Department Knowledge Engineering
University of Vienna

Workshop on Ontology and Enterprise Modelling: Ingredients for Interoperability

Proceedings

In Conjunction with 5th International Conference on Practical Aspects of Knowledge Management
December 2, 2004
Vienna, Austria

Edited by
Harald Kühn
BOC Information Systems GmbH

Supported by
Network of Excellence INTEROP
www.interop-noe.org

Program Committee

Yannis Charalabidis, Singular Software, Greece
Norbert Gronau, University Potsdam, Germany
Manfred Jeusfeld, University of Tilburg, Netherlands
Dimitris Karagiannis, University of Vienna, Austria
Michele Missikoff, Italian National Research Council, Italy
Dimitris Plexousakis, University of Crete, Greece
Benkt Wangler, University of Skövde, Sweden

Organization

Harald Kühn, BOC Information Systems, Austria

Contents

Preface

"There exist various definitions on interoperability. According to the Oxford Dictionary, interoperable means 'able to operate in conjunction'. The word "interoperate" also implies that one system performs an operation on behalf of another system. From software engineering point of view, interoperability means that two co-operating software systems can easily work together without a particular interfacing effort. It also means establishing communication and sharing information and services between software applications regardless of hardware platform(s). The interoperability is considered achieved if the interaction can, at least, take place at the three levels: data, application and business process with the semantics defined in a business context."
(From: Model driven and dynamic, federated enterprise interoperability architectures and interoperability for non-functional aspects, Deliverable D9.1, State of the Art of Interoperability Architecture Approaches, Version 1.0, page 13, November 2004, Network of Excellence INTEROP, www.interop-noe.org)

"Ontology represent a new technology with the purpose to improve electronic information organization, management, and understanding. An ontology is a conceptual information model that describes "the things that exist" in a domain (hence the name): concepts, properties, facts, rules and relationships. An ontology acts as a standardized reference model to support information integration and knowledge sharing. Hence, the role of ontology is twofold:

- *it supports human understanding and communication,*
- *in machine-processable form, it facilitates content-based access, communication and integration across different information systems.*

These roles are both achieved by explicating and formalizing the meaning, or semantics, of organization and enterprise application information resources."
(From: Ontology-based integration of Enterprise Modelling and Architecture & Platforms, Deliverable D8.1, State of the Art and State of Practice Including Initial Possible Research Orientations, Version 1.2, page 12, November 2004, Network of Excellence INTEROP, www.interop-noe.org)

In this workshop we explore important ingredients for interoperability in enterprise systems: enterprise modelling and ontology. The workshop is structured into four thematic sessions:

1. Ontology Building
2. Model Driven Development
3. Web-based Systems
4. Ontology-based Frameworks

Dr. Harald Kühn
Vienna, December 2, 2004

Modelling semantic relations to build an inferential engine for a complex knowledge domain.

Fiorella De Cindio, Laura Anna Ripamonti, Alfio Ferrara, Cristian Peraboni

D.I.Co. – Università di Milano – Italy

fiorella.decindio@unimi.it, [ripamonti, ferrara]@dico.unimi.it, cris.peraboni@tin.it

ABSTRACT

Retrievability and reusability of Learning Objects (LOs) in a shared distributed context is a relevant issue, primarily due to the resource-consumption that the production of on-line courses implies. Ontologies can be effectively exploited in order to provide a mean of searching based on a widely recognised cataloguing criteria, that couples the rigorous directions provided both by the SCORM and the ACM Computer Curricula standards, while enriching them with meaningful semantic relations among stored LOs' contents. The resulting ontology-based searching engine will be embedded into a portal for distributing PhD courses in Informatics of several Italian Universities[1].

KEYWORDS

E-learning, SCORM, ACM Computer Curricula, ontology, semantic relations, Semantic Web

[1] This work has been partially supported by the Italian Ministry of Education, Universities, and Research in the framework of the FIRB "Web-Minds" project N. RBNE01WEJT_005.

1. Introduction

The WEB-MINDS (Wide-scalE, Broadband, MIddleware for Network Distributed Services) project involves a number of Italian Universities, and is aimed at the development of an effective middleware framework for accessing information and services (both with mobile and not devices) using a peer-to-peer architecture. In this work, we focus on the project work package that addresses the issue of e-learning in a distributed context, with special attention to the definition of an ontology representing the Computer Science knowledge domain.

The aim is to support the development and distribution of Computer Science courses for PhD students through the net. After a testing phase, the searching and downloading capabilities of the developed portal will be extended also to undergraduate courses. Since the environment will be embedded into a general middleware platform and integrated with different existing *Learning Management Systems* (LMSs), a strong emphasis has been put on the modularity and scalability aspects, defining the general structure shown in Fig.1: a central catalogue contains referrals to the whole set of Learning Objects (LOs) stored in several local repositories (eventually linked to a specific LMS). An user accessing the portal can search for specific contents through an interface onto the ontology-based catalogue and then download the LOs s/he is interested into. We won't dig any further into the overall structure, since the focus of this work is on the definition of the very core of the system, that is to say the ontology upon which we are building the project's searching engine. Actually, since e-learning courses production is an expensive activity, it is fundamental to produce contents that, if necessary, can be easily retrieved and reused: this implies the application of universally used standards and the definition of a tool able to allow detailed search among existing shared LOs. To accomplish this goal, at least two dimensions need to be included in whichever cataloguing system we can imagine: a mean of standardizing the LOs format and a universally shared representation of the specific knowledge domain (in our case the Computer Science disciplines) for classifying LOs content. Ontologies are generally recognized as an essential tool for supporting a common representation and understanding of a domain of interest (an ontology allows users to classify information in terms of concepts, properties, and semantic relations). We will combine the above two dimensions, and then enrich them by adding semantically significant relations, for creating an ontology-based cataloguing system, able to guarantee: that the catalogue is understood, shared and accepted universally by the domain's actors (teachers, students, …), an exhaustive coverage of the domain; and an easy extendibility of the catalogue itself. Moreover, since the portal uses web-based technologies, for representing the ontology we adopt OWL (Web Ontology Language), that in the Semantic Web context [Berners-Lee, et al., 2001], has been recently proposed as a standard capable of joining the Description Logics semantics with the syntactical freedom of RDF (Resource Description Framework) [Smith (Ed.) et al., 2004].

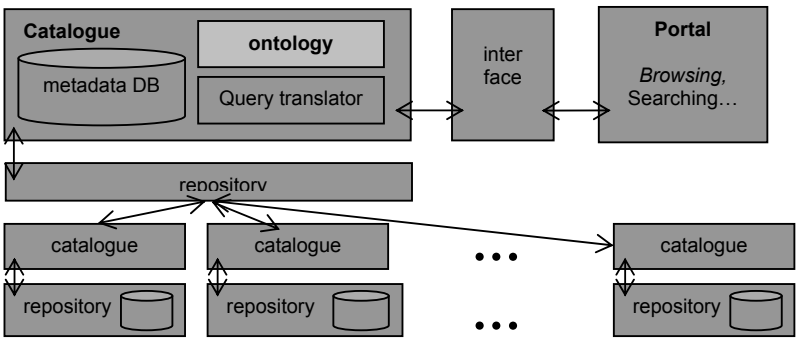

Figure 1 – WEB-Minds project portal overall structure

2. An ontological representation of LOs based on SCORM and ACM-CC standards

Several definition of ontology have been proposed [Guarino and Giaretta, 1995]; in this paper, according to [Gruber, 1993] and [Studer et al., 1998], we define an ontology as *a formal specification of a shared conceptualization*, since we are interested in emphasizing the role of ontologies in providing a definition of concepts that could be shared by a wide community of users that have the same, or similar, needs. Moreover, our approach is aimed at providing a reusable skeleton for several possible knowledge bases, as suggested in [Swartout and Gil, 1995].
In particular, we address the problem of building a single "global" ontology by combining the ontological representation of the SCORM (Shareable Content Object Reference Model initiative) and ACM-Computer Curricula (ACM-CC) [ACM, 2001] models. As a matter of fact, the growing interest around e-learning issues has made indispensable the creation and adoption of standards able to guarantee that learning contents are sharable and portable through different LMSs. The standardisation process has crystallized into the SCORM (Sharable Content Object Reference Model) standard [Advanced Distributed Learning, 2003]. Hence, we included a representation of SCORM metadata into the ontology. On the other hand, our aim is providing the portal with cataloguing capabilities that go beyond a mere "taxonomisation" of its content. To provide a mean for selecting, navigating and exploiting semantic relations referred to LOs' content, we have adopted the ACM Computer Curricula (ACM-CC) standard for classifying LOs on the basis of their knowledge content (for the specific reasons for choosing ACM-CC instead of the ACM standard see [De Cindio et al., 2004]).

2.1 Design methodology: *synthesis approach* and *one shot strategy*

To design the "global" ontology we have adopted the *synthesis approach* [Holsapple & Joshi 2002], that allows the creation of an ontology by synthesising

common characteristics between two (or more) ontologies. Thus, before undertaking the *synthesis* phase, it has been necessary to define the two "input" ontologies, describing respectively the ACM-CC taxonomy, and the SCORM metadata. To this extent the *one-shot strategy* (Fig.2) [Gomez-Perez, Fernandez-Lopez and Corcho, 2003] has been adopted. This approach, that has been followed till the *formalization* phase, allowed us to build the ontologies starting from the two standards' specifications and a bunch of textual information. The *data analysis* activity allowed us to select relevant elements and their properties among the information sources. During the *conceptualization* phase we have analysed the meaning of concepts, the relations among them, and types and cardinalities of properties. Finally the ontologies have been *formalized*, formally defining concepts, relations among them, properties and relations among properties and concepts, while building specific restrictions where necessary.

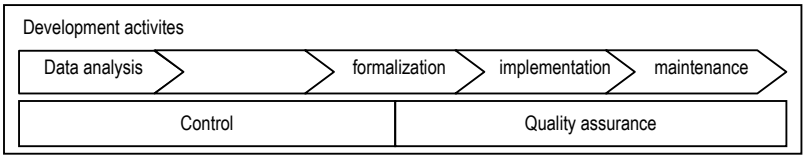

Figure 2 – one-shot strategy activities

2.2 Deriving the SCORM ontology from XML-schema

SCORM structures learning contents in *Content Model Component* that are then packaged into *Content Packages* for sharing them among SCORM-conformant systems. Each *Content Model Component*, and each *Content Package* (or *Content Aggregation*), should be described using metadata compliant with the IEEE 1484.12.1-2002 standard [http://ltsc.ieee.org/wg12/20020612-Final-LOM-Draft.html]. Thus we can imagine each *Content Component* stored in the system as a LO, and we have modeled the LOMs (*Learning Objects Metadata*) structure considering them as autonomous objects able to describe whichever *Content Component* can enter the system. In the *Data Analysis* activity phase that has been conducted on the IEEE document, we have selected representative concepts among metadata. We have reconstructed the metadata eXtensible Markup Language (XML)-schema in order to obtain a more structured view, and then we have collected information on each element belonging to a LOM - so to determine its role and nature -, and thus defining the whole set of concepts belonging to the ontology and their properties. It is important to point out that not the whole set of concepts is derived directly from LOMs: some of them have been modelled as concept specifically for our purposes (e.g. the *vocabulary item* concept has been derived from the *vocabulary* data type, that represents any term in any vocabulary in the SCORM standard). The following *Conceptualization* phase has focused on determining semantic relations existing among concepts and properties. This activity, coupled with the definition of the nature (*datatype property* or *object property*), the type (for the *datatype properties*) and the cardinality of each property, allowed us to draw the H-MODEL [Castano et al., 2004] schematically represented in Fig.3 (for figure readability only the principal items have

been sketched). Then, during the *Formalization* phase, we have transformed the conceptual model into a formal one, using Protégé [http://protege.stanford.edu]. Each concept has been translated into an OWL *class*, and each property has been modelled, according to its nature, into a *DataType* or an *Object Property*. Then properties have been associated to related classes, imposing appropriate restriction on cardinalities values. Finally, we have transformed semantic relations among concepts into relations among classes.

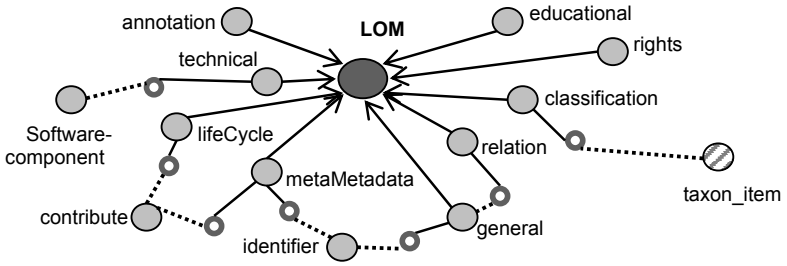

Figure 3 - H-MODEL of the SCORM ontology

2.3 Representing the Computer Science knowledge domain through an ACM-CC-based ontology

As a starting point for building the ACM-CC ontology we adopted an existing DAML+OIL (DARPA Agent Markup Language + Ontology Reference Layer) ontology developed by the University of Trento [Saini P., Ronchetti M., 2003]. Since that representation of the Computer Science (CS) domain of knowledge was only partially adequate for our purposes, several modifies and improvements have been introduced. We have deeply analysed the ACM-CC standard document [ACM, 2001], with an eye to the use we would do of its taxonomy: for this reason we have chosen to base our ontology only on the Appendix A of the standard, that describes in very details the Computer Science Body of Knowledge, disregarding Appendix B, that focuses mainly on defining how undergraduate courses should be organized in order to be consistent with the ACM standard[2].

Appendix A taxonomy is organized into three hierarchical levels (Fig.4):

1. **Area**: the higher level, representing a specific discipline;
2. **Unit**: representing a single module within an area;
3. **Topic**: the lower level f the hierarchy, details its unit's content.

In our ontological representation each of these levels has been modelled in such a way that relevant semantic relations among items (area, unit, topic) can be easily traced.

[2] The very reason for this choice is that undergraduate courses structure often has particularities depending on the specific Country, while Appendix B is specially tailored on USA courses structure.

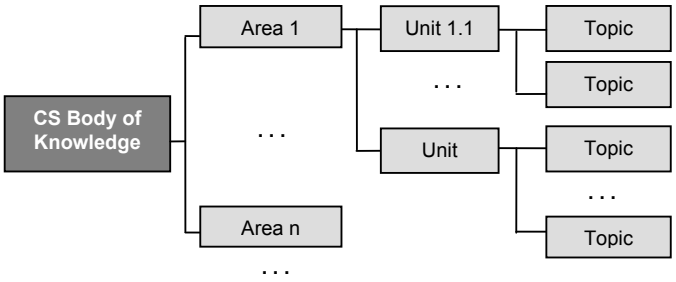

Figure 4 – ACM-CC taxonomy structure

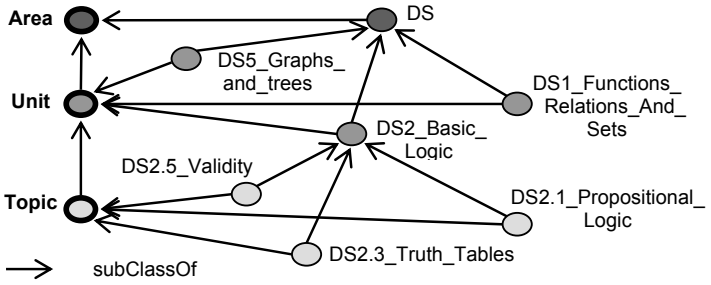

Fig. 5 – H-MODEL of the ACM CC ontology

Actually, since the solution developed by the University of Trento modelled the Unit class as a subclass of the Area class (and thus each *area* of the ACM-CC taxonomy was an instance of the Area class, each *unit* an instance of the Unit class and each *topic* a property of its *unit*), it was very difficult to trace new semantic relations among items. Our solution implies the use of three general classes: *Area*, *Unit* (subclass of Area) and *Topic* (subclass of Unit). Hence, each topic is no longer modelled as an instance of its generic class, but as a subclass. For this reasons, e.g., the DS – Discrete Structure Area is a child class of the generic class *Area* (see Fig.5). Moreover, each class referred to an item is a subclass of the class referred to the item that precedes in the hierarchy of the taxonomy. Thus, e.g., the Propositional Logic topic is represented as a child class of the *Topic* class and of the class related to the DS2 – Basic Logic unit, that, in its turn, is a child of the *Unit* class and of the class related to the DS – Discrete Structure area, that, as we said, is a subclass of the *Area* class. Last but not least, the taxonomy has been extended by adding an identifier to each topic. As a result our ontological representation do contains the whole set of the taxonomy items modelled as *concepts* endowed with an *identifier*, thus allowing to trace navigable semantic relations among them quickly and easily. The ontology has then been completed by selecting, defining and adding semantic relations.

One of the main aim of the portal is to make possible researches among the LOs stored in the distributed repositories that will give as output not only the results of the

specific query (e.g. all the LOs containing theoretical lessons on "OS7.3 – security methods and devices"), but also one or more lists containing LOs semantically related to the query subject. To achieve this goal it has been necessary to identify all the different types of semantic relations and what kind of items each of them may connect. Tab.1 summarises the results. Once established all the possible type of relations, the text of the ACM-CC standard has been carefully read in order to identify and formalize semantic relations among its items applying *a collaborative approach*. This process, still going on, has been broken down into two phases: in a first step, semantic relations have been extrapolated exclusively through the cues explicitly present in the text. Then, experts of the field are being involved, in order to validate these relations. This approach should minimize the presence of subjective perceptions into the ontology, that must preserve the maximum universality in the description of its specific domain.

Table 1 - possible semantic relations among the ACM-CC based ontology items

SEMANTIC RELATION TYPE	ITEMS RELATED	DESCRIPTION
is subClassOf	Area-unit, unit-topic, topic-subtopic	Describes ACM-CC taxonomy structure
is suggested for	any	item A should be a prerequisite for learning item B
is required for	any	item A do is a prerequisite for learning item B
is detailed by	any	item A may be further deepened by learning item B

2.4 Synthesising the global ontology from SCORM and ACM-CC ontological representations

To merge the two ontologies in a single "global" one, a deep analysis of the concepts present into each of them has been conducted in order to select common concepts. We have focused on the *TaxonItem* concept belonging to the SCORM-based ontology and on its relations with other concepts, since it represents the structure of any generic item in the classification and characterises the item through three properties:

- *source*: that identifies the classification system to which the item belongs;
- *id*: that identifies one specific item into the classification system;
- *taxon entry*: that contains the label associated to the item.

This concept is referred by the *Classification* class through the *TaxonPath* attribute, that identifies possible classification paths for the LO of which is a metadata. Hence, the global ontology is composed by an orthogonal connection between the two ontologies through *TaxonItem*, that "contains" a navigable representation of the Computer Science knowledge domain. As a result, the searching engine that will be embedded into the Webminds portal will be able to provide an answer to different type of queries, producing at the meantime a list of all the other LOs related to the subject of the answer, distinguishing among those which are related to issues that are suggested or compulsory prerequisite for studying a specific topic, or are useful for further deepening it.

3. Conclusion & future developments

The present work outlines the design of a shared and agreed upon classification system for cataloguing LOs of PhD courses in Informatics stored in a distributed structure, providing - at the same time - a mean for supporting not-trivial queries on the stored LOs, and thus assuring both retrievability and reusability. The next step of our work will be the design and implementation of an appropriate searching engine able to exploit the semantic richness provided by the ontology.

References

ACM, 2001, *Computing Curricula 2001 Computer Science*, Final Report.

Advanced Distributed Learning, 2003, *Sharable Content Object Reference Model (SCORM) – The SCORM Content* Aggregation Model, Version 1.3 Working Draft 1.

Berners-Lee T. , Hendler J. and Lassila O., 2001. The Semantic Web. *Scientific American*. May.

Castano S., Ferrara A., Montanelli S., Racca G., 2004, From Surface to Intensive Matching of SemanticWeb Ontologies. In: Proc. of the *3th Int. Workshop on Web Semantics (WEBS) at DEXA 2004*, Zaragoza, Spain, IEEE Computer Society (to appear).

De Cindio F., Ripamonti L.A., Ferrara A., Peraboni C., 2004, Combining SCORM metadata and ACM Computer Curricula to create an ontology for cataloguing learning objects. In Proc. of *IADIS International Conference on Cognition and Exploratory Learning in Digital Age (CELDA 2004)*, 15-17 Dec., Lisbon, Portugal (to appear).

Gomez-Perez A., Fernandez-Lopez M., Corcho O., 2003, *Ontological Engineering Advanced Information and Knowledge Processing series*, Springer Verlag.

Gruber R., 1993, *Toward Principles for the Design of Ontologies Used for Knowledge Sharing*, Stanford Knowledge Systems Laboratory Technical Report KSL 93-04.

Guarino N., and Giaretta P., 1995. *Ontologies and Knowledge Bases: Towards a Terminological Clarification*. In Mars N. (Ed.) Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing. University of Twente, Enschede, The Netherlands. IOS Press, Amsterdam, The Netherlands, pp.25-32.

Holsapple C.W., Joshi K.D., 2002, A collaborative approach to Ontology design. *Communication of the ACM*, Vol.45, N.2, pp.42-47, February 2002.

IMS, 2003, IMS Content Packaging Information Model, Version 1.1.3 Final Specification

Saini P., Ronchetti M., 2003, Deriving *ontology-based metadata for e-learning from the ACM computing curricula*, # DIT-03-017 University of Trento – Department of Information and Communication Technology.

Smith M. K. (Ed.), et al., 2004. OWL Web Ontology Language Guide, http://www.w3.org/TR/2004/REC-owl-guide-20040210.

Studer R., Benjamins V.R., Fensel D., 1998. Knowledge Engineering: Principles and Methods. *Data Knowl. Eng*. N.25, Vol.1-2, pp. 161-197.

Swartout W., Gil Y., 1995, Expect: Explicit representations for flexible acquisition. 9th Workshop on Knowledge Acquisition for Knowledge Based Systems (KAW'95), Banff, Canada.

Similarity for Ontologies - a Comprehensive Framework

Marc Ehrig[1], Peter Haase[1],
Mark Hefke[2], Nenad Stojanovic[1]

[1]Institute AIFB, University of Karlsruhe
[2]FZI Research Center for Information Technologies at the University of Karlsruhe
{ehrig, haase, stojanovic}@aifb.uni-karlsruhe.de
hefke@fzi.de

Abstract. In this paper we present a comprehensive framework for measuring similarity within and between ontologies as a basis for the collaboration across various application fields. In order to define such a framework, we base our work on an abstract ontology model that allows to adhere to various existing and evolving ontology standards. The main characteristics of the framework is its layered structure: We have defined three levels on which the similarity between two entities (concepts or instances) can be measured: data layer, ontology layer, and context layer, that cope with the data representation, ontology meaning and the usage of these entities, respectively. In addition, in each of the layers corresponding background information is used in order to define the similarity more precisely. The framework is complete in the sense of covering the similarity between all elements defined in the abstract ontology model by comprising similarity measures for all above-named layers as well as relations between them. Moreover, we have validated our framework with several practical case studies in order to prove benefits of applying our approach compared to traditional similarity measures. One of these case studies is described in detail within the paper.

1 Introduction

The importance of ontologies as ”an explicit specification of a shared conceptualization” [1] increased drastically in the last years, especially for the applications that require integration of heterogenous data, like knowledge management. Indeed, ontology becomes a very important technology for the improvement of the inter/intra-organizational exchange of knowledge and services. Moreover, ontology-based knowledge management enables a variety of new retrieval services, like personalization and cooperative answering. The key issue is that an ontology supports more granular views on the knowledge items that should be exchanged, enabling in that way the more context-sensitive retrieval process than in the traditional knowledge management systems. For example, if a user makes the query ”fast Jaguars”, an ontology-based system, by using the conceptual model of a domain, can distinguish between the various interpretations of the term Jaguar (e.g. car and animal) and enable the user to find only relevant items. However, due to the ambiguity in translating a user’s information need in a query, an efficient knowledge management system should enable the user to find some items that do not match perfectly his query, but that are relevant for his need. For example, if a user wants to find information about the safety issues of Jaguar X, the

relevant information could be documents about safety issues of the Ferrari Y, which has showed the same characteristics as Jaguar X in several crash tests. It is clear this calculation of the similarity between two items has to be performed very carefully in order to ensure that all relevant items and no irrelevant items will be retrieved. Moreover, the similarity computation depends on personal rules and preferences as well as on personal interpretation. To get a grip on arising problems from the above-described examples, it is in our opinion necessary to combine ontology-based technologies with novel approaches for similarity computation. Therefore we have developed a general framework for calculating the similarity within and between ontologies by also focusing on domain knowledge that has direct influence on the similarity. The main characteristics of the developed comprehensive framework is its layered structure: We have defined three layers on which the similarity between two entities, can be measured: data layer, ontology layer, and context layer, that cope with the data representation, ontological meaning and the usage of these entities, respectively. In that way, different layers consider different aspects of the nature of entities which are combined in the final judging about their similarity. Moreover, in each layer corresponding background information is used in order to define the similarity more precisely. Our intention is not only to develop a collection of existing methods for measuring similarity, but rather to define a framework that will enable their effective systematization regarding the task of comparing ontologies. The framework is complete in the sense of covering the similarity between all elements defined in the abstract ontology model and comprises several new methods in order to achieve such a completeness. Moreover, it provides some inter-/intralayer relations between methods, that enable the methods to be applied more efficiently. The framework has been validated with several practical case studies which are directly focusing on every single layer as well as their interaction in order to deliver a proof of concept.

The paper is organized as follows: In Section 2 we first introduce a formal model for ontologies and similarity in ontologies and thendescribe our general framework comprising three different layers for similarity computation. In Section 4 we validate our framework with a selected case study. After a discussion of related work in Section 5 we conclude with remarks outlining some future work.

2 Definitions

In this section we introduce basic definitions of ontologies and similarity. Upon these we build our similarity framework.

2.1 Ontology Model

In our framework we will adhere to the Karlsruhe Ontology Model[2], which we adapt to also accommodate datatypes:

Definition 1 (Ontology with Datatypes). *An ontology with datatypes is a structure*

$$O_T := (C, T, \leq_C, \leq_T, R, A, \sigma_A, \sigma_R, \leq_R, \leq_A)$$

consisting of a set of concepts C aligned in a hierarchy $\leq_C$, a set of relations R with $\leq_R$, the signature $\sigma_R\colon R \to C \times C$, a set of datatypes T with $\leq_T$, a set of attributes A with $\leq_A$, and the signature $\sigma_A : A \to C \times T$. For a relation $r \in R$, we define its domain *and its range by* $\mathrm{dom}(r) := \pi_1(\sigma_R(r))$ *and* $\mathrm{range}(r) := \pi_2(\sigma_R(r))$.

Ontologies formalize the intensional aspects of a domain. The extensional part is provided by a knowledge base, which contains assertions about instances of the concepts and relations.

Definition 2 (Knowledge Base with Datatypes). *A* knowledge base with datatypes *is a structure*

$$KB_T := (C_{KB}, T_{KB}, R_{KB}, A_{KB}, I, V, \iota_C, \iota_T, \iota_R, \iota_A)$$

consisting of fours sets C_{KB}, T_{KB}, R_{KB} and A_{KB} as defined above, a set of instances I, a set of data values V, the concept instantiation $\iota_C\colon C_{KB} \to \mathfrak{P}(I)$, the data value instantiation $\iota_T : T_{KB} \to \mathfrak{P}(V)$, the relation instantiation $\iota_R\colon R_{KB} \to \mathfrak{P}(I^2)$, and the attribute instantiation $\iota_A\colon A_{KB} \to \mathfrak{P}(I \times V)$.

2.2 Similarity

Common sense tells that two entities need common characteristics (or attributes) in order to be considered similar. Formalizing the concept of similarity, we refer to the definition of a similarity function introduced by [3]:

Definition 3 (Similarity Measure). *A similarity measure is a real-valued function $sim(x, y) : S^2 \to [0, 1]$ on a set S measuring the degree of similarity between x and y.*

Though there may be split opinions about the properties of sim, it is generally agreed that sim ought to be reflexive and symmetric, i.e.

$$\forall x, y \in S \text{ it holds} \quad \begin{array}{ll} 1.\ sim(x,x) = 1 & \text{(reflexivity)} \\ 2.\ sim(x,y) = sim(y,x) & \text{(symmetry)} \end{array}$$

Similarity for Ontologies

Definition 4. *Now given two knowledge bases with datatypes and the corresponding ontologies (KB_i, O_i) and (KB_j, O_j), we can compare elements of $KB_i^{O_i}$ as given by the following family of functions:*

$$sim : (E_i) \times (E_j) \to [0..1],$$

where $E_i \in \{O_i, C_i, T_i, R_i, A_i, I_i, V_i\}$ and $E_j \in \{O_j, C_j, T_j, R_j, A_j, I_j, V_j\}$ and E_i and E_j are of the same kind, i.e. both are instances or both are concepts, etc.

3 General Framework

Since an ontology represents a conceptualization of a domain, comparing two ontology entities goes far beyond the representation of these entities (syntax level). Rather, it should take into account their relation to the real world entities they are referencing, i.e. their meaning, as well as their purpose in the real world, i.e. their usage. In order to achieve such a comprehensive comparison, we use a semiotic view (theory of signs) on ontologies and define our framework for similarity in three layers, as shown in Figure 1: Data-, Ontology-, and Context Layer. We further enhance these by an additional orthogonal field representing specific domain knowledge. Initial blueprints for such a division in layers can be found in the semiotics (theory of signs) for example in [4], where they are called *symbolic*, *semantic* and *pragmatic* layer, respectively.

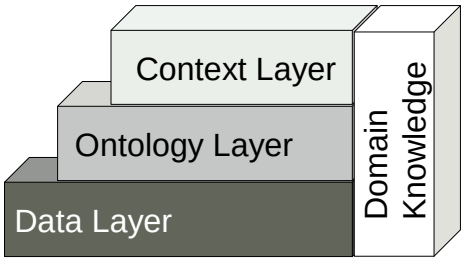

Fig. 1. Layer Model

Data Layer On this first layer we compare entities by only considering data values of simple or complex data types, such as integers and strings. To compare data values, we may use generic similarity functions such as the edit distance for strings. For integers we can simply determine a relative distance between them. The complex data types made up from simple data types would also require more complex measures, but which are effectively completely based on simple measures.

Definition 5 (Similarity based on Data).

$$sim_{data} : (E_i) \times (E_j) \rightarrow [0..1]$$

computes the similarity of entities based on the corresponding data values V_i and V_j occurring in KB_i and KB_j.

Ontology Layer In the second layer, the ontology layer, we consider semantic relations between the entities. In fact, we use the graph structure of the ontology to determine similarity. For specific predefined relations such as taxonomies or restrictions we can use specific heuristics. For example, certain edges could be interpreted as a subsumption hierarchy. It is therefore possible to determine the taxonomic similarity based on the number of *is-a* edges separating two concepts. Besides intensional features we can also rely on the extensional dimension i.e. assess concepts to be the same, if their instances are similar. The similarity measures of the ontology layer can include similarity measures of the data layer to determine the basic similarities.

Definition 6 (Similarity based on Ontology Structures).

$$sim_{ontology} : (E_i) \times (E_j) \rightarrow [0..1]$$

computes the similarity of entities based on the ontological structures of (KB_i, O_i) *and* (KB_j, O_j).

