Managing Multiple Ontologies and Ontology Evolution in Ontologging

A. Maedche, B. Motik, L. Stojanovic, R. Studer, R. Volz

FZI Research Center for Information Technologies at the Univ.of Karlsruhe,Germany
email: {maedche, motik, stojanov, studer, volz}@fzi.edu

Abstract: Ontologging is an ontology-driven environment to enable next generation knowledge management applications building on Semantic Web technology. In this paper we first present the conceptual architecture underlying Ontologging. Second, we focus on two important challenges for ontology-based knowledge management, namely the supporting multiple ontologies and managing ontology evolution. We will provide a general approach for handling these two essential issues within the Ontologging architecture.

Key words: Knowledge management, ontology mapping, ontology evolution.

1. INTRODUCTION

"People can't share knowledge if they do not speak a common language" [7]. This simple insight accurately characterizes what makes knowledge management a challenging task. Its goal to reach global knowledge access within different departments of an enterprise is usually difficult due to the fact that different departments usually encompass different vocabularies, which hinders commu-nication. Consider the case of a large company consisting of different departments, e.g. Human Resources, Production, Sales, etc. Under optimal circumstances we can assume that the first problem of collecting, organizing, and distributing the knowledge within one department has been solved. Ontologies have shown to be the right answer to these structuring and modelling problems by providing a basis for the definition of meaning. They can be used to provide the conceptual basis for communication among humans and machines [2].

However, we have been confronted with several problems when using ontologies in real-world applications. In this paper we consider the following two important problems: First, the traditional approach to design one large-scale ontology, which covers all departments, has shown to be difficult due to effort, scale and maintainability. To facilitate

communication between multiple communities one approach is to rely on multiple ontologies, where individual ontologies are defined, e.g. for each department, and mappings between ontologies establish the linkage between the individual domains. Second, knowledge management systems typically operate in changeable environments. Business dynamics result in several necessary changes within the applications. Consequently, the underlying ontology changes and evolves over time [2]. Clearly, the ontological changes have to be propagated to the depending artefacts in order to keep an overall consistency. This question of managing the necessary evolution of ontologies has not yet sufficiently been approached in literature and practice.

In this paper we introduce a novel approach that tackles these difficulties by allowing the management of multiple ontologies and introducing means for ontology evolution. Ontologging, an ontology-based environment targets beside other research questions these two core problems introduced above. It builds on Semantic Web standards to enable the next generation of knowledge management applications. Based on a short introduction of the comprehensive Ontologging architecture we will present an approach for handling the two essential issues of managing multiple ontologies and supporting ontology evolution.

2. ARCHITECTURE

To enable the described enhanced of existing ontology-based knowledge management approaches a comprehensive architecture is required. We pursue a clearly separated three layered architecture within our Ontologging system. On the top layer, the presentation layer, relevant information is accessed, browsed, queried and edited. We distinguish between the two different users, namely the knowledge manager and the normal end user. In this paper we mainly focus on the knowledge manager and the corresponding methods and tools supporting him in his work with respect to dealing with multiple ontologies and managing ontology evolution. Normal end users access the system via different clients, e.g. a MS Office-based connector and a Web-based browsing and querying interface.

Presentation clients access the different the systems backend via an integrated SOAP-based Web service interface[1] hiding the complexity of the different APIs to application programmers. There exist dedicated interfaces for multiple ontology and metadata management, user management and documents management. Additionally, services in the middleware layer provide value adding functionality on top of the core data, e.g. we provide intelligent services for user personalization and different kinds of agents. Finally, on the lowest layer, the data layer, data

[1] http://www.w3.org/2002/ws/

relevant for the overall system is stored. There, we mainly rely on relational database technology.

