aforementioned we consider a navigational path being finished when the user browses a page for at least 2 minutes. We express this with the following *behavioural* rule:

```
When Navigation.Link(NM.CurrentPath path ,1,120) do
```

```
Foreach a in (UM.UserToDirectPath) do
        If (a.path=path) then
                SetContent(a.path, a.path &link &finished)
                SetContent(a.clicks, a.clicks+1)
        endIf
endForeach
endWhen
```

This behavioural rule is also triggered by any link activation (represented by the *Link* ID). In this case we add to the path (list) variable of the user model the last link visited and the keyword *finished*, knowing it is an ended path. We need now a rule to update the clicks on the finished path when the user goes through it:

```
When Navigation.Link(NM.CurrentPath path) do
Foreach a in (UM.UserToDirectPath) do
        If (a.path=path and last(a.path)=finished) then
                SetContent(a.clicks, a.clicks+1)
        endIf
endForeach
endWhen
```

This rule checks if the current visited path has been already detected as a finished one (checking the last element of the list in which the path is stored) and if so the number of clicks on it is increased.
What is left now is to check if the direct path pattern is detected and perform the proper action. We consider that if a user has more than 100 clicks on a navigational path the direct path is detected, and what we do is to add in the home page a shortcut to that path.

```
When SessionStart do
Foreach a in (UM.UserToDirectPath) do
        If (a.clicks>100) then
                Add(shortcut(a.path),homepage)
        endIf
endForeach
endWhen
```

5 Execution Architecture

In OO-H we have created a prototype software application called OPWAC (OO-H Personalized Web Applications Creator) with which we give credit to what we have seen in the previous sections. This tool generates from the OO-H models (the four previously described views), the final (personalized) web pages. Moreover it allows evaluating and performing the set of PRML rules attached to the OO-H models to personalize the final web sites. In figure 5 we show the architecture that follows the OPWAC prototype.

Our main goal has been to generate (personalized) web applications from OO-H based models. We have a web server that interacts with the user, gathering the requests and giving back the response. This website engine generates (on demand) the web pages dynamically from the models (represented in XMI[13] format),

capturing the events produced by the user actions; these events are sent to the PRML evaluator module responsible of evaluating and performing the personalization rules attached to the events that cause the modification of the OO-H diagrams. After these modifications the web engine properly generates the new pages.

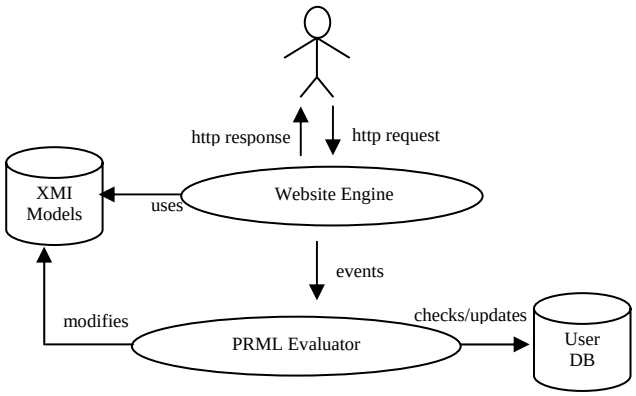


Figure 5. OPWAC architecture

Next we describe in detail the technologies used for implementing this architecture.

5.1 Website Engine

The Website Engine, as we previously said, generates the final (personalized) web pages from the OO-H models. These models are the input of our tool and are represented by means of XML elements (in XMI [13] format). The reason for choosing an XMI representation of the models is that this format can be easily generated from UML models[4]. The only problem is that the OO-H diagram is not UML compliant, but it can be represented by the UML modeling language. To transform the OO-H navigation diagram (NAD) to UML we have extended the syntax and semantics of the UML 2.0 modelling language to express the specific concepts of the NAD, thanks to the UML profiles. To define these profiles UML 2.0 uses stereotypes, which would represent the NAD elements that we want to include in the profile.

To read and process the OO-H models (specified in XMI) for the generation of the final web pages we have used the .NET technology. This technology provides us with the DOM class (XML Document Object Model), with which we can represent in memory the XML documents.

5.2 PRML Evaluator

This module is the responsible for analyzing the events that the Website engine sends for the execution of the personalization rules that will perform modifications

[4] Most UML tools allow this transformation

in the OO-H models. In the NAD components we have added information (thanks to the UML profile created) about the PRML rules to execute (which are stored in a separated file). When a rule is triggered, to evaluate the rule conditions and perform the proper actions we have implemented a .NET component using the ANTLR Parser generator [1]. From the PRML grammar we can generate the syntactic trees which help us to evaluate the rule conditions and perform them if necessary. Finally to execute the actions of the rules, we have implemented in C# the different actions types that we can find in PRML. These actions will modify the XMI models to generate the new web pages from them.

6 Conclusions and future work

In this paper we have presented an approach to model behaviour aware web applications. We use a method independent language called PRML (Personalization Rules Modeling Language) to specify behaviour aware websites. PRML evolved out the experience of OO-H and was designed to be a generic personalization specification method that can be reused for different web design approaches. PRML has been extended to support personalization when a complex behaviour event is triggered (i.e. a sequence of links) and also allows the definition and recognition (at runtime) of user behaviour patterns. We can define such a pattern once and reuse it for its recognition in several websites.

References

1. ANTLR, *ANother Tool for Language Recognition*, http://www.antlr.org/
2. Casteleyn, S., De Troyer, O., Brockmans, S.: *Design Time Support for Adaptive Behaviour in Web Sites*, In Proc. of the 18th ACM Symposium on Applied Computing, Melbourne, USA (2003), pp. 1222 – 1228.
3. Casteleyn, S., Garrigós, I., Plessers, P.: *Pattern Definition to Refine Navigation Structure in Hypermedia/Web Applications* , In Proceedings of the IADIS International Conference WWW/Internet 2004 (ICWI2004), Volume II, pp. 1199-1203, Eds. Isaias, P., Karmakar, N., Publ. IADIS PRESS, ISBN 972-99353-0-0, Madrid, Spain (2004)
4. Dayal U.: *Active Database Management Systems*, In Proc. 3rd Int. Conf on Data and Knowledge Bases, pp 150–169, 1988.
5. De Troyer, O., Casteleyn, S.: *Designing Localized Web Sites*, In Proceedings of the WISE 2004 Conference pp. 547 - 558. Springer-Verlag, Brisbane, Australia (2004)
6. Facca F. M., Ceri S., Armani J. and Demaldé V., *Building Reactive Web Applications*. Poster at WWW2005, Chiba, Japan (2005).
7. Garrigós I., Gómez J., Barna P., Houben G.J.: *A Reusable Personalization Model in Web Application Design.* International Workshop on Web Information Systems Modeling (WISM 2005) July 2005 Sydney, Australia.
8. Gómez, J., Cachero, C., and Pastor, O.: *Conceptual Modelling of Device-Independent Web Applications*, IEEE Multimedia Special Issue on WE (2001) pp 26-39.