Context Layer On this layer we consider how the entities of the ontology are used in some external context. This implies that we use information external to the ontology itself. Although there are many contexts in which an ontology can be considered (for example the context in which an ontology is developed, or in which it has been changed), from the point of view of determining the similarity, the most important one is the application context, e.g. how an entity of an ontology has been used in the context of a given portal. An example for this is the *amazon.com* portal in which, given information about which people buy which books, we can decide if two books are similar or not in a given context. We assume that an ontology can be used for annotating content/documents in an information portal. Therefore, the similarity between two ontology entities can be easily determined by comparing their usage in an ontology-based application. A naive explanation is that similar entities have similar patterns of usage. However, the main problem is how to define these usage patterns in order to discover the similarity in the most efficient way. In order to generalize the description of such patterns we reuse the similarity principle from CBR (*similar problems have similar solutions*) in the terms of the usage: *similar entities are used in similar context*. We use both directions of the implication in discovering similarity: if two entities are used in the same (related) context then these entities are similar and vice versa: if in two contexts the same (related) entities are used then these contexts are similar.

Definition 7 (Similarity based on Context).

$$sim_{context} : (E_i) \times (E_j) \rightarrow [0..1]$$

computes the similarity of entities based on some context external to the ontologies.

Domain Knowledge Special shared ontology domains e.g. the bibliographic domain, have their own additional vocabulary. The right part of Figure 1 therefore covers domain-specific aspects. As this domain-specific knowledge can be situated at any level of ontological complexity, it is presented as a box across all of them. Just like we use

general similarity features to compare ontologies we can also do so with domain specific features.

Amalgamation For the computation of the overall similarity between two entities we use an amalgamation function that combines the results of the individual similarity functions of the layers described above, i.e.

Definition 8 (Amalgamation of similarity functions).

$$sim(e_a, e_b) = \mathcal{A}(sim_1(e_a, e_b), \cdots, sim_n(e_a, e_b))$$

where sim denotes the overall similarity, and $\mathcal{A}$ the amalgamation function composing individual similarities sim_i ($i \in \{1, \cdots, n\}$).

4 Application Scenario

We have validated our similarity framework in varoius different application scenarios, such as Case-based Reasoning [5] and Usage Mining [6]. To illustrate the application of our similarity framework, we present Bibster, a semantics-based bibliographic Peer-to-Peer system. It can be treated as a Peer-to-Peer-based knowledge management system, since it enables the efficient access to the information stored in a Peer-To-Peer network. Bibster addresses researchers in a community that share bibliographic metadata via a Peer-to-Peer system. Many researchers own hundreds of kilobytes of bibliographic information, in dozens of BibTeX files. At the same time, many researchers are willing to share these resources, provided they do not have to invest work in doing so. Bibster enables the management of bibliographic metadata in a Peer-to-Peer fashion: It allows to import bibliographic metadata, e.g. from BibTeX files, into a local knowledge repository, to share and search the knowledge in the Peer-to-Peer system, as well as to edit and export the bibliographic metadata.

In Bibster[1] we make use of two common ontologies for the representation of bibliographic metadata: The first ontology is the Semantic Web Research Community Ontology (SWRC)[2], which models among others a research community, its researchers, topics, publications, and properties between them [7]. The second ontology is the ACM topic hierarchy[3], according to which our publications are classified.

4.1 Usage of Similarity

Ontology-based similarity measures are used for a variety of functionalities in the Bibster system:

[1] http://bibster.semanticweb.org/
[2] http://www.semanticweb.org/ontologies/swrc-onto-2001-12-11-daml
[3] http://www.acm.org/class/1998/

Duplicate detection Due to the distributed nature and potentially large size of the Peer-to-Peer network, the returned result set for a query might be large and contain duplicate answers. Furthermore, because of the heterogeneous and possibly even contradicting representation, such duplicates are often not exactly identical copies. The ontology based similarity function allows us to effectively determine the similarity between the different answers and to remove apparent duplicate results. Instead of confronting the user with a list of all individual results, we are able to present query results grouped by semantic duplicates.

Peer Selection with Semantic Topologies In the Bibster system, the user can specify the scope of a query: He can either query the local knowledge, direct the query to a selected set of peers, or can query the entire peer network. For the latter option, the scalability of the Peer-to-Peer network is essentially determined by the way how the queries are propagated in the network. Peer-to-Peer networks that broadcast all queries to all peers do not scale – intelligent query routing and network topologies are required to be able to route queries to a relevant subset of peers that are able to answer the queries. In the Bibster system we apply the model of expertise based peer selection as proposed in [8]. Based on this model, peers advertise semantic descriptions of their expertise specified in terms of the ACM topic hierarchy. The knowledge about the expertise of other peers forms a semantic topology, in which peers with a similar expertise are clustered. That means, a semantic link between two peers is established, if their expertise is similar according to the similiarity function. To determine an appropriate set of peers to forward a query to, a matching function determines how closely the semantic content of a query that references an ACM topic matches the expertise of a peer.

Recommendations Bibster features recommender functionality that allow personalized access to the bibliographic metadata available in the Peer-to-Peer network according to the particular needs of the users. In a nutshell, the recommender functions are based on the idea that if a publication is known to be relevant, a similar publication might also be relevant.

In more detail, the recommender functions build upon the semantic representation of the available metadata, including content and usage information: The bibliographic metadata is represented according to the two bibliographic ontologies (SWRC and ACM). These ontological structures are then exploited to help the user formulate semantic queries. Query results again are represented according to the ontology. These semantic representations of the knowledge available on the peers, the user queries and relevant results allow us to directly create a semantic user profile. The semantic similarity function determines how well a publication matches the user profile. Potentially interesting publication are then recommended to the user.

4.2 Methods

We will now describe the specific methods applied in the system grouped by the layers *Data Layer*, *Ontology Layer*, and *Context Layer*. We also show how we exploit *Domain Knowledge* in the individual layers.

Data Layer On this layer we compare the literal values of specific attributes of the publication instances. For example, to detect typical differences in the representation of a publication title for the duplicate detection, we use the *Syntactic Similarity* function and are thus able to handle spelling errors or mismatches in capitalization, which is important for example for duplicate detection:

Syntactic Similarity [9] introduced a measure to compare two strings, the so called edit distance. For our purposes of similarity we rely on the syntactic similarity of [4] which is inverse to the edit distance measure:

$$sim_{syntactic}(v_1, v_2) := max(0, \frac{min(|v_1|, |v_2|) - ed(v_1, v_2)}{min(|v_1|, |v_2|)}) \quad (1)$$

The idea behind this measure is to take two strings and determine how many atomic actions are required to transform one string into the other one. Atomic actions would be addition, deletion, and replacement of characters, but also moving their position.

Further, we are using domain specific background knowledge to define more meaningful similarity measures. For example, for bibliographic metadata we know that attributes such as first and middle names are often abbreviated: In these cases we compare only the characters in front of the abbreviation dot. For other attributes expansion of the abbreviations makes sense before comparing them.

Ontology Layer To compare the classifications of two publications according to the ACM topic hierarchy, we use the *Taxonomic Similarity for Concepts*. We can then build the semantic topology of the Peer-to-Peer network according to the taxonomic similarity of the peers' expertise, i.e. the classified topics of the publications shared by the peer. This is necessary for efficient peer selection and routing of queries.

Taxonomic Similarity One possible generic function to determine the semantic similarity of concepts C in a concept hierarchy – or taxonomy – $\leq_C$ has been presented by Rada et al. in [10]:

$$sim_{taxonomic}(c_1, c_2) := \begin{cases} e^{-\alpha l} \cdot \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}}, & \text{if } c_1 \neq c_2, \\ 1, & \text{otherwise} \end{cases} \quad (2)$$

$\alpha \geq 0$ and $\beta \geq 0$ are parameters scaling the contribution of shortest path length l and depth h in the concept hierarchy, respectively. The shortest path length is a metric for measuring the conceptual distance of c_1 and c_2. The intuition behind using the depth of the direct common subsumer in the calculation is that concepts at upper layers of the concept hierarchy are more general and are semantically less similar than concepts at lower levels. This measure can be easily used analogously for relation R comparisons through $\leq_R$.

SWRC Concept Similarity For our specific scenario with the SWRC ontology as domain ontology, we have further background knowledge that allows us to define a simpler, but also more appropriate similarity function. There are many subconcepts of publications:

articles, books, and technical reports to just name a few. We know that if the type of a publication is not known, it is often provided as Misc (e.g. in Citeseer). We therefore use the following function:

$$S(c_1, c_2) = \begin{cases} 1, & \text{if } c_1 = c_2, \\ 0.75, & \text{if } (c_1 = \text{Misc} \vee c_2 = \text{Misc}) \wedge c_1 \neq c_2 \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

Furthermore, we analyze the graph structure of the metadata, specifically we check how publication instances are structurally linked with person instances, e.g. authors. Thus we can compare two publications on the basis of the similarity of the sets of authors using the function for *Set Similarity*, which is useful for example to detect duplicates or to recommend publications that are similar based on co-authorship:

Set Similarity Often it is necessary to compare not only two entities but two sets of entities. As the individual entities have various and very different features, it is difficult to create a vector representing whole sets of individuals. Therefore we use a technique known from statistics as multidimensional scaling [11]. We describe each entity through a vector representing the similarity to any other entity contained in the two sets. This can easily be done, as we rely on other measures which already did the calculation of similarity values $[0..1]$ between single entities. For both sets a representative vector can now be created by calculating an average vector over all individuals. Finally we determine the cosine between the two set vectors through the scalar product as the similarity value.

$$sim_{set}(E, F) := \frac{\sum_{e \in E} \boldsymbol{e}}{|E|} \cdot \frac{\sum_{f \in F} \boldsymbol{f}}{|F|} \quad (4)$$

with entity set $E = \{e_1, e_2, \ldots\}$, $\boldsymbol{e} = (sim(e, e_1), sim(e, e_2), \ldots sim(e, f_1), \ldots)$; F and f are defined analogously.

Context Layer On the context layer, we exploit information about the usage of the bibliographic metadata. The usage information includes recently relevant results (i.e. publications that have for example been stored by the user into his local knowledge base), and queries the user has performed. As the recently relevant results are bibliographic metadata themselves, they can be directly compared. For the recent queries, the situation is similar: A query can be represented as an underspecified publication with the attribute-value pairs that have been specified in the query (As known from *Query-by-Example*). With the similarity function one can then determine, how closely a publication matches a query, instead of only considering exact matches. The context layer is of big value for the recommender functionality.

Amalgamation Function To combine the local similarities to the global similarity Sim, we use a weighted average by assigning weights w_i to all involved local similarities:

$$\mathcal{A}(sim_1, \ldots, sim_n) = \frac{\sum_{i=1}^{n} w_i \cdot sim_i}{\sum_{i=1}^{n} w_i}$$

The weighted average allows a very flexible definition of what similar means in a certain context. For example, to detect duplicate publications, the similarity based on the title has a high weighted, and the global similarity needs to be close to 1. For the recommendation of potentially relevant publications on the other hand, one might set the weights to consider similarity based on the co-authorship or the topic classification. Additionally, one certainly does not want to recommend duplicate publications.

5 Related Work and Conclusion

5.1 Related Work

Similarity measures for ontological structures have been widely researched, e.g. in cognitive science, databases, software engineering and AI. Though this research covers many areas and application possibilities, most applications have restricted their attention to the determination of the similarity of the lexicon, concepts, and relations within one ontology.

The nearest to our comparison between two ontologies come [12] and [13]. In [12] the attention is restricted to the conceptual comparison level. In contrast to our work the new concept is described in terms of the existing ontology. Furthermore, he does not distinguish relations into taxonomic relations and other ones, thus ignoring the semantics of inheritance. [13] computes description compatibility in order to answer queries that are formulated with a conceptual structure that is different from the one of the information system. In contrast to our approach their measures depend to a very large extent on a shared ontology that mediates between locally extended ontologies. Their algorithm also seems less suited to evaluate similarities of sets of lexical entries, taxonomies, and other relations.

MAFRA[14] describes a framework for mapping ontologies. Detecting similarities among entities constitutes one module in the mapping framework. In this sense, our framework can be seen as a complementary to MAFRA.

Research in the area of database schema integration has been carried out since the beginning of the 1980s. Schema comparison analyzes and compares schema in order to determine correspondences and comes therefore near to our approach. The most relevant to our framework is the classification of schema matching approaches given in [15]. The authors distinguish three levels of abstraction. The highest level differs between schemata- and instance-based information. The second level distinguishes the similarity among elements and among structures. On the third level the calculation can be based on linguistic or information about a model's constraints. On the other hand our approach uses a conceptual decomposition: if the similarity of entities can be discovered on the data representation level (e.g. two strings are similar), then it can be expanded to the semantic level (e.g. if these strings are label for two concepts, then it can be an evidence that the concepts are similar) and finally this information can be propagated on the level of the usage of these concepts (e.g. if they are used similarly, then there is more evidence for their similarity). In that context our framework is more "compact" and goal-oriented, whereas all methods mentioned in [15] can be found in our framework. Moreover, we use background information about the given domain and not only

"auxiliary" linguistic information (like synonyms, hypernyms) in all layers. Further, we base our framework on a formal ontology model, that enables us to define all methods formally. Finally, none of the related approaches, as known to the authors, had the intention to define a framework which will drive the comparison between ontologies, e.g. none of them consider the context layer as a source for discovering similarities.

5.2 Conclusion

In this paper we present a general framework for calculating similarity among ontologies for various application fields. In order to define a general framework we base our work on an abstract ontology model that allows to adhere to various existing and evolving ontology standards. The main characteristics of the framework is its layered structure: we have defined three levels on which the similarity between two entities can be measured: data layer, ontology layer, and context layer, that cope with the data representation, ontology meaning and the usage of these entities, respectively. In that way, different layers consider different aspects of the nature of entities which are combined in the final judging about their similarity. Moreover, in each of the layers corresponding background information (like the list of synonyms of a term) is used in order to define the similarity more efficiently (precisely). Our intention was not only to develop a collection of existing methods for measuring similarity, but rather to define a framework that will enable their efficient systematization regarding the task of comparing ontologies. The framework is complete in the sense of covering the similarity between all elements defined in the abstract ontology model. We developed several new methods in order to achieve such a completeness. Moreover, the framework provides some inter-/intralayer relations between methods, that enables more efficient applications. In a case study regarding searching for bibliographic metadata in a Peer-to-Peer network we showed the advantages of using our approach in knowledge management applictions. Currently, we are evaluating our framework in several new application areas. The future work will be oriented to the more formal treatment of the context layer which will enable the reasoning about the similarity of the contexts as well.

5.3 Acknowledgements

Research reported in this paper has been partially financed by the EU in the IST projects SWAP (IST-2001-34103), SEKT (IST-2003-506826), and Dot.Kom (IST-2001-34038), and the BMBF project SemiPort (08105939). Many thanks also to our colleagues for the fruitful discussions, especially Philipp Cimiano and Thomas Gabel, who helped to create the first draft.

References

1. Gruber, T.R.: Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In Guarino, N., Poli, R., eds.: Formal Ontology in Conceptual Analysis and Knowledge Representation, Deventer, The Netherlands, Kluwer Academic Publishers (1993)

2. Stumme, G., Ehrig, M., Handschuh, S., Hotho, A., Maedche, A., Motik, B., Oberle, D., Schmitz, C., Staab, S., Stojanovic, L., Stojanovic, N., Studer, R., Sure, Y., Volz, R., Zacharias, V.: The Karlsruhe view on ontologies. Technical report, University of Karlsruhe, Institute AIFB (2003)
3. Richter, M.M.: Classification and learning of similarity measures. Technical Report SR-92-18 (1992)
4. Maedche, A., Staab, S.: Measuring similarity between ontologies. In: Proceedings of the European Conference on Knowledge Acquisition and Management (EKAW), Springer LNCS (2002)
5. Hefke, M.: A framework for the successful introduction of km using cbr and semantic web technologies. In: Proceedings of the 4th International Conference on Knowledge Management (I-KNOW'04), Graz, Austria (2004) 731–739
6. N. Stojanovic, L.S., Gonzalez, J.: More efficient searching in a knowledge portal - an approach based on the analysis of users queries. In: PAKM 2002, Springer Verlag (2002)
7. Handschuh, S., Staab, S., Maedche, A.: CREAM - creating relational metadata with a component-based. In: Proceedings of the First International Conference on Knowledge Capture K-CAP 2001. (2001)
8. Haase, P., Siebes, R., Harmelen, F.: Peer selection in peer-to-peer networks with semantic topologies. In: International Conference on Semantics of a Networked World: Semantics for Grid Databases, 2004, Paris. (2004)
9. Levenshtein, I.V.: Binary codes capable of correcting deletions, insertions, and reversals. Cybernetics and Control Theory (1966)
10. Rada, R., Mili, H., Bicknell, E., Blettner, M.: Development and application of a metric on semantic nets. In: IEEE Transactions on Systems, Man and Cybernetics. (1989) 17–30
11. Cox, T., Cox, M.: Multidimensional Scaling. Chapman and Hall (1994)
12. Bisson, M.: Learning in FOL with a similarity measure. In: Proceedings of the Tenth National Conference on Artificial Intelligence. (1992) 8287
13. Weinstein, P., Birmingham, W.P.: Comparing concepts in differentiated ontologies. In: Proceedings of the Twelfth Workshop on Knowledge Acquistion, Modeling and Management (KAW'99), Banff, Alberta, Canada (1999)
14. Maedche, A., Motik, B., Silva, N., Volz, R.: Mafra - a mapping framework for distributed ontologies. In: Proceedings of the EKAW 2002. (2002)
15. Rahm, E., Bernstein, P.: survey of approaches to automatic schema matching. VLDB Journal **10** (2001) 334350

Building an Ontology of Competencies[1]

Vlad Posea[1], Mounira Harzallah[2]

[1] University Politehnica of Bucharest
Vladposea@yahoo.com
[2] 2, rue de la Houssinière BP 92208 44322 Nantes Cedex 03
mounira.harzallah@iut-nantes.univ-nantes.fr

Abstract. A competence ontology seems necessary to share the human capital across organization units or corporations. This paper presents a method for building an ontology of competencies. This method uses a model for competence representation, ontology reasoning and semantic distances. It allows to define the similarity of competencies in order to share them and to simplify their management. A brief synthesis on ontology characteristics and building are presented on this paper. Then, a method for building a competence ontology is proposed, its application is illustrated by a case study.

1 Introduction

Competence management is becoming more than ever a key success factor in many organizations to make a better appraisal of human capital [1], [2]. However, competence management is complex, considering the numerous abilities and individuals to manage in a company. In addition, it includes several heavy processes, as for instance: the identification of domain required competencies, the evaluation of individuals for their acquired competencies, the study of acquired and required competency adequacy. These processes are usually realised by several organisation unit mangers. Each one defines competence lists with his method and gives them his proper semantic. The same case for a corporate network, each company defines its list of competencies in a proper way. In this situation, the share of competencies is difficult. The interoperability of these different lists can be possible using an ontology-based approach. A competence ontology offers multiple functionalities like the unification of competence meaning, the share of competence list across business units or organisations, the integration of different competence lists.
The difficulty in building the competence ontology is that a competence exists only by its direct link with one or more concepts of a domain. In the competence ontology there is also the need to find similarities between concepts in order to find those are

[1] [1]This work is partially supported by the Commission of the European Communities under the Sixth Framework Programme (INTEROP Network of Excellence, Contract N° 508011, <http://www.interop-noe.org>).

close enough for their competencies to be close too. This allows then to define the competence similarity.

To develop a competence ontology, our approach is based on the ontology of the domain. Once this ontology is built, each organisation unit defines its competencies according to it (Fig. 1). Then, competencies have the same semantic in all organisation units, they can be shared.

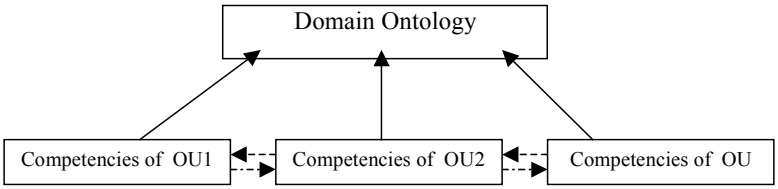

Fig. 1. Interoperability of competencies of OUs based on the domain ontology

This paper is focused on the definition of a domain ontology for the comptence definition. After a brief state of the art about ontology and competence modelling, a competence ontology buiding method is proposed. Then, a case study illustrates the application of this method on the computer sciences domain.

2 Ontology, Reasoning and Semantic distance

We have studied the possibility to use an ontology in order to represent competencies and their relationships. We were interested in the definition of the concept of ontology, their usage, what they can offer us. We were also interested in the existing methodologies for building an ontology.

The best known definition of the concept of ontology is Gruber's "an ontology is an explicit specification of a shared conceptualization"[3]. This is also the definition that unites the most important ideas that can be found in other definitions [4], [5], [6], [7].

Ontology has the following characteristics: it describes a domain, it is made of concepts and relationships between them, it is build for a specific purpose, it represents the point of view of a community and it is usually formal.

The advantages of an ontology are, among others, to share common understanding of the structure of information among people or software agents, to enable reuse of domain knowledge, to analyze domain knowledge [7].

There are many methods to build an ontology. They include especially the following steps: the definition of the goals of the ontology, the knowledge acquisition it building, the informal specification of it, its formalization, its implementation and the evaluation and validation of the results.

About reasoning on ontology, two methods are distinguished: internal reasoning and external reasoning. Internal reasoning concerns automatic validation (detection of inconsistencies) of concepts and automatic classification of concepts based on theirs

characteristics. The external reasoning involves obtaining results by reasoning using the ontology. In this domains, there are used also the measures of semantic similarity.

Several semantic distances were used in ontology to measure the similarity between concepts [8], [9], [10], [11], [12]. They are based on measuring the number of arcs between concepts in the ontology graph. Some of them are taking also into consideration the type of relations between concepts, the distances to the closest common ancestor or the number of relations attached to a concept.

3 Competence Modelling

A competence is a way to put in practice some knowledge, know-how and also attitudes (i.e., resources of the competence) inside a specific context. For example, the competence "To be competent in the improvement project in a company" can demand the following resources: "To know the characteristics of an improvement project", "To know the objectives of an improvement project", "To know how to define improving solutions for an improvement project", "To know how to define a planning of an improvement project", "To know how to solve a problem", "To know how to lead a meeting".

We have considered the competence concept through the CRAI model (Competency, Resource, Aspect and Individual) (Fig. 2) [13], [14]. This model represents the essential characteristics of this concept:

1. A competence concerns an aspect of the domain studied;
2. A competence is a set of resources (knowledge, know-how and behavior);
3. A resource is related to an aspect;

In the CRAI model, two entities are specific to the competence domain: Competency (for instance, "to be competent for machine X") and C-Resource for competence resources (for instance, "to know how to remove components on machine X"). C-resource can be understood as basic knowledge, or know-how or behavior concerning a specific enterprise aspect and that can be used for precisely identifying and understanding what a competency is. Another entity, Individual, represents the personnel set of the enterprise. The entity Aspect represents the contextual information, i.e., the enterprise components and feature comprising several additional concepts, especially business processes, organizational aspects, economic aspects, information aspects, etc., as developed in the enterprise modeling field. In the CRAI model, it is possible to link a resource to an occurrence of the Competency entity by means of one of the specific relationships named To-Know, To-Know-how, and To-Behave.

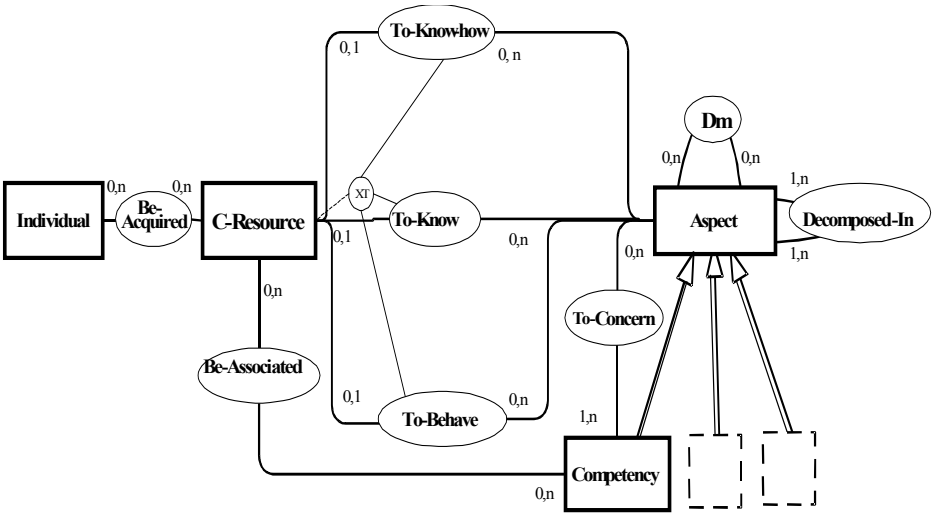

Fig. 2. The Entity-Relationship schema of the model CRAI

4 Competence ontology building

We have established the importance of competence management and also the role of the ontology in the process of sharing and reusing knowledge.

To build a competence ontology, we have to define the ontology's concepts and relationships. The concepts of this ontology (the competencies) can be described only through the concepts that are referring and through the resources that are linked to these concepts. The relationships between competencies are different than the relationships between concepts of a domain (is-a, part-of) but they still however depend on them.

For these reasons, in order to build a competence ontology, we consider that we have to base it on a domain ontology.

4.1 Domain ontology

The domain ontology that is required for building a competence ontology is a little different that a usual domain ontology. The main difference is that this ontology needs to have some resources defined for the concepts of the domain. This characteristic is not usually needed but because a competence is defined by its set of resources we have to add that data in the domain ontology.

The domain ontology is also composed by concepts and relationships. The relationships that we considered are "is-a" and "part-of. The concepts of the domain ontology are the elements of the domain. These elements have as attributes the competency resources that are attached to them. For example, the concept

Unix can have attached a resource ("know-how") called "To know how to install Unix", and Oracle has resources like "To know the components of Oracle" ("knowledge") and "To know how to start the database server" ("know-how"). We have also considered the fact that all the resources attached to a concept cannot be defined manually because there is a large quantity of resources and also a very large quantity of concepts. That is why a method of automatically deducting the missing resources is proposed. This method is based on the concepts of the domain ontology, on the relationships between them and on a measure of semantic relatedness between these concepts

The idea of this method is that a concept is likely to "have" the resources attached to its very close neighbors. To find out which are its close neighbors, similarity between concepts is measured using a measure based on Sussna's measure of semantic distance [10]. The difference between our formula and Sussna's is that our measure doesn't need to be symmetrical. The measure does not need to be symmetrical because for example if we have the concepts "Operating System" and "Unix" than "having the resource "to know how to install an operating system" almost surely can be inherited by Unix but not the other way around. This also means that the relations in the domain ontology are not symmetrical. So if we consider the types of relations "is-a" and "part-of" then we have four different relations: "Father-Son", "Son-Father", "Whole-Part", "Part-Whole".

The Sussna's measure implies establishing a coefficient for every relationship. That is why it is very important to define relationships used and to observe that they aren't symmetrical.

Our measure of the similarity between two concepts c1 and c2 is:

$$dist_S(c_1, c_2) = \frac{w(c_1 \rightarrow_r c_2)}{d}$$

where $w(c_1 \rightarrow_r c_2) = \max_r - \frac{\max_r - \min_r}{nr(c_1)}$, nr(c1) is the number of concepts to which c1 is related, maxr and minr are coefficients that depend of the type of relationships between c1 and c2 and d is the height of the tree.

So, the particularities of the ontology concepts, the relationships between them are defined.

4.2 The Competence Ontology

The competence ontology is composed, like the domain ontology, by concepts and relations between them. The concepts of this ontology are the competencies. The competence is linked directly to a concept of the domain ontology (as in the CRAI model) and is composed by resources that are directly linked to the concept of the domain ontology or by resources that are close to this concept. The distance between concepts is measured the same way as in the previous section.

The relationships of the competence ontology are based on the sets of resources of each competence, and especially on the inclusion of these sets. To express these relationships, axioms are defined as follows:

Co1, Co2 are competencies and R11, …, R1N are resources of Co1 and R21, …, R2M are resources of C2. Then E(Co1)={ R11, …, R1N } and E(Co2)={R21, …, R2M }. E(Coi) is the set of resources that belongs to the competence Coi.

Axiom 1: if $E(Co1) \subset E(Co2) \Rightarrow$ Co2 demands Co1

This means that if Co2 is acquired than Co1 must be already acquired too or if we must acquire Co2 than we must before acquire Co1. If Co1 is acquired only the difference between Co1 and Co2 must still be acquired.

Axiom 2: If $E(Co3) \not\subset E(Co1) \wedge E(Co3) \not\subset E(Co2) \wedge E(Co3) \subset (E(Co1) \cup E(Co2)) \Rightarrow$ Co1 and Co2 demand Co3 (composed demand).

This means that a competence can be demanded by two other competencies even if it is not individually demanded by any of the other two.

Axiom 3: if $E(Co1) \subset E(Co2) \wedge E(Co2) \subset E(Co1) \Rightarrow$ Co2 is identical with Co1

This axiom helps identify the competencies that have different names but the same definition.

Axiom 4: $E(Co1) \cap E(Co2) = \varnothing \Rightarrow$ Co2 and Co2 are disjoined

This axiom tells us when two competencies are independent.

These rules are 100% certain. Other rules and relations can be obtained my measuring the similarity between the sets of resources. By applying these axioms to the identified competencies the relations between them are defined.

5 Case study

To experiment our method for building a competence ontology, an application was developed in cooperation with Cap Gemini Ernst &Young (Nantes/France). The needs of Cap Gemini were to obtain a tool that could help them to measure differences of competencies between jobs, between employees and between technologies in order to propose development strategies on training, station change or reorganization for the enterprise that they were advising. A particular need was, for example, to change the IT structure of a big enterprise without affecting the personnel. In order to do that, they had to train the personnel in the new technologies. By defining the competencies acquired by the employees for the old technology and the competencies required for the new one and by finding the relations between them they could find out easier the training requirements.

In order to do that a tool for the building of the competence ontology was developed. This tool is a plug-in for the Stanford's Protégé software. This plug-in requires an ontology of a domain to be built in Protégé [15]. To some of concepts of this ontology knowledge and know-how facts must be attached.

The software ontology that was built is based on documents from Cap Gemini, on other ontologies like CALO [16], and on classifications on the web (especially download sites like download.com, tucows.com, softpedia.com). The ontology was built having as top concepts "System" with the sons "Operating System" "Database System" and "Office System" and "Software program". Between these concepts there are the relations previously defined ("Father-Son", "Son-Father",

"Whole-Part", "Part-Whole"). The instances of these concepts are, among others, the operating systems (Unix, Windows) and the database systems (Oracle, MS Sql).

To these concepts and to these instances there are attached resources (know-how and knowledge).

The plug-in allows to identify other resources (knowledge and know-how) besides the ones already defined, based on the semantic distance that is already defined. Fig. 3 shows an example of how to obtain the resources of the concept Calc supposing that just two resources were attached directly to it: R1 (to know how to use tables) and R2 (to know the Calc components). The distance between Calc and its neighbors is measured. A threshold is chosen and the resources of all the concepts closer than this threshold are gathered in the set of resources of Calc. The following resources are added: R3 (to know how to use formulas in spreadsheet applications), R4 (to know how to use the Open Office interface) and R6 (to know the Excel commands).