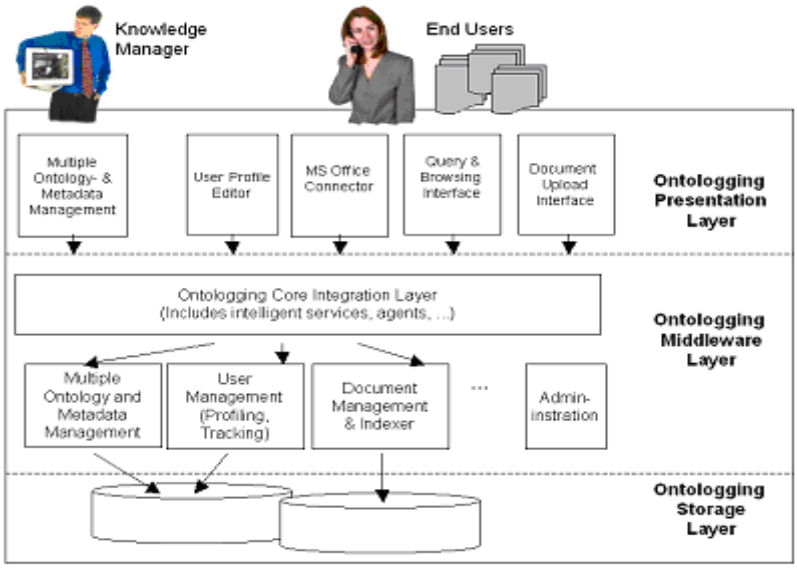

Figure 1. Overall Ontologging Architecture

In the following we will mainly focus on the multiple ontology and metadata management parts within Ontologging. First, we will discuss the underlying modules enabling multiple ontologies. Second, we will discuss the internal support and strategies we provide for the overall ontology evolution process. It is important to mention that both methods and components are driven by meta-ontologies, a so-called semantic bridging and an evolution ontology that capture the relevant information to support multiple ontologies and evolving ontologies.

3. MULTIPLE ONTOLOGIES

An ontology mapping process is the set of activities required to transform instances of a source ontology into instances of a target ontology. By studying the process and analysing different approaches from the literature [10] we observed a set of commonalities and assembled them into the our MAFRA mapping conceptual framework, outlined in Figure 2. The framework consists of five horizontal modules describing the phases that we consider fundamental and distinct in a mapping process. Four vertical components run along the entire mapping process, interacting with horizontal modules.

Within the horizontal dimension, we identified following five modules:

Lift & Normalization. This module focuses on raising all data to be mapped onto the same representation level, coping with syntactical, structural and language heterogeneity. Both ontologies must be

normalized to a uniform representation, in our case RDF(S), thus eliminating syntax differences and making semantics differences between the source and the target ontology more apparent. To facilitate that, we developed a LIFT approach providing means to bring DTDs, XML-Schema, and relational databases[2] to the structural level of the ontology. Lift is not further elaborated in this paper - we shall simply assume that the source and target ontologies are already represented in RDF-Schema with their instances in RDF.

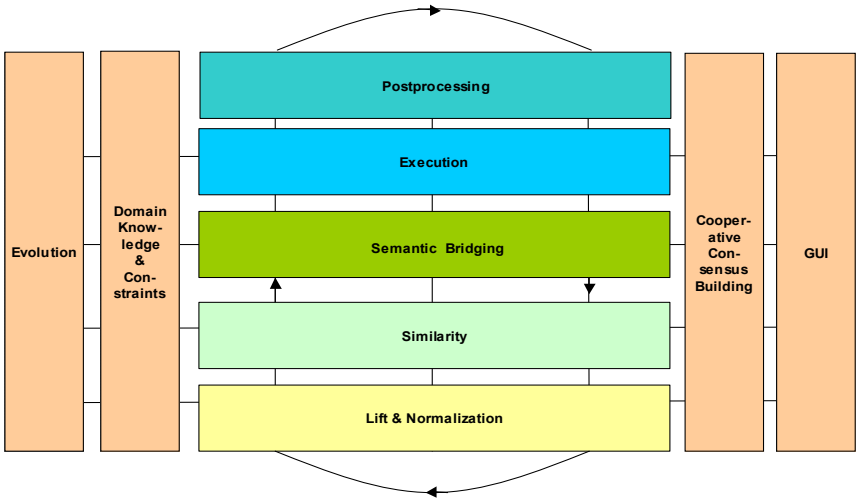

Figure 2. Conceptual Mapping Architecture

Similarity. This module establishes similarities between entities from the source and target ontology. Similarity between conceptual models is hard to measure and often establishing a suitable similarity measure is a very subjective task. Several different similarity measures have been proposed in literature [1, 10, 12] focusing on different aspects of ontology entities. We don't further elaborate on this issue, as it is not in scope of this paper.

Semantic Bridging. Based on the similarities computed in the previously described phase, the semantic bridging module is responsible for establishing correspondence between entities from the source and target ontology. Technically, this is accomplished by establishing semantic bridges[3] - entities reflecting correspondence between two ontology entities [9]. Apart from the semantic correspondence, additional "procedural" information is needed to further specify the transformation to be performed, e.g. translation of measures like currencies.