9. Houben, G.J., Frasincar, F., Barna, P, and Vdovjak, R.: *Modeling User Input and Hypermedia Dynamics in Hera* International Conference on Web Engineering (ICWE 2004), LNCS 3140, Springer-Verlag, Munich(2004) pp 60-73.

10. Hubert Baumeister, Alexander Knapp, Nora Koch and Gefei Zhang. *Modelling Adaptivity with Aspects*. International Conference on Web Engineering (ICWE 2005), Sydney, Australia, LNCS 3579, Springer Verlag, 406-416, July 2005.

11. Nielsen, J.:*Finding usability problems through heuristic evaluation*.. In Proc. of the SIGCHI Conf on Human factors in computing systems.Monterey, California, United States pp: 373 – 380. 1992.

12. Rossi, G., Schwabe, D. and Guimaraes M. *Designing Personalized Web Applications*. In Proc. of the World Wide Web Conference (WWW'10). Hong Kong May 2001.

13. XML Metadata Interchange www.omg.org/technology/documents/formal/xmi.htm

14. Wu, H., Houben, G.J., De Bra, P.:*AHAM: A Reference Model to Support Adaptive Hypermedia Authoring*, In Proc. of the "Zesde Interdisciplinaire Conferentie Informatiewetenschap", pp. 77-88, Antwerp, 1998.

A Platform for Managing Term Dictionaries for Utilizing Distributed Interview Archives

Kenro Aihara and Atsuhiro Takasu

National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
{kenro.aihara,takasu}@nii.ac.jp

Abstract. This paper proposes a platform that aims to support the whole process and facilitate archiving tasks at museums and galleries.
When we try to preserve tacit knowledge or skills of artists or masters by interview and utilize interview videos, it is important to support the whole process of archiving; capturing interview, digitizing it to a machine readable format, authorizing metadata, and exporting them to be accessed from others.
To interoperate distributed archives, rich meta information is required. In this paper, we focus our tools to facilitate authoring metadata of interview, such as transcription and annotation task for interview, with speech processing or natural language processing functions. The important point now is that the performance of the speech processing or text processing for specific domain depends on well-fitted language models to it. We, therefore, design our tools as an networked software package that can download up-to-date term dictionary and language models for speech processing or text processing from a server of the platform. Users can utilize tools with updated models. Meanwhile, created metadata including terms occurred in interview can be uploaded to the server and they will be used for update of the models hereafter.
In this paper, we first overview our project. Then, we propose an platform and our tools to be distributed.

1 Introduction

For the field of arts and crafts, domain knowledge or individual knowledge, such as tacit knowledge or skills of artists or masters, are not inherited for lack of successors or effectual methods of communications to the next generation. We, therefore, must consider how to support patrimony of such knowledge or skill of human creativity.

We suppose that text is not enough because skill and knowledge are sometimes beyond description and we need more excellent form for recording them. Therefore interviews with masters and artists are effective way to record their skill and knowledge instead of textbook type description. We think that interview videos can keep not only master's words but also his/her gestures, atmosphere, savvies, and context of them. We have been constructing an interview video

archive for preserving such knowledge of masters in the field of lacquer arts and crafts for this purpose.

We must consider to integrate such interview archives efficiently when archives are created individually and its number gets grown. This paper focuses this issue and proposes our approach for improve information integration by sharing fundamental resources.

In this paper, we propose a platform for management of term dictionary for utilizing distributed interview archives. Section 2 describes the background of this research. In Section 3 , our project called MONO is overviewed. Then, Section 4 describes our platform for managing term dictionary. Conclusions are given in Section 5.

2 Background

2.1 Sharing Skill and Knowledge of Artistic Creativity

Sharing skill and knowledge of master workmen and artists is one of important and challenging issues. Usually disciples acquire the skill and knowledge by oral communication with masters and watching the master's works. They, therefore, can be conveyed to limited number of disciples and they are sometimes lost when masters and artists pass away. A networked mechanism for sharing skill and knowledge must be needed to preserve them and convey it to a large amount of people.

Since skill and knowledge inherent in masters and artists, the first step to construct such a mechanism is to externalize them and represent them in an appropriate form. We want to emphasize that this means that this kind of knowledge sharing needs not only an efficient file sharing mechanism but also support of digitization to externalize them.

Text is not appropriate because skill and knowledge are sometimes beyond description and we need more excellent form for recording them. Interviews with masters and artists are effective way to record their skill and knowledge instead. It can record various kinds of information such as emotional behavior, procedure of creative activity as well as textual information in the form of conversation. Interview has another advantage that it enables to obtain the information from masters and artists quickly without heavy mental load. It will take a year or more for masters to write a book of their skill and knowledge. However it takes only several hours to have interviews. The latter advantage of interview is very effective to solve the bottle-neck problem of information capturing not only for the skill and knowledge of masters and artists but also for externalizing the knowledge of human beings in many fields.

2.2 Language Models for Processing

When we try to integrate distributed resources, we must consider two aspects: data type and domain.

In recent competitive solutions for multimedia retrieval, textual metadata or annotation is attached first to each non-textual data and then an index for image retrieval is made[3]. This suggests that creating appropriate textual metadata corresponding multimedia data is critical. Once textual metadata is attached, multimedia contents can be handled as the same as text.

In text retrieval, feature of data is usually defined with frequency of term occurrence, represented by Salton's TF-IDF weighting[5]. We must consider that controlled term sets are necessary if we try to improve efficiency of such weighting. In other words, sharing domain specific terms and its maintenance is critical to integrate distributedly generated contents. Construction of domain ontology can be regarded as one of approaches for this purpose. However, we cannot utilize such effective and flexible ontologies for now.

On the other hand, recent researches on both natural language processing (NLP) and speech recognition get advanced in corpus-based approach, which applies language models learned from data collection.

It is important to note that efficient processing for text and multimedia data depends on shared terms and language models corresponding each domain. Especially, they are necessary if the domain is highly technical and there are relatively small amount of digital contents.

We, therefore, have been developing a platform for sharing fundamental resources of specific domain, such as technical terms.

3 Overview of MONO project

In this section, we describe our project called MONO[1] to archive skill and knowledge of artists [1]. In this project, we adopted interview videos as the main form of information representation. To the best knowledge of authors, only a few digital libraries handle interview videos such as a VHF's archive of interviews to survivors of the Holocaust [4]. The MONO project will clarify new technical challenges and derive a new framework for constructing archives.

3.1 Purpose

The purpose of MONO project is to create a platform for archiving skill and knowledge of artists in the field of the arts and crafts and providing it for students in this field as well as visitors to museum. The goals of this project are:

- Constructing an archive of skill and knowledge of masters and artists to preserve the knowledge otherwise it may be lost with masters in near future.
- Developing information processing technologies for archiving, e.g., editing interview videos and attaching metadata to them.
- Developing an interview archive utilization methods such as efficient interview video search and effective browsing methods.