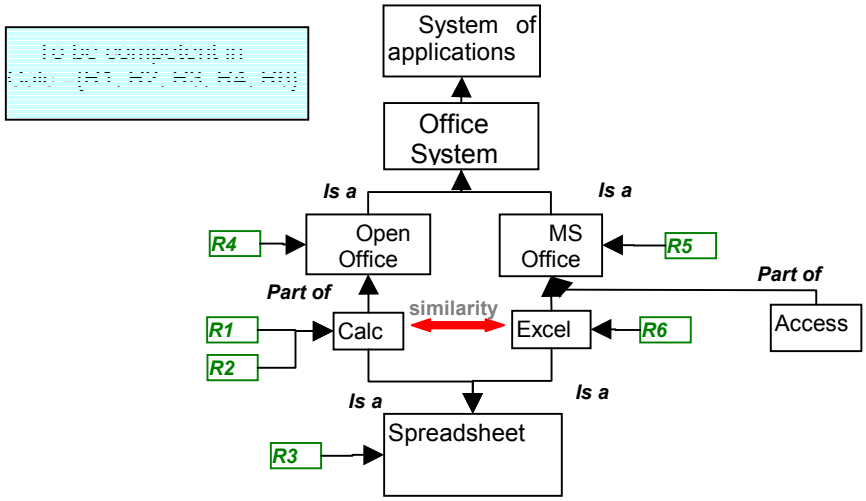

Fig. 3. Excerpt from a competence ontology of computer sciences

The resources identified at this step need to be validated by an expert because at this level many error sources appear. The error sources are given by the difficulty of deciding the granularity of the domain ontology and by the fact that we apply the same way to measure the distance on two resources that are attached to the same concept without taking in consideration their particularity.

The following step defines the competence as set of resources attached to a concept. For example: the competence "to be competent in Calc" has the set of resources belonging to Calc: {R1, R2, R3, R4, R6}. The similarity measure can show that just the similarities between Calc and Open Office, Calc and Excel, Calc and Spreadsheet are over the defined threshold. In this case the "to be competent in Calc" competence gathers the resources from Open Office, Excel and

Spreadsheet and not from MS Office, for example. That's how the competence "to be competent in Calc" is defined.

Fig. 4 shows the competence "To be competent in Unix" as it is obtained in the plug-in.

The competence has attached resources that were directly attached to the concept "Unix" and also other resources that were obtained using the similarity measure like "To know Operating Systems" and "To know how to use the Unix Terminal".

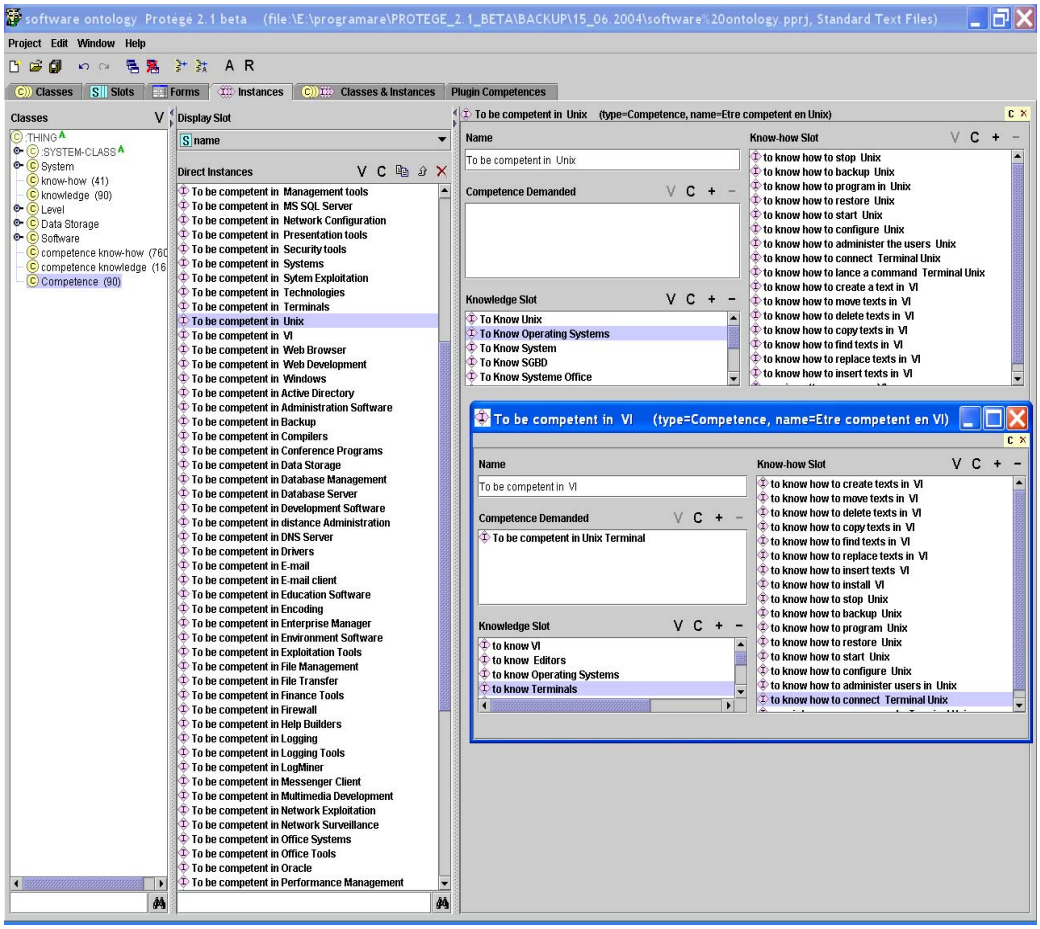

Fig. 4. Defining competencies as set of resources in the Protégé plug-in

The last step is to identify relationships between competencies based on theirs sets of resources according to the axioms defined in section 4. For example, a relation obtained from the first axiom is displayed in Fig. 5. This example shows that to be competent in mail tools you have to be competent in mail clients and in mail servers. This relation is obtained by comparing the sets of resources attached to these compe-

tences and obtaining that the set of resources belonging to “To be competent in e-mail client” and “To be competent in e-mail server” are included in the set of resources of “To be competent in e-mail tools”.

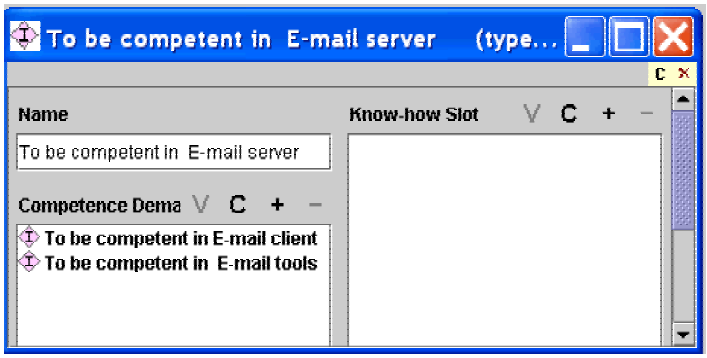

Fig. 5. Relation between competencies

6 Conclusion

In the context of building a system for improving the competence and knowledge management, we are interested in a method for a unified competence representing. Ontology is considered to be appropriate for defining competencies, identifying new competencies rules between them, and allowing their share.

The proposed method is based on the domain ontology. Comptencies are defined as sets of resources attached to ontology concepts. This idea is based on the relationships existing in the CRAI model.

In order to experiment this method, an ontology of computer science domain is developed and also a plug-in for Protégé that identified the competencies of the domain and some relationships between them.

The improvements that can be brought to this method are the finding of a better measure of the semantic distance. This better measure must include more types of relationships between concepts. It also must consider the particularities of a resource like the difficulty of acquisition. These improvements will make easier the expert’s task of validating the resources that were automatically identified.

This project will be integrated with another project that will discover by inferences other rules and relations between the competencies. The inferences will be made on the competencies defined in this part of the project. The project obtained by the integration of these two parts will be an important part of the system for the management of competencies and knowledge developed in the Inf3C project.

References

1. Dubois D., Competency-based performance improvement: a strategy for organizations. Amherst: HRD Press Inc. International society for Performance Improvement, Washington, (1993).
2. Prahalad C.K. and G. Hamel, Competing for the future, Boston: Harvard Business School Press, (1994).
3. Gruber, T. R. A Translation Approach to Portable Ontology Specifications (1993).
4. Sowa J.F. http://www.jfsowa.com/ontology/, (2001).
5. Guarino, N. and Giaretta, P. 1995. Ontologies and Knowledge Bases: Towards a Terminological Clarification. In N. Mars (ed.) Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing, IOS Press, Amsterdam: 25-32. (1995)
6. Uschold, M. "Building Ontologies: Towards A Unified Methodology", Proc. Expert Systems 96, Cambridge, December 16-18th, (1996).
7. Noy N., D. McGuinness - Ontology Development 101: A Guide to Creating Your First Ontology http://protege.stanford.edu/publications/ontology development/ontology101-noy-mcguinness.html, (2004).
8. Rada R., H. Mili, E. Bicknell and M. Blettner. Development and application of a metric on semantic nets. IEEE Transactions on Systems, Man, and Cybernetics, (1989).
9. Hirst G., and D. St-Onge. Lexical chains as représentations of context for the detection and correction of malapropisms. In Christiane Fellbaum, editor, WordNet: An Electronic Lexical Database, chapter 13, pages 305 - 332. The MIT Press, Cambridge, MA, (1998).
10. Sussna M. J. Text Retrieval Using Inference in Semantic, Metanetworks. PhD thesis, University of California, San Diego, (1997).
11. Wu Z. and M. Palmer. Verb semantics and lexical selection. In Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics, pages 133-138, Las Cruces, New Mexico, June (1994).
12. Zargayouna H., S. Salotti. Mesure de similarité dans une ontologie pour l'indexation sémantique de documents XML dans Actes de la conférence Ingénierie des Connaissances, IC'2004, Mai (2004).
13. Harzallah M. and F. Vernadat, IT-Based competency modeling and management: from theory to practice in enterprise engineering and operations, Computers in Industry, 48 (2): 157-179, (2002).
14. Harzallah, M. and Berio, G. Competency Modeling and management: A case study, The 6th international conference on Enterprise Information Systems (ICEIS'04), University of Portucalense, pp. 350-358, Porto, April 13-16, (2004).
15. Protégé web site – http://protege.stanford.edu, (2004).
16. CALO Computer Ontology Texas University - http://www.cs.utexas.edu/users/mfkb/RKF/tree/specs/ontologies/Computer-onto.html (2003)

Achieving Enterprise Application Interoperability: Design Patterns and Directives

Yannis Charalabidis[1], David Chen[2]

[1]Singular Software SA, 23rd KM Athens-Lamia Road, Ag. Stefanos, 14565, Greece,
yannisx@singular.gr

[2]University Bordeaux 1, 351 Cours de la liberation, 33405 Talence cedex, FRANCE
chen@lap.u-bordeaux1.fr

Abstract. The paper elaborates on the various approaches to achieve Enterprise Application Interoperability (EAIo), where Enterprise Applications such as SCM, ERP and CRM systems are concerned. These approaches have been formulated within the INTEROP Network of Excellence, along the evolutions in the areas of Enterprise Modelling, Ontologies and Model-Driven Architecture. After looking at the general causes and remedies for the present interoperability lack, a detailed analysis of the various processes, standards, and solutions has been performed leading to the formulation of an Enterprise Application Interoperability Maturity Matrix (EAIoMM). Within this modeling context, levels of interoperability of existing applications are clearly defined, while certain solutions, practices or directions lead to state changes within the maturity matrix. In parallel, interoperability-oriented high level design directives are providing the means for guiding the development of new, more interoperable Enterprise Applications.

1. Introduction

This work is partly carried out within the frame of WP6 (Design Principles for Interoperability) under the umbrella of INTEROP Network of Excellence (NoE)[1]. INTEROP NoE aims at integrating expertise in relevant domains for the sustainable structuring of European Research on Interoperability of Enterprise Applications [1]. Research in the interoperability domain in Europe is badly structured, fragmented and sometimes overlapping. There is no global vision on the consistency due to the sporadic co-ordination between research centres, laboratories, industries or other bodies. This situation is not only true for research, but also in the training and education areas. Consequently, the primary goal of INTEROP is the emergence of a lasting European research community on interoperability of enterprise software applications. The overall NoE activities are structured in 4 areas: (i) Integrating Activities (IA), (ii) Joint Research Activities (JRA), (iii) Spreading Excellence Activities (SEA), and (iv) Management Activities (MA). For each activity area, several workpackages are defined and launched. The work presented in this paper fits within the frame of Joint Research Activities (JRA) and is carried out by Workpackage 6 - Design Principles and Patterns for Interoperability.

[1] INTEROP (Interoperability Research for Networked Enterprises Applications and Software) NoE is partly financed by E.C. under FP6 framework, Contract n° IST-508011

INTEROP WP6 research activity focuses on the design stage of the system lifecycle. It is concerned with the development of principles and patterns that one can use at design stage to make appropriate design decisions so that the applications so designed are more easily interoperable than if these principles/patterns were not used. A design principle is a rule to follow to orient design decision-making. A pattern is a proven solution to a problem in a context. When following some principles to design; one finds that one repeats the same structures over and over again. These repeating structures of design are known as design patterns.

There are basically two complementary approaches to deal with interoperability problems: (a) Develop reference architectures, models, methodologies, tools to help establishing interoperability between existing heterogeneous enterprise applications (b) Build interoperability capability inside applications at the design stage. Most of the on-going research focuses at the first approach, (see also IST Integrated Project ATHENA [2]). The present paper is analysing given solutions and results of the first approach in order to build structures and knowledge within the second approach. Problem statement considers that existing software and applications were not designed to interoperate. However, hypothesis can be formulated that the interoperability of enterprise applications can be designed-in. It has been considered that some existing applications are more interoperable than others and these applications may have common features. Consequently, this research activity aims to identify generic features, define generic principles and patterns for designing interoperable applications, as design principles and patterns are independent of technologies and methodologies so applying across industry.

The objective of the paper is to address the use of some patterns for developing better interoperable enterprise applications. The paper will discuss certain patterns which appear in various levels of the modeling representations, both in the problem and in the solution space, formulating an initial, structured set of concepts: interoperability types and levels, models for representing given solutions, as well as ontologies for patterns and principles for achieving interoperability. After having given the basic concepts in section 2, section 3 will present the context, problem and patterns to use in the context to solve various problem situations. Section 4 concludes the paper outlining strengths of the proposed approach, open issues and future work.

2. Concepts and definitions

Enterprise applications and relevant business ICT systems are highly complex and difficult to analyse, due to their ever-extending functionality, covering more and more business processes. In parallel, systems and enterprises interoperability is also affected by numerous factors and becomes extremely demanding to measure if not in a structured way. In order to provide a proper ontological framework for analyzing and categorizing systems according to their interoperability level, in extending recent research on the ontological representation of interoperability issues [5], the following definitions are made:

The **Enterprise Applications Interoperability Maturity Matrix (EAIo-MM)** is a set of levels L = (L_1, L_2, L_3, L_4, L_5), where each level corresponds to a different interoperability level for a set of Enterprise Applications. Initially five levels are defined, as following:

- Level 1: Foreign (no interoperability)
- Level 2 : Known
- Level 3 : Connected
- Level 4 : Collaborating
- Level 5 : Integrated (adequate interoperability, for Enterprise Applications)

It has to be noted that the absolute number of Levels may be extended, in order to increase the granularity of the matrix.

Enterprise Applications Interoperability Maturity Matrix	**Foreign**	**Known**	**Connected**	**Collaborating**	**Integrated**
	Level_1 (0-2)	Level_2 (3-4)	Level_3 (5-6)	Level_4 (7-8)	Level_5 (9+)
Processes & Business					
A1. Business model exists					*
A2. Processes match				*	*
A3. Applications match		*	*	*	*
Applications					
A4. Data formats match		*	*	*	*
A5. A2A control established			*	*	*
A6. UI matches	*			*	
Platforms					
A7. Input/output exists		*	*	*	*
A8. Connectivity exists			*	*	*
A9. Overall architecture sound				*	*
Non-functional aspects					
A10. Performance adequate			*		*
A11. Documentation adequate			*		*
A12. Security adequate					*
EXAMPLES	Monolithic applications – same UI	Typical data-centric, batch process, ASCII input/output	Daemon-controlled ASCII input/output, well documented and operational	Service-based Enterprise Application Integration – low standardisation	Internet based A2A integration, ebXML based, productised and COTS

Figure 1 : EAIo-MM Levels and attributes with examples

The categorization of a specific system in a specific EAIo-MM Level follows scoring in a vector of **Interoperability Attributes**, that is a set of 12 attributes analysed as following:

- *Processes and Business* (characteristics concerning the interoperability at process and business level, such as the process matching between the analysed enterprises)
- *Applications* (characteristics concerning information, control and user interface)
- *Platforms* (attributes concerning connectivity, openness and input/output features)
- *Non-functional Aspects* (such as security, performance or documentation)

The categorization of a system according to the EAIo-MM is based on the number of the 12 attributes that are valid or not, thus allowing for different configurations belonging to the same level. As shown in Figure 1, scoring in the various attributes with yes/no creates a set of attributes that corresponds to a specific Level, also indicating a possible instantiation of example configurations for each level.

Extending the definitions concerning the interoperability status of enterprise applications, the following structures are defined:

A **Situation** (or state) of a system, concerning its interoperability level, is a specific vector $S_n=(A_1, A_2, \ldots, A_{12})$, that is denoting the exact attributes that are present or not within the analysed system.

A **Direction** (or state transition) is the change needed so that a system moves from a Situation to another: D_{12} depicts the change from Situation S_1 to Situation S_2.

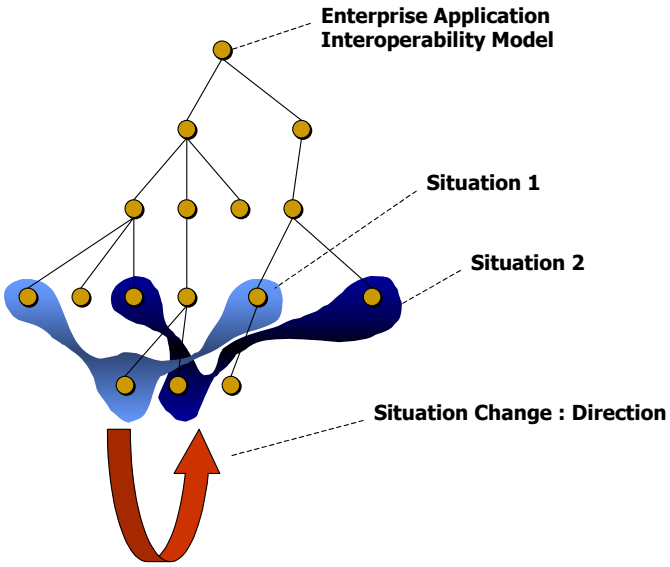

Figure 2 : Situations and Directions of interoperable enterprise systems

The change from a Situation to another can usually be achieved by one ore more **Paths**, that is specific directives or principles aiming at changing certain attributes of the systems.

This notation allows the proper definition of patterns and principles for both the interoperability problem solving approaches presented in Chapter 1:

- The various "retrofit" solutions, patterns and principles, aiming at increasing the interoperability level of existing systems are represented as Directions or Paths between certain, defined Situations.
- Patterns and principles aiming at the design of new, interoperable systems are represented as Directions or Paths leading directly to a specific Interoperability Level or Situation.

Figure 3, that is based on the Levels and specific Situations of Figure 1, illustrates the following case:

- Two specific situations and have been identified, pertaining to Levels 2 and 3 (Known and Connected), both with specific vectors S1 and S2, respectively. S1 corresponds to a set of enterprise applications with a typical data-centric, batch process, ASCII input/output interconnection. S2 corresponds to a daemon-controlled, ASCII input/output interconnection that is well documented and provides adequate performance.
- The change from S1 to S2 (Direction D12) can be achieved by following more than one specific methodological patterns, that is Paths P121, P122, P123. In the example these Paths correlate to Process-driven, Data-driven or Connectivity-driven approaches for enhancing interoperability of the system towards Situation S2.

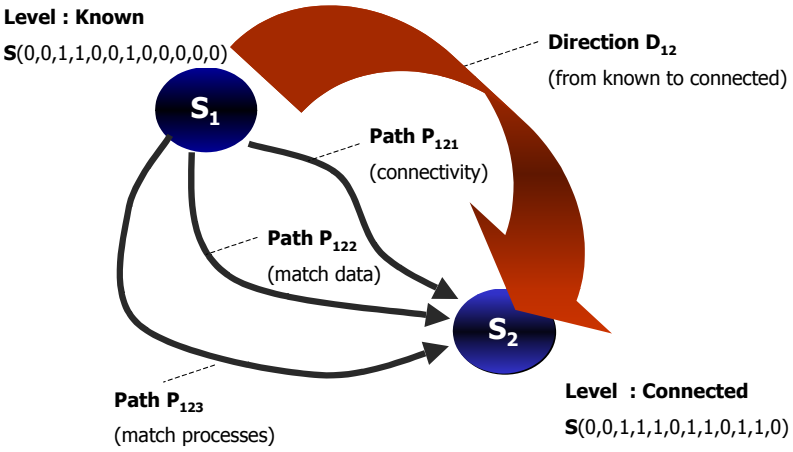

Figure 3 : Situations, Directions and Paths in enterprise application interoperability

3. Directions and Paths for Interoperability

Both within INTEROP Network of Excellence, as well as from current State-of-the-Art in Enterprise Applications interoperability research [2,3,4,5], various approaches have been collected and analysed, towards specifying their exact positioning in the interoperability problem or solution ontologies. Following the concepts of Chapter 2, a set of Situations have been constructed that relate to real-life systems configurations. Figure 4 depicts the EAIo-MM levels and stores relevant Situations both for Systems as well as for Enterprises – the latter having a different attribute set that is outside the scope of the present paper.

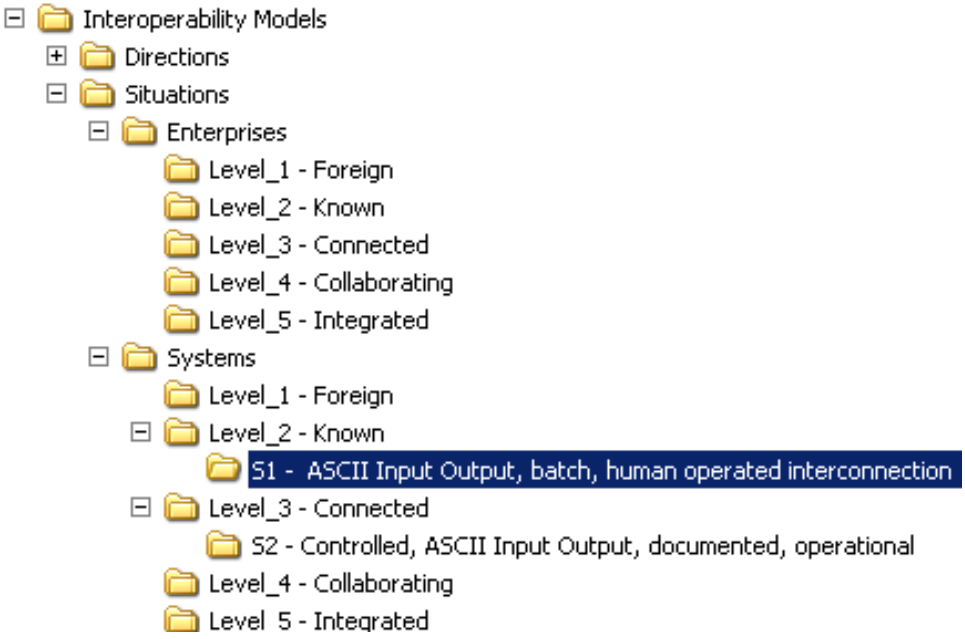

Figure 4 : EAIo-MM Levels and Situations for Systems and Enterprises

In a similar way, a variety of approaches have been analysed, belonging to different systems states or information technology methodologies and techniques.

This Directions ontology (that is storing various approaches categorized as Solutions, Practices, Patterns and Principles going from specific to generic), is constructed with the following containers:

A. Directions affecting Interoperability between Enterprises

These directions are typically related to making the enterprises themselves more interoperable, through motivating, organizing and guiding their collaboration. Studied approaches include Enterprise Modelling (EM) methodologies and tools, aiming at representing and integrating organizational structures or processes, yielding the following examples [3],[4]:

- Holonic Representation of Organisations
- Supply Chain ICT Maturity Matrix (SCIMM)
- M2EE Architecture Description Framework
- MACCIS Architecture Description Framework
- ArchiMate Methodology / Tool
- IEM Methodology / MO^{2}GO Tool

It has to be noted that various approaches have also been presented during the last years for increasing interoperability between enterprises with the use of interoperability-aware modeling methodologies, such as the following:

- UEML, Unified Enterprise Modeling Language [11]
- GRAI Enterprise Modeling Methodology [12]

B. Directions affecting Interoperability between Systems

These directions govern the state change of a system between an interoperability Situation Sn to a Situation Sj. The applied, specific examples, or Solutions, are as following:

- Data-oriented approaches, such as the Services-to-Businesses approach for public administration
- Connectivity-oriented approaches for integrating information flows between ERP and CRM applications, [6]
- Internet-oriented approaches, such as interconnecting ERP systems over the Internet through using B2B/XML standardized data flow, [8]
- Application of M2EE architecture description framework [9]
- Application of TEMoD architecture integration framework [10]
- Process-oriented approaches, such as the alignment of business processes between heterogeneous enterprise applications

Extracted from approaches as the above or transferred from existing practice, various directives can be classified as Practices or Patterns, helpful at the development of new Enterprise Applications and interoperable systems including:

- Architecture-related practices (such system coupling / decoupling modes, system granularity, system connectivity standards)
- Modelling-related practices (model abstraction layers, model transformations and patterns, etc.)
- Ontology-related practices and patterns (data dictionaries, ontological schemas, etc)

- Software engineering practices leading to increased interoperability (encapsulation rules, object coupling, etc)

As illustrated in Figure 5, both the enterprise-oriented and the systems-oriented Solutions, Practices, Patterns and Principles co-exist in the common tree of Directions, leading from a Situation to another.

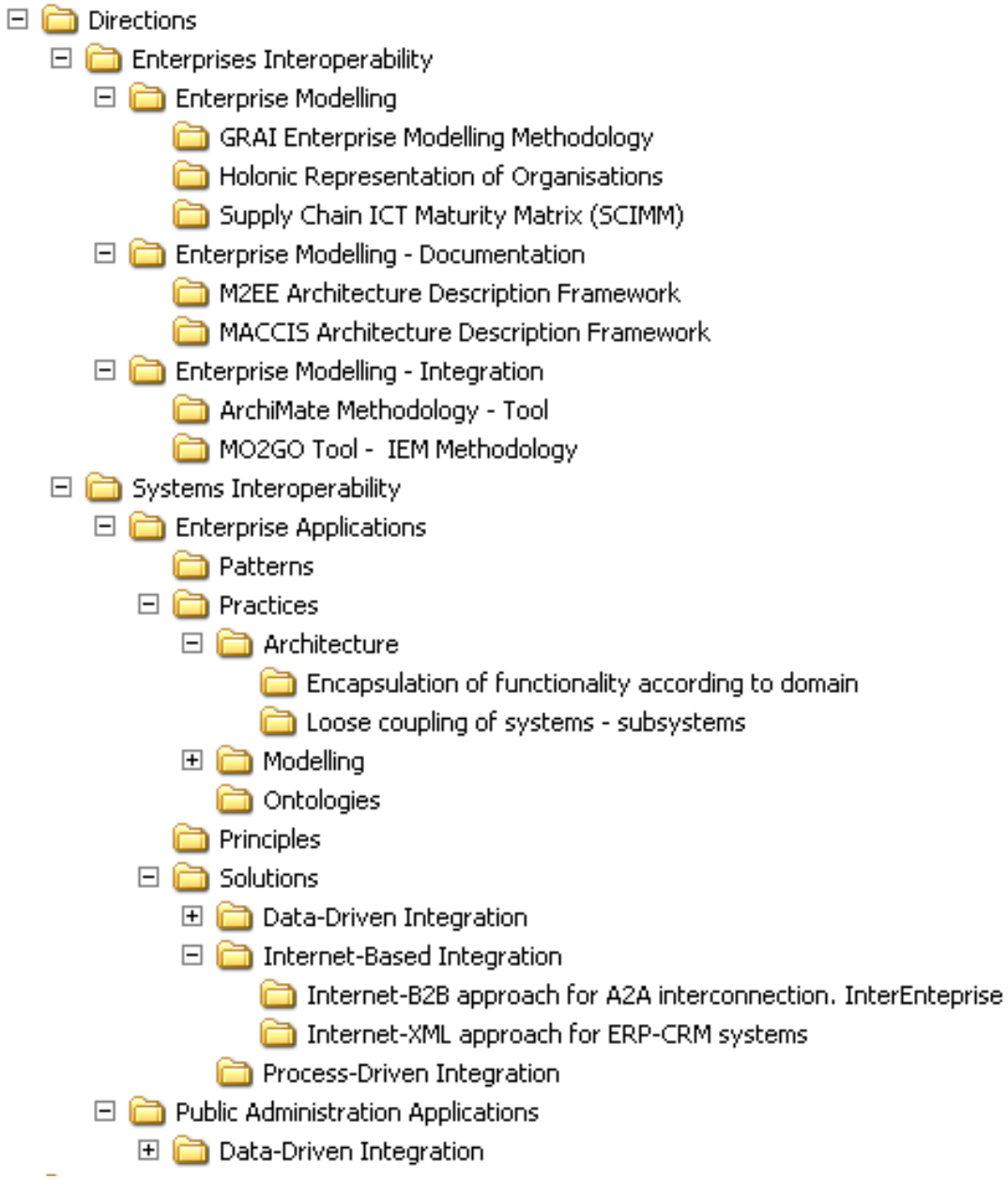

Figure 5 : Classification of Directions regarding enterprises and systems interoperability

4. Conclusions & Discussion

The present work has introduced an ontology framework for connecting the interoperability levels of a system or an enterprise with the performed or needed actions that increase the interoperability between enterprises or enterprise applications. This extendible framework is capable of capturing the internal characteristics of each interoperability situation that an enterprise may adhere to at a certain time point, together with the directions that may be followed in order to achieve a better level. Also, design

directives for the development of new enterprise applications are mapped, so that decisions can be made for their adoption.

The initial population of the above modeling structure showed the difficulty in collecting, analyzing and finally extracting the needed core descriptions from various existing approaches, due to the complexity of the provided solutions and the convolution of various patterns in each real-life scenario. Future relevant research topics include:

- The mostly accurate definition of Interoperability Attributes of existing and new systems, that will lead to the more analytical structure of Levels and Situations of the Enterprise Applications Interoperability Maturity Matrix (EAIo-MM).
- The application of a novel approach for analyzing interoperability solutions and patterns, so that directives and principles are more easily de-convoluted, categorized and presented.

5. References

[1] INTEROP (2004), Interoperability Research for Networked Enterprises Applications and Software, IST Network of Excellence, www.interop-noe.org

[2] ATHENA (2004), Advanced Technologies for interoperability of Heterogeneous Enterprise Networks and their Applications, IST Integrated Project, www.athena-ip.org

[3] Interoperability Development for Enterprise Application and Software (IDEAS, IST project) (2002). http://www.ideas-roadmap.net

[4] INTEROP Network of Excellence, Deliverable 6.1: "Good Practices And Solutions For Interoperability", ed. Chen David, 2004, www.interop-noe.org

[5] INTEROP Network of Excellence, Deliverable 6.2 : "Design Principles / Patterns for Interoperability", 2004, ed. Chen David, www.interop-noe.org

[6] Charalabidis Y., Pantelopoulos S., Koussos Y.: "Enabling Interoperability of Transactional Enterprise Applications", Workshop on Interoperability of Enterprise Systems, 18th European Conference on Object-Oriented Programming (ECOOP), Oslo, 14–18 June 2004

[7] Osterwalder, C. Parent, Y. Pigneur: "Setting up an ontology of business models", Workshop on Enterprise Modeling and Ontologies for Interoperability, 16th International Conference on Advanced Information Systems Engineering (CAISE), Riga, 2004

[8] Charalabidis Y., Karakoidas V., Theotokis S., Spinelis S.: "Enabling B2B Transactions over the Internet through Application Interconnection: The PRAXIS Project", e-Challenges Conference, European Commission, Vienna, 27 - 29 October 2004

[9] Aagedal J. Ø., Berre A.-J., Bjanger B. W., Neple T., Roark C. B., "ODP-based Improvements of C4ISR-AF", Proceedings of the 1999 Command and Control Research and Technology Symposium, 1999.