Execution. This module actually transforms instances from the source ontology into target ontology by evaluating the semantic bridges defined earlier. In general two distinct modes of operation are possible, namely offline (static, one-time transformation) and online (dynamic, continuous mapping between source and the target) execution.

[2] http:// kaon.semanticweb.org/REVERSE
[3] http://www.fzi.de/wim/staff/Nuno/bridges

Post-processing. The post-processing module takes the results of the execution module to check and improve the quality of the transformation results. The most challenging task of post-processing is establishing object identity - recognizing that two instances represent the same real-world object. Furthermore, by computing statistical properties of transformed instances, it is possible to check whether semantic bridges were under specified.

The vertical dimension of MAFRA contains modules that interact with horizontal modules during the overall process. Following four modules have been identified and will be only shortly mentioned in this paper:

Evolution. This modules focuses on keeping semantic bridges obtained by the “Semantic Bridge" module, which must be kept in synchrony with the changes in the source and target ontologies. Evolving ontologies on the Semantic Web result in an update requirement of the corresponding semantic bridges. Although this may be achieved by reapplying the mapping process, this is probably not the most efficient or accurate way. Thus, the mapping process must have an evolution component that will reuse the existing semantic bridges in adapting them to new requirements.

Cooperative Consensus Building. The cooperative Consensus Building module is responsible for establishing a consensus on semantic bridges between two communities participating in the mapping process. This is a requirement as one has to choose frequently from multiple, alternatively possible mappings .The amount of human involvement required to achieve consensus may be reduced by automating the mapping process as much as possible.

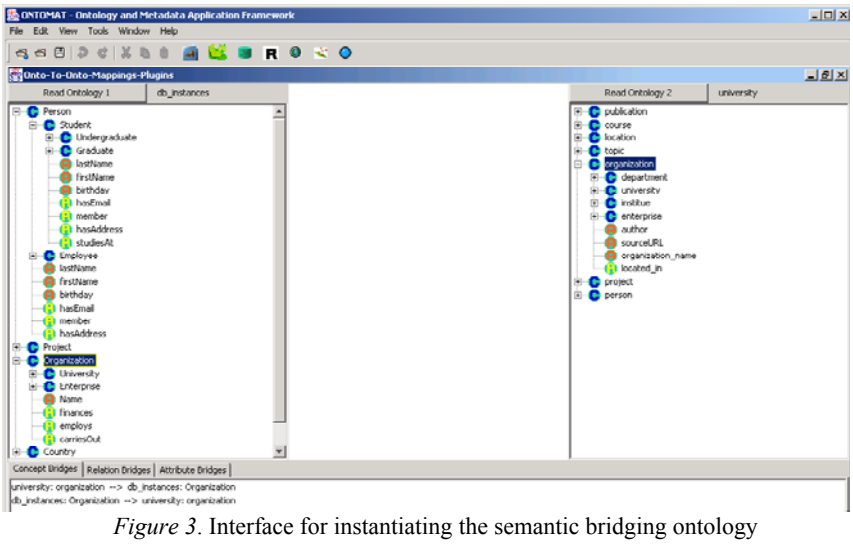

Figure 3. Interface for instantiating the semantic bridging ontology

Domain Constraints and Background Knowledge. The quality of similarity computation and semantic bridging may be dramatically improved by introducing background knowledge and domain constraints, e.g. by using glossaries to help identify synonyms or by using lexical

ontologies, such as WordNet or domain-specific thesauri, to identify similar concepts.

Graphical User Interface. Mapping is a difficult and time consuming process, which is not less difficult than building an ontology itself, i.e. deep understanding of both conceptualisations required on human side, thus extensive graphical support must be given and it is a separate issue how this can be achieved in an optimal way. The graphical user interfaces modules (Figure 3) allows the users drive the mapping process, provide domain constraint and background knowledge, create semantic bridges, refine bridges according to the results of the execution module, etc.

4. EVOLVING ONTOLOGIES

Ontology evolution (OE) is the timely adaptation of an ontology to changed business requirements, to trends in ontology instances and patterns of usage of the ontology-based applications, as well as the consistent management/propagation of these changes to dependent elements. A modification in one part of the ontology may generate subtle inconsistencies in other parts of the same ontology, in the ontology-based instances as well as in depending ontologies and applications [6]. This variety of causes and consequences of the ontology changes makes OE a very complex operation that should be considered as both, an organizational and a technical process [13]. It requires a careful analysis of the types of the ontology changes that can trigger evolution as well as the environment in which the whole OE process is realized.