[1] General meaning of MONO is an object in Japanese, but this word also means an valuable object created by an excellent workman or artist.

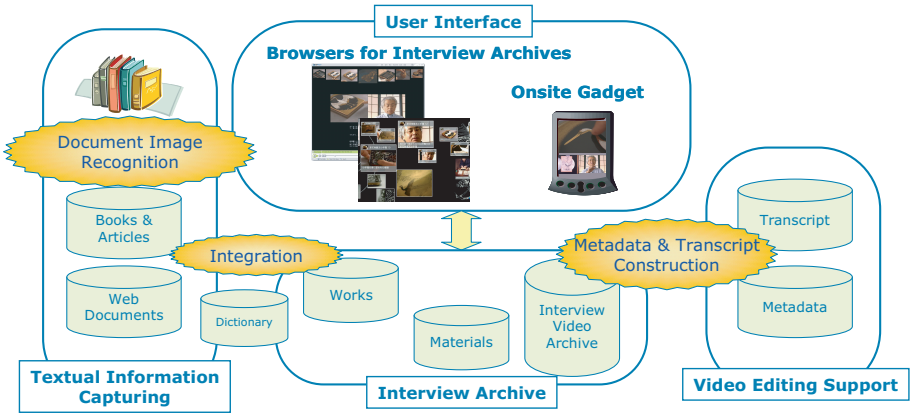


Fig. 1. An Overview of MONO project

– Constructing a test bed for multimedia processing technology such as speech recognition, video processing and information extraction.

For the test bed construction, we are attaching metadata such as transcripts for interview videos, scene change of videos, topic change of the interviews for evaluation.

3.2 Architecture

A system of the MONO project consists of data collection and three modules as shown in Fig. 1.

Collection The system contains various kinds of information. They are categorized into two groups: interview videos and complementary textual information. Interview videos is the main part of the collection. In the interview, masters and artists explain their creative works using their artifacts. Currently the collection contains 21 interviews of artists and masters in the field of lacquer arts and crafts. In this project, we have been recording some interviews with two cameras. One is for recording interviewee's face and the other is for his/her hands or works which he/she is handling. The format of interview video is AVI with the resolution of 720 × 480 frame, 29.97 frames per second. Total length of interviews is 2,088 minutes. We continue interviews and plan to collect 30 interviews within this year.

Although an interview is a very effective way to record the skill and knowledge, it is sometimes too specific for users who do not have sufficient knowledge in this field. In order to help users to comprehend this field and understand the interview contents well, the collection contains complementary textual information such as dictionary, articles, books and catalogs of exhibitions.

Fig. 2. Snapshot Images of "Lacquer Arts and Crafts" Collection

Functional Modules Current system has three functional modules: a video editing support module (VESM), a textual information capturing module (TICM) and a user interface module (UIM). VESM and TICM provide archiving functions and UIM provides an information utilization function.

Usually the interview contains several topics. In editing process, interview videos are segmented into sections and each section is further segmented into scenes. On the other hand, conversation in the interview video is converted to transcript. Video segmentation and transcription generation are basically done by hand. This part is most labor intensive in archive construction. Cost reduction of this process is a key to construct large archive. Currently VESM supports this process in two ways: scene change detection and transcript generation. There are several techniques for detecting scene change in videos. These techniques usually use the change of color frequency distribution between frames. However, in interview videos, the color frequency distribution does not always change drastically even if the scene changes. VESM detects and provides the candidate scene changes for a video editor based on the speech signals. Second, VESM has speech recognition ability. It recognizes the conversation in interview videos. Since the recognition accuracy is not high enough to use the resultant transcript as the final one, editors need to correct the recognition error. As for the speech recognition, the goal of this project is to establish a video search technique without manually produced transcript.

As described above, our collection contains complementary textual information as well as interview videos. TICM supports to capture these textual information. Currently TICM has the following functions. Books and articles sometimes need digitization from printed paper. TICM has OCR and document image analysis submodule [7] to digitize them. In order to support dictionary

edition, TICM has a natural language processing module to extract important words from documents based on a statistical measure [2]. Dictionary editors can select entry words from these candidates in dictionary edition. We are now developing a collaborative dictionary editing environment where dictionary editors collaboratively write the meaning of the entry words and set links to related words.

UIM provides effective access to the archive.

Technical Challenges Interview video archive is a kind of video archive and it has same technical problems as general video archives. In this subsection, we focus on the technical problems specific to interview videos.

Metadata Construction Metadata is very important to utilize videos. Our system has two level structure of meta data for interview video: *scenes* and *sections*. A section is a logically minimum unit of video corresponding to one verbal phrase. And a scene is a continuous parts of video. This metadata structure is similar to general video archives. However, in the case of interview video, we can control the interview in the section level in some extent. Currently we are designing a standard interview manual. It will support to make section level metadata. In VHF's interview archive [4], pre-interview questionary is carried out. Resultant documents are attached to interviews and used in interview retrieval. In our system, the manual is combined to interview video tightly and it will be used to retrieve interview videos at the section level. The point of metadata construction for interview video is pre-interview story configuration which derives homogeneous video archive.

As mentioned above, in the interview video, it is difficult to extract scenes based on color distribution of frames because the main object of the interview is a master or an artist, and this feature tends not to change during shooting. Currently our system detects the scene changes based on the speech signals. However, in order to reduce the editing cost we need a more accurate method. In order to improve the accuracy of scene change detection, we have been developing an authoring tool to cut scenes detailed later and also are planning to develop an equipment with which interviewer can record marks at scene change during the interview.

Speech Processing and Recognition In the interview videos, the conversation is the dominant information sources. Therefore, speech processing technology is much more important than video processing technology. Accuracy of current speech recognition is not sufficient enough, and recognized textual information is too erroneous. Therefore we need a robust technique to utilize conversation in the form of speech signals.