[10] Di Leva, D. Occhetti, C. Reyneri: "The PRADIGMA Project: and Ontology-based Approach for Cooperative Work in Medical Domain", Workshop on Enterprise Modeling and Ontologies for Interoperability, 16th International Conference on Advanced Information Systems Engineering (CAISE), Riga, 2004

[11] M. Petit, P. Heymans: "Perspectives on the scope and definition process of the Unified Enterprise Modeling Language", Workshop on Enterprise Modeling and Ontologies for Interoperability, 16th International Conference on Advanced Information Systems Engineering (CAISE), Riga, 2004

[12] GRAI Enterprise Modeling Methodology (2004), Graisoft company, www.graisoft.com

Defining Model Transformations for Business Process Models Graphically*

Marion Murzek[1], Gerhard Kramler[2]
[1]Women's Postgraduate College for Internet Technologies (WIT),
[2]Business Informatics Group (BIG),
Institute for Software Technology and Interactive Systems
Vienna University of Technology, Austria
E-Mail: murzek@wit.tuwien.ac.at, kramler@big.tuwien.ac.at

Abstract

Because of today's diversity of models and heterogeneous modelling languages in the area of business modelling there is a need to automate model transformations on the level of business process models. To define such transformation processes in a simple way, a graphical modelling approach is required. In previous works a textual model transformation tool, the BMT (BOC Model Transformer) has been introduced. It enables the transformation of business process models which are instances of different meta models. A rule file which contains XML-coded instructions controls the transformation process. To simplify the requirements for creating such a rule file this paper introduces a graphical approach for modelling transformation processes. For this purpose a meta model for the transformation language of the BMT is introduced. Furthermore transformation processes which represent model instances of this meta model are illustrated and discussed.

1. Introduction

Organisations require business process management tools that support model transformation between different business process meta-models, because of the following business requirements:

Model exchange: Multinational organisations use different modelling methods for modelling their business processes. They need to exchange their models in order to provide transparency of their organisational structures.

Method change: Due to organisational needs companies change or update their modelling method. After updating, they want to automatically transfer their whole stock of models from the old method into the new one.

Model deployment: A current request on model transformations in the area of business modelling is to derive basic IT infrastructure models from already existing business process models.

The BOC Model Transformer (BMT) has been designed to support these kinds of model transformations [5, 8]. It is a transformation tool which supports the idea of Enterprise Model Integration (EMI) [6] and of Model Driven Architecture (MDA) [9]. The BMT transforms business process models based on meta model X (for example event driven process chains) into business process models based on meta model Y (for example UML action diagrams).

* "This research has been partly funded by the Austrian Federal Ministry for Education, Science, and Culture, and the European Social Fund (ESF) under grant 31.963/46-VII/9/2002."

The transformation uses an XML-based rule file, which contains the description of the transformation process and the transformation rules.

The utilisation of the BMT in a number of projects has shown that for most users it is too complicated to define the whole transformation process textually. Basically, knowledge of XML and the structure of model representation in the meta modelling tool are needed. Additionally the usage of the rule file elements must be studied intensively to write efficient and executable rule files.

A graphical transformation language could reduce the requirements for constructing rule files for the BMT. Therefore this work proposes to graphically model the transformation process based on a meta model. For the implementation of the meta model we use the Business Process Management Tool ADONIS® [3].

The remainder of the paper focuses on the meta model and the graphical representation. Chapter 2 gives overview of the implementation process, the meta model and examples for graphical model transformations. Chapter 3 related work. The paper concludes with a summary and a position statement.

2. Graphically Modelling a Transformation Process

The goal of offering a possibility to *graphically* model transformation processes based on the BMT is realized in three steps, as shown in Fig. 1.

The first step is to create a meta model for transformation processes by abstracting away the XML-specifics out of the syntax of the rules file. The next step is to define the meta model in an appropriate way using the metamodel-enabled tool ADONIS®. Based on the meta model, the tool provides the user a graphical possibility to model the transformation process. The last challenge is to generate the rules file automatically out of ADONIS®’ file format.

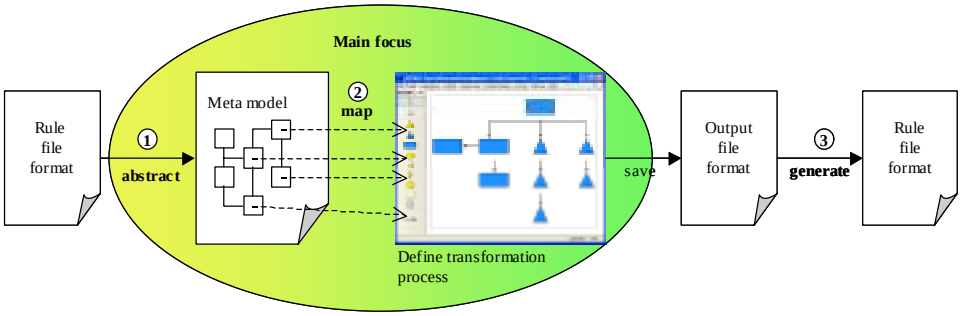

Fig. 1: Three steps to enable the graphical creation of rule files

2.1 Characteristics and Features of the BMT

To get familiar with the BMT and its mode of operation this section gives a short description of the architecture and main elements.

The BMT consists of a transformation engine and a transformation language. The language is used for describing the transformation process in a rules file. During the transformation process the transformation engine reads the rules file and executes the instructions step by step. The main instructions which are available to construct rule files are: navigations, definitions, functions, rules and conditions.

Navigations are used to match model fragments in the source context. *Definitions* are comparable with variables. The values assigned to definitions can either be fixed values or source values selected by functions. Beside selecting values, functions are also used to manipulate values. Therefore functions can be subdivided in two groups, *manipulation functions* and *selection functions*.

To generate the target models and model fragments *rules* are used. Rules are able to create model fragments in the target context, e.g. models, classes, relations, attributes etc. Depending on how a model fragment is transformed we distinguish *copy-rules* and *create-rules*. *Conditions* are used to decide if rules should be executed or not, e.g. depending on a comparison of two attribute values.

For further details regarding to the BMT and its rules file refer to [5] and [8].

2.2 Meta Model for Transformation Processes of BMT

The meta model shown in figure 2 describes the main classes of a transformation process and their relationships. The meta model itself is an instance of the meta^2-model of ADONIS® [3]. Consequently all three participating meta models – source-, target- and transformation meta model – are instances from the same meta^2-model.

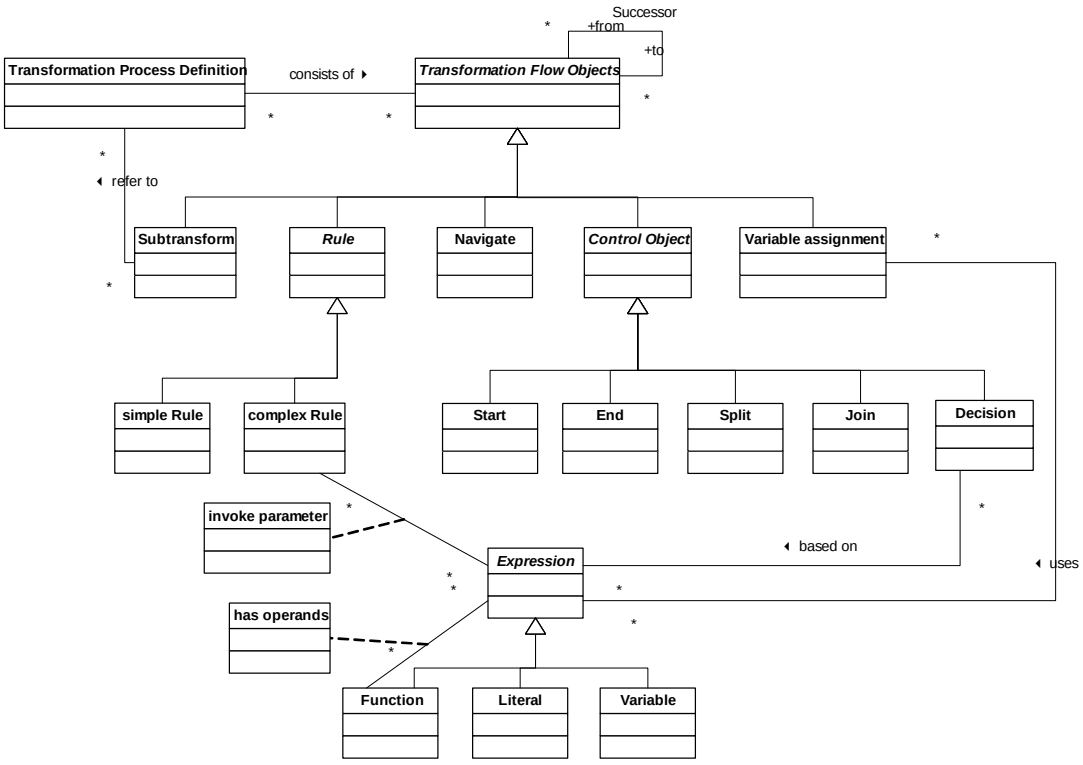

Fig. 2: Part of the meta model for transformation processes

In the following the entities are described in detail, including the most important of their attributes.

The *Transformation Process Definition* describes the process itself. It provides header-information for the transformation process, e.g. the *name* of the process, *description*, *author* etc. A process definition consists of *Transformation Flow Objects*.

The flow object *Subtransform* represents the invocation of a transformation sub process. Beside its attribute *name*, it also has a relation *"refer to"* which points to the sub process. It is used to structure the whole transformation process in logical parts.

The class *Navigate* provides the possibility to explicitly model navigations within the source context. The attribute *OfType* specifies the types of elements to be matched in the source context.

The class *Rule* is designed to create the destination output. It is used to copy existing model elements or create new ones in the target context. The abstract class *Rule* generalises the classes *simple Rule* and *complex Rule*. The attribute *type* specifies which kind of rule it is, for example model-, instance-, relation-, attribute rule etc. Each rule has parameters specifying further details of the target element to be created. Options are to either take values of the concerning source element or provide literal values for the target element. A complex Rule is characterised by its additional relation *"invoke parameter"*, which is used to generate parameter values via functions.

Start, end, split, join and decision describe the *control objects*. Together with the relation *"successor"* these objects describe the control flow of the transformation process.

The abstract class *Expression* describes the computation of a certain value. It generalises the classes *Function*, *Literal* and *Variable*. The relation *"has operands"* expresses that a function could use further expressions to evaluate the result value.

2.3 Graphical Representation

Given the different meta classes, relations and their attributes we now have to find an adequate graphical representation for these concepts.

The main problems of a graphical representation of transformation processes are:

- Simple/intuitive representation of non-trivial transformations [1].
- The balanced proportion of graphical elements and text.
- The elements should be self-explanatory. This means the concepts represented should be more concise and intuitive in graphical form compared to the textual one [10].
- Not too many different models and model elements should be provided.

The most difficult issue in graphically representing transformation processes is the first one – how to represent non-trivial transformations.

In the following two examples of models instantiated from our meta model are provided. These two examples show graphical representations of parts of transformation processes addressing the problems stated above.

First the problem is described. Then the graphical representation and its description is provided. Finally the XML-code representation of the solution is presented.

2.3.1 Example 1: Replacement of Strings

The first case deals with a problem which often occurs in the case of business model transformations, replacement of substrings. The problem is to create new models, one for each model of type "BusinessProcessModel" and replace all Ü, Ä and Ö by Ue, Ae and Oe in the model name. Fig. 3 illustrates the solution of this problem.

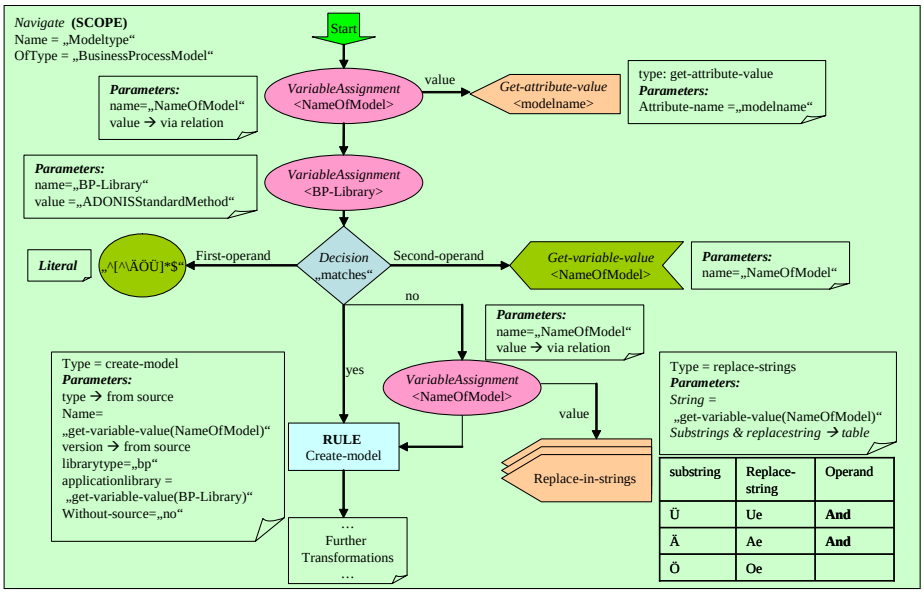

Fig. 3: Creation of models and replacing letters in the model name – graphically

The rectangle containing all of the other elements represents the navigation to each model of type “BusinessProcessModel” within the source context. This implies that during the whole process fragment - contained in this rectangle - the source relation are models of the type mentioned above. Navigations are comparable with “foreach”-loops used in programming languages. That means for each model of type “BusinessProcessModel” in the source context the process inside the rectangle is executed. Therefore the visualization as rectangle has been chosen.

Within the navigation element, first of all two variables are created and values are assigned. The variable-assignment for “NameOfModel” uses a function to get the value out of the source-model-attribute “modelname”. To the second variable “BP-Library” the literal “ADONISStandardMethod” is assigned.

The next step is a decision whether or not the name of the model contains Ä, Ö or Ü. The comparison is visualized with two related objects, a literal which contains a regular expression and a function which gets the value out of the variable “NameOfModel”. If the name does not contain any of these letters then the model will be created. Otherwise a variable assignment which uses the function “replace-in-strings” precedes the creation of the model.

The multipart object “replace-in-strings” illustrates that the string-replacement has to be done more than once, for each substring which is stated in the table. This kind of representation has been a solution of the problem “how to represent a complex function intuitively”. In preceding graphical drafts many single functions have been used to express this circumstance. This has shown that a consolidation of equal functions is necessary.

The object “create-model” indicates that a target model will be created. Two of the parameters (librarytype and Without-source) of the object “create-model” are filled with literals, two (type and version) are taken out of source and two (name and applicationlibrary) use the expression “get-variable-value” to get the assigned values.

Fig. 4 shows the same transformation process written in XML statements. Due to shortage of space, only one of the three IF-statements, implementing the string-replacement is stated.

```
<MODELTYPE name=„BusinessProcessModel“>
<NAMEVALUEMAP>
  <ELEM name=„NameOfModel“>
    <select-value>
      <PARAM name="attribute">modelname</PARAM>
    </select-value>
  </ELEM>
  <ELEM name=„BP-Library“>ADONISStandardMethod</ELEM>
</NAMEVALUEMAP>
<IF>
  <COMPARE>
    <LEFT-VALUE>
      <get-map-value>NameOfModel</get-map-value>
    </LEFT-VALUE>
    <CONDITION>matches</CONDITION>
    <RIGHT-VALUE>^[^\ÄÖÜ]*$</RIGHT-VALUE>
  </COMPARE>
  <THEN/>
  <ELSE>
    <CALL name="replace-in-strings"/>
  </ELSE>
</IF>
    <RULE type="create-model">
      <PARAM name="Modeltype">Ownerpool</PARAM>
      <PARAM name="Modelapplib">
        <get-map-value>BP-Library</get-map-value>
      </PARAM>
      <PARAM name="Modellibtype">bp</PARAM>
      <PARAM name="Modelname">
        <get-map-value>NameOfModel</get-map-value>
      </PARAM>
    </RULE>
…
</MODELTYPE>
```

```
<RULESCONTAINER name="replace-in-strings">
  <IF>
    <COMPARE>
      <LEFT-VALUE>
        <str-find>
          <param name="string">
            <get-map-value>NameOfModel</get-map-value>
          </param>
          <param name="substring">Ü</param>
        </str-find>
      </LEFT-VALUE>
      <CONDITION>equals</CONDITION>
      <RIGHT-VALUE>-1</RIGHT-VALUE>
    </COMPARE>
    <THEN/>
    <ELSE>
      <NAMEVALUEMAP>
        <ELEM name="string2replace">
          <str-replace>
            <param name="string">
              <get-map-value>NameOfModel</get-map-value>
            </param>
            <param name="substring">Ü</param>
            <param name="replacestring">Ue</param>
          </str-replace>
        </ELEM>
      </NAMEVALUEMAP>
    </ELSE>
  </IF>
  <IF>
    … repeat for each string to replace
  </IF>
</RULESCONTAINER>
```

Fig. 4: Creation of models and replacing letters in the model name – textually

2.3.2 Example 2: Getting Attribute Values from Related Objects

The second example again demonstrates the use of navigations (see Fig. 5). The goal is to create a copy of all connectors (= relation between two model instances) of type “successor” in the destination context. For each successor pointing away from an “XOR”-instance the value of the attribute “transitioncondition” is filled with the value of the attribute “description” of the instance the successor points to.

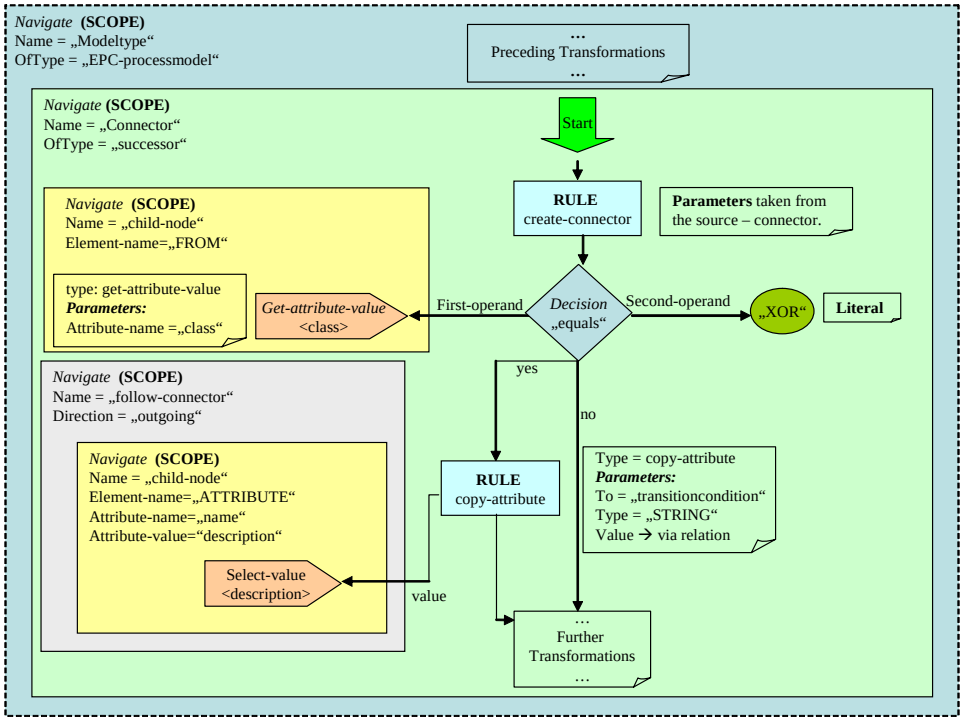

Fig. 5: Creating connectors with attribute values from related instances

The outermost rectangle illustrates the navigation to each model of type “EPC-processmodel” within the source context. The rectangle within shows the navigation to each connector of

type “successor” within a model of type “EPC-processmodel”. Then a connector is created (RULE create-connector) in the target context. The parameter values for the creation of the new connector are taken from the source connector.

The following decision compares the literal “XOR” with the name of the instance from which the connector is pointing away. This value received by navigating to the FROM-Element of the connector itself and selecting the value of the class-attribute.

If the value does equal “XOR” the attribute “transitioncondition” is created within the target connector. In doing so the value is evaluated as follows: First navigate to the instance where the connector is pointing to. Then navigate to the attribute “description” of this instance and select the value, visualized with the function “select-value”.

The same transformation process written in XML statements can be found in [5].

2.3.3 Graphical vs. Textual Representation

In the graphical representation the complexity is reduced by visualizing the language elements by the means of different colours and shapes. The visualization of navigations as rectangles makes sure that the user always knows in which part of the source context he or she is actually operating. Compared to the textual representation the graphical representation highlights the process flow, which is delimited by the elements *Start* and *End*. Furthermore the readability of the graphical representation is supported by summarizing repeating instructions, for example the “replace-in-strings” function in Fig. 3. People who are not familiar with the syntax of XML will develop transformation processes faster in the graphical way.

2.5 Related Work

Krzysztof Czarnecki and Simon Helsen identified design features to categorize Model transformation approaches [2]. According to these features the BMT could be categorized as *Hybrid Model-to-Model Approach*.

One of the graphical representations of transformations which has been tested was the Bidirectional Object oriented Transformation Language (BOTL) [7]. It is a *Graph-Transformation-Based approach* which is already implemented in a tool called BOTL. It offers the possibility to specify rules graphically. The rules are structured in LHS/RHS in form of two graphs. The information in which order the rules are executed can not be influenced, it is left to the implementation itself. This is one of the main differences to the BMT which is focusing on modelling the transformation process itself.

A *Pattern-Based approach* for graphically representing transformations is MOLA (MOdel transformation LAnguage) [4]. It represents the transformations as structured flowcharts with pattern-based rules. They combine rule patterns with a graphical loop concept. The visualization of loops in MOLA is very similar to the concept of navigations in the BMT. But the representation of the contained elements differs completely.

The focus of the most graphical representations of model transformations lies on visualizing the source-element and the resulting target-element which represents a rule. The graphical language in this paper highlights the transformation process itself and how the rules are applied. The similar structure of graphical transformation processes and the underlying source- and target models make it easier for experienced business process modellers to create model transformations this way.

4. Summary and Statement

To cope with the complexity of the requirements for transforming business process models we introduced the graphical approach to model transformation processes with the BMT.

The advantages of the graphical representation are that no knowledge of XML is required. The graphical representation is concise and intuitive for both kind of users, developers and business experts. The user is able to control the transformation process by arranging the transformation elements due to his or her requirements.

This graphical modelling approach involves the vision of MDA [9] in two ways. First with the BMT, which is a model transformer for business models and secondly with the idea of creating a model of the transformation process and transform it into XML-code for the transformation tool.

Next steps are to evaluate the graphical modelling language of the BMT in practical use and transforming the output file of ADONIS® into an according rule file to automate the graphical transformation.

5. Acknowledgment

We would like to thank BOC Information Technologies Consulting GmbH for providing the BMT[1] and the Business Process Modelling Tool ADONIS® in which the meta model has been configured.

References

[1] Bettin, J.: Ideas for a Concrete Visual Syntax for Model-to-Model Transformations. OOPSLA'03, Workshop on Generative Techniques in the Context of Model-Driven Architecture.
[2] Czarnecki, K.; Helsen, S.: Classification of Model Transformation Approaches. OOPSLA'03, Workshop on Generative Techniques in the Context of Model-Driven Architecture.
[3] Junginger, S.; Kühn, H.; Strobl, R.; Karagiannis, D.: Ein Geschäftsprozessmanagement-Werkzeug der nächsten Generation – ADONIS: Konzeption und Anwendungen, Wien im April 2000. An abridged version of this BPMS-report was published in WIRTSCHAFTS-INFORMATIK 42 (2000) 5, S. 392-401 (in German).
[4] Kalnins, A.; Barzdins, J.; Celms, E.: Basics of Model Transformation Language MOLA. ECOOP 2004, Workshop on Model Driven Development.
[5] Kühn, H.; Murzek, M.; Bayer, F.: Horizontal Business Process Model Interoperability using Model Transformation. Workshop INTEREST at ECOOP 2004, Oslo, June 2004.
[6] Kühn, H.; Bayer, F.; Junginger, S.; Karagiannis, D.: Enterprise Model Integration. In: Bauknecht, K.; Tjoa, A M.; Quirchmayr, G. (Hrsg.): Proceedings of the 4th International Conference EC-Web 2003 - Dexa 2003, Prague, Czech Republic, September 2003, LNCS 2738, Springer-Verlag, pp. 379-392.
[7] Marschall, F.; Braun, P.: Model Transformations for the MDA with BOTL. In Rensink, A., ed.: CTIT Technical Report TR-CTIT-03-27, Enschede, The Netherlands, University of Twente (2003) 25–36
[8] Murzek, M.: Methodenübergreifende Modelltransformationen am Beispiel von ADONIS. Diploma Thesis, University of Vienna, April 2004 (in German).
[9] Object Management Group: MDA Guide, Version 1.0.1, 12. Juni 2003. http://www.omg.org/cgi-bin/apps/doc?omg/03-06-01.pdf, access 12 April 2004.
[10] Sendall, S.; Kozaczynski, W.: Model Transformation – the Heart and Soul of Model-Driven Software Development. Software, IEEE , Volume 20 , Sept.-Oct. 2003, Pages: 42 - 45

[1] The BOC Model Transformer (BMT) was designed and implemented by one of the authors during her employment at BOC Information Systems GmbH.

INTEGRATION APPROACHES FOR METAMODELLING PLATFORMS

Harald Kühn
BOC Information Systems GmbH
Platform Development
Rabensteig 2
A-1010 Vienna
Austria
Tel.: ++43-1-513 27 36 10
Fax: ++43-1-513 27 36 28
E-Mail: harald.kuehn@boc-eu.com

Franz Bayer
BOC Information Systems GmbH
Service Development
Rabensteig 2
A-1010 Vienna
Austria
Tel.: ++43-1-513 27 36 10
Fax: ++43-1-513 27 36 28
E-Mail: franz.bayer@boc-eu.com

ABSTRACT

Metamodelling approaches are an active research field since the past 15 years and have found serious application areas in the software and information technology industries. Since metamodelling platforms gain widespread industrial and research usage, the integration and interoperability of metamodelling platforms is moving into focus of applied research and product-quality implementations. This paper presents various metamodelling platform integration approaches, namely the organizational integration, the transformation-based integration, the online integration and the repository integration, and evaluates the advantages and disadvantages of each approach.

1. INTRODUCTION

Metamodelling approaches are an active research field since the past 15 years and have found serious application areas in the software and information technology industries [12]. Typical domains of application are Enterprise Model Integration (EMI) [15] in the context of Enterprise Application Integration (EAI) [17], Model Integrated Computing (MIC) [16], domain specific modelling languages such as the Unified Modelling Language (UML) [20] based on Meta Object Facility (MOF) [22] and model-driven development approaches such as Model Driven Architecture (MDA) [23]. Additionally, metamodelling approaches serve as valuable base technology to merge different modelling approaches into a problem specific modelling language, e.g. to mix UML and the Business Process Execution Language (BPEL) [6] to gain a process-oriented software configuration language.

Metamodelling platforms are modelling environments allowing the problem-specific definition of method elements such as metamodels, the definition of mechanisms working on models and their underlying metamodels, and the definition of procedure models describing the processes applying the metamodels and the corresponding mechanisms. Their major functional and non-functional requirements are multi-product ability, web-enablement, multi-client ability, adaptability, and scalability [12].

Since widespread industrial and research usage of metamodelling platforms such as ADONIS [1, 11], MetaEdit+ [13] or METIS [18], the integration and interoperability of metamodelling platforms is moving into focus of applied research and product-quality implementations [19]. Based on our experiences in applying metamodelling technology in business and technical domains, various integration approaches for metamodelling platforms will be presented.

The remainder of the paper is structured as follows: chapter 2 introduces a generic architecture of metamodelling platforms. Based on this architecture, chapter 3 presents and evaluates four metamodelling platform integration approaches. Chapter 4 discusses related work and chapter 5 concludes with a summary and an outlook to future developments.

2. METAMODELLING PLATFORM ARCHITECTURE

Metamodelling platforms should be realised based on a component-based, distributable, and scalable architecture. Figure 1 presents a *generic architecture* of functional aspects of metamodelling platforms [12]. Additional architectural aspects in the context of metamodelling platforms such as multi-tier architectures and repository architectures can be found in [14, p. 181 ff].

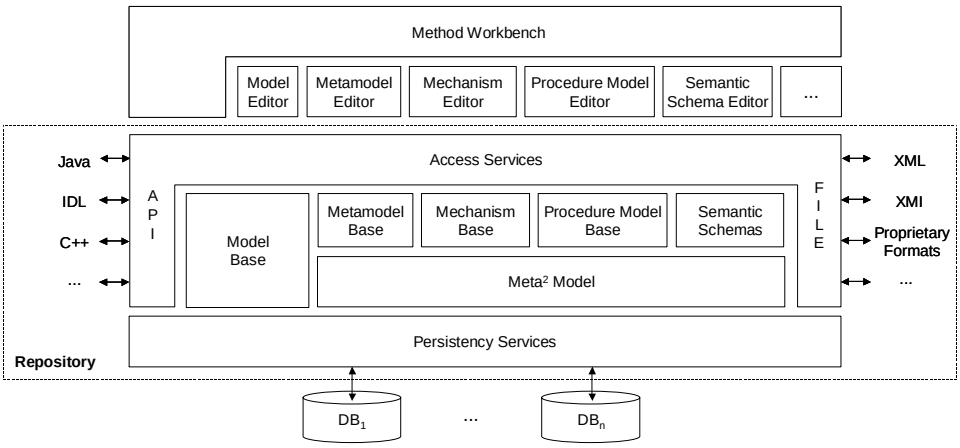

Figure 1: Generic Metamodelling Platform Architecture

An important element of generic metamodelling architectures is the *meta-metamodel* (meta^2 model) [8]. The meta^2 model defines general concepts available for method definition and method usage such as "metamodel", "model type", "class", "relation", "attribute" etc. *Semantic schemas* are tightly coupled with the meta^2 model. They describe the semantics of each method element defined by using the meta^2 model. Semantic schemas can be described by using approaches such as ontology [10], semantic engines ("mechanisms") etc.

The *metamodel base* contains metamodels of concrete modelling languages. Metamodel editors are used for the definition and maintenance of metamodels. The metamodel base is based on the meta^2 model. The metamodel base forms the foundation of the *model base*, in which all models are stored. Models can be created, changed and visualized by using appropriate editors.

All mechanisms and algorithms used for evaluating and using models are stored in the *mechanism base*. Mechanism editors are used for definition and maintenance of mechanisms. The mechanism base is based on the meta^2 model.

Procedure models describe the application of metamodels and mechanisms. They are stored in the *procedure model base*. Procedure model editors are used for definition and maintenance of procedure models. The procedure model base is based on the meta^2 model.