The overall OE process is presented in Figure 4. It has a cyclic structure, since validation of realized changes may induce new changes in order to obtain model consistency or to satisfy users' expectations. In the following we will shortly elaborate on each of the phases.

Change Representation. To resolve changes, they have to be identified and represented in a suitable format. Elementary changes in the ontology shown in Table 1 are derived from our ontology definition [7] given in specifying fine-grained changes that can be performed in the course of OE. However, this granularity of OE changes is not always appropriate. Often, intent of the changes may be expressed on a higher level. For example, the may need to generate a common superconcept of two concepts. He may bring the ontology into desired state through successive application of a list of elementary evolution changes. However, there is an impedance mismatch between the intent of the request and the way the intent is achieved. Moreover, a lot of unnecessary changes may be performed if each change is applied alone. To avoid these drawbacks, it should be possible to express changes on a more coarse level, with the intent of change directly visible. We introduce the composite changes (e.g. Merge_concepts, Extract_subconcepts, Extract_related_concept) representing a group of elementary changes applied together.

Semantics of Change. Application of an elementary change in the ontology can induce inconsistencies in other parts of the ontology. We distinguish syntax and semantic inconsistency. Syntax inconsistency arises when undefined entities at the ontology or instance level are used or ontology model constraints are invalidated. Semantic inconsistency arises when meaning of an ontology entity is changed [17]. For example, removal of a concept which is the only element of domain set for some property results in syntax inconsistency [5]. Resolving that problem is treated as a request for a new change in the ontology, which can induce new problems that cause new changes and so on. If an ontology is large, it may be difficult to fully comprehend the extent and meaning of each induced change. The task of 'semantics of change' phase is to enable resolution of induced changes in a systematic manner, ensuring consistency of the whole ontology. To help in better understanding of effects of each change, this phase should contribute maximum transparency providing detailed insight into each change being performed.

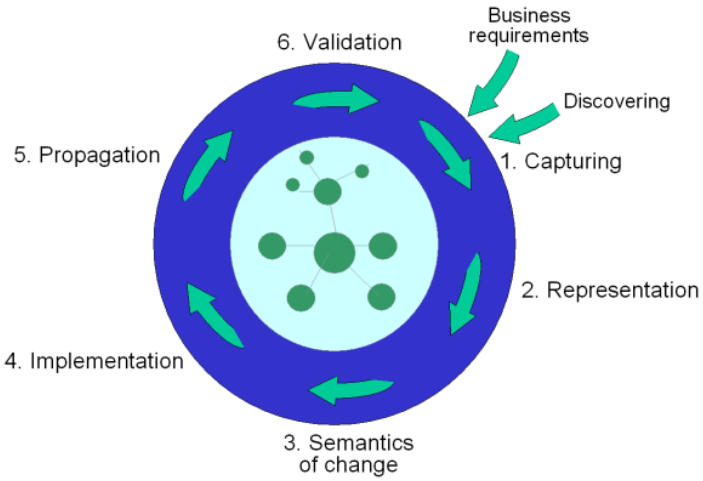

Figure 4. Cycle Ontology Evolution Process

Table 1. Elementary changes in the ontology and associated metadata

Change	Elementary change
Add	Add_Concept; Add_SubConceptOf Add_Property; Add_SubPropertyOf; Add_Domain; Add_Axiom; Add_InstanceOf; Add_PropertyInstanceOf
Delete	Delete_Concepì; Delete_SubConceptOf; Delete_Property; Delete_SubPropertyOf; Delete_Domain; Delete_Axiom; Delete_InstanceOf; Delete_PropertyInstanceOf
Modify	Set_Property_Range

However, for each change in the ontology, it is possible to generate different sets of additional changes, leading to different final consistent states. Most of existing systems for the ontology development provide only one possibility for realizing a change and this is usually the simplest one. For example, the deletion of a concept always causes the deletion of all its subconcepts.