Information Integration In interview archives, complementary textual information is especially important to provide comprehensive view of the archive for users because the information in the interview is usually very specific. Therefore,

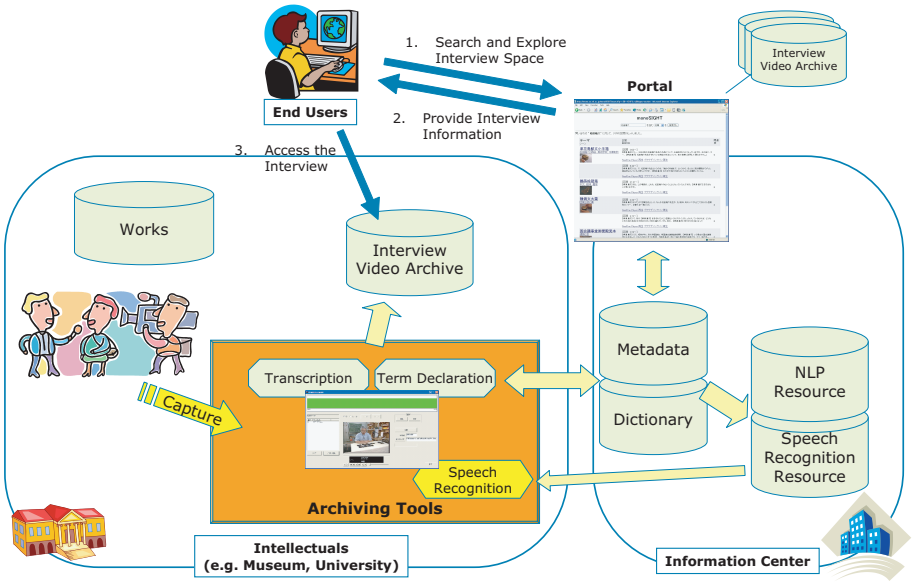


Fig. 3. Overview of the Proposed Platform

related textual information such as articles, books, exhibition catalog and web pages should be associated with interview video. From the technical viewpoint, transmedia information integration technology is a key to solve this problem. In MONO project, we first construct a dictionary, then associate related textual information with scenes and sections of interview videos using the dictionary as a base of information integration.

As mentioned in Section 3.2, this paper focuses a collaborative dictionary edition in the MONO project.

4 Proposed System

4.1 Overview

Our methodology aims not only at construction of portal site but also at supporting capture of digital contents transformed from interview videos with intellectuals. We assume that a reciprocal relation between portal and local repositories must be needed for successful portal construction.

Fig. 3 illustrates an overview of our proposed platform for construction of portal service and local repositories.

We provide a support tool package for archives for local repository sites, such as museum. The package includes manuals for interview, software to edit videos and to create metadata that includes transcripts of interview, and server program for online service.

The orange box in Fig. 3 indicates the software to support transcription and editing term dictionary. While terms in transcripts can be tagged manually in metadata creation phase by users, such as curator at museum, terms and description about them also should be added into local term dictionary.

In addition, the package also includes speech recognition module. Unfortunately performance of current speech recognition tools depends on the quality of language models and dictionaries. Our portal of "Information Center" in the figure,therefore, provides language models for speech recognition which are incrementally constructed with collected metadata and term dictionaries from local repositories. Users can download up-to-date data for speech recognition and apply it to their videos when they transcribe them.

The portal site can aggregate high quality metadata and language resource from each local repository. Aggregated language resources can be used not only for speech recognition but also natural language processing or information retrieval.

At first, a user has to register him/herself at the portal site and then download the archiving package.

After creation of an archive, the local site can provide its own video service. Typically, videos and metadata are integrated as SMIL contents. SMIL enables us to describe the temporal behavior of a multimedia presentation, associate hyperlinks with media objects and describe the layout of the presentation on a screen.[2] Fig. 4 illustrates a snapshot image of browsing interview video as SMIL content. The window of the SMIL browser contains multimedia objects as follows:

- the scene being played in the middle of the window
- caption under the current scene, which scrolls up synchronously
- sequence of thumbnails of scenes in chronological order on the top, while the current scene is located in the middle of the sequence

End users can be navigated to an appropriate archive when they search for contents at the portal.

We think that the sequence of scenes can help you grasp the context of the interview intuitively. It is obvious that scrolling caption is helpful to understand more correctively and easily than without it. Words in interviews often includes technical terms.

4.2 Archiving

Metadata Structure For interview videos, we define the whole interview as *commentary.* One commentary can includes some comments about one or more works or topics. One commentary consists of some *themes.* Theme is a time region of the interview and usually corresponds to one work or topic. We, therefore,

[2] SMIL, the Synchronized Multimedia Integration Language, is a standard language for simple authoring of interactive audiovisual presentations. W3C published SMIL 2.0 Recommendation in August 2001. http://www.w3.org/TR/smil20/

Fig. 4. Accessing Interview Video with an SMIL browser

regard a theme as a comment of each work. One theme consists of some *scenes* which are continuous parts of recorded video. And also one theme is segmented into *sections* by verbal gap or topical change. Section, therefore, is an atomic unit of contents here. Fig. 5 illustrates that one theme consists of one or more scenes of multiple cameras.

Utterance of interviews is transcribed and used as caption. This transcribed text is also used for indexing each section.

Archiving Interview Fig. 6 shows process flow of archiving.

At first, a user of this annotation tool captures and digitize interview videos, and the tool shows possible points of scene change. The user has to fix the points interactively.

Next, the user needs to segment each scene into sections. Viewing candidates of section according to verbal gap, the user fixes sections.

Then, the tool runs speech recognition function with language models provided at the portal to show a candidate of transcript for each section. While the user edits transcripts, he/she also declares terms by tagging and its description. The tool highlights terms which are already registered in the current dictionary to facilitate the term declaration task.

After the authoring process, the tools produces an interview video archive including videos and metadata and also uploads metadata with local term dic-

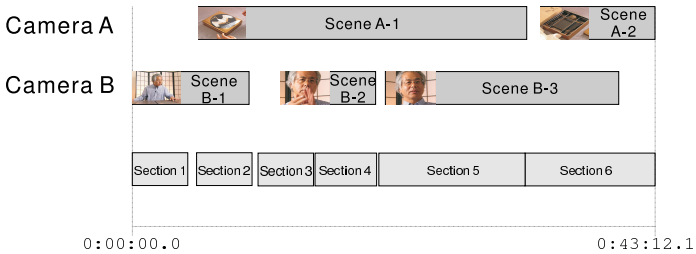


Fig. 5. Theme

tionary to the server of the portal. The gray parts in Fig. 6 indicates shared resources.

4.3 Shared Language Models

In order to support dictionary edition, we first have to create an initial dictionary. Books and articles sometimes need digitization from printed paper. TICM of Fig. 1 has OCR and document image analysis submodule [7] to digitize them. TICM has a natural language processing module to extract important words from documents based on a statistical measure [2]. Dictionary editors can select entry words from initial candidates in dictionary edition.

The models can be updated when local term dictionaries are uploaded.

Metadata including transcripts and term dictionary can be used as a fundamental resource of its own field; lacquer arts and crafts. It may be exploited in natural language processing of search engines of the portal.

5 Conclusion

This paper overviews our project called MONO which aims to reveal a mechanism for sharing knowledge or skill of artists or masters. In particular, we focus maintenance of shared term dictionary of the proposed platform in this paper because sharing fundamental resources for natural language processing and speech recognition is critical for integration of distributed contents.

Distribution of prototype tools and feasibility studies are future issues.

Acknowledgments

This study is partly supported by Grand-in-Aid Scientific Research on Priority Area “Informatics” (Area #006).