Persistency services support the durable storage of the various bases. These services abstract from concrete storage techniques and permit filing of modelling information in heterogeneous databases, file systems, web services etc.

Access services serve two main tasks: on the one hand they enable the open, bi-directional exchange of all metamodelling information with other systems using API or file based. interfaces. On the other hand they cover all aspects concerning security such as access rights, authorization, en-/decryption etc.

3. INTEGRATION APPROACHES

Figure 2 provides an overview of the presented *metamodelling platform integration approaches*. All approaches can be applied both on the metamodel level and on the model level.

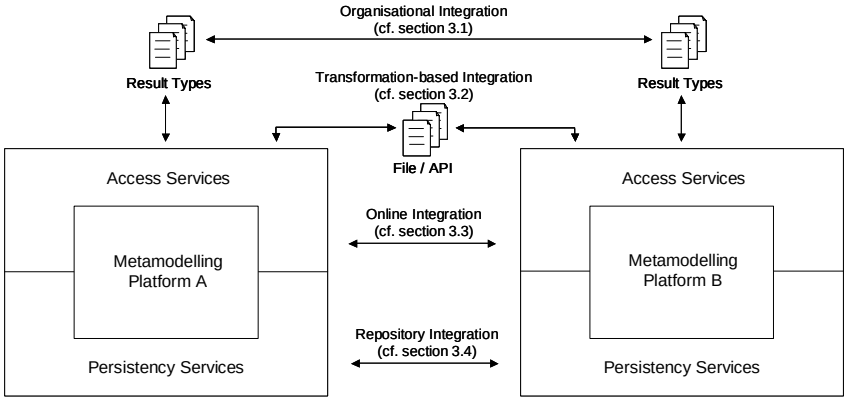

Figure 2: Overview of Integration Approaches

3.1 Organizational Integration

Organizational integration will be applied, if no technical interfaces are available, but two metamodelling platforms should be used integrated. Organizational integration is based on organizational steps, describing who has to do what and when. Typical tasks in the context of organizational integration are manual exchange of metamodel and model information and cross checks to ensure consistency of integrated information. For this, often documents are used, which were generated by each platform for mutual information exchange.

An aid often used in practice is shown in figure 3. There, the metamodelling platforms itself are not integrated, but the result types generated by each platform. Prerequisites of this solution are (a) the possibility of generating documents in formats such as HTML or XML, (b) the generation of stable and durable file names and (c) the possibility to enter links to external documents in at least one metamodelling platform.

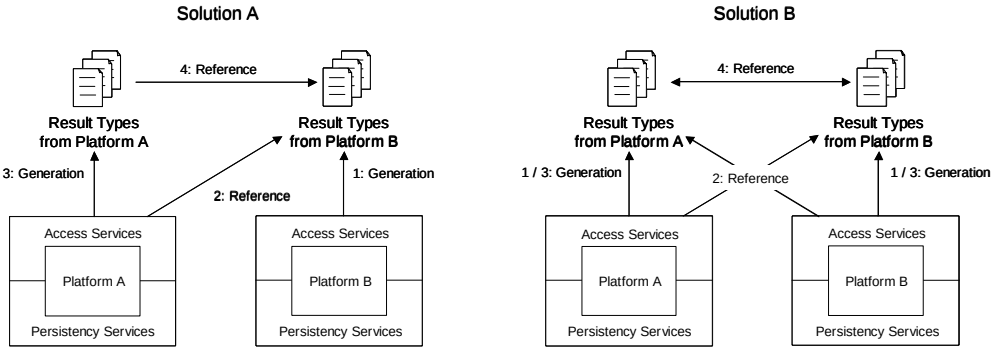

Figure 3: Organizational Integration

In solution A platform B knows how to generate stable and durable file names, e.g. using global unique identifiers (GUID). Platform A is not able to do so (step 1). The generated documents, i.e. result types, are referenced from platform A using the stable file names (step 2). The result types generated by platform A (step 3) are integrated by the stable file references with result types from platform B (step 4).

In solution B both metamodelling platforms can generate stable and durable result type file names. Following an iterative process, documents are generated with both platforms (step 1), the generated documents are referenced by the other platform (step 2) and the resulting cross

references are generated again (step 3). The result of repeated steps 1 to 3 are completely and bidirectional integrated result type documents from both platforms (step 4).

3.2 Transformation-based Integration

The *transformation-based integration* approach transforms one or more method elements from platform A to platform B. Basically, two kinds of technical transformation interfaces exist: file-based interfaces and API-based interfaces.

Currently, file-based interfaces are the most common. The information to be transformed is written in a file, which is then imported into the target platform. A well-known example for file-based model transformation is the XMI format used for XML-based exchange of UML models [21].

Using API-based interfaces, the information to be transformed is directly transferred from platform A to platform B via programmatic interfaces. Among other things, in the API approach the dependency to platform suppliers has to be taken into account. If a supplier changes his API, this can have negative influence on an existing integration, potentially resulting in a non-operational integration.

In both kinds of interfaces, *structural and content transformation* can be distinguished. The structural transformation transforms the structure of a method element from platform A to platform B. At simplest case in both platforms the same method definition exist resulting in 1:1 transportation of the corresponding method elements. If the method definitions differ, also the method structure has to be transformed. The content transformation focuses on the properties (attributes) of the structural elements.

A weak spot of the transformation-based integration is the redundancy of information. If information from metamodelling platform A will be transformed to platform B, in both platforms the transformed information will be changed separately and the transformation will be repeated, normally the changes in platform B will be lost. Approaches for consolidation of redundant information exist e.g. in reverse engineering and round-trip-engineering. An active research field in solving the problem of lost information in the context of model transformation is the MDA [9, 23, 24].

3.3. Online Integration

Applying *online integration*, interdependent information of method elements should be set into relation spanning each platform. In contrast to transformation-based integration, no import/export of method information from platform A to platform B will be done, but a platform-spanning online-linking of information. The information stored in each platform should be as disjunctive as possible, to minimize the problem of redundant information (see section 3.2). The information linking follows the hyperlink concept from section 3.1. In comparison to the hyperlink concept known from the Internet, where links are pure pointers which can break if the referenced target will be renamed, online integration uses "stable links". In this approach, the global unique identifier of the referenced information is used, which will be kept unchanged during lifetime of the information.

Furthermore, the online integration can be distinguished into unidirectional and bidirectional. In *unidirectional online integration*, one metamodelling platform is the leading platform from which links to method information in the target platform are set. Link information are only stored in the leading platform and can only be activated in the leading platform (see figure 5, left). In *bidirectional online integration*, links are stored in both participating platforms. This gives the advantage, that in both platforms the links can be queried, evaluated, and used (see figure 5, right).

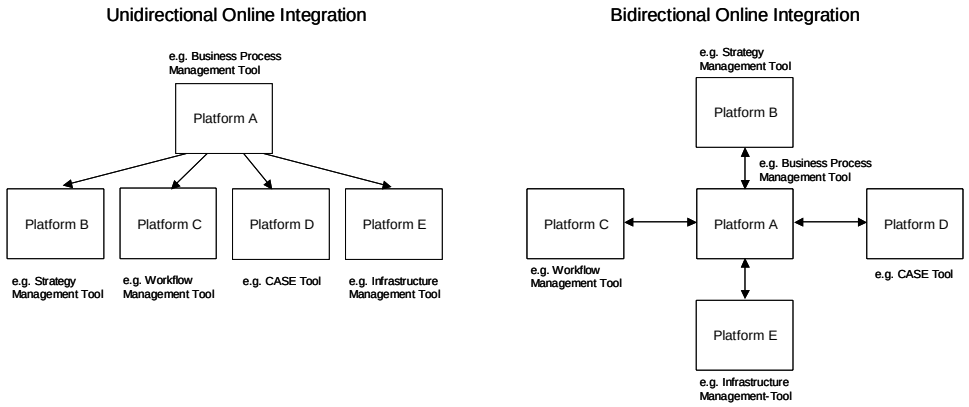

Figure 5: Online Integration

For technical implementation, programmatic interfaces such as COM or middle ware technology can be used. In normal case, the platforms participating in online integration are installed on a single workstation to enable the necessary programmatic access. In some cases, e.g. in distributed work groups, it can be necessary to integrate the metamodelling platforms in a distributed environment. In this case, the platforms have to support APIs based on Remote Method Invocation (RMI) or Web Services [2]. This can raise the integration and implementation effort considerably. Therefore, a careful cost-benefit-analysis should be executed.

3.4 Repository Integration

In *repository integration* the exchange of integration-relevant information does not happen via the access service as in the before mentioned approaches but via the persistency service. The idea of this integration approach is the uniform usage of a repository by different platforms. This way, the drawbacks of redundant storage of method information or the information loss in transformation-based approaches should be avoided.

Generally, two kinds of repository integration can be distinguished. Using a *virtual repository*, the original repositories of each metamodelling platform will be kept. Each platform accesses with its persistency service not directly its own repository, but uses for repository access the virtual repository (see figure 6, left). The virtual repository acts as a proxy delegating requests to each responsible repository and return the corresponding results to the correct platform. If a *common repository* is used, the "local repositories" of each platform are obsolete. Each platform sends its repository requests only to the common repository (see figure 6, right).

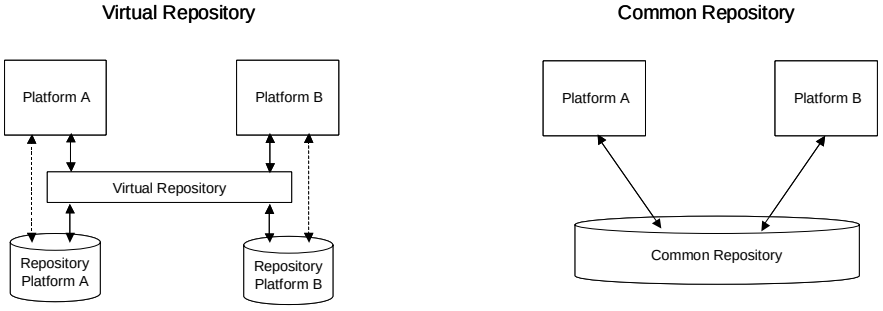

Figure 6: Repository Integration

In comparison to the other integration approaches, in practice repository integration is the approach less used. This has several reasons:

- Until now, no international standard and no industry standard for definition and operation of a uniform repository is established. There are several proprietary solutions such as the

Universal Repository of UNISYS [28] or the Microsoft Repository [3], but none of them has gained supplier-spanning dissemination. As a possible starting point for virtual repositories we see the OMG activities around MOF [22].

- The creation of a common repository depends on a number of non-trivial problems and conceptually demanding issues. Some selected aspects are the mapping and integration of different instances of method elements in a common schema, the potentially necessary transformation between different modelling languages, and the definition of operational semantics for the method element instances, which are not certain at repository implementation time.
- A main reason for not using a common repository from a business perspective is the loss of influence and the limited autonomy of a metamodelling platform supplier. The repository is the core of a metamodelling platform and one of the central competitive advantages of each supplier is the functionality of its repository.

3.5 Evaluation of Integration Approaches

Figure 7 provides a comparison of the metamodelling platform integration approaches. For each approach the advantages and disadvantages are presented and the typical application scenarios described.

Integration Approach	Advantage	Disadvantage	Application Scenario
Organizational Integration	• "Linking" of result types generated from different platforms.	• Manual effort for transformation/ integration.	• If no other integration approach is available. • If only low demand for transformation/ integration.
Transformation-based Integration	• Avoidance of double work. • Can often be realized by platform user itself.	• Often available only unidirectional. • Normally no "delta transformation".	• If strong intersections of method elements. • If sequential procedure model, unless (bidirectional) "delta transformation" is available. • If different method users.
Online Integration	• Integration of method element instances via linking. • Normally tool support to detect inconsistencies.	• Can often be realized only by supplier. • Often there are certain redundancies in integrated method elements. • Normally, platforms have to be installed on a single workstation.	• If method element information are largely disjunctive. • Especially for iterative procedure model.
Repository Integration	• Strong avoidance of redundancies. • Normally strong interrelationships of platforms.	• Is rarely available (often only if platforms come from single supplier). • Can often be realized only by supplier.	• Useful for sequential as well as for iterative procedure model.

Figure 7: Evaluation of Integration Approaches

4. RELATED WORK

ADONIS is a meta business process management tool [1, 11]. It offers a three-step modelling hierarchy with a rich meta^2 model. Meta models can be customized as instances of the meta^2 model. Mechanisms such as "simulation" or "analysis" are defined on the meta^2 level and can be redefined on the metamodel level. The scripting language AdoScript provides mechanisms to define specific behaviour and functionalities.

MetaEdit+ offers also a three-step modelling hierarchy [13]. The meta^2 model forms the "GOPRR" model, offering the basic concepts "Graph", "Object", "Property", "Relationship" and "Role". A diagram editor, object and graph browsers and property dialogs support the definition of a new modelling language without hand coding. Furthermore MetaEdit+ includes XML import and export, an API for data and control access and a generic code generator.
OMG´s *Meta Object Facility* (MOF) [22], the open source *Eclipse Modelling Framework* (EMF) [26] and the *Graphical Editor Framework* (GEF) [25] are no metaCASE tools themselves. With the MOF the OMG created a meta^2 model standard, which provides a basis for defining modelling frameworks. Interoperability issues concerning the meta model and the model domain are addressed by the ongoing standardisation of MOF Query/Views/Transformations (QVT) [24] which will provide mechanisms for mappings between models and meta models.
The *eXecutable Metamodelling Facility* (XMF) is a metamodelling tool supporting model transformation [7]. The central part of the XMF is the XCore which is comparable to the MOF model. To support mappings between models two further languages are defined: XMap which is a unidirectional pattern based mapping language and XSync which is a bidirectional synchronisation language.
Additional related work can be found e.g. in [5], [8], [18], [21], and [27].

5. CONCLUSION

Metamodelling platforms are getting more and more base technology [12] and models are getting "first class citizens" [4]. Additionally, domain specific languages, model transformation approaches, and lifecycle management within large model bases are active research issues. The integration of metamodelling platforms becomes an important aspect in managing corporations' knowledge assets. This paper presented four metamodelling platform integration approaches and described their advantages and disadvantages.
In the near future, we see three important trends in the area of metamodelling platforms:

- *Metamodelling gets commodity*: metamodelling provides suitable concepts for flexible modelling platform solutions. Furthermore, metamodelling concepts diffused more and more in other domains such as MOF, into language design such as the language definition of UML 2.0 or in Software Engineering e.g. in product family-based approaches. We expect, metamodelling will also attract more attention in domains such as Workflow Management, IT Architecture Management and Knowledge Management. Saying this, challenging interoperability issues will have to be solved.
- *Integration of business-oriented and IT-oriented methodologies*: we see strong demands integrating approaches such as Strategy Management, Process Management and IT Management into single, integrated methods. A promising approach is MOF and MDA. Nevertheless, their focus is currently focused on systems development. Upper-level models such as business specifications and computation independent models (CIM) are not well represented until now. Here, research need in areas such as low-loss transformations and treatment of heterogeneous semantics is expected.
- *Method Integration and Knowledge Management*: our society is seen as a "knowledge society". Methods represent experts knowledge, they represent the know how to do and process things in a certain way. As a future research domain we see the investigation of interdependencies of Knowledge Management and Method Engineering and corresponding issues in integrating both.

REFERENCES

1. ADONIS Homepage. http://www.boc-eu.com
2. Alonso, G.; Casati, F.; Kuno, H.; Machiraju, V.: Web Services. Springer-Verlag, 2004.

3. Bernstein, P. A.; Bergstraesser, T.; Carlson, J.; Pal, S.; Sanders, P.; Shutt, D.: Microsoft Repository Version 2 and the Open Information Model. In: Information Systems, Vol. 24, Nr. 2, 1999, S. 71-98. http://www.research.microsoft.com/~philbe/Info Sys on MS Repository for Web.pdf, access 30 September 2004.
4. Bézevin, J.: From Object Composition to Model Transformation with the MDA. In: Proceedings of TOOLS'USA, Volume IEEE TOOLS-39, Santa Barbara, California, USA, 2001.
5. Blanc, X.; Gervais, M.; Sriplakich, P.: Model Bus: Towards the Interoperability of Modelling Tools. Proceedings of Model Driven Architecture: Foundations and Applications, Uwe Aßmann (Ed.), Linkoeping, Sweden, June 2004.
6. BPEL4WS (Business Process Execution Language for Web Services) Version 1.1 May, 5 2003. http://www-106.ibm.com/developerworks/library/ws-bpel/, access 30 September 2004.
7. Clark, T.; Evans, A.; Sammut, P.; Willans, J.: Applied Metamodelling – A Foundation for Language Driven Development. Xactium, 2004.
8. Electronic Industries Association - CDIF Technical Commitee: CDIF CASE Data Interchange Format - Overview. EIA-IS-106, 1994.
9. Gardner, T.; Griffin, C.; Koehler, J.; Hauser, R.: A review of OMG MOF 2.0 Query / Views / Transformations Submissions and Recommendations Towards the Final Standard. OMG, IBM, July 2003.
10. Gruber, T. R.: Toward principles for the design of ontologies used for knowledge sharing. In: Guarino, N.; Poli, R. (Hrsg.): Proceedings of the International Workshop of Formal Ontology, Padova, Italien, August 1993.
11. Junginger, S.; Kühn, H.; Strobl, R.; Karagiannis, D.: Ein Geschäftsprozessmanagement-Werkzeug der nächsten Generation - ADONIS: Konzeption und Anwendungen. In: WIRTSCHAFTSINFORMATIK 42 (2000) 5, pp. 392-401.
12. Karagiannis, D.; Kühn, H.: Metamodelling Platforms. Invited Paper. In: Bauknecht, K.; Min Tjoa, A.; Quirchmayer, G. (Eds.): Proceedings of the Third International Conference EC-Web 2002 – Dexa 2002, Aix-en-Provence, France, September 2-6, 2002, LNCS 2455, Springer-Verlag, Berlin, Heidelberg, p. 182.
13. Kelly, S.; Lyytinen, K.; Rossi, M.: MetaEdit+ - A Fully Configurable Multi-User and Multi-Tool CASE and CAME Environment. In: Constantopoulos, P.; Mylopoulos, J.; Vassiliou, Y. (Eds.): Advanced Information System Engineering, 8th International Conference, CAiSE'96, Heraklion, Crete, Greece, May 20-24, 1996, Proceedings. Springer-Verlag, LNCS 1080, Berlin et al., 1996, pp. 1-21.
14. Kühn, H.: Methodenintegration im Business Engineering. PhD Thesis, University of Vienna, Austria, April 2004.
15. Kühn, H.; Bayer, F.; Junginger, S.; Karagiannis, D.: Enterprise Model Integration. In: Bauknecht, K.; Tjoa, A M.; Quirchmayr, G. (Eds.): Proceedings of the 4th International Conference EC-Web 2003 - Dexa 2003, Prague, Czech Republic, September 2003, LNCS 2738, Springer-Verlag, pp. 379-392.
16. Ledeczi, A.; Maroti, M.; Bakay, A.; Karsai, G.; Garrett, J.; Thomason, C.; Nordstrom, G.; Sprinkle, J.; Volgyesi, P.: The Generic Modeling Environment. In: Proceedings of the Workshop on Intelligent Signal Processing, Budapest, Ungarn, May 2001. http://www.isis.vanderbilt.edu/publications/archive/Ledeczi_A_5_17_2001_The_Generi.pdf, access 30 September 2004.
17. Linthicum, D. S.: Enterprise Application Integration. Addison-Wesley, 2000.
18. METIS Homepage. www.computas.com, access 30 September 2004.
19. NoE INTEROP, Interoperability Research for Networked Enterprises Applications and Software, IST Network of Excellence, www.interop-noe.org, 2004.

20. Object Management Group: OMG Unified Modeling Language Specification, Version 1.4, September 2001. http://www.omg.org/cgi-bin/doc?formal/01-09-67.pdf, access 30 September 2004.
21. Object Management Group: OMG XML Metadata Interchange (XMI) Specification, Version 1.2, Januar 2002. http://www.omg.org/cgi-bin/doc?formal/02-01-01.pdf, access 30 September 2004.
22. Object Management Group: Meta Object Facility (MOF) Specification, Version 1.4, April 2002. http://www.omg.org/cgi-bin/doc?formal/02-04-03.pdf, access 30 September 2004.
23. Object Management Group: MDA Guide, Version 1.0.1, 12. June 2003. http://www.omg.org/cgi-bin/apps/doc?omg/03-06-01.pdf, access 30 September 2004.
24. Object Management Group: Revised submission for MOF 2.0 Query/Views/ Transformations RFP, OMG Specification/2003-08-18, August 2003, http://www.omg.org/docs/ad/03-08-08.pdf, access 30 September 2004.
25. The Graphical Editing Framework (GMF). http://www.eclipse.org/gef/
26. The Eclipse Modeling Framework (EMF). http://www.eclipse.org/emf/
27. UEML – Unified Enterprise Modelling Language, http://www.ueml.org, access 30 September 2004.
28. UNISYS: Universal Repository - Technical Overview. Version 3.0, 8807 6971-001. Unisys Corporation, 1998.

Extracting Ontologies for the Semantic Web from HTML Forms

Irina Astrova[1], Bela Stantic[2]

[1] Tallinn University of Technology, Ehitajate tee 5,
19086 Tallinn, Estonia
irina.astrova@cellnetwork.com
[2] Griffith University, PMB 50 Gold Coast Mail Center,
QLD 9726 Brisbane, Australia
b.stantic@griffith.edu.au

Abstract. We propose a novel approach to reverse engineering of relational databases to ontologies. Our approach is based on the idea that semantics of a relational database can be inferred, without an explicit analysis of relational schema, tuples, and user queries. Rather, these semantics can be extracted by analyzing HTML forms – popular user-friendly interface to access relational databases on the current Web. The semantics are supplemented with the relational schema and user "head knowledge" to build an ontology. Our approach can be applied to migrating the current Web, which is usually based on relational databases, to the ontology-based Semantic Web.

1 Introduction

The Semantic Web [1] is an extension of the current Web on which information is given a well-defined meaning better enabling machines and humans to work in cooperation. Ontologies are the key to this cooperation, as they determine the common vocabulary software agents will need to communicate with each other and users will need to communicate with the software agents.

The Semantic Web is fueled by a hope that it will solve many problems [2, 3] of the current Web; e.g.:

- The Web content is not machine-understandable, as Web pages are usually represented by HTML. This language describes how information looks like and not what it is
- A large number of Web pages (which are usually referred to as the Deep or Hidden Web [2]) are not included in search engine indexes, as these pages are generated dynamically from relational databases, by sending user queries through HTML forms. The Deep Web is estimated to be around 500 times the size of the Surface Web, which consists of static Web pages.

The Semantic Web is a solution for many problems of the current Web. But it is the solution that presupposes machine understandability of the Web content. That is, Web

pages will be annotated with ontologies that provide semantics of the Web content. The Semantic Web will create an environment, where software agents roaming from page to page can readily carry out sophisticated tasks for users [1]. Thus, the users will find that the Semantic Web more useful and more powerful than the current Web.

With all these advantages, what keeps the Semantic Web from being brought to reality[1]? One reason is that manual semantic annotation [4] is time-consuming, subjective and error-prone. It is even impossible on the Web scale – the current Web contains more than 1.5 billion pages.

If we could automatically (or at least semi-automatically) extract semantics of relational databases from Web pages and reconstitute them as ontologies annotating those pages, we would be able to migrate from the current Web to the Semantic Web. Reverse engineering of relational databases to ontologies is the key to this migration.

However, because of the novelty of that area, there are few approaches that consider ontologies as the target for reverse engineering; e.g. [3, 5, 6, 7]. These approaches usually analyze a relational schema, tuples, and user queries to build an ontology, thus being limited in terms of:

- Requiring more input information than it is possible to provide in practice, as the complete information about a relational database is usually unavailable; and
- Making unrealistic assumptions about the input. E.g. the relational database is in third normal form (3NF).

As an attempt to overcome these limitations, we propose a novel approach to reverse engineering of relational databases to ontologies.

2 Our Approach

Our approach is based on the idea that semantics of a relational database can be inferred, without an explicit analysis of relational schema, tuples, and user queries. Rather, these semantics can be extracted by analyzing HTML forms and search results[2]. The semantics are supplemented with the relational schema and user "head knowledge" to build an ontology.

Our approach uses Web pages as the main input and goes through three basic steps: (1) analyzing HTML forms and search results to extract a form model schema; (2) transforming the form model schema into an ontology; and (3) creating ontological instances from data in the search results.

To illustrate these steps, Fig. 1 shows a Web page. This results from the search for a used vehicle on `http://www.bobhowardhonda.com`.

[1] The Semantic Web does not exist except in isolated environments, mostly in research labs.

[2] Since HTML forms present only a sketch of (part of) the relational database [2], our approach also analyzes information contained in the search results.

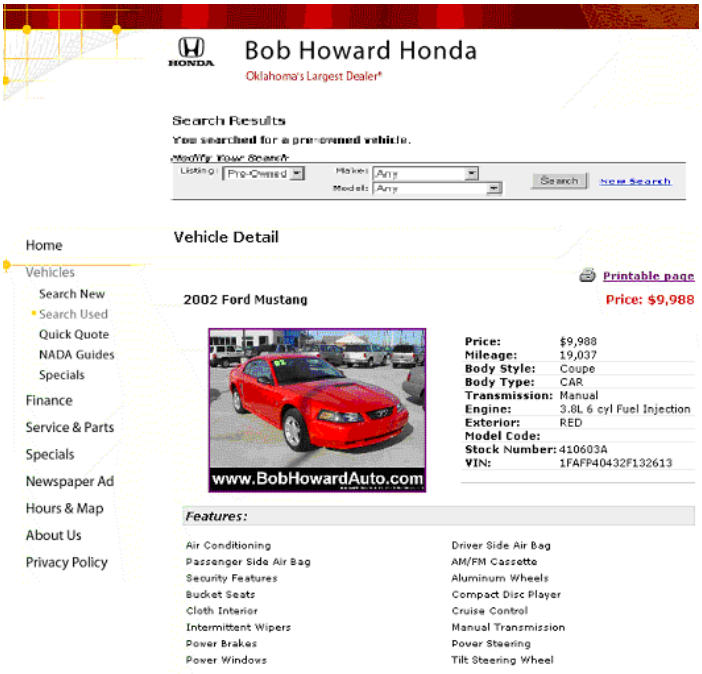

Figure 1. Web page

2.1 Extracting Form Model Schema

A form model schema [8] was originally proposed to extract an entity-relationship schema from database forms. Basically, it consists of:

- *Form field:* This is an aggregation of *name* and *entry* associated to it[3]. A name is pre-displayed and serves as a clue to what is to be entered by the user or displayed by the system. An entry is the actual data; it roughly corresponds to an attribute in the relational database. We use the term of *linked attribute* for such an entry to distinguish it from other entries, which are computed or simply unlinked with the relational database
- *Structural unit:* This is a logical group of closely related form fields (e.g. a table in the Web page). It roughly corresponds to a relation in the relational database
- *Relationship:* This is a connection between structural units that relates one structural unit to another (or back to itself). There are two kinds of relationship: *association* and *inheritance*

[3] However, we can also identify a form field, where there is no name for the entry; e.g. the photo, year, make, and model in Fig. 1.

- *Constraint:* This is a rule that defines what data is valid for a given form field. A *cardinality constraint* specifies for an association relationship the number of instances that a structural unit can participate in
- *Underlying source:* This is a structure of the relational database (i.e. a relational schema), which defines relations and attributes along with their data types
- *Form type:* This is a collection of empty form fields
- *Form template:* This is a particular representation of form type. Each form template has a layout (i.e. its graphical representation) and a title, which identifies the Web page and provides its general description
- *Form instance:* This is an occurrence of form type, when its template is filled in with the actual data. Fig. 1 is an instance of the form type.

In reverse engineering of relational databases to ontologies, the first step is analyzing HTML forms and search results to extract a form model schema. This schema expresses semantics of a relational database behind the HTML forms.

2.1.1 Analysis of HTML Form and Search Result Structure

Basically, this means identifying constructs in the form model schema and giving them names using table understanding techniques [9].

Identifying form instances: Web pages typically consist of multiple frames. This complicates the process of extracting the form model schema. Further complications arise when Web pages contain advertisements and navigational menus, which can be viewed as "noisy" data [2]. Thus, given a Web page, the first task is to identify a form instance or data-rich section. We can do this in two ways. First, we can use the DSE (Data-rich Section Extraction) algorithm [2]. Second, we can search through all the frames in the page to find the largest one. This approach typically implies that frames occupying larger display areas will be more interesting to users [10]. E.g. from Fig. 1 we would identify that `Vehicle Detail` represents all data that are the subjects of interest to users.

Identifying structural units: We can take two basic approaches to this. First, we can examine HTML code for structural tags such as `<table>`, `<ul>` and `<ol>`[4] [9]. E.g. from Fig. 1 we would identify two structural units. One contains information about a used vehicle (`Year`, `Make`, `Model`, `Price`, `Mileage`, …, and `VIN`); while another structural unit lists the vehicle features (`Air Conditioning`, `Passenger Side Air Bag`, …, and `Tilt Steering Wheel`). This approach assumes that HTML code is well designed, correct and stable. However, Web pages typically change their layout more than twice a year [11]. Different HTML code can result in the same layout of Web pages. This is usually because of errors in HTML code and common misuse of the structural tags [10]. Second, we can use machine learning techniques [12].

Identifying linked attributes: We can take two basic approaches to this. First, we can examine HTML code for structural tags such as `<thead>` and `<th>` [9]. This approach typically is viable only if the linked attributes are separated with the struc-

[4] It is necessary to examine the tags `<ul>` and `<ol>`, because structural units are sometimes represented by lists and not by tables (e.g. the features in Fig. 1).

tural tags; it does not work for merged data. E.g. the year, make, and model in Fig. 1 are all merged data, meaning that they are encoded into a single text string: "2002 Ford Mustang". Second, we can use visual cues [2, 10]. This approach typically implies that there will be some separators (e.g. blank areas) that help split the merged data. E.g. the year, make, and model in Fig. 1 are separated with a space.

Identifying relationships: Related structural units will appear at the same form instance. E.g. from Fig. 1 we would identify an association relationship between Vehicle and Feature: a used vehicle Has a feature.

Naming structural units: Structural units are usually given names of the corresponding relations in the relational database. But it is generally less confusing to users if the names are more meaningful. Notice the adaptation of the name Feature to the features in Fig. 1. This can better convey the meaning of the structural unit than the original relation name Detail would.

Naming linked attributes: There are three basic approaches to this. One is to give the linked attributes names of the corresponding attributes in the relational database. A difficulty is that the relational schema can be unknown [9], while "the original attribute names are usually not encoded in Web pages" [2]. E.g. the photo, year, make, and model in Fig. 1 are given no names at all. Another approach is to give the linked attributes field names. This approach typically assumes that Web pages are user-friendly and thus, field names are more explicit and meaningful than the original attribute names. E.g. there is a possibility that the mileage in Fig. 1 actually were named m in the relational database. However, it is difficult to expect that users to deduce the meaning of that attribute from the name m. Yet another approach is to give the linked attributes data type names [2]. E.g. a linked attribute represented by the image in Fig. 1 might be named image.

Naming relationships: Relationships are usually given names that are either names of the corresponding relations (for many-to-many relationships) or foreign key names (for one-to-one or one-to-many relationships). Again, users can give more meaningful names to the relationships.

The end result for the first step of reverse engineering is shown in Fig. 2.