Thus, to resolve a change, the evolution process needs to determine answers at many *resolution points* – branch points during change resolution where taking a different path will produce different results. Each possible answer at each resolution point is an *elementary evolution strategy*. Common policy consisting of a set of elementary evolution strategies, each giving an answer for one resolution point, is an *evolution strategy* and is used to customize the OE process. Thus, an evolution strategy unambiguously defines the way how elementary changes will be resolved. Typically a particular evolution strategy is chosen by the user at the start of the OE process (the left part of Figure 5).

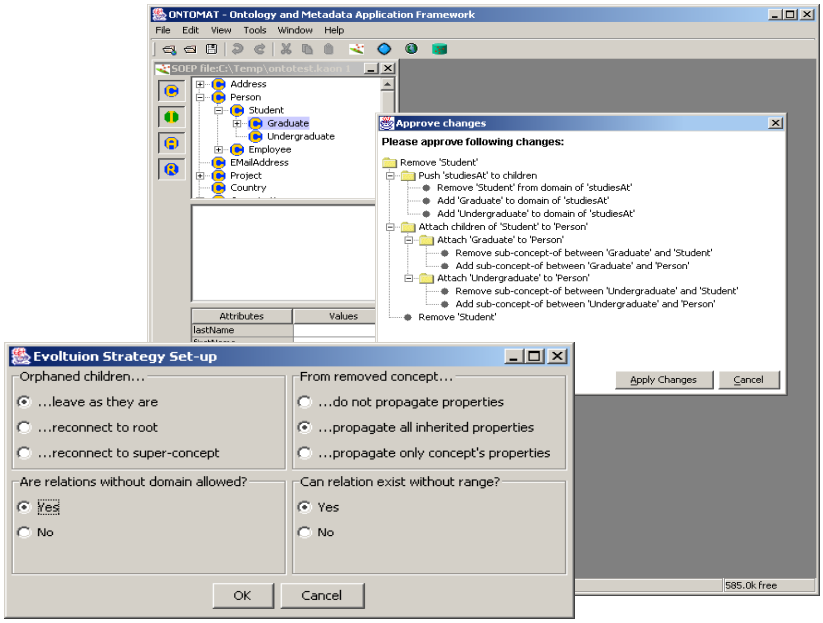

Figure 5. Ontology Evolution in KAON[4] framework: Evolution Strategy Set-up and Ontology Evolution User Interface in OntoMat-SOEP

Change Implementation. In order to avoid performing undesired changes, before applying a change to the ontology, a list of all implications to the ontology should be generated and presented to the user [16]. He should be able to comprehend the list and approve or cancel the change (the right part of Fig. 5). When the changes are approved, they are performed by successively resolving changes from the list. If changes are cancelled, the ontology should remain intact.

Change Propagation. First, when the ontology is modified, ontology instances need to be changed to preserve consistency with the ontology [6]. This can be performed in three steps. If the instances are on the Web, they are collected in the knowledge base[5]. In the second step, modification of instances is performed according to the changes in the ontology [15]. In

[4] http://kaon.semanticweb.org/SOEP
[5] http://kaon.sematicweb.org/CRAWL

the last step "out-of-date" instances on the Web are replaced with corresponding "up-to-date" instances. Second, ontologies often reuse and extend other ontologies. Therefore, an ontology update might also corrupt ontologies that depend on the modified ontology and consequently, all artefacts that are based on these ontologies. This problem can be solved by recursive applying the OE process on these ontologies. However, besides of the syntax inconsistency, the semantic inconsistency can also arise when, for example, the dependent ontology already contains a concept that is added in the original ontology. Third, when an ontology is changed, applications based on the changed ontology may not work correctly. An OE approach has to recognize which change in the ontology can affect the functionality of dependent applications [11] and to react correspondingly [14].

Validation. When working on an ontology collaboratively, different ontology engineers may have different ideas about how the ontology should be changed. Moreover, the ontology engineer may fail to understand the actual effect of the change and approve the change that shouldn't be performed. It may be desired to change the ontology for experimental purposes. In order to enable recovering from these situations, we introduce the validation phase in the OE process (see Figure 4). It enables validation of performed changes and undoing them at user's request. It is important to note that reversibility means undoing all effects of some change, which may not be the same as simply requesting an inverse change manually. For example, if a concept is deleted from a concept hierarchy, its subconcepts will need to be either deleted as well, attached to the root concept, or attached to the parent of the deleted concept. Reversing such a change is not equal to recreating the deleted concept – one needs, also, to revert the concept hierarchy into original state. The problem of reversibility is typically solved by creating evolution logs. An evolution log, based on the evolution ontology described in the following, tracks information about each change, allowing to reconstruct the sequence of changes leading to current state of the ontology.