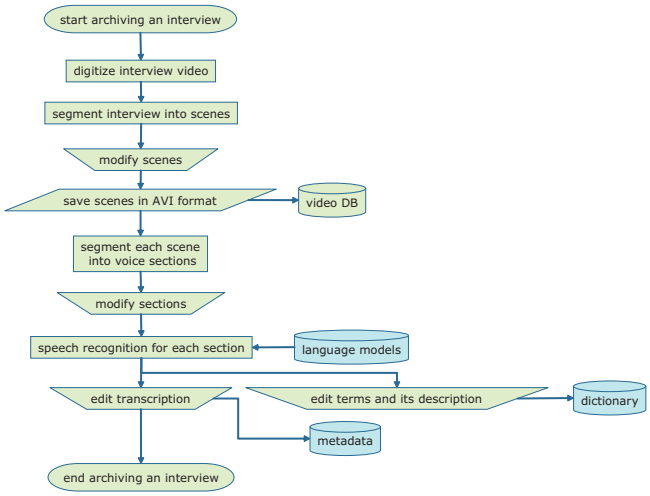


Fig. 6. Flowchart of Archiving

References

1. K. Aihara and A. Takasu. A reciprocal platform for archiving interview video about arts and crafts. In *Joint Conference on Digital Libraries*, page 363, June 2005.
2. A. Aizawa. An information-theoretic perspective of TF-IDF measures. *Information Processing and Management*, 39(1):45–65, 2003.
3. P. Clough, M. Sanderson, and H. Müller. The CLEF cross language image retrieval track (ImageCLEF) 2004. In *Working Notes for the CLEF 2004 Workshop*, 2004.
4. S. Gustman, S. Soergel, D. W. Oard, and W. J. Byrne. Supporting access to large digital oral history archives. In *Joint Conference on Digital Libraries*, pages 18–27, 2002.
5. G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1998.
6. S. Srinivasan and D. Petkovic. Phonetic confusion matrix based spoken document retrieval. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 81–87, 2000.
7. A. Takasu. Probabilistic interpage analysis for article extraction from document retrieval. In *Proceedings of the 14th International Conference on Pattern Recognition*, pages 932–935, 1998.
8. A. Takasu. Bibliographic attribute extraction from erroneous references based on a statistical model. In *Joint Conference on Digital Libraries*, pages 46–60, 2003.
9. A. Takasu and K. Aihara. DVHMM: Variable length text recognition error model. In *Proceedings of the 15th International Conference on Pattern Recognition*, volume III, pages 110–114, 2002.
10. M. Wechsler, E. Munteanu, and P. Schäuble. New techniques for open-vocabulary spoken document retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 20–27, 1998.

Aligning Software Architectures of Mobile Applications on Business Requirements

Volker Gruhn and André Köhler

University of Leipzig, Department of Computer Science,
Chair of Applied Telematics / e-Business, Klostergasse 3, 04109 Leipzig, Germany
{gruhn, koehler}@ebus.informatik.uni-leipzig.de

Abstract. The support of mobile workers with mobile IT solutions can create dremendous improvements in mobile business processes of a company. The main charateristic of such a mobile system is the ability to connect via a (mobile) network to a central server, e.g. in order to access customer data. The frequency and the location of the use, data topicality, interaction requirements and many more are central aspects when developing a suitable system architecture. This paper provides a detailed decription of the four main software architectures for mobile systems and their main charateristics. Beyond, typical business requirements are developed, the implications for the system architecture for each of them is shown.

Keywords. Mobile system, System Architecture, Always-online systems, Business alignment

1 Introduction

Since the availability of mobile broadband networks and the reduced costs for mobile devices the use of mobile applications has become an interesting opportunity in several fields. Companies with large divisions of mobile employees (e.g. service technicians, sales representatives, healthcare services) can use mobile applications to gain access to corporate applications and databases at the point of service (POS). Therewith better coordination of mobile employees, rapid task assignment, the avoidance of error-prone format conversion, instant access to customer data and many more becomes feasible [1], [2].

The architecture of a mobile system can range from always-online systems using browser-based clients to fat client systems synchronizing with a central server occasionally. Which software architecture fits best for a specific mobile task depends basically on business needs. The frequency of movement, the probability of network availability, requirements for data topicaility, update mechanisms, synchronisation procedures and many more play a crucial role. Beyond, the costs for the development of a mobile system depend strongly from the choosen type of architecture.

As these issues are of particular relevance in the decision process of software architects and IT project managers, this paper explains the main types of mobile

system architectures with their advantages and disadvantages, typical business requirements for mobile systems and a matching scheme in order to identify the suitable architecture for a given set of business requirements.

This paper is organized as follows: Section 2 gives an overview about related work. Section 3 introduces four main types of architectures for mobile systems, showing their structure with component diagrams and explaining their main charateristics as well as their advantages and disadvantages. In section 4, typical business requirements for mobile systems are given and interdependencies between with the shown system architectures are described. Section 5 draws a conclusion.

2 Related Work

The changes for the discipline of software engineering when developing systems for mobile environments are discussed in [3]. The authors state that "mobility represents a total meltdown of all stability assumptions [...] associated with distributed computing". A comprehensive overview of software engineering for mobile systems is given, regarding issues like models, algorithms, applications and middleware to solve in the future. Our paper adresses the some of these issues. In [4] an architectural model that identifies the components representing the essential aspects of a mobile agent system is described. The interaction design for mobile information systems is subject of [5]. The authors developed a platform that supports the rapid prototyping of multi-channel, multi-modal, context-aware applications and describe how it was used to develop a tourist information system.

A lot of work is done regarding system architectures and other technical aspects of mobile system. An example for this work is [6], where a three-layer software architecture for distributed and mobile collaboration is presented. [7] presents an approach for the modeling and performance evaluation of mobile multimedia systems using generalized stochastic petri nets. The author focuses on verifying the optimal performance achievable under some QoS constraints in a given setting of design parameters. In [8] an approach for modeling software architectures for mobile distributed computing is presented. The modeling method aims at the verification of the correctness of both the functional and non-functional properties of the resulting mobile system.

The authors of [9] report on a project that aims for providing a system architecture simplifying the task of implementing mobile applications with adaptive behaviour. Temporal, spatial and personal mobility are considered in this approach. A comparison of different architectures for mobile systems is given in [10], where the client/server model, the agent-based client/server model and the mobile agent model is considered. In [11] is discussed, how the approach of service-oriented architectures can be applied within the development of mobile systems. It is shown, how to compose mobile services and how to apply these services in the system architecture.

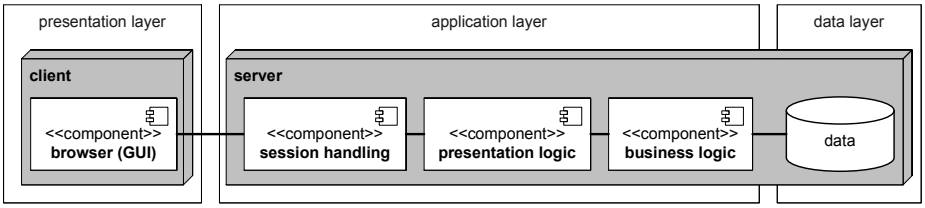


Fig. 1. Web-based always-online architecture

The work described in [12] is an essential basis for this paper. The authors give an overview about different types of software architectures for mobile systems, based on an analysis of different types of user and device mobility.