Form model schema	**Ontology**
-- Structural units Structural-Units ::= { Feature(**-- Linked attributes** airConditioning : BIT, passengerSideAirBag : BIT, … tiltSteeringWheel : BIT) Vehicle(**-- Linked attributes** year : INTEGER, make : VARCHAR, model : VARCHAR, price : FLOAT, mileage : FLOAT, … vin : VARCHAR)} **-- Relationships**	**// Classes** Feature::Object[**// Attributes** airConditioning =>> Boolean, passengerSideAirBag =>> Boolean, … tiltSteeringWheel =>> Boolean]. Vehicle::Object[**// Attributes** year =>> Integer, make =>> String, model =>> String, price =>> Float, mileage =>> Float, … vin =>> String, **// Relationships**

```
Relationships ::= {
 Has(Vehicle, Feature)}
-- Constraints
NotNull(Vehicle, mileage)



-- Underlying source
Underlying-Source ::= {
 Detail(
  -- Attributes
  airConditioning : BIT,
  PassengerSideAirBag : BIT,
  …
  tiltSteeringWheel : BIT)
 Vehicle (
  -- Attributes
  year : INTEGER,
  make : VARCHAR,
  model : VARCHAR,
  price : FLOAT,
  mileage : FLOAT,
  …
  vin : VARCHAR)}
```

```
feature =>> Feature].
// Axioms
NotNull(Vehicle, mileage).
  Forall C,A NotNull(C, A) <-
   Forall IC Exists IA,IC:C And
    IC[A ->> IA].
// Instances
aFeature:Feature[
 // Attributes
 airConditioning, // true value
 passengerSideAirBag,
 …
 tiltSteeringWheel].
aVehicle:Vehicle[
 // Attributes
 year ->> 2002,
 make ->> "Ford",
 model ->> "Mustang",
 price ->> 9988,
 mileage ->> 19037,
 …
 // Relationships
 feature ->> aFeature].
```

Figure 2. Summary of reverse engineering

2.1.2 Data Analysis

In addition to the structure of HTML forms and search results, data in the search results are analyzed to identify constraints. A data analysis includes a strategy of learning by examples borrowed from machine learning techniques [13, 14]. In particular, it is performed as a sequence of learning tasks from the relational database. Each task is defined by: (1) *task relevant data* (e.g. the data in the search results), (2) *problem background knowledge* (e.g. the application domain knowledge), and (3) *expected representation of results of learning tasks* (e.g. the first order predicate logic) [8]. The results of learning tasks are related to a current state of the relational database. Thus, they are generalized into knowledge about all states through an induction process. This process combines the semantics extracted from the search results with the application domain knowledge, which is provided by users (i.e. the user "head knowledge"). Such knowledge controls the learning tasks to come to the best inductive conclusion, the conclusion that will be consistent with all states of the relational database.

E.g. from Fig. 1 we would identify a constraint `NotNull` on the linked attribute `mileage`. This contains non-null values for any used vehicle.

2.2 Schema Transformation

The second step of reverse engineering is transforming the form model schema into an ontology (i.e. "schema transformation"). Basically, this means mapping constructs in the form model schema to constructs in the ontology using mapping rules [15]. The ontology is formulated in F-Logic (Frame Logic) [16]. This language provides classes, attributes with domain and range definitions, inheritance hierarchies of classes and

attributes, and axioms that can be used to further characterize relationships between ontological instances.

As an example, consider the form model schema in Fig. 2. Here schema transformation is straightforward. First, we create a class for each structural unit in the form model schema. E.g. we create two classes: `Vehicle` and `Feature`. Second, within each class, we create an attribute for each linked attribute in the structural unit. E.g. for `Vehicle`, we add attributes `year`, `make`, `model`, `price`, `mileage`, …, and `vin`. Third, if an association between the structural units is binary, we create an attribute to represent that association. Otherwise, we create a class for the association. E.g. for `Vehicle`, we add an attribute `feature`; this associates `Vehicle` with `Feature`. Fourth, we create an axiom for each constraint in the form model schema. E.g. we add an axiom `NotNull` to the ontology. The end result for the second step of reverse engineering is shown in Fig. 2.

2.3 Data Migration

The third step of reverse engineering is creating ontological instances from data in the search results (i.e. "data migration"). Basically, this means assigning values to the appropriate attributes in the ontology using table understanding techniques [9].

As an example, consider again the Web page in Fig. 1. Here data migration is easy for the attributes `year`, `make`, `model`, `price`, `mileage`, …, and `vin`. However, we meet with a difficulty when trying to find values for `airConditioning`, `passengerSideAirBag`, …, and `tiltSteeringWheel`. We overcome this difficulty by representing the features in Fig. 1 through Boolean attributes and assigning them true values. A used vehicle can have many other features (e.g. `Anti-Lock Brakes`, `Automatic Transmission`, `Front Wheel Drive`, etc.). However, these features are not displayed in Fig. 1, because the appropriate attributes in the relational database will have false values for any tuple corresponding to `Ford Mustang`. The end result for the third step of reverse engineering is shown in Fig. 2.

3 Conclusion

We have proposed a novel approach to reverse engineering of relational databases to ontologies. Our approach has two important advantages. First, it requires only minimal information about a relational database. Second, it makes no assumption about the relational database. The relational database can be bad-designed, optimized and denormalized. Both advantages are derived from an analysis of HTML forms – popular user-friendly interface to access relational databases on the current Web.

In the future, our approach will be applied to migrate from the current Web to the Semantic Web. The main reason for this migration is to make the Web content machine-understandable.

Acknowledgement

This research is partly sponsored by ESF under the grant nr. 5766.

References

1. Berners-Lee, T.: XML 2000 – Semantic Web Talk, http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html (2000)
2. Wang, J., Lochovsky, F.: Data Extraction and Label Assignment for Web Databases, In: Proceedings of 12th International Conference on World Wide Web (WWW) (2003) 187–196
3. Stojanovic, L., Stojanovic, N., Volz, R.: Migrating Data-intensive Web Sites into the Semantic Web, In: Proceedings of the 17th ACM Symposium on Applied Computing (SAC) (2002) 1100–1107
4. Erdmann, M., Maedche, A., Schnurr, H., Staab, S.: From Manual to Semi-automatic Semantic Annotation: About Ontology-based Text Annotation Tools, Linköping Electronic Articles in Computer and Information Science Journal (ETAI), Vol. 6, No. 2 (2001)
5. Astrova, I.: Reverse Engineering of Relational Databases to Ontologies, In: Proceedings of the 1st European Semantic Web Symposium (ESWS), LNCS Vol. 3053 (2004) 327–341
6. Kashyap, V.: Design and Creation of Ontologies for Environmental Information Retrieval, In: Proceedings of the 12th Workshop on Knowledge Acquisition, Modeling and Management (KAW) (1999)
7. Dogan, G., Islamaj, R.: Importing Relational Databases into the Semantic Web, http://www.mindswap.org/webai/2002/fall/Importing_20Relational_20Databases_20into_20the_20Semantic_20Web.html (2002)
8. Mfourga, N.: Extracting Entity-Relationship Schemas from Relational Databases: A Form-Driven Approach, In: Proceedings of the 4th Working Conference on Reverse Engineering (WCRE) (1997) 184–193
9. Embley, D.: Toward Semantic Understanding – An Approach Based on Information Extraction, In: Proceedings of the 15th Australasian Database Conference (ADC) (2004) 3–12
10. Yang, Y., Zhang, H.: HTML Page Analysis Based on Visual Cues, In: Proceedings of the 6th International Conference on Document Analysis and Recognition (ICDAR) (2001) 859–864
11. Knoblock, C., Kambhampati, S.: Information Integration on the Web, http://rakaposhi.eas.asu.edu/aaai-i3-tut-all.pdf (2002)
12. Wang, J., Hu, J.: A Machine Learning Based Approach for Table Detection on the Web, In: Proceedings of the 11th International Conference on World Wide Web (WWW) (2002) 242–250
13. Paredis, J.: Learning the Behavior of Dynamical Systems from Examples, In: Proceedings of the 6th International Workshop on Machine Learning (ICML) (1989) 137–140
14. Michalski, R.: A Theory and Methodology of Inductive Learning, In: Machine Learning: An Intelligence Approach, Vol. 1 (1983) 83–134
15. Astrova, I., Stantic, B.: Reverse Engineering of Relational Databases to Ontologies: An Approach Based on an Analysis of HTML Forms, In: Proceedings of the Workshop W6 on Knowledge Discovery and Ontologies (KDO), 15th European Conference on Machine Learning (ECML), 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD), eds. Buitelaar, P. et al. (2004) 73–78
16. Kifer, M., Lausen, G., Wu, J.: Logical Foundations of Object-oriented and Frame-Based Languages, Journal ACM, No. 42 (1995) 741–843

Ranking Web Application Compositions Based on Ontology

Tetsuya Osawa[1], Naoki Fukuta[2], Tadashi Iijima[3], and Takahira Yamaguchi[3]

[1] Graduate School of Science for Open and Environmental Systems, Keio University, 3-14-1 Hiyoshi Kohoku-ku Yokohama 223-8522, JAPAN,
tetsuya@ae.keio.ac.jp,
[2] Department of Computer Science, Shizuoka University, 3-5-1 Johoku Hamamatsu Shizuoka 432-8011, JAPAN,
fukuta@cs.inf.shizuoka.ac.jp,
[3] Department of Administration Engineering, Faculty of Science and Technology, Keio University, 3-14-1 Hiyoshi, Kohoku-ku, Yokohama, 223-8522, JAPAN,
{iijima,yamaguti}@ae.keio.ac.jp,

Abstract. Today, there are too many Web Applications on Internet and it is difficult for a user to find appropriate applications. Furthermore, a task demanded by a user is rather abstract compared to a Web application and a user need to use two or more Web applications to demand a task and do copy and paste to combine them. In this paper, we propose a Web application composition method based on ontology. We have built a system which creates application coordination paths for the target task with the ontology, and composes Web applications automatically for a user not to do copy and paste process. In the method, quality of the composed service is important. The system supports a user by calculating scores of generated paths using evaluation function in order to select the paths which are more useful for a user. In order to evaluate the validity of the proposed method and the system proposed in this paper, we prepared ontology and wrapper of some existing Web applications that cover two case studies we will present and conducted the experiment.

1 Introduction

A Web Application is an application program that has an HTML-based front-end for users to utilize services provided by a remote HTTP server. Today, there are too many Web Applications on Internet and it is difficult for users to find appropriate applications. Furthermore, a task demanded by a user is rather abstract compared to a Web application and users need to search two or more appropriate Web applications to demand a task and do copy and paste to combine applications. For example, a user wants to find a restaurant near a hotel. For demanding this task, they basically query on "Restaurant" in a search engine. And they find that it is necessary to input "Address" in order to search a restaurant with the Web application. Next they must search a Web Application to find the hotel address. Finally, in order to combine two Web applications, users

need to open these two Web pages, input the hotel name on first page, copy the hotel address in part of a result, and paste this copy into the input-form on the page of restaurant search service. Internet users cannot combine existing Web applications into a single application. Considering the situation that many users in a company want to demand a similar task, it drives up substantial costs for the company. In this study, we aim to reduce such costs to support users' "search" and "copy and paste" activity by searching, coordinating and composing Web applications automatically on users' demands.

On the other hand, Web services have been proposed to be used as software components that employ Internet technology effectively. Semantic Web Services[3], which prepare meta-data to services on the Web, are intended to often used to implement their composition and cooperation. OWL-S[4] becomes a standard of denoting services and their coordination.

In this paper, we propose a Web application composition method based on ontology. This ontology is defined by OWL-S. We have built a system which creates application coordinating paths for the target task with the ontology by discovering and coordinating, and composes Web applications which have been wrapped to transfer values. In the method, quality of the composed service is important. The system supports users by calculating scores of generated paths using evaluation function in order to select the paths which are more useful for user. In order to evaluate the validity of the proposed method and the system proposed in this paper, we prepared ontology and wrapper of some existing Web applications that cover two case studies we will present. We show the possibility that users easily look for and carry out much higher quality application compositions by adding and rearranging the evaluation value into the system.

2 Decompose User Demand to Service and Compose Web Application

We define three levels of granularity(Fig. 1): "task level" that represents user's task, "service level" for representing specifications of each service, and "application level" that represents real Web applications provided for end-users. Agent automatically decomposes a task to appropriate services, and make mapping among services and Web applications. We show a method to compose useful service execution paths by coordinating Web applications and ranking the composed paths.

2.1 Decomposition from the Task Level to Ontology Structure

A task demanded by a user is rather abstract compared to a single Web application. Therefore, it is important to establish a technique to make unique and direct mapping of the task level and the application level. This study seeks a solution to this subject by preparing OWL-S based ontology that fills the gaps between the task level and the application level.

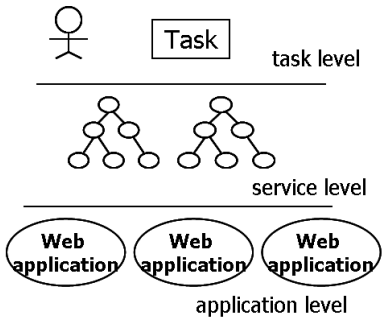

Fig. 1. Three-level of granularity scale

Usually, a user's task is constructed by multiple subtasks. For example, a task "Search restaurant near a hotel" can be decomposed to several subtasks such as :(1) finding address of the hotel, (2)searching restaurant with the address. Information needed to decompose a task to respective subtasks is described in ontology as the ProcessModel of OWL-S.

This study employs the "Precondition" and "Effect" of each part for the component strategy of the OWL-S structure. The OWL-S structure is composed by repeated decomposition of the task structure in terms of Preconditions and Effects. After the OWL-S structure is determined, each structural component is mapped onto the corresponding Web applications on the application level.

2.2 Ontology Construction

We prepared two types of ontology to describe the service level; service ontology and object ontology. Each ontology is outlined below.

Service ontology We collected some existing Web applications which have appropriate facilities for two case studies we will present. A Web application is described as a model with an input and an output parameter. Fig 2 shows a part of the constructed service ontology.

Object ontology The I/O parameter of each service should be identified to constitute coordination path. The object ontology that deals with all I/O parameters of the target services was constructed. All attributes that would appear in the I/O parameters of Web applications were specified in the constructed object ontology. The hierarchical structure of all the extracted attributes is based on WordNet[6]. Fig 3 shows a part of the constructed object ontology.

2.3 Design of Service Cooperation System

The I/O parameters may be verified using the following two procedures: the first is "strict matching" (Fig 4). All attribute sets are mapped onto the same

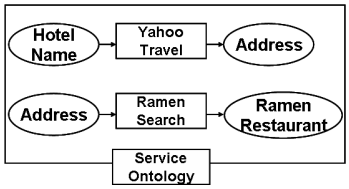

Fig. 2. Part of service ontology

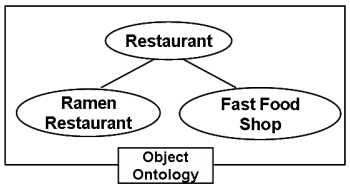

Fig. 3. Part of object ontology

nodes in the object ontology. In this case, the input attributes set of the service performed earlier and belongs to the common set of two services. The second is "extended matching"(Fig 5). All attribute sets, wherein the input attribute set of the service performed later is no subset of the output attribute set of the service performed earlier. Thereby, a path wherein the output attributes of the service performed earlier is included in the child node of the input attributes of the service performed later. The extended matching verifies only the child nodes of the input attributes of the services performed later; neither a child's child (grandchild) node nor a child of the parents (brother) node is included.

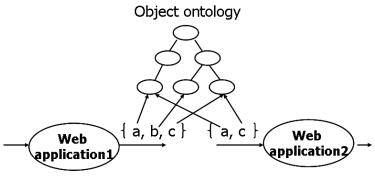

Fig. 4. Strict matching

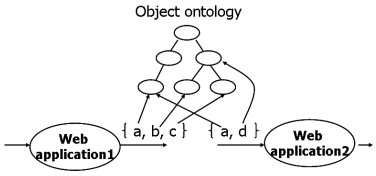

Fig. 5. Extended matching

2.4 Criteria of Evaluating Service Execution Paths

The system generates a lot of service execution paths along with increasing Web applications. This means that the paths include a lot of meaningless composition which is consisted only by matching input and output attribute. In this paper, the system supports users by calculating scores of generated paths using evaluation function in order to select the paths which are more beneficial for a user. We set three criteria for making evaluation function to evaluate the composed service execution paths.

- Length of execution path

The number of web application in a execution path. The short length of a service execution path was considered to be important because a long and complicated service execution path may require time cost and deteriorate output information.

- The kind of coordination

Whether existence of extended matching or not. There is a possibility if a service execution path that allows extended verification may create a semantically wrong path.

- Information provided by the Web application

In this study, we give the evaluation value to all web application from a stand point of abundance of information, response time.

Based on these criteria, we defined the evaluation function to assign the scores to service execution paths.

3 Experimental Results of Service Composition

In this chapter, we describe the experiment conducted by using the system. First we show the outline of the system. After that we also describe the application coordination path in service level, an evaluation function, and experimental result in application level. The system creates application coordination paths and composes Web applications according to a user's requirement for a user not to do copy and paste process. In that case, the experiment aims at verifying whether the candidate of Web application coordination path can be created with based on two ontology and evaluation function in service level, and also whether the system create composed Web application according to candidate path in application level. We collected 10 Web applications which cover two case studies, and prepared two ontology and wrappers of each Web application. Case studies are as follow. Case study #1: input attribute is "Company name", and output attribute is "Stock value in Yen". It means the situation that a Japanese invester wants to know stock quotes of some major US companies in Japanese Yen. Case study #2: input attribute is "Hotel name", and output attribute is "Restaurant". It means that a person who goes on a business trip want to find good restaurant near the hotel.

3.1 System Outline

In this section, we describe the process of composing Web application as a system outline. System outline is Fig 6.

First, a user input I/O parameter into the system in order to demand their task (Task level). The system creates the candidates of Web application coordination paths based on I/O parameter and ontology as described in 2.s, calculates scores of all generated paths based on evaluation function, shows the ranked candidate paths to a user. Next, the user selects a candidate path in ranked paths and input the value in input form (Service level). The system composes the Wrapped Web application according to the coordination selected by the user, and inputs the value to first Web application. So the value is changed by Web applications by being passed between them, and finally the system shows the worked value to the user as a result to demand task (Application level)

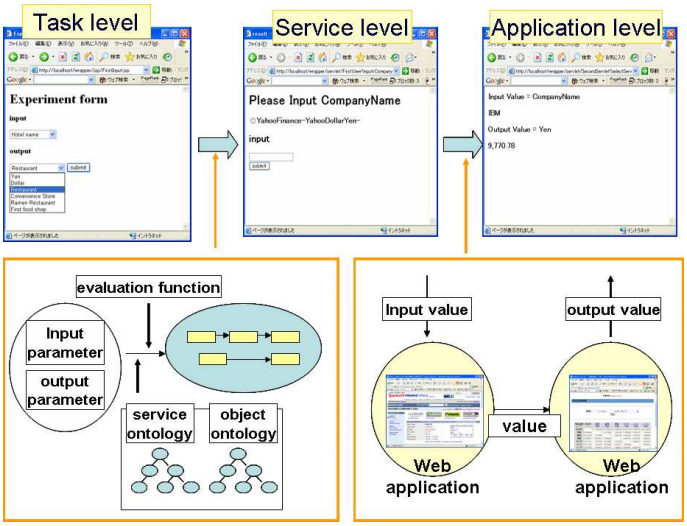

Fig. 6. System Outline

3.2 Generation Experiment of Service Execution Path

In this section, we verify the way in which an Web application coordination path is generated dynamically by using service ontology and object ontology. First, we define what is given by a user when carrying out a task (input attribute set), and the target item for task achievement (output attribute set). Then, an application coordination path is assembled dynamically, extending from the input attribute set, until it outputs a set of target attribute. The deeper a search become, the larger the search space grows and the harder it will be to process it using a computer. In order to prevent this, a maximum search depth is restricted to 3.

3.3 Evaluation Function

In this section, we evaluate the qualities of all the execution paths which are obtained in Section 3.2 by calculating scores of all generated paths. We define the evaluation function based on the criteria shown in Section 2.4 to calculate scores of generated paths. In this case study, we set these evaluation criteria of a path; the length of generated path, the kind of coordination, abundance of information, and response time. Moreover, each evaluation criterion is weighed as shown in Table 2. In this case study, the short length of a service execution path was considered to the most important because a long and complicated service execution path may require time cost and deteriorate output information. Next importance weight is assigned to the abundance of the information which can be treated by the constituted Web application. In coordination of the Web application, it will be thought that in case to pass the information from the one Web

application to another, the cooperation will stop if constituted Web application treat little amount of information. Moreover, nonexistence of extended matching was assigned to high weight, for a service execution path that allows extended verification may create a semantically wrong path.

Table 1. Weight of each evaluation criterion

	Path length	Abundance of information	Response time	Extended matching
Weight	1.0	0.8	0.6	0.7

In this experiment, the system calculates scores of generated application coordination paths by using the following evaluation function.

$$V = \frac{I \times 0.8 + R \times 0.6 + E \times 0.7}{L \times 1.0} \quad (1)$$

Here, "L", "I", "R" denote the path length, the amount of information, and the user evaluation of response time. "E" is 0.5 if the next service is extended verification - otherwise it is 1.0. Table 3 shows the each Web applications' evaluation value of abundance of information and evaluation of response time. The value is based on a questionnaire completed by members of our laboratory.

Table 2. Web application information

Service Name	Information	Response	Explanation of Service
YahooFinance	0.90	0.87	Company name → Stock value in dollar
YahooDollarYen	0.73	0.87	Dollar → Yen
TabimadoAddress	0.83	0.83	Hotel name → Address
YahooTravel	0.70	0.80	Hotel name → Address
Aitel	0.63	0.60	Hotel name → Address
MapfanSearchStation	0.73	0.73	Address → Nearest station
Gurunabi	0.83	0.73	Address or Station → Restaurant
YahooRestaurant	0.90	0.90	Address → Restaurant
RamenBank	0.67	0.77	Address → Ramen Restaurant
RamenBaca	0.30	0.64	Address → Ramen Restaurant
MacDonald	0.70	0.70	Address → Fast food shop

3.4 Result of Each Case Study in Application Level

In this section, we verify whether the candidate paths coordinate correctly as a composed application. We have already shown the result view of case study #1 in Section 3.1, so we describe the result of each case study #2 in Fig 7. The result verify that the system can creates and ranks application coordination paths based on ontology and evaluation function, and it can compose Web applications for transferring input value and outputting final result which a user searched to demand a task.

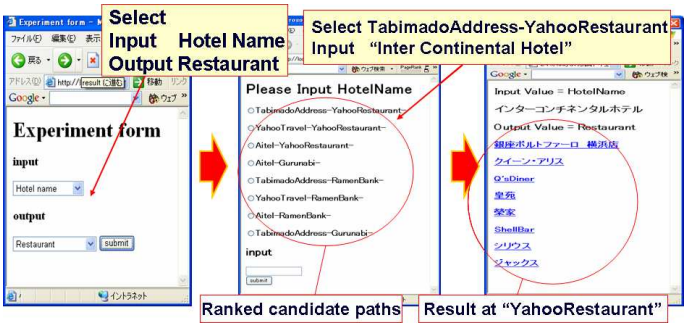

Fig. 7. Result flow in #2

4 Conclusion and Future Work

In this paper, we proposed a Web application composition method based on service ontology and object ontology. We described the outline of dynamic generation of application coordination paths using an automatic Web applications composition system. The system also supports users by calculating scores of generated paths using evaluation function in order to select the paths which are more beneficial for users.

The experimental result in Section 3.2 shows that Web applications can be coordinated and composed automatically. Furthermore, the result in Section 3.4 suggest the possibility that the composed application can be selected which are more beneficial for user by making two type of simple ontology and using evaluation function based on criteria shown in Section 2.5.

We are now registering wrapped Web applications into the system and constructing service ontology. As a future work, we will conduct more datail experiment by using the system and refine the evaluation function reflecting user favor and evaluation.

References

1. Kimihito Ito, and Yuzuru Tanaka: "A visual environment for dynamic web application composition", ACM PressPages: pp.184-193, 2003
2. A. Ankolekar, F. Huch, and K. Sycara: "Concurrent Execution Semantics of DAML-S with Subtypes", *In Proc. of 1st International Semantic Web Conference(ISWC2002)*, pp.318–332, 2002.
3. Sheila A. McIlraith et al.,"Semantic Web Services", IEEE Intelligent Systems, Mar./Apr.2001, pp.46-53.
4. OWL-S 1.0: http://www.daml.org/services/
5. Liangzhao Zeng, Boualem Benatallah, Marlon Dumas, Jayant Kalagnanam, and Quan Z. Sheng: "Quality driven web services composition", *International World Wide Web Conference*, pp.411–421, 2003.
6. WordNet: http://www.cogsci.princeton.edu/ wn/

Applying MDA in Enterprise Application Interoperability: The PRAXIS Project

Vassilios Karakoidas[1], Stephanos Androutsellis-Theotokis[1], Diomidis Spinellis[1], Yannis Charalabidis[2]

[1] Athens University of Economics and Business, 76 Patission Str., Athens, 10434, Greece, {stheotok, bkarak, dds}@aueb.gr

[2] Singular Software SA, 23rd km Athens-Lamia Road, Ag. Stefanos, 14565, Greece, yannisx@singular.gr

Abstract: This paper elaborates on the application of the MDA approach for achieve interoperability among existing Enterprise Applications, such as Enterprise Resource Planning (ERP) and Customer Relationship Management (CRM) systems, through utilizing XML/B2B interconnection standards, developing components and modifying existing systems where necessary. We present the PRAXIS project design as a specific case study of this approach, describing the way in which interoperability-oriented high level model mappings are providing the means for guiding the model-driven development of interoperable Enterprise Applications. Specifically, we focus on the transformation processes that drive the Computation Independent System model into the Platform independent and finally the Platform Specific Model, including our final design decisions for the PRAXIS system.

The problem of Interoperability

Since the 1960s, Enterprise Applications development suffered from a lack of available technological solutions for organizing business processes [19]. To deal with this issue, software engineers worked in the direction of developing software in the areas of Relational Database Management Systems (RDBMS) and Rapid Applications Development IDE (RAD). Researchers also developed methodologies for formalizing the design of Information Systems such as the Soft Systems Methodology (SSM) [21] [22] [23] etc. Recently, IEEE provided a series of guidelines for efficient system design and development [20].

During the last two decades, many sophisticated systems were also developed to provide common ground for enterprise application development. These Enterprise Resource Planning systems (ERP) are widely used nowadays in the design of modern Information Systems (IS).

In the last few years, however, the focus has shifted from the development of software systems to the integration between them, in other words dealing with the problem of interoperability.

Interoperability is defined as [1]:

"… the ability of two or more systems or components to exchange information and to use the information that has been exchanged".

This is a particularly complicated task. Most systems developed and in use today by organizations were deliberately alienated with each other, making it very complex to find common ways to share business related information.

Driving Forces of Interoperability

While working on the PRAXIS system [2], we made some interesting observations regarding the problem of interoperability between enterprise applications.

Figure 1 shows an attempt to categorise the driving forces that affect interoperability in Enterprise Applications development.

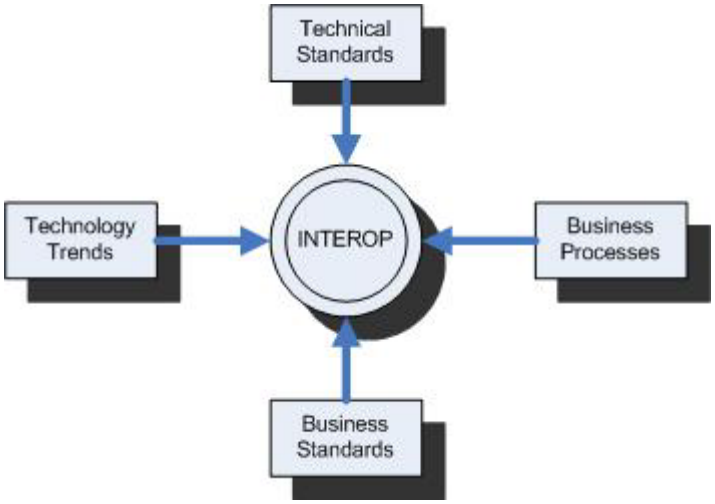

Figure 1 – Driving Forces of Interoperability in Enterprise Applications

Technical Standards: Enterprise Applications are based upon modern or legacy technical standards, such as XML [8], CORBA, and Java. In trying to make a system interoperable, one has to consider a wide set of standards in order to implement the solution.

Technology Trends: Technology trends pose serious restrictions to interoperability. In fact, it is common practice that the implementation of an interoperable system is attempted with the use of the most popular technology instead of the most effective one. For example, the way in which XML [8] and Web Services [13] affected Business 2 Business standards such as ebXML [9] and UN/CEFACT [10]. Similarly, according to recent research papers, ontologies are expected to be used as key technology for Enterprise Modeling [18].

Business Standards: These are often used as the common language between enterprise systems. Attempts to design an interoperable system often depend on the usage of one or more business standard, for example, EDI [11], ebXML [9], RosettaNet [12] etc.

Business Processes: It is a common case for enterprises to achieve a level of interoperability in the business process level. In fact, in order to design such a bridge, one must reengineer a set of common business processes between the enterprises. In order to face interoperability issues, an enterprise must perform some “light” Business Process Rengineering (BPR). EbXML [9] is a widely accepted standard that is trying to solve this problem.

Considering the aforementioned facts, we attempted to find a formal and concrete way to design the PRAXIS system.

Case Study: The PRAXIS Project

The main objective of the PRAXIS system [2], which is particularly targeted towards SMEs, is to allow the interconnection of existing Enterprise Applications, such as Enterprise Resource Planning and Customer Relationship Management (CRM) systems, as well as their seamless interconnection with corresponding software applications of the public sector and financial institutions. This is achieved through the utilization of XML/B2B interconnection standards to support interoperability, the development components and the modification of existing systems where necessary.

The goal is to support a variety of transactions, ranging from invoicing and sales cycle support to bank and tax payments.

It is expected that the SMEs which will adopt the PRAXIS system will experience a significant reduction in the required time and the rate of errors involved in carrying out the various transactions, which in turn will offer an overall competitive advantage.

Using a Model-Driven Architecture approach

The Model Driven Architecture (MDA) concept aims to facilitate the design and development of wide-scale enterprise applications in an evolvable and flexible way, with particular emphasis on interoperability [3].

MDA defines an approach that separates the system functionality specification from its implementation on a specific technology platform. In this way, the system architecture can be language, vendor and middleware neutral.

Through the MDA approach, systems are modeled at the following three different levels of abstraction and platform/technology dependency:

- The Computation Independent (or business domain) Model (CIM) is one in which the computational details are hidden or as yet undetermined.

- The Platform Independent Model (PIM), expressed in UML [15], describes computational components and their interactions in a

platform-independent manner. The PIM represents the logical view in which the composition and behaviour of all components (but not their implementation) are fully specified.

- The Platform Specific Model (PSM) is expressed in terms of the software development platforms, software standards and network protocols of the specification model of the target platform.

Since the PIM, by definition, does not contain technical details, it is envisaged that the PIM will be mapped to one or more platform-specific models (PSM). The PSM is a refinement of the PIM to target platforms such as Microsoft .NET (COM+), Enterprise JavaBeans (EJB) or the CORBA Component Model (CCM). The PSM represents the source code or its UML representation. There can be as many PSM as there are different implementations of a given PIM.

System design and modeling

Our system was designed using three levels of abstraction as defined in the MDA methodology.

Computation Independent Model Level

At the most abstract, Computation Independent System (CIS) Model level, the proposed system would be described as shown in Figure 2.