Change Discovery and Capture. In OE we may distinguish two types of changes: top-down and bottom-up, whose generation is part of the "capturing phase" in the OE process. Top-down changes are explicit changes, driven, for example, by top-manager who want to adapt the system to new requirements and can be easily realized by an OE system. However, some changes in the domain are implicit, reflected in the behaviour of the system and can be discovered only through analysis of its behaviour. For example, if a customer group doesn't contain members for a longer period of time, it may mean that it can be removed. This second type of change mined from the set of ontology instances are called bottom-up changes. Another source of bottom-up changes is the structure of the ontology itself. Indeed, the previously described "validation phase" results in an ontology which may be in a consistent state, but contains some redundant entities or can be better structured with respect to the

domain. For example, multiple users may be working on different parts of an ontology without enough communication. They may be deleting subconcepts of a common concepts at different points in time to fulfil their immediate needs. As a result, it may happen that only one subconcept is left. Since classification with only one subclass beats the original purpose of classification, we consider such ontology to have a suboptimal structure. To aid users in detecting such situations, we investigated the possibilities of applying the self-adaptive systems principles and proactively make suggestions for *ontology refinements* – changes to the ontology with the goal of improving ontology structure, making the ontology easier to understand and cheaper to modify. As known to authors, none of existing systems for ontology development and maintenance offer support for (semi-) automatic ontology improvement.

4.1 Evolution Ontology

The backbone of the whole evolution process is a meta-ontology for evolution that enables representation, analysis, realization and sharing ontological changes in a more systematic and consistent way. It is a specific ontology that is designed to support all phases in the evolution process of an ontology.

The evolution ontology consists of three parts. First part is about mechanisms to represent changes (see Table 1). Ontological changes [7] are represented using the top level concept "*Change*" and its relations. For every change, it is also useful to know who is *author* of the change and when it is happened (*date*). The *cause* of the change is used to represent the source of the change (business requirements or the learning process) and the *relevance* of the change describes whether and how it can fulfil the requirements. Also, OE is a managerial process and it needs some properties to support decision-making like *cost*, *priority*, etc. The order of changes is also very important as it enables recovery of implemented changes (if the result of the validation phase is unsatisfied) and/or mining trends (patterns) to improve the OE process. To solve semantics of change problem, the evolution ontology contains axioms that derive additional changes. The derived change and the required change are connected using *parentChange* relation.

The second part of the evolution ontology containing relations like *prototypical*, *primary_key*, etc. represents semantic information about the domain ontology explicitly [17], because the conceptual structure of the evolution ontology aims to provide enough mechanisms to deal with problems of syntax as well as semantic inconsistencies. The third part of the evolution ontology aims to support data-driven self-improvement of the domain ontology. We enforce formal discovering of changes by representing some heuristics as axioms in the evolution ontology. For example, if all subconcepts have the same property, the property may be moved to the parent concept.

5. RELATED WORK AND CONCLUSION

In the last decade, there has been much active research in the area of ontology-based systems. However, there are very few approaches investigating the problems of changing in the ontologies.

Heflin [5] points out that ontologies on the Web will need to evolve and he presents SHOE, a web-based knowledge representation language that supports multiple versions of ontologies. Although good design may prevent many ontological errors, some errors will not be realized until the ontology is put to use. However, this problem as well as the problem of the change propagation are not treated. Moreover, the user cannot customize the way of performing the change and the problem of the identification of the change is not analysed. In contrast to the OE that allows access to all data (to ontology itself and to dependent artefacts) only through the newest ontology, ontology versioning allows access to data through different version of the ontology. Thus, OE can be treated as a part of the ontology versioning. Ontology versioning is analysed in [6]. Authors provide an overview of causes and consequences of the changes in the ontology. However, the most important flaw is the lack of a detailed analysis of the effect of specific changes on the interpretation of data which is a constituent part of our work.

Other research communities also have influences our work. The problem of schema evolution and schema versioning support has been extensively studied in relational and database papers [11]. However, there are several differences that steam from different knowledge models and different usage paradigms. Research in OE can also benefit from the many years of research in knowledge-based system evolution. The script-based knowledge evolution [16] that identifies typical sequences of changes to knowledge base and represents them in a form of scripts, is similar to our approach. In contrast to the knowledge-scripts that allow the tool to understand the consequences of each change, we go step further by allowing the user to control how to complete the overall modification and by suggesting the changes that could improve the ontology.