3 Types of Software Architectures for Mobile Systems

When analyzing the communication behaviour of a mobile application, its architecture is of particular relevance. According to [12], mainly four different types can be distinguished. First, a complete offline architecture could be used where (nearly) no communication via a network occurs. As we focus mainly on network issues, we do not consider this type of architecture. Second, an offline architecture is conceivable, where the mobile user synchronizes the mobile application occasionally with a central server. Third, a hybrid architecture could combine the advantages of an offline and an online architecture: If a network is available the mobile application communicates online with a central server, if the network is unavailable, the mobile application works offline and can be synchronized later with the central server. Fourth, with an always online architecture, the mobile application would communicate with a central server exclusively. This architecture is typical for web-based systems. In the following, we present a component-based view onto these architectures and explain their main characteristics.

3.1 Web-based Always-online Architecture

Whitin this architecture (see Fig. 1), the client works always online via a (mobile) network. The presentation layer is completely realized at the client side, where only a browser component is needed to realize the graphical user interface (GUI). The application layer is located server-side and contains a session handling component as well as components for presentation and business logic. The data layer contains the databases at the server side.

The main advantage of this architecture is, that only a browser is needed at the client side. Thus, the systems architecture allows the cooperation with a wide range of client systems independently from the client' operating systems or other client-side conditions. Furthermore, as all the data and the logic is located server-side, no update or synchronization mechanisms are needed. All clients can work on the same central data base, using the recent data as well as the

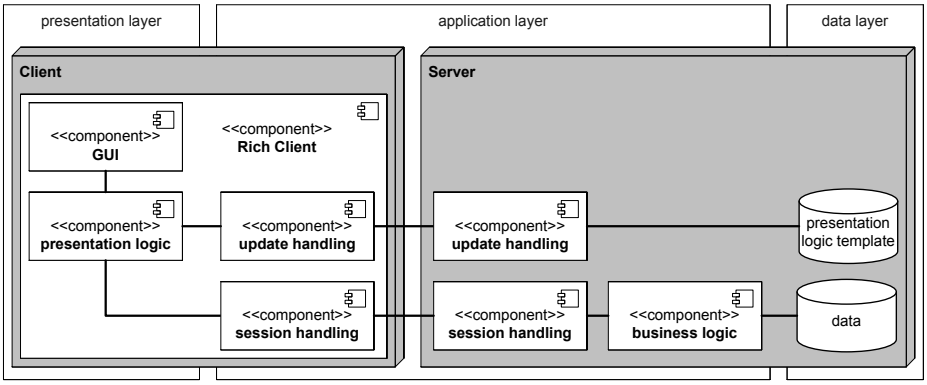


Fig. 2. Rich Client Always-online Architecture

recent presentation and business logic. The effort for the administration of such a systems is very small compared to other system architectures.

The main disadvantage of this solution is, that always a (mobile) network is needed, otherwise the client is unusable. When using a mobile network, the coverage in certain areas might not be given. Beyond, mobile networks usually offer just a small bandwidth causing a probably unsatisfying performance of the client. Furthermore, browser-based applications are limited in terms of user interface design to the facilities of HTML, which is quite less than users known from fat client applications. The simultaneous connection of a large number of clients to the central server also causes high requirements for the central server regarding its performance as well as its operational availability.

3.2 Rich Client Always-online Architecture

Within this architecture (see Fig. 2), one disadvantage of the web-based always-online architecture is adressed: Using a rich client at the client side, the limitations of the user interface design to the facilities of HTML can be overcome, as rich clients offer the full variety of interaction elements for the interface design. Beyond, rich clients offer the advantage of moving presentation logic to the client and therewith offering perfomance and costs improvements through a reduction of data traffic via the mobile network.

The presentation layer is completely realized on the client, containing a GUI and a presentation logic component. Additionally, the client contains some elements of the application layer, i.e. an update component for the presentation logic as well as a session handling component. Both components have an equivalent at the server side, where also a component for the business logic is located (application layer). The data layer is completely realized at the server side containing templates for the presentation logic as well as other databases.

The main advantages of this solution are the full user interface design capabilities and the reduced data traffic, still having a thin client. Furthermore,

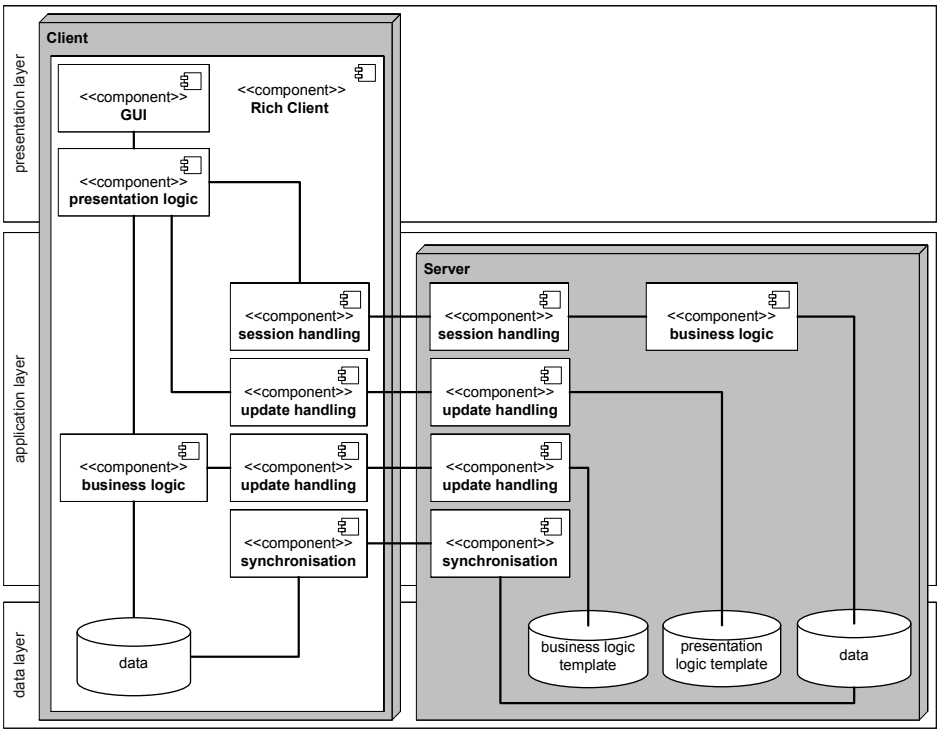


Fig. 3. Rich client hybrid architecture

all clients can work on the same central data base, using the recent data as well as the recent presentation and business logic. The effort for the administration of such a systems is small compared to other system architectures (except the web-based always-online architecture). Furthermore, with rich clients partly asynchronous communication becomes feasible, producing an improved user interaction. Short-time network disconnection can be intercepted through the client-side session component.