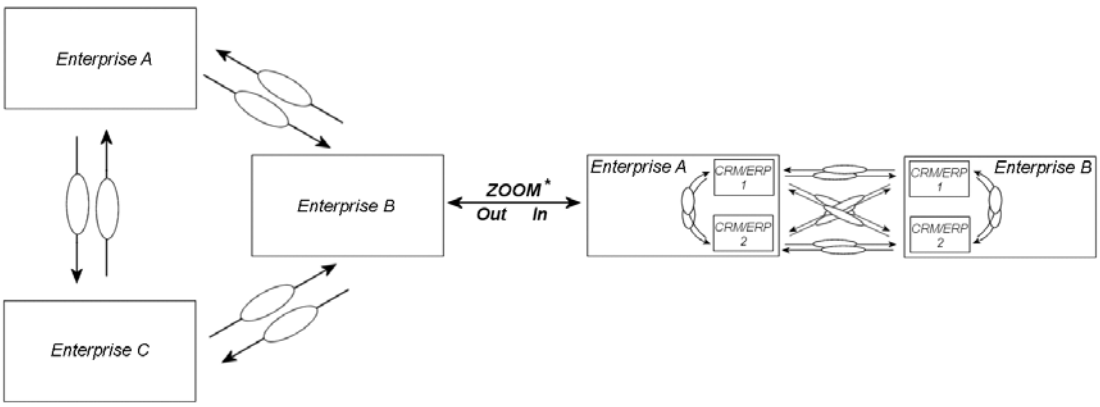

Figure 2 - ***The Computation Independent System Model for the PRAXIS system***

This abstract model illustrates the interoperability requirements for different software applications owned by the same or different enterprises that are expected to interact with each other. The various transactions schematically illustrated in Figure 2, entail a variety of interactions between the applications and lay out a set of complex functional and non-functional requirements for the system to be designed and implemented, which must be examined at the lower modeling levels.

It should be noted that the CIM model is abstract enough to encompass any system architecture, including both client-server and distributed approaches. The use of MDA describes and represents interactions between entities, and not the way in which they will be implemented. The choice of architecture and components that will be utilized is up to the next model level.

Platform Independent Model Level

In refining the CIM model to produce the Platform Independent Model (PIM), the top-level system architecture needs to be designed. In the PRAXIS system design phase it was decided [2] to adopt a client-server architecture, consisting of a central server performing communication, mediation and store-and-forward activities, and a set of client applications that will run on the connected enterprise sites.

The server, which does not appear in the CIS model, needs to be part of the PIM model, in order to produce a valid system description that will be further refined at the platform-specific level.

Figure 3 illustrates the PIM level model for (a subset of) the PRAXIS system (simplified for clarity). The model is divided in two main modules, the server and client.

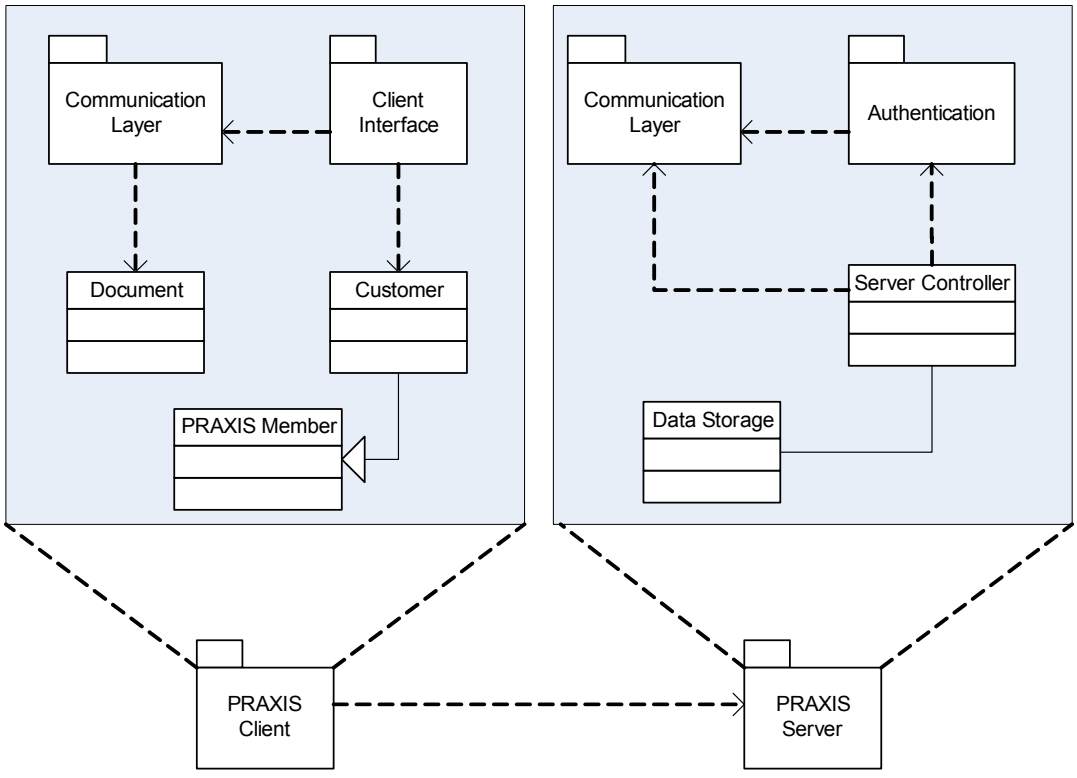

Figure 3 - ***Platform Independent Model for the PRAXIS system***

The PIM design is based on UML notation [15] and provides the bridge between the abstraction and the realization.

Platform Specific Model Level

The transition from the PIM to the PSM model consists of defining specific technologies and infrastructures for the various components of the system. These mappings fill the gap between the abstract model and the Platform Specific Model. The realization of our system is depicted in Figure 4.

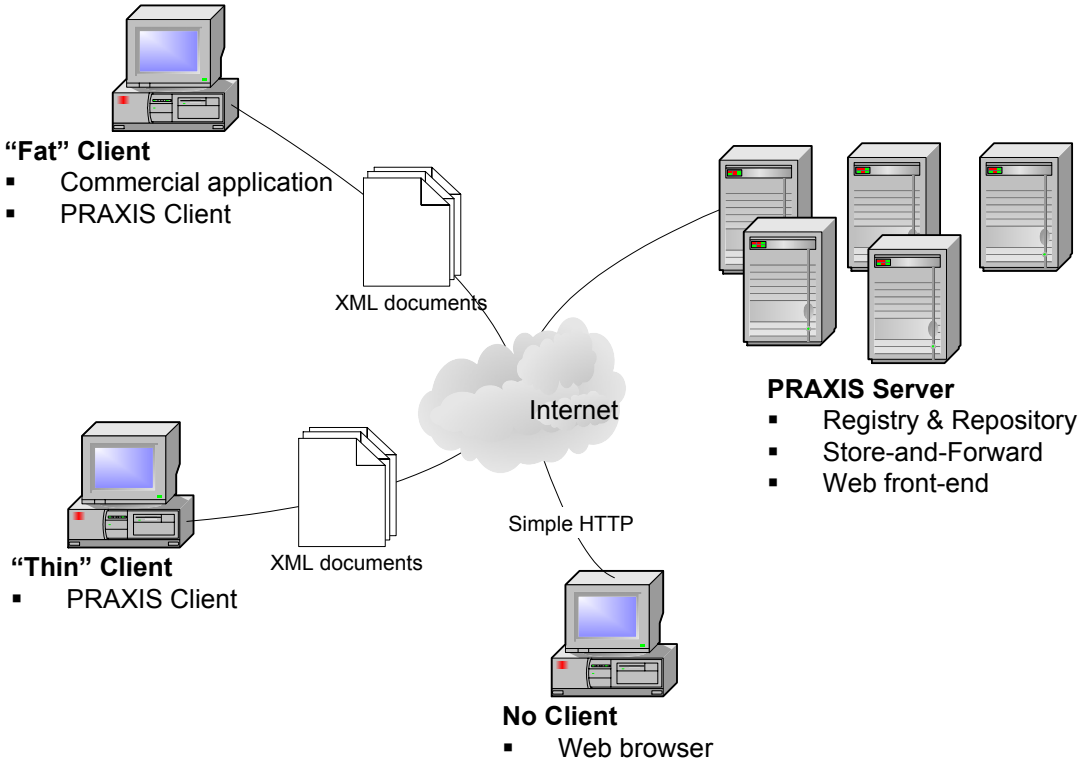

Figure 4 – PRAXIS System Architecture

In the case of the PRAXIS project, the central choices were the following:

- The server was implemented on the .NET platform, using a mixture of C++ and C# code.
- Data is stored in an MS SQL Server Relational Database.
- The following three types of clients were implemented:
  - A *Fat (Thick) Client*, based on an existing ERP system.
  - A *Thin Client, a* typical win32 application that implements the basic functionality of the system. This will be implemented in Visual Basic .NET.
  - A *Web Client*, a web application that provides limited access for enterprises, which do not have an ERP system installed.

These three types of clients were provided in order to reach out for SMEs that do not have the capability to support expensive ERP systems.

A more detailed architecture schema is provided in
Figure 5.

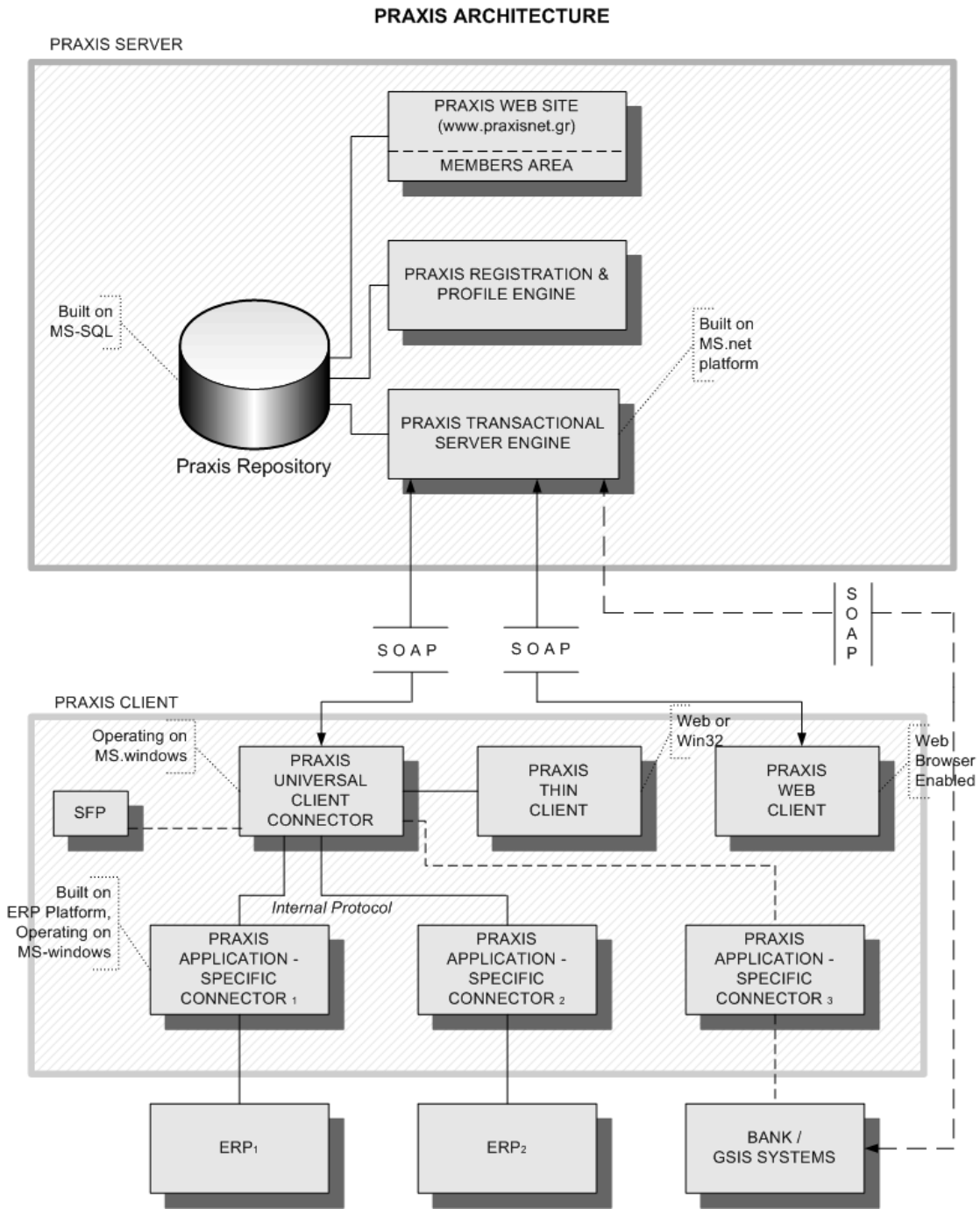

Figure 5 – PRAXIS system architecture in detail

The communication is layered as illustrated in Figure 5. The base protocol is based on SOAP. Other protocols that might be needed to achieve interoperability between distinct systems are supported through layering them on top of SOAP, making the necessary transformations where appropriate.

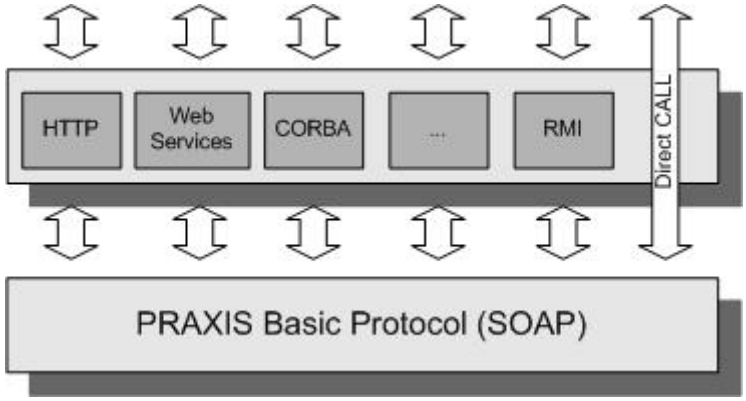

Figure 6 – PRAXIS Protocol Layers

Conclusions and Future Work

This paper has presented the application of the MDA approach in the design of a client-server application to support interoperability among enterprises. It has illustrated the refinement steps that lead from the more abstract to the technology-specific models, pointing out the specific patterns of the design. We are convinced that these patterns can be extracted and later used as common practice in similar situations. A direct extension of our work may include the employment of Ontology-based approaches to be defined in order to provide common knowledge [14] [16] [17] [18].

References

[1] Institute of Electrical and Electronics Engineers. *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries.* New York, NY: 1990.

[2] Yannis Charalambidis, Vassilios Karakoidas, Stephanos Androutsellis-Theotokis and Diomidis Spinellis, *Enabling B2B Transactions over the Internet through Application Interconnection : The PRAXIS project*, e-Challenges 2004, Vienna, 27-29 October.

[3] Joaquin Miller and Jinshu Mukerji, *Model Driven Architecture*, July 2001, Available online at http://www.omg.org/mda/

[4] John Daniels, *Modeling with a sense of purpose*, IEEE Software, January / February 2002,

[5] Alan Brown, *An introduction to Model Driven Architecture Part I : MDA and today's systems*, January 2004, Available online at http://www-106.ibm.com/developerworks/rational/library/3000.html

[6] B. Selic, *The pragmatics of Model-Driven Development*, IEEE Software, Vol. 20, September 2003

[7] D. Frankel, *Model Driven Architecture : Applying MDA to Enterprise Computing,* Wiley Press, 2003

[8] Tim Bray, Jean Paoli, C.M. Sperberg-McQueen, Eve Maler, and Francois Yergeau. *Extensible markup language (XML) 1.0. Technical report*, W3C, February 2004. Available online at http://www.w3.org/TR/2004/REC-xml- 20040204

[9] ebXML Requirements Team. *ebXML requirements specification*

v1.06. Technical report, ebXML, May 2001. Available online at http://www.ebxml.org/specs/ebREQ.pdf

[10] UN / CEFACT. United nations centre for trade facilitation and electronic business. Available online at http://www.unece.org/cefact/

[11] UN / EDIFACT: UN directories for electronic data interchange for administration, commere and transport (UNECE). Available online at http://www.unece.org/trade/untdid/texts/d100 d.html

[12] RosettaNet. *Rosetta Implementation Framework (RNIF): Core specification* Technical report, RosettaNet, March 2002. Available online at http://www.rosettanet.org/

[13] David Booth, Hugo Haas, Francis McCabe Eric Newcomer, Iona Michael Champion, Chris Ferris David Orchard, *Web Services Architecture,* Technical Report, W3C, February 2004. Available online at http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/

[14] F. Vernadat, *Enterprise Modeling: objectives, constructs and ontologies*, Workshop on Enterprise Modeling and Ontologies for Interoperability, 16th International Conference on Advanced Information Systems Engineering (CAISE), Riga, 2004

[15] Berio, G., Petit , M., *Enterprise Modelling and UML: (sometimes) a conflict without a case*. In Proc. Of Concurrent Engineering Conference 03, July 26-30, Madeira Island, Portugal, 2003

[16] Wand, Y,*Ontology as a Foundation for Meta-Modelling and Method Engineering*. In Information and Software Technology, Vol. 38 (1996) 281-287

[17] G. Guizzardi, G. Wagner, *A Unified Foundational Ontology and some Applications of it in Business Modeling*, Workshop on Enterprise Modeling and Ontologies for Interoperability, 16th International Conference on Advanced Information Systems Engineering (CAISE), Riga, 2004

[18] M. Lenzerini, M. Missikoff: *Ontologies for Interoperability*, Workshop on Enterprise Modeling and Ontologies for Interoperability, 16th International Conference on Advanced Information Systems Engineering (CAISE), Riga, 2004

[19] Linthicum S.D., *B2B Application Integration – e-Business-Enable Your Enterprise*, Addison-Wesley, 2001

[20] IEEE Guide for Developing System Requirements Specifications Std 1233, *IEEE Standards Software Engineering: Volume One Customer and Terminology standards* 1999 Edition,Available online at http://www.standards.ieee.org/

[21] Checkland, P, *Towards a system-based methodology for real-world problem-solving*, Journal of Systems Engineering, Vol. 3, No. 2 - reprinted in *Systems Behaviour*, Third Edition (1981), The Open Systems Group, Open University Publishing, UK

[22] Checkland, P. & Scholes, J.(1990) *Soft Systems Methodology In Action,* John Wiley & Sons, Chichester UK

[23] Checkland, P. *Information, Systems and Information Systems: Making Sense of the Field,* John Wiley & Sons, Chichester UK.

Ontology based Framework for Adaptive Web System

Daisuke Kanjo[1], Yukiko Kawai[1], and Katsumi Tanaka[2]

[1] National Institute of Information and Communications Technology,
3-5 Hikaridai,Seika-cho,Soraku-gun,Kyoto,Japan,
{kanjo,yukiko}@nict.go.jp,
[2] Kyoto University, Department of Social Informatics,
Graduate School of Informatics,
Yoshida Honmachi, Sakyo-ku, Kyoto, Japan,
tanaka@dl.kuis.kyoto-u.ac.jp

Abstract. We describe a framework for sharing and reusing user ontologies, which we call "Adaptation Anywhere & Anytime(A3)". A user ontology is defined as a representation of a user's knowledge and interests which can be used as a user's profiles. The A3 enables a web system with methods for sharing and reusing user ontologies. Individual web systems can adapt their output to each user through such methods.

1 Introduction

Today, a vast amount of information is available from the web, but most of this information is verbose and not interesting to the user. Thus, techniques for an adaptation or personalization are becoming more important to locate and present an appropriate information. For such adaptation or personalization, it requires a user's profile which represents a user's preference, knowledge and so on. To acquire a user's profile, some systems explicitly require each user to input her/his profiles, or complete a questionnaire regarding personal information. However, users find it tedious to repeat such tasks for each of the many websites they visit. Automatic acquisition of user profiles can solve this problem. In addition, it is important to share and reuse a user's profile. If a user is required to input her/his profiles to individual system, she/he must perform a similar procedure repeatedly. This makes a user boring. So sharing or reusing of user profiles become more important.

To enable automatic acquisition of user profiles, and the sharing and reuse of such user profiles by various web sites or web applications(in the following, we describe both of these as web systems), we propose a framework called "Adaptation Anywhere & Anytime(A3)". This framework provides web systems with methods to acquire a user's profile based on the interaction between a user and a web system, and to locate appropriate contents using the acquired user profile. A web system can easily adapt its output to a user using methods provided by A3. Within the A3 framework, a user's profile is represented by a user ontology, which is a classified tree consisting of web resources and categories. A web resource is an item which has been presented to the user -for example, an item of merchandise from an e-shopping site or news story from a news site. A user

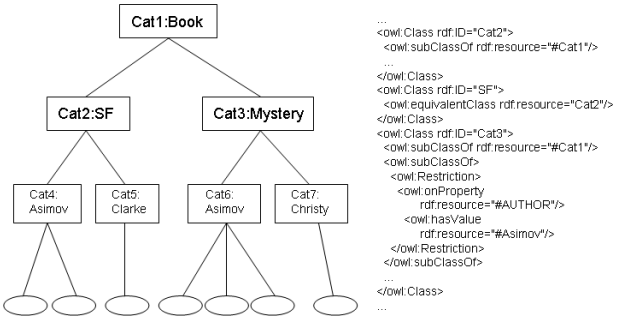

Fig. 1. Example of user ontology

ontology is automatically constructed by adding a resource to the user ontology whenever it is reasonable to presume that the user acquired knowledge about the resource, for example, when the user has input data about the corresponding item. Then, A3 categorizes it to a suitable category and reconstructs the user ontology when necessary. The user is not required to perform any action. The user ontologies thus created can be shared by web systems built on A3 framework, which allows a web system to adapt its output according to a user ontology created by other system.

Furthermore, for easy implement of an adaptive web system through the A3 framework, we provide web system builders with implementation methods using XSLT and its extension[4]. If a builder writes an XML document containing resources which can be presented to a user and writes an XSLT stylesheet specifying target resources by an XSLT and its extensions, A3 components can select and sort resources according to each user's preferences based on the user ontologies, and can transform these into an another document. The selected resources will then appear in the transformed document. An original XSLT transforms all resources matching a template represented by an element such as `xsl:template`. In cotrast, with A3, when an XML document is transformed into a different document, target resources can be selected and sorted based on the user preferences represented by user ontologies.

2 User Ontology

Figure1 shows an example of a user ontology. We defined a user ontology as a classified tree which can represent a user's knowledge, interests, and so on. Each user has his or her own user ontology. It consists of two elements: a resource(shown as an ellipse in Fig1) and its category(shown as a rectangle). Each category has its own restriction, which is an attribute that any resource categorized into the category must have. Furthermore, a category inherits restrictions from higher categories.

A resource is an item included in content selected and presented to the user. For example, it can be an item of marchandise from a ne-shopping site, or news stories from a news sites. A user ontology is constructed by adding each resource as an instance to the user ontology, when it can be reasonably presumed that the user acquired some knowledge about the resource. When adding a resource, the appropriate A3 component(the "UserOntology Manager", described in Section 3.) searchs for a suitable category to which the resource can be assigned, and then categorizes the resource accordingly. Then, the A3 framework reconstructs the user ontology to ensure the restrictions of a upper-categories have precedence, for example, by making a new category when necessary. A resource added to a user ontology can be considered an instance, and a category can be considered a class.

User ontologies are written using RDF/S[1, 3] and OWL[5] (Fig.1, right),as are resources and their attributes. RDF/S and OWL are common descriptions proposed for use in a Semantic Web[2]. These descriptions enable the sharing and reuse of user ontologies.

2.1 Semantics of a user ontology

In this section, we describe what kinds of information can be acquired from a user ontology.

What the user knows: As explained above, a user ontology is constructed by adding a resources whenever it can be reasonably presumed that the user has acquired some knowledge about the resource. The existence of an instance representing a resource `r` in a user ontology thus means that "the user `u` has knowledge regarding the resource `r`" – that is, `know(u, r)` is true[3]. When constructing a user ontology, the attributes of a resource are used to assign the resource into a suitable category. This requires an assumption that "the user `u` knows that the resource `r` has attributes `a1, a2, ... , a`$_n$" – that is, `know(user, has(r, a1)), know(user, has(r, a2)), ..., know(user, has(r, a`$_n$`))` are true. Moreover, because each category inherits its restrictions from the upper-categories, it is reasonable to consider that a relation between resources is known.

What the user prefers: Generally, users have more knowledge about resources they consider interesting than about resources in which they are not interested. Consequently, a user will probably be more interested in a resource categorized into a category containing many other resources than in one categorized into a sparsely populated category. The common attributes shared by such interesting resources are also likely to be important to a user. Therefore, a user will probably be interested in resources with many of these common attributes.

[3] Strictly speaking, this means"The system s believes that the user u has knowledge regarding the resource r" – that is, `bel(`s`, know(`u`, `r`))`

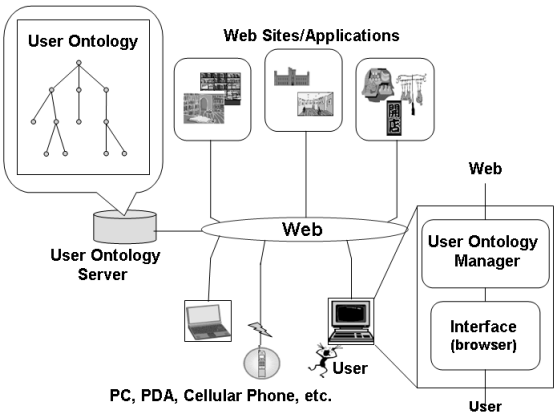

Fig. 2. Overview of A3 framework

When constructing a user ontology, a restriction of an upper category should become a common attribute of resources in the user ontology whenever possible. Thus, the restrictions of upper-categories are more important than lower-category restrictions.

3 A3 Framework

Figure2 shows an overview of the A3 framework. A user accesses web systems through an interface on a device such as a PC, PDA, or cellular phone. The framework consists of two main components: "User Ontology Server", and "User Ontology Manager".

3.1 User Ontology Server

"User Ontology Server(UOS)" has two primary functions; the storage and delivery of user ontologies. When a user begins to use a web system, the UOS specifies the user and sends a stored user ontology to the "User Ontology Manager(described in the following)". The user ontology is used as a user profile to adapt the output to the user, and then is reconstructed by adding resources. When the user finishes using the web system, the user ontology is returned to the UOS for storage.

Each user ontology can be shared among the web systems built on the A3 framework and repeatedly used to adapt output for that user. And web systems built on A3 can use user ontologies constructed by another web systems. This sharing and reuse of user ontologies helps solve the problem where a system cannot adapt its output until after it acquires appropriate user information(i.e. the *cold-start problem*).

3.2 User Ontology Manager

"User Ontology Manager(UOM)" has two premary functions: user ontology construction and resource selection using user ontologies. In the following, we consider an example where an XSLT and its extensions are used to implement an adaptive web system.

When a user begins to use a web system, an XML document and an XSLT stylesheet is sent to the UOM from the web system. These documents are written by the builder of the web system. Also, XML documents can be generated dynamically from a database of the web system. The XML document contains target resources – that is, items such as marchandise or news stories which the web system wants to present to the user. The XSLT stylesheet also specifies target resources and how the XML document should be transformed. While transforming the XML document, the UOM selects resources appropriate to the user's interests or knowledge, and adds a new CGI function to add a resource.

The UOM can calculate weights for individual resources and rank them accordingly. Target resources are specified by the `xsl:apply-template` element which has the`a3:sort` element as a child element. This is written as follows in an XSLT stylesheet:

```
<xsl:apply-templates>
    <a3:sort maxN="20" />
</xsl:apply-templates>
```

The weight of a resource in category C_n is given by

$$\sum_{k=1}^{n} (\frac{1}{2})^{k-1} * w_k * d_{tmp}$$

where w_k is given by p/q. Here, p is the number of resources which satisfying the restriction of C_k in the user ontology, and q is the number of resources in the user ontology with the attributes $Attrs$($Attrs$ are attributes common to all resources matching `xsl-template` element). d_{tmp} is a numerical value from 0 to 1 determined by elapsed period since a resource was last categorized into C_k. The highest ranked resources are then selected and presented as the result of an XSLT transformation. Since resource selection is done at the UOM, builders of web systems do not need to invent a ranking algorithm and implement it within their systems. Note that this is just a example of the calculation and there are many other conceivable ranking methods and algorithms.

A web system built on A3 must be able to determine when a resource is added to a user ontology. Generally, this decision should be made through input from the user. In popular web systems with an HTML format, the `input` element is used to receive an input. If the builder of a web system specifies an `input` element with an `a3:add-resource` element as shown below, the UOM adds the new CGI function(``example.cgi'', in this example) to the document generated by XSLT translation while the translation is occuring.

```
01:  <a3:add_resource resource="example">
02:     <form ... action="cgi-bin/example.cgi">
03:       <div>attribute A: <input ... name="ATTR_A"></div>
04:       <div>attribute B: <input ... name="ATTR_B"></div>
```

```
05:            ...
06:       <input type="submit" ...>
07:            ...
08:     <\form>
09:  </a3:add_resource>
10:  <a3:resource-template select="example">
11:     <ATTR_A>_ATTR_A</ATTR_A>
12:     <ATTR_B>_ATTR_B</ATTR_B>
13:            ...
14:  </a3:resource-template>
```

This CGI function, `example.cgi`, is executed and informs the UOM that a resource should be added when the user inputs data using the `input` element with the `a3:add-resource` element. The resource and its attributes are defined by the `a3:resource-template` element. When it receives this information, the UOM categorizes the resource into a suitable category and reconstructs the user ontology if necessary.

Note that user ontologies are not provided to individual web systems. This is to protect each user's privacy. A user ontology is far reaching representation of a user's knowledge and interests, so allowing web systems freely access user ontologies would enable a serious invasion of users privacy.

4 Discussion

The current A3 framework gives web systems a limited ability to adapt their output for each user. Despite the current limitations of the A3 framework, we can see certain benefits that arise from implementing a web system on A3.

One example of such an application is a web system which needs to be able to select an appropriate resource from among many resources; for example, e-shopping sites. Users who visit an e-shopping site want only information regarding merchandise consistent with their preferences. Figure3 shows an example of an XML document and XSLT stylesheet for a book recommendation on a book-shopping site. These documents generate web pages(written by HTML) showing books appropriate for the user's preferences. In this way, users can easily get information on books they will be interested in.

To create this book recommendation system, the builder would only have to write these documents. In many cases, the XML document containing target resources can automatically written from a database. This eliminates the need to invent a book ranking algorithm, or implement an algorithm within the system. In other wards, implementation of a system to recommend products becomes very easy.

As an other example, we consider a news site. Most users do not want to read the same news story several times. A user ontology can reveal to a news site which news stories a user has already read so that these news stories are not output to the same user again. When a user does want to reread a news story, the news site can find the desired news story by retrieving it from the news stories previously added to the user ontology as an instance.

4.1 Related work

Personis[7] is a system for creating user models(user profiles) and sharing them among various systems. To construct a user model, though, Personis explicitly requires users to answer questions.

User modeling is discussed in [6] and discussion about ontology mapping or merging is discussed in [8]. These discussions focus on the construction of user models, mapping one ontology onto another, or creating tools to construct user models or ontologies. They do not consider the application of the constructed ontologies.

5 Conclusion

We have proposed and developed the "Adaptation Anywhere & Anytime(A3)" framework to enable sharing and reuse of user ontologies A user ontology is defined as a representation of each user's knowledge and interests. By enabling the automatic construction and sharing of user ontologies among web systems, we hope to relieve users of the tedious task such as one of repeatedly registering and completing questionnares for each web system they use. And we have described a simple method for web site implementation using XSLT. This method makes it easy to create and implement an adaptive web system easy.