There is only little work concerning the support of using multiple ontologies. Again, our approach is motivated by classical work on federated database and mediators done by the database community [4]. Nevertheless, our approach goes beyond classical techniques, as it provides an integrated view on the overall multi-ontology scenario, from discovering mappings, representing mappings [1] to processing mappings.

In this paper we have presented Ontologging, the corporate ontology modeling and management system. Ontologging is an ontology-based environment to enable next generation knowledge management applications building on Semantic Web standards. In this paper we have mainly focused on two important challenges for ontology-based knowledge management: First, the management of multiple ontologies and, second, the handling of ontology evolution in dynamic environments.

Both approaches rely on heavily using meta-primitives, also represented in the form of ontologies.

ACKNOWLEDGEMENTS

The research presented in this paper was profited from fruitful discussion with our Ontologging project partners from Insead (France), Meta4 (Spain), Deltatec (Belgium), Archetypon (Greece) and Indra (Spain). Research for this paper was financed by European Commission, IST, project "Ontologging" (IST-2000-28293).

REFERENCES

1. A. Doan, J. Madhavan, P. Domingos, and A. Halevy: *Learning to map between ontologies on the semantic web*. In Proc. of the World-Wide Web Conference 2002.
2. D. Fensel, *Ontologies: Dynamics Networks of Meaning*, In Proceedings of the 1st Semantic web working symposium, Stanford, CA, USA, July 30th-August 1st, 2001
3. E. Franconi, F. Grandi, and F. Mandreoli, *A semantic approach for schema evolution and versioning in object-oriented databases*, Proc. CL2000, 2000
4. W. Gio and M. Genesereth, *Basis for Mediation*, Proc. COOPIS'95 Conference, Vienna Austria, available from US West, Boulder CO, May 1995.
5. J. Heflin, *Towards the Semantic Web: Knowledge Representation in a Dynamic, Distributed Environment*, Ph.D. Thesis, University of Maryland, College Park. 2001.
6. M. Klein and D. Fensel, *Ontology versioning for the Semantic Web*, Proc. International Semantic Web Working Symposium (SWWS), USA, 2001.
7. A. Maedche, S. Staab, N. Stojanovic, R. Studer, and Y. Sure. *SEmantic PortAL - The SEAL approac*h. to appear: In Creating the Semantic Web. D. Fensel, J. Hendler, H. Lieberman, W. Wahlster (eds.) MIT Press, MA, Cambridge, 2001.
8. A. Maedche and S. Staab, *On Comparing Ontologies*, Internal Report 403, Institute AIFB, University of Karlsruhe, 2001.
9. A. Maedche, B. Motik, N. Silva, R. Volz, MAFRA – An Ontology Mapping FRAmework in the Context of the Semantic Web, Internal Report, FZI, 2002.
10. N. Rahm, P. Bernstein: A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.
11. J.F. Roddick, *A Survey of Schema Versioning Issues for Database Systems*, Information and Software Technology, 37(7):383-393, 1996.
12. S. Staab, A. Maedche: *Comparing Ontologies - Similarity Measures and a Comparison*, Internal Report 408, Institute AIFB, Karlsruhe University.
13. S. Staab, H.-P. Schnurr, R. Studer and Y. Sure, *Knowledge Processes and Ontologies*, IEEE Intelligent Systems. 16(1), 2001. Special Issue on Knowledge Management.
14. L. Stojanovic, N. Stojanovic, S. Handschuh, *Evolution of the Metadata in the Ontology-based Knowledge Management Systems*, In Proc. of Experience Management 2002, Berlin, March 7-8, 2002.
15. L. Stojanovic, N. Stojanovic and R. Volz, *Migrating data-intensive Web Sites into the Semantic Web*, ACM Symposium on Applied Computing SAC, Madrid, 2002.
16. M. Tallis, Y. Gil, *Designing Scripts to Guide Users in Modifying Knowledge-based Systems*, AAAI/IAAI 1999: 242-249
17. V.A.M. Tamma and T.J.M Bench-Capon, *A conceptual model to facilitate knowledge sharing in multi-agent systems*, In Proc. of the OAS 2001. Montreal, 2001, pp. 69-76.