The disadvantage of this solution is the permanent need for a (mobile) network, as otherwise the client is unusable. When using a mobile network, the coverage in certain areas might not be given. Beyond, mobile networks usually offer just a small bandwidth, causing a probably unsatisfying performance of the system. The simultaneous connection of a large number of clients to the central server also causes high requirements for the central server regarding its performance as well as its operational availability. Additionally, components for the session and update handling need to be developed, update mechanisms are needed.

3.3 Rich Client Hybrid Architecture

The two architectures described before suffer from one main disadvantage: If no mobile network is available, the application is not usable for the mobile worker. The rich client hybrid architecture adresses this issue (see Fig. 3). The aim of this architecture is to keep the client as thin as possible but to assure that the application works also in the emergency when no mobile network is available.

The client consists of a GUI and a presentation logic component (presentation layer). Additionally, a business logic component as well as session handling, update handling and data synchronization components are located at the client side (application layer). Also a database is needed at the client (data layer). The business logic, session handling, update handling and synchronization components are also needed at the server side. The server-side data layer consists of templates for business and presentation logic and other databases. In the normal case the application works always-online, using business logic and data from the server side (like in the rich client always-online scenario). In case of loosing the network connection, the application would use the equivalent components at the client. Thus, a synchronization mechanisms is needed.

The main advantage of this architecture is the ability to work always-online but having also the capability to work offline when no network is available. Through the use of a rich client, the full user interface design capabilities are available and reduced data traffic can be achieved. When establishing a network connection, all clients can work on the same central data base, using the recent data as well as the recent presentation and business logic. Furthermore, with rich clients partly asynchronous communication becomes feasible, producing an improved user interaction. Short-time network disconnection can be intercepted through the client-side session component.

The main disadvantage of this solution is, that when using a mobile network, the coverage in certain areas might not be given. Beyond, mobile networks usually offer just a small bandwidth causing a probably unsatisfying performance of the system. The simultaneous connection of a large number of clients to the central server also causes high requirements for the central server regarding its performance as well as its operational availability. Additionally, components for the session and update handling need to be developed, update mechanisms are needed. When working offline, a synchronization mechanism needs to transfer data stored at the client, resolving possible conflicts with the server-side data base. In this architecture, the client is a thick client as many components needed to realize different functionality. All these components need to be developed and updated regularly, which causes a high administration effort.

3.4 Fat Client Offline Architecture

The fat client offline architecture is quite similar to the rich client hybrid architecture except that no online connection is provided (see Fig. 4). The mobile worker uses the application always offline, the client represents the whole application. The data stored on the client is occasionally synchorized with a central

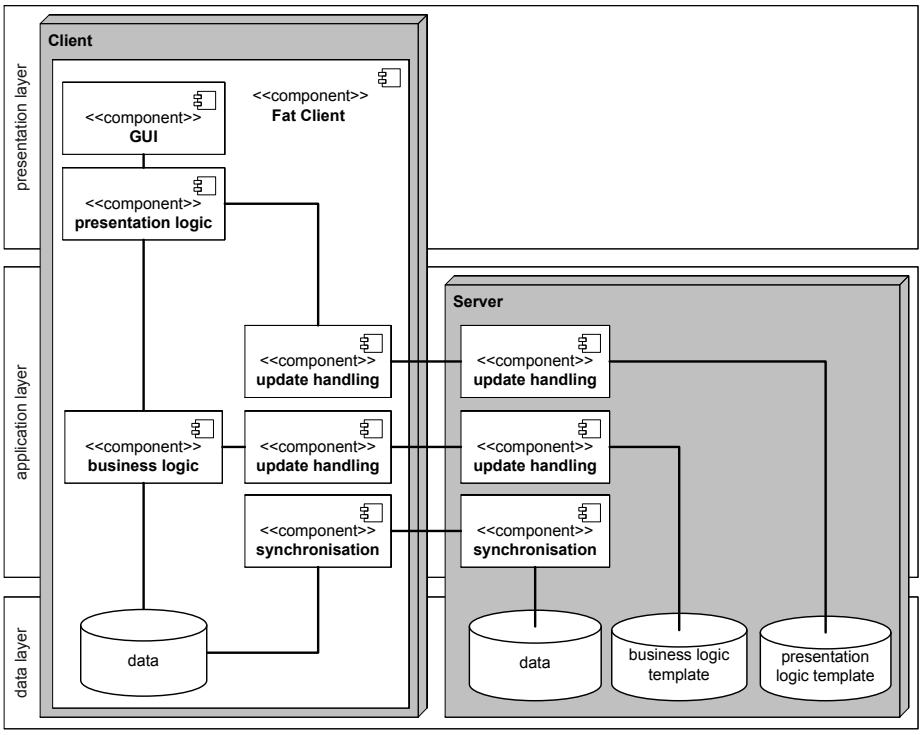


Fig. 4. Fat client offline architecture

server, e.g. via a stationary network at the mobile workers office. At these points, also the components for business and application logic need to be synchronized.

The main advantage of this architecture is, that through the use of a fat client, the full user interface design capabilities are available. As no network connection is needed, the application can be used very flexible in nearly every environmental situation. No costs or performance problems due to the restrictions of mobile networks occur.

The main disadvantage of this solution is, that components for the update handling need to be developed and deployed regularly. A synchronization mechanism needs to transfer the data stored at the client, resolving possible conflicts with the central server. The synchronization intervalls are completely influenced by the user, the data and component topicality can not be assured.

3.5 Architecture Comparison

The web-based always-online architecture is obviously a very small and leightweight architecture, causing relatively small effort for its development and administration. But, it has also some significant shortcomings in connectivity and usability issues. The other three architectures aim on the improvement of these

	always-online web-based	always-online rich client	hybrid rich client	offline fat client
point of service				
office	+	+	+	+
urban areas	0	0	+	+
rurally areas	–	–	+	+
data topicality				
real–time data required	+	+	–	–
relatively recent data required	+	+	+	–
other topicality required	+	+	+	+
source code redundancy				
large redundancy accpeted	+	+	+	+
small redundancy accepted	+	+	–	–
no redundancy accepted	+	–	–	–
software distribution				
offline distribution accepted	+	+	+	+
online distribution accepted	+	+	+	0
distribution not accepted	+	–	–	–
user interface design and interaction techniques				
should be extensive	–	+	+	+
can be restricted to HTML	+	+	+	+
security issues				
decentral organisation accepted	+	+	+	+
central organisation demanded	+	+	0	–

Table 1. Business requirements and their architectural implications

shortcomings. But, as more as they improve connectivity and usability, the effort for development and administration grows rapidly. The main task for IT project managers and software architects is, to decide how much effort for the developement and the administration of such a solution is justifiable for the given business needs. The following section shows some of the typical business requirements for mobile solutions and explains, how the appropriate system architecture can be deduced.