Future work remains to be done. The current algorithm for ranking items is very simple and needs to be refined. In addition, the A3 framework does not provide a rationale showing why a particular resource is selected. Generally, users like to know why resources are preferentially selected, so we plan to incorporate a means of providing such information.

References

1. Beckett,D.,RDF/XML Syntax Specfication, http://www.w3c.org/TR/rdf-syntax-grammar/

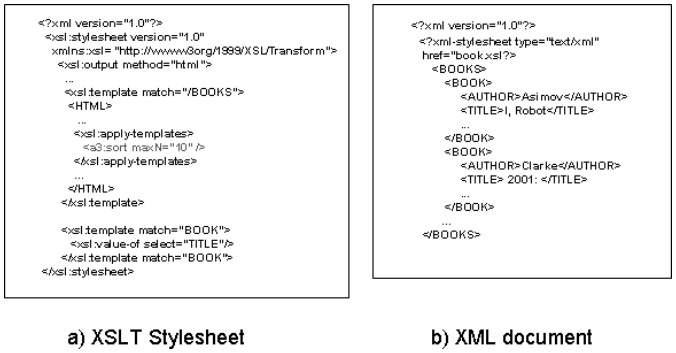

Fig. 3. Example of an XML document and XSLT stylesheet

2. Berners-Lee,T.,Hendler,J.,Lassile,O., The Semantic Web,Scientific American,2001
3. Brickley,D.,Guha,R.V., RDF Vocabulary Description Language 1.0:RDF Schema, http://www.w3.org/TR/rdf-schema/
4. Clark,J., XSL Transformations(XSLT) Version1.0, http://www.w3.org/TR/xslt, 1999
5. Dean,M., Connolly,D., Harmelen,F., Hendler,J., Horrocks,I., McGuinness,D., Schneider,P., Stein,L., OWL Web Ontology Language 1.0 Reference, http://w3.org/TR/owl-ref/
6. Dolog,P.,Nejdl,W., Challenges and Benefits of the Semantic Web for User Modeling, Adaptive Hypermedia and Adaptive Web-Based Systems,2003
7. Kay,J.,Kummerfeld,B.,Lauder,P., Personis: A Server for User Models, In Proc. of Adaptive Hypermedia and Adaptive Web-Based System AH2002, 2002
8. Noy,N.F.,Musen,M.A., The PROMPT suite: interactive tools for ontology merging and mapping. International Journal of Human-Computer Studies, 59(6),2003

Towards an Organizational Knowledge Representation Framework

Rinaldo Pietrantonio[3,4], Massimo Ruffolo[1,2]

[1] Istituto di Calcolo e Reti ad Alte Prestazioni del Consiglio Nazionale delle Ricerche – ICAR-CNR, Università della Calabria, 87036 Rende (CS), Italy
ruffolo@icar.cnr.it
[2] Exeura s.r.l., Università della Calabria, 87036 Rende (CS), Italy,
ruffolo@exeura.it
[3] Dipartimento di Ingegneria Economico-Gestionale, Università di Napoli "Federico II" Piazzale Tecchio, 80, 80131 Napoli, Italy
r.pietrantonio@unina.it
[4] CIES Centro di Ingegneria Economica e Sociale – Scuola Superiore Majise Università della Calabria, C.da Vermicelli, 87036 Rende (CS), Italy
r.pietrantonio@majise.it

Abstract. Organizations are characterized by complex business processes in which different kinds of knowledge are produced and used along a lifecycle. This paper describes an ontology-based organizational knowledge representation framework focused on the specification of so called *core organizational knowledge entities*. The framework constitutes a theoretical support in knowledge management systems and strategies analysis and design.

1 Introduction

In the last years the technological innovations and the social and economic transformations have deeply changed the global market and the enterprises structure all over the world. Knowledge has become one of the most important economic resources with respect to the competitive advantage acquisition turning the traditional enterprises into knowledge intensive organizations [1]. These are characterized by complex managerial, operational and decisional processes [2] involving a number of different forms and kinds of knowledge following a "Knowledge Life-Cycle".

In this scenario Knowledge Management (KM) can really increase the efficiency and effectiveness of the organizational business processes and it can also contribute into the creation of value and the growth of intellectual capital and all the intangible assets within the enterprises. Therefore efficient KM Systems (KMS) and coherent KM strategies are needed to support the organizations in managing knowledge created, stored, distributed and applied within the business process; in particular specific methods and instruments for organizational knowledge elicitation and representation are required to design and implementation of KMS and KM strategies.

Nowadays a number of models, languages and tools coming from computer science and information technology area [5] like ontology and workflow allow organizational knowledge representation directed to build KM organizational and technological infrastructures.

This paper describes an Organizational Knowledge Representation Framework (OKRF) organized as a two level family of ontologies: the first level ontologies formally represent the *Core Organizational Knowledge Entities* (COKE) (i.e. human resources, business processes, knowledge sources, knowledge objects, knowledge-based services), the second level (top level) ontology represents the different topics of the business activities and aims.

The framework provides a representation of the different knowledge forms and kinds following an extension of the KLC model Fraunhofer IPK institute [3], [4] – the representation aims at contributing as theoretical base in supporting the analysis and design of Knowledge Management Systems (KMS) and strategies.

The different forms and kinds of the organizational knowledge are presented in section 2, the KLC phases are present in section 3. In section 4 the organizational knowledge representation framework is shown.

2 Organizational Knowledge Forms and Kinds

Many different kinds of organizational knowledge are wide spread within the enterprises under different forms and distributed in several sources (humans and systems) inside and outside the organization. The most relevant distinctions about the concept of "knowledge" are presented in the follows.

In an operational-based environment Ryle [5] distinguishes between:

- "*know how*", as the practice knowledge, or knowledge used in a operational way. This is task-specific and related (but not similar) to the individual ability in applying tasks;
- "*know that*", as the theoretical knowledge related to the deep (and hidden) causes of the phenomena. The "know-how" is essentially different from the theory because it is expressed in a formal form using rules;

A generally accepted classification is due to Polanyi [15], [16] and extended by Nonaka and Takeuchi [9], [10]:

- "*implicit knowledge*", the knowledge resulting from personal learning processes, present within each organization in terms of its members' personal knowing. This is strictly related to the practice action and the use of brain and body applying the tasks. This kind of knowledge is characterized by "individual" ("know-how" and "know-that") "social" aspects (relationships among people inside and outside of the organization);
- "*explicit knowledge*", generally shared and publicly accessible within the organization through formal storing and processing infrastructures. Explicit knowledge can also be classified basing of the following forms: "structured" (available in database), “semi-structured" (available in intranet and internet web sites: HTML pages, XML documents, etc.) and "unstructured" (available as textual documents: project documents, procedures, white papers, etc.).

In the "Artificial Intelligence" field the traditional classes are:

- "*detail knowledge*", addressing some causal models and relationships based on natural laws;

- "*surface knowledge*", concerning the practice rules that people can learn applying his own task in an efficient way (human experts). The differences between the "surface" and "detail" knowledge concern the knowledge classification based on: "static knowledge", describing domain specific facts, concepts, constraints, states, relationships without references to ongoing actions or in progress processes, "dynamic knowledge", processes and procedures within a specific domain and describing them in their own ongoing progress.

A further classification concerns where the knowledge is coming from: *"internal knowledge"* belongs to the enterprise and/or its own members and system, whereas the *"external knowledge"* is not present inside the enterprise while available by external systems and people or by Internet.

The traditional information systems are able to process only a small portion of the whole organizational knowledge (i.e. explicit knowledge under structured form). Instead a KMS must be able to support the generation, discovery, capture, store, distribution and application of a wide variety of knowledge (i.e. explicit knowledge under structured, semi-structured and unstructured forms and individual and social aspects of implicit knowledge) through related knowledge-based services. Therefore a KMS need knowledge representation capabilities that can be provided by ontology formalisms able to specify the different organizational knowledge forms and kinds and to carry out the same knowledge-based services.

3 The Organizational Knowledge Life-Cycle

Within the business processes different forms and kinds of knowledge are generated, stored, distributed and applied along a life-cycle. A KMS has to provide suitable knowledge-based services to support the KLC. The proposed organizational knowledge representation framework aims at allowing to design KMS knowledge-based services realizing the KLC phases. A so designed KMS improves the performances of the business process and increases the value creation capability of enterprises.

For each phase of the KLC the framework provides a representation of managed knowledge and point out which knowledge-based services are needed to the same KLC phase realization.

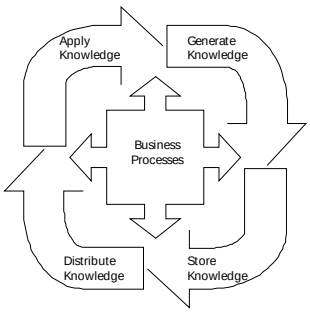

Fig. 1. The Knowledge Life-Cycle

In the following the KLC phases (fig. 1), based on Fraunhofer's IPK KLC model [3], [4], are shown.

"*Generate Knowledge*", aiming at making available the (new) knowledge as generated in several ways at individual (training, learning by doing, problem solving, etc.) and social (communities of practice, project team, etc) level. Just a part of it is directly available under explicit form while the remaining implicit part requires proper capturing methods (questionnaires, lessons learning writing, best practices writing, etc.). When embedded under explicit form in huge amount of structured, semi-structured and unstructured data and information it can be made available through knowledge discovery and classification methods. To realize this phase the KMS knowledge-based services must be based on knowledge discovery, content management, information retrieval, reasoning, etc.;

"*Store Knowledge*", focuses on knowledge extraction and acquisition from all different sources distributed across the organization structures. Knowledge representation methods provided by the framework facilitate the storing of both declarative and procedural knowledge into knowledge bases. In particular the structure of these is provided by the top level ontology while knowledge sources description is provided by the second level ontologies For each kind of knowledge the related sources are specified. To realize this phase the KMS knowledge-based services must by based on ontology and workflow management functionalities and on wrapping, crawling, data warehousing techniques, etc.;

"*Distribute Knowledge*", concerning the knowledge distribution to organizational knowledge workers. The framework represents individual and group profiles in term of required knowledge to be shared within the organization among the several different actors with respect their own specific competencies and needs. To realize this phase the KMS knowledge-based services can be based on two main approaches: the stock approach (adding to databases and distributing documents) and the flow approach (share and public knowledge by synchronic and asynchrony communication system, chat, forum, blog, etc.);

"*Apply Knowledge*", concerning the use of the codified knowledge into the business processes where required. To realize this phase the KMS knowledge-based services can be based on business intelligence, decision support, customer relationship management, etc.

The KLC phases involve the core organizational knowledge entities contained into and described by the proposed organizational knowledge representation framework presented in the next section.

4 The Organizational Knowledge Representation Framework

The proposed OKRF is organized as a two level family of ontologies (fig. 2). The first level ontologies formally represent the COKE as the main elements characterizing the organization structure and playing a fundamental role in the KLC: human resources, business processes, knowledge sources, knowledge objects, knowledge-based services. The second level (top level) ontology represents the different topics of the business activities and aims.

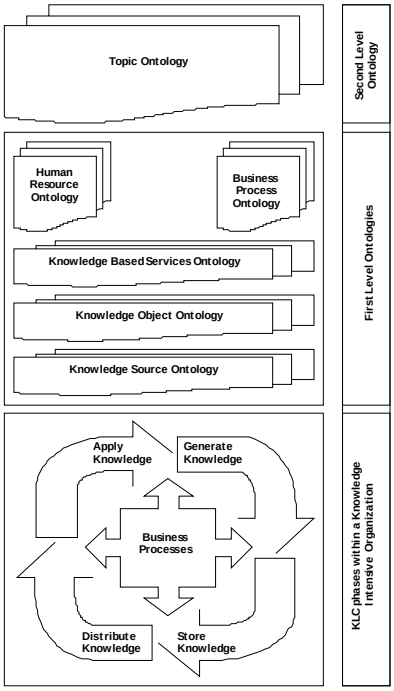

Fig. 2. The Organizational Knowledge Representation Framework

The *Topic Ontology* at the second level of the framework contains the concepts characterizing the typical background of an organization. In particular it specify the explicit and implicit organizational declarative knowledge concerning the concepts characterizing a specific organizational domain: e.g. an IT enterprise background is founded on concepts coming from computer science field. As top level ontology it provides the second level ones with concepts to formally annotate other COKE's contents.

The first level ontologies are the following:

- *Human Resource Ontology*, represents individuals working in the organization (knowledge workers) and social groups they are involved in. Each individual profile is represented in term of implicit, explicit, individual and social knowledge, organizational role, social group membership, required knowledge-based services, etc. Each social group (community of practice, project team, organizational group, etc.) profile is represented in term of its members profiles, the knowledge interest topics, the required knowledge-based services, etc.;
- *Business Processes Ontology*, represents procedural knowledge related to the managerial, operational and decisional processes. Each of them is described in terms of activities, sub-processes, transition states and conditions, involved actors, treated topics, etc. This can be a simple representation of business pro-

cess or a complex ontology where taxonomic and non-taxonomic relations between processes are represented using a workflow specification [17], [18];

- *Knowledge Sources Ontology*, describes the physical sources where knowledge is accessible as people (implicit knowledge owners) or system containing explicit knowledge stored in structured, semi-structured or unstructured machine-readable form (e.g. database management systems, web sites, document management systems, legacy systems, etc.). Knowledge sources, internal or external to the organization, are annotated by topic ontology concepts to indicate where is located the knowledge needed into the business processes;
- *Knowledge Objects Ontology*, maps the structure of logical objects (e. g. database schema, database tables, textual documents, web pages) containing explicit knowledge under structured, semi-structured or unstructured form [9]. These are used in the business processes and handled by the KMS through the knowledge-based services. Knowledge objects are physically contained in the knowledge sources. Moreover each of them can be annotated using the topic ontology concepts to facilitate the handling and the retrieval;
- *Knowledge-Based Service Ontology*, describes all specific KMS knowledge-based services to generate, acquire, store, discover, share, apply the different knowledge forms and kinds during the business process running.

All the ontologies of both levels are strictly connected by relations between their own elements. An implementation of them based on the Ontology Web Language (OWL) [19] and the BPMI workflow language [28] is in progress.

5 Conclusions

The presented organizational knowledge representation framework is proposed as a theoretical base to support the analysis and design of KMS and KM strategies by elicitation and formal representation of core organizational knowledge entities.

In particular the framework provided representation of the COKE helps in designing and implementing KMS knowledge-based services that realize the KLC phases.

To carry out the KMS analysis, design and implementation with respect to a real enterprise a specific knowledge audit and assessment methodology is needed in order to elicit all COKE to represent by the framework. A knowledge audit methodology development is in progress.

References

1. Alvesson M.: Organizations as Rhetoric: Knowledge-Intensive Firms and the Struggle with Ambiguity. In Journal of Management Studies, 30:997—1015, (1993)
2. Davenport T., Prusak L.: Working Knowledge. How Organization Manage what they Know. Boston Harward Business School Press, (1998)

3. Birk A., Kroschel F.: A knowledge management lifecycle for experience packages on software engineering technologies. Technical Report IESE-Report No. 007.99/E, Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern, Germany, (1999)
4. Heisig P.; Mertins K.; Vorbeck J.: Knowledge Management Concepts and Best Practices. Springer-Verlag, Berlin, Heidelberg GmbH & Co. KG, (2003)
5. Ryle, G.: The concept of mind. London, Hutchinson & Company, (1949)
6. Merrill, M. D.: Second Generation Instructional Design. Available at: http://www.id2.usu.edu/id2/index.htm. (2002)
7. Merrill, M. David: "Knowledge Objects", CBT Solutions, March/April, pp. 1-11, (1998)
8. Alavi M., Leidner D. E.: Knowledge Management Systems: Issues, Challenges, and Benefits. In Communications of the Assocation for Information Systems, 1, (1999)
9. Nonaka I.:A Dynamic Theory of Organizational Knowledge Creation. In Organization Science, 5, (1994)
10. Nonaka I., Takeuchi H.: The Knowledge-Creating Company. Oxford University Press, (1995)
11. Tiwana A.: Knowledge Management Toolkit, The: Practical Techniques for Building a Knowledge Management System, Prantice Hall, (1999)
12. van Elst L., Abecker A.: Domain Ontology Agents in Distributed Organizational Memories. In Working Notes of the Workshop on Knowledge Management and Organizational Memories (IJCAI-2001), Seattle, (2001)
13. Staab S., O'Leary D. (eds.): Proceedings of 2000 AAAI Spring Symposium: Bringing Knowledge to Business Processes. AAAI Press, (2000)
14. Papargirys A., Poulymenakou A., Samiotis K.: Knowledge Processes Embedded in Task Structures: Implications for the Design of a Technical and Organizational Solution. PAKM, (2002)
15. Polanyi M.: Tacit Knowledge. Chapter 7 in *Knowledge in Organizations*, Laurence Prusak, Editor. Butterworth-Heinemann, Boston, (1997)
16. Polanyi M.: The Tacit Dimension, Routledge and Kegan Paul, (1966)
17. Arkin A.: Business Process Modeling Language, http://www.BPMI.org, (2004):
18. The Workflow Management Coalition. http://www.wfmc.org, (2004)
19. The Web Ontology Language OWL. http://www.w3.org/2004/OWL (2004)

Distributed autonomous Knowledge Acquisition and Dissemination ontology based framework

Petros Belsis[1]♣, Stefanos Gritzalis[1]

[1] Department of Information and Communication Systems Engineering University of the Aegean, Karlovasi, Samos, Greece**, {**pbelsis, sgritz}@aegean.gr

Abstract In this paper, we present our arguments about security enhanced Knowledge Management (KM) systems, which enable -transparent to the user - diffusion of knowledge, filtered through security policy mechanisms. We provide identification and access to security related knowledge assets, based on the use of software agents, which interact between several organizational domains and authorize access to knowledge resources. The authorization process is based on an automated policy framework, which handles negotiations between different organizational domains, and provides transparent access to knowledge assets. Users benefit from the system by acquiring knowledge not only from their domain, but by being able to query different organizations or different domains on grounds of a common for all co-operating domains security policy framework. The role of ontologies is eminent in exploiting heterogeneous knowledge sources.

1. Introduction.

Knowledge Management (KM) has become lately an emerging deployed discipline, that promises to capitalize on organisation's intellectual capital [7]. KM consists of processes and strategies for identifying, capturing and leveraging knowledge [8]. Many organizations have benefited from deploying knowledge management related activities. Organizations can benefit from harvesting knowledge from several heterogeneous sources and assets, varying from old data repositories, till the knowledge that resides inhuman assets of an organization, namely its employee's experience [9].

Although KM has emerged for more than a decade, little work has been done relative to security issues and protection of intellectual assets within an organization. We attempt to address the issues that arise concerning information security and KM, and how a flexible mechanism that allows transparent to the user access to knowledge resources can coexist with an automated security policy established framework.

KM technologies evolved for almost more than a decade. Both theoretical efforts as well as practical applications have focused on exploiting tacit and explicit knowledge –according to Polanyi's definition [10] - that resides within the organizational borders. Though, the real challenge and boosting in knowledge exploitation is inter-organizational knowledge exchange. Different organizations though, have different security policies and different security restrictions to knowledge assets.

Our work focuses on establishing a security enhanced KM framework, which overcomes the limitations in knowledge sharing and reuse, by making use of heterogeneous, distributed knowledge sources, and by providing a transparent, automated access control mechanism based on the use of security policy languages.

The rest of the paper is organized as follows: section 2 makes a brief introduction in security mechanisms on distributed environments and presents the main concepts upon which the authorization process is based. Section 3 presents the key-role of ontologies on discovering assets upon this framework, section 4 presents an overall overview of the under development prototype, section 5 presents a comparison with related work, while section 6 concludes the paper, providing at the same time directions for further work.

♣ Corresponding author

2. Managing security in distributed environments.

2.1 Key concepts

From the very early years of the emergence of distributed environments concepts, the necessity of establishing security mechanisms emerged. In the early years of mainframes, the term computer security was referring to operating system control mechanisms. The seminal work by Lampson [1] established the ground rules for access control policy specification and implementation mechanism. The formulation of access control in terms of client naming has its roots in existing role based access control architectures, such as those described in [2]. These distributed access control mechanisms, provide with better flexibility and easier enforcement of security controls.

We will refer in brief to a few terms relative to a few main concepts relative to the effective management of distributed inter-organizational, large-scale systems [11]:

- Domains: group of objects to which a common policy applies.
- Roles which identify the rights, duties, functions and interactions, associated with a position such as president, doctor or nurse in hospital, security administrator and so on. A role is the set of authorization and obligation policies, which have a particular role position as a subject [3].

The advantage of using roles for specifying enterprise policies is that individuals can be assigned to roles or withdrawn without having to respecify the policies applying to the role. Domains provide flexible means for partitioning objects in a large system in terms of geographical boundaries, object types, or management needs [4].

These concepts are very useful in order to manage effectively and securely large scale systems. Furthermore, it is easy to represent organizational roles as objects, concluding to an object oriented approach implementation of policies.

Policies can be expressed formally by adopting a policy specification language and can be expressed in the appropriate formalism which can be encoded in machine interpretable form. Based to the aforementioned concepts, domains represent organizations, roles represent organizational structure and rights can be expressed as authorization policies.

2.2 System activities description

By incorporating the notions of domains, we imply that many objects or users may exhibit common characteristics with respect to some criteria, so it is useful to specify policies that apply to a *group* of objects rather than individual ones. Additively, We use the concept of a Role associated with a position so that policies can be specified with respect to organizational positions and describe the duties and access rights of the individuals assigned to the positions[11]. This means that the policies do not have to be respecified when individuals are assigned to new positions. An administrator with the appropriate authority, can edit, modify or delete policies (Fig 1), which are applied to objects [12].

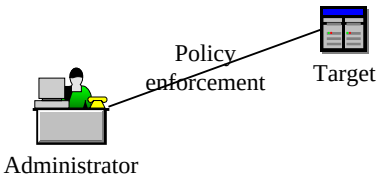

Figure 1 Policy rules application scheme

In our system we handle several KM related activities:

- Knowledge assets discovery, which is handled by security agents and is based on the use of an appropriate ontology,
- Authorization process, which identifies according to the user that requests access to an asset if he has the appropriate level of classification
- Negotiation, when it comes to inter-organizational knowledge transfer

A key concept to the process of identification of assets is the use of ontology, which is essential for the management of heterogeneous knowledge sources.

3. Ontology exploitation

Ontology is “an explicit specification of conceptualization” [14]. It belongs to a family of concepts and tools, such as metadata and meta-knowledge, used to achieve better content description in context. Ontology provides a set of concepts and terms for describing some domain [16][15]. Domain ontologies as defined by Guarino [15] “provide a vocabulary for describing a given domain”. Using domain ontology, we can model entities in KM, their attributes, their role and relationships [13].
The realization of interoperable systems is weighty process, as a consequence of two main system characteristic - distributed data sources and their heterogeneity [5]. Information systems heterogeneity may be considered as structural (schematic heterogeneity), semantic (data heterogeneity), and syntactic heterogeneity (database heterogeneity) [6]. Syntactic heterogeneity means that various database systems use different query languages (SQL, OQL, etc). Structural heterogeneity means that different information systems store their data in different structures. Semantic heterogeneity considers the content of an information item and its meaning. Semantic conflicts among information systems occur whenever information systems do not use the same interpretation of the information. Semantic heterogeneity of the data sources causes serious problems. Ontologies seem a promising discipline towards the alleviation of this problem. We attempt to overcome heterogeneity barriers by implementing an ontology based on the RDF syntax [17]. For simplicity, we provide with a common for all domains ontology. Even though ontology merging seems lately to be a very active and promising scientific area.

4. System Architecture

In this section, we discuss the implementation issues of our prototype, which is still under the construction phase, though the main design concepts have been determined and are currently implemented. In our system, two procedures are being handled, mainly performed through the use of software agents: Knowledge assets discovery and authorization. As it concerns to knowledge discovery process, by querying the RDF-based ontology we identify the URL (Unified Resource Locator) while on the several other descriptive tags –such as the resource creator or a more detailed description of its content- provided on the ontology files, the user is provided with a number of choices concerning the most suitable knowledge assets. Upon request, the authorization process is activated, where the user by providing his id and password for local authentication, an authentication agent, handles the authorization process according to

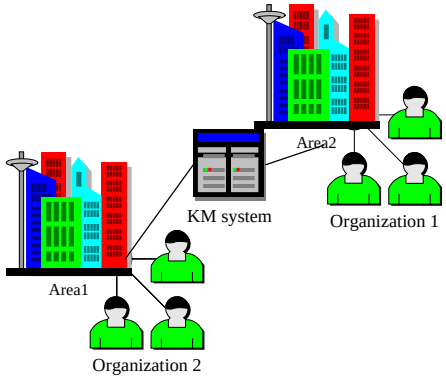

Figure 2 Inter-organizational knowledge exchange framework

the credentials provided for his role. In order for inter-organizational knowledge exchange (in our case inter-domain) knowledge exchange, a negotiation procedure is undertaken, where a correlation between different organization roles procedure is undertaken. The system is developed under the JADE [25] agent platform, while the ontology being utilized was developed by using the XML-spy editor.

5. Related work

Recently, there is enough interest on grounds of providing large-scale distributed organizations with a flexible, interoperable policy driven framework [19][4][3]. Knowledge Management is highly dependent on intercommunicating organizational domains or cooperating organizations. Connectivity and communication is a necessity for most of today's computing environments, enabling them with access to vast amounts of knowledge and on lesser time. Not all the access attempts are benign, as the number of security related incidents and consequent financial losses continuously tend to increase in magnitude, as well as in severance. (See for example the latest CSI/FBI Computer Crime and Security Survey [20]. The networked world is dynamic and undergoes continuous change. Effective management becomes a real challenge as well as a nightmare for security experts. Certain solutions have been proposed towards the facilitation of effective management of distributed systems. KM systems provide several additive challenges against the automated policy based management: they intend to provide with access to the maximum extent to the available resources, while they have to preserve the basic attributes of information, namely confidentiality, integrity, availability [21].

In [22][23] an autonomous system that attempts to provide the problem of making access control decisions in distributed fashion is presented. This system attempts to provide access to devices and services in a distributed environment without distinct organisational boundaries. In order to avoid overloading or failure of centralised decision scheme, a decentralised approach is proposed, where trust decisions are controlled on local node level. This system is characterised by acceptable degree of scalability, it proves out though to be non-applicable to critical environments with high vulnerability where attack resistance remains vital and prior to scalability issues. Such environments, like e-Government or health care medical environments, demand high resistant metrics and proclaim penetration resistance as prior issue.

Levien [24] introduces the notion of group trust metrics. Advocado is a prototype implementing the notion of trust metrics, still there is too much work to be done, but the main deficiency of this approach is the extremely low scalability potential.

Our system, can apply the notion of inter-organizational exchange with scalable, effective, and reliant procedures, without leaving any possibilities for misuse and without embedding risk in the authentication and negotiation procedure as in most trust management systems.

6. Conclusions – Further work

This paper presented a conceptual description of a distributed, automated, security enhanced Knowledge Management system. The architecture is agent-based, and handles heterogeneity of knowledge sources located at different organizational domains through the use of appropriate domain ontologies. Agents, developed in JADE agent development kit, carry the necessary knowledge assets discovery functions and perform the necessary authorization procedures, through the use of security policies.

In the recent future, we intend to expand the policy based framework in order to handle more complex negotiation processes and we attempt to experiment with ontology integration, when different domains make use of different ontologies as described in [18] and no general ontology scheme is pre-established, as it stands for the current implementation of our system.

Acknowledgments

This work was co-funded by 75% from E.E. and 25% from the Greek Government under the framework of the Education and Initial Vocational Training Program – Archimedes.

References

1. B. W. Lampson,"Protection", Fifth Princeton Symposium on Information Sciences and Systems, pp.437-443, Princeton University, March 1971, Reprinted in Operating Systems Review, 8(1), pp.18-24, January 1974
2. Ravi S.Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman "Role based access control models", IEEE Computer 29(2), pp. 38-47, February 1996

3. Sloman M., “Policy driven management for distributed systems, “Journal of Network and Systems Management, vol. 2, no. 4, pp. 333-360, Dec. 1994.
4. Lupu E., Sloman M. “Conflicts in Policy Based Distributed Systems Management”, IEEE Transactions on Software Engineering vol. 25, No 6, 1999.
5. Genesereth M., et al., «Infomaster: An Information Integration System», *ASIGMOD Conference*, pp.539-542, 1997.
6. Stoimenov L., S. Dordevic-Kajan Framework for semantic GIS interoperability, Ser. Mathem. Informatics Journal, 17 (2002).
7. Rus I., Lindvall M. “Knowledge Management in Software Engineering, IEEE Software, vol. 3, pp.26-38, 2002.
8. McCampbell A., Mordhead Clare L., Howard Gitters S. “Knowledge management: the new challenge for the 21st century”, Journal of Knowledge Management, vol. 3, No 3, 1999, pp 172-179.
9. Bhatt G., (2002). “Management strategies for individual knowledge and organizational knowledge”, Journal of Knowledge Management, vol. 6, number 1, 2002, pp. 31-39.
10. Polanyi (1966). “The Tacit Dimension”, Routledge & Kegan Paul, London.
11. Yialelis N., Sloman M. A Security Framework Supporting Domain-Based Access Control in Distributed Systems, proceedings of SNDSS 96, IEEE.
12. Yialelis N., Lupu E., Sloman M. Role-Based Security for Distributed Object Systems, 1996, IEEE Workshops on Enabling Technology.
13. Weinberger H., Te’eni D. Frank A. Ontologies of Organizational Memory as a basis of evaluation, The knowledge engineering review, “Putting ontologies to view”, 2001.
14. Gruber, T.R. (1995), ‘Toward Principles for the Design of Ontologies used for Knowledge Sharing’, Int. J. of Human-Computer Studies, vol. 43, pp. 907-928.
15. Guarino N. (1997). ‘Understanding, Building and Using Ontologies’. Int. J. of Human-Computer Studies, vol. 46, pp. 293-210.
16. Go’mez-Pe’rez, A. (1998), ‘Knowledge Sharing and Reuse’. in: The Handbook of Applied Expert Systems, Liebowitz, J. (ed.) CRC Press, LLC Roca Raton, 10, pp. 1-36.
17. S. Decker, S. Melnik, F. van Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann, I. Horrocks, The semantic web: the roles of XML and RDF, IEEE Internet Comput. 4 (5) (2000) 63–74.
18. Pinto, H. S. 1999a. Towards Ontology Reuse. In AAAI99’s workshop on Ontology Management, WS-99-13, 67–73 AAAI Press.
19. Damianou N. (2002). “A Policy Framework for management of Distributed Systems”, Phd Thesis, Imperial College, London.
20. CSI/FBI (2003), 2003 CSI/FBI Computer Crime and Security Survey, Computer Security Institute, USA.
21. ISO 17799, “Information Technology – Code of Practice for information security management”, ISO/IEC 17799
22. Seleznyov A., Hailes S. “An Access Control Model on Distributed Knowledge Manegement” AINA (2) 2004: 403-406
23. Seleznyov A., Mohamed A., Hailes S. “ADAM: An agent-based Middleware Architecture for Distributed Access Control” Twenty-Second International Multi-Conference on Applied Informatics: Artificial Intelligence and Applications, 2004.
24. Levien R. “Attack Resistant Trust Metrics” draft Phd Thesis University of Berkeley, 2003.
25. **http://jade.tilab.com/**