4 Typical Business Requirements for Mobile Systems and their Architectural Implications

In the following, typical business requirements for mobile applications are described and evaluated regarding their effects on the system architecture. The results are shown in Table 1. If an architecture is thoroughly suitable to fulfill a given business requirement, it is marked with '+', if it is suitable with some restrictions it is marked with '0'. If an architecture is not suitable for a given business requirement, this is indicated by '–'.

4.1 Point of Service

The typical location of the POS is of particular relevance for deriving a suitable system architecture. If the application is used mostly in a stationary environment, all kind of architectures are conceivable. If the application is used mobile in urban environments, mobile networks will probably be available most of the time. Thus, the web-based as well as the rich client always-online architecture will be suitable in most of the cases, the other two architectures will work of course in every situation. If the mobile application is used in rurally environments, the availability of a mobile network can frequently not be assured. In these cases, only the hybrid rich client architecture as well as the fat client offline architecture is conceiveable.

4.2 Data Topicality

The requirements for data topicality have also a main influence on the system architecture. If the mobile worker needs always real-time data topicality, only the always-online architectures can be considered. The rich client hybrid architecture can be used, if the requirements for data topicality are not so strict (e.g. data from the previous day are sufficiant). The fat client offline architecture is only conceivable, if the requirements for data topicality are not critical (e.g. a couple of days or weeks).

4.3 Sorce Code Redundancy

The architecture of a mobile system influences also the source code redundancy. If the business logic of an application is used in different contexts (e.g. client and server side), the business logic component often needs to be developed twice because of different system requirements. Each later change in the business logic component also need to be implemented twice. If the complete avoidance of source code redundancy is demanded (single source, single instance), only the web-based always-online architecture is conceivable. If a small source code redundancy is accepted (single source, multiple instances), the always-online rich client architecture should be choosen. If no restrictions regarding the source code redundancy exists, the hybrid rich client architecture and the fat client offline architecture are feasible.

4.4 Software Distribution

The distribution of new releases for mobile applications often causes a high effort. To avoid this, the web-based always-online architecture should be choosen, as within this architecture the update process can be conducted for a single server instance of an application. If an online update process is accepted, the always-online and the hybrid rich client architecture can be used. As the update process for fat client offline systems usually requires a high amount of data to be transferred, often only offline updates (e.g. via CD, DVD) are possible, causing very high costs and organizational effort.

4.5 User Interface Design and Interaction Techniques

Both the rich client and the fat client architecture offer the full range of elements for designing the user interface. If the application is not to complex and only simple user interactions are needed (feasible with HTML), the web-based always-online architecture can be used.

4.6 Security Issues

Within mobile applications often confidential data is processes and transfered. Thus, a couple of security mechanisms are needed. From the administrators point of view, a centralized security management would be desirable. This would be feasible with both the always-online architectures. The hybrid rich client architecture as well as the fat client offline architecture require also security effort at the client side.

5 Conclusion

In this paper we presented the four basic software architectures conceivable for mobile systems. The decision, which architecture is the most suitable in a given project situation depends on the business requirements on the one hand side as well as on the restrictions for the development and administration effort on the other hand side. Both aspects have strong interdependencies. The higher the business requirements are, especially in terms of connectivity and usability, the higher the costs for the development and the administration of the solution are.

Considering this, no general recommendation for a system architecture of a mobile system can be given. It is rather necessary to asses single aspects of business requirements like the exact needs for connectivity, redundancy and update preferences and many more. For each single aspect the best system architecture can be derived, but often the result will vary over all considered aspects. Then, a compromise need to be found for the given situation. The above given business requirements and their relation to the developed system architetures can help to support this process.

6 Acknowledgements

The Chair of Applied Telematics/e-Business is endowed by Deutsche Telekom AG. The results presented in this paper were partly developed within a research project in cooperation with the company inverso GmbH [13].

References

1. Gruhn, V., Köhler, A., Klawes, R.: Modeling and analysis of mobile service processes by example of the housing industry. In van der Aalst, W.M., Benatallah, B., Casati, F., Curbera, F., eds.: Business Process Management, Springer LNCS 3649 (2005) 1–16

2. Nah, F.F.H., Siau, K., Sheng, H.: The value of mobile applications: a utility company study. Communications of the ACM **48** (2005) 85–90
3. Roman, G.C., Picco, G.P., Murphy, A.L.: Software engineering for mobility: a roadmap. In: ICSE '00: Proceedings of the Conference on The Future of Software Engineering, New York, NY, USA, ACM Press (2000) 241–258
4. Schoeman, M., Cloete, E.: Architectural components for the efficient design of mobile agent systems. In: SAICSIT '03: Proceedings of the 2003 annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology, South African Institute for Computer Scientists and Information Technologists (2003) 48–58
5. Belotti, R., Decurtins, C., Norrie, M.C., Signer, B., Vukelja, L.: Experimental platform for mobile information systems. In: MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking, New York, NY, USA, ACM Press (2005) 258–269
6. Dustdar, S., Gall, H.: Architectural concerns in distributed and mobile collaborative systems. Journal on System Architecture **49** (2003) 457–473
7. Tsang, T.: Modelling and performance evaluation of mobile multimedia systems using qos-gspn. Wireless Networks **9** (2003) 575–584
8. Issarny, I., Tartanoglu, F., Liu, J., Sailhan, F.: Software architecture for mobile distributed computing. In: WICSA '04: Proceedings of the Fourth Working IEEE/IFIP Conference on Software Architecture (WICSA'04), Washington, DC, USA, IEEE Computer Society (2004) 201
9. Augustin, I., Yamin, A.C., Barbosa, J.L.V., Geyer, C.F.R.: Isam, a software architecture for adaptive and distributed mobile applications. In: ISCC '02: Proceedings of the Seventh International Symposium on Computers and Communications (ISCC'02), Washington, DC, USA, IEEE Computer Society (2002) 333
10. Kan, Z., Luo, J., Hu, J.: The design of a software technology architecture for mobile computing. In: HPC '00: Proceedings of the The Fourth International Conference on High-Performance Computing in the Asia-Pacific Region-Volume 2, Washington, DC, USA, IEEE Computer Society (2000) 762
11. van Thanh, D., Jorstad, I.: A service-oriented architecture framework for mobile services. In: AICT-SAPIR-ELETE '05: Proceedings of the Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunications Workshop (AICT/SAPIR/ELETE'05), Washington, DC, USA, IEEE Computer Society (2005) 65–70
12. Book, M., Gruhn, V., Hülder, M., Schäfer, C.: A methodology for deriving the architectural implications of different degrees of mobility in information systems. In H. Fujita, M., ed.: New Trends in Software Methodologies, Tools and Techniques, IOS Press (2005) 281–292
13. http://www.inverso.de.