A case study in supporting DIstributed, Loosely-controlled and evolvInG Engineering of oNTologies (DILIGENT)

Christoph Tempich[2] & **Sofia Pinto**[1] & **Steffen Staab**[2] & **York Sure**[2]
[1]Dep. de Engenharia Informática, Instituto Superior Técnico, Lisboa, Portugal
sofia.pinto@dei.ist.utl.pt
[2]Institute AIFB, University of Karlsruhe, 76128 Karlsruhe, Germany
{sure,staab,tempich}@aifb.uni-karlsruhe.de

Abstract: Knowledge management solutions relying on central repositories sometimes have not met expectations, since users often create knowledge ad-hoc using their individual vocabulary and using their own decentral IT infrastructure (e.g., their laptop). To improve knowledge management for such decentralized and individualized knowledge work, it is necessary to, first, provide a corresponding IT infrastructure and to, second, deal with the harmonization of different vocabularies/ontologies. In this paper, we briefly sketch the technical peer-to-peer platform that we have built, but then we focus on the harmonization of the participating ontologies.

Thereby, the objective of this harmonization is to avoid the worst incongruencies by having users share a core ontology that they can expand for local use at their will and individual needs. The task that then needs to be solved is one of distributed, loosely-controlled and evolving engineering of ontologies. In this paper we present a corresponding process template and a case study.

Key Words: Distributed Knowledge Management, Methodology, Case Study
Category: H 4.1 I 2.4 J C 2.4 I 2.6

1 Introduction

The knowledge structures underlying today's knowledge management systems constitute a kind of ontology that may be built according to established methodologies *e.g.* the one by [9]. These methodologies have a centralized approach towards engineering knowledge structures requiring *knowledge engineers*, *domain experts* and others to perform various tasks such as *requirement analysis* and *interviews*. While the user group of such an ontology may be huge, the development itself is performed by a — comparatively — small group of domain experts who *represent* the user community and ontology engineers who *help structuring*.

In Virtual Organizations [2], organizational structures change very often, since organizations frequently leave or join a network. Therefore, working based on traditional, centralized knowledge management systems becomes infeasible. While there are some technical solutions toward Peer-to-Peer knowledge management systems (e.g., [1]) — and we have developed a technically sophisticated solution of our own [4] — traditional methodologies for creating and maintaining knowledge structures appear to become unusable like the systems they had been developed for in the first place.

Therefore, we postulate that ontology engineering must take place in a Distributed, evolvInG and Loosely-controlled setting. With DILIGENT we here provide a process

template suitable for distributed engineering of knowledge structures and intend to extend it towards a fully worked out and multiply tested methodology in the long run. We here show a case study we performed using DILIGENT in a virtual organization.

The case study (cf. Section 4) suggests that the resulting ontology is indeed shared among users, that it adapts fast to new needs and is quickly engineered. With some loose control we could ensure that the core ontology remained consistent, though we do not claim that it gives a complete view on all the different organizations.

In the following, we briefly introduce the organizational and technical setting of our case study (Section 2). Then we sketch the DILIGENT process template (Section 3), before we describe the case study.

2 Problem setting

2.1 Organizational setting at IBIT case study

In the SWAP project, one of the case studies is in the tourism domain of the Balearic Islands. The needs of the tourism industry there, which accounts for 80% of the islands' economy, are best described by the term 'coopetition'. On the one hand the different organizations *compete* for customers against each other. On the other hand, they must *cooperate* in order to provide high quality for regional issues like infrastructure, facilities, clean environment, or safety — that are critical for them to be able to compete against other tourism destinations.

To collaborate on regional issues a number of organizations now collect and share information about *indicators* reflecting the impact of growing population and tourist fluxes in the islands, their environment and their infrastructures. Moreover, these indicators can be used to make predictions and help planning. For instance, organizations that require *Quality & Hospitality management* use the information to better plan, *e.g.*, their marketing campaigns. As another example, the governmental agency IBIT[1], the Balearic Government's co-ordination center of telematics, provides the local industry with information about *new technologies* that can help the tourism industry to better perform their tasks.

Due to the different working areas and objectives of the collaborating organizations, it proved impossible to set up a centralized knowledge management system or even a centralized ontology. They asked explicitly for a system without a central server, where knowledge sharing is integrated into the normal work, but where very different kinds of information could be shared with others.

To this end the SWAP consortium — including us at Univ. of Karlsruhe, IBIT, Free Univ. Amsterdam, Meta4, and empolis — have been developing the SWAP generic platform and we have built a concrete application on top that allows for satisfying the information sharing needs just elaborated.

[1] `http://www.ibit.org`

2.2 Technical setting: The SWAPSTER Platform for Peer-to-Peer KM

The SWAP environment (Semantic Web And Peer-to-peer; short SWAPSTER) [4] is a generic platform which was designed to enable knowledge sharing in a distributed network. Nodes wrap knowledge from their local sources (files, e-mails, etc.) and they ask for and retrieve knowledge from their peers. For communicating knowledge SWAPSTER transmits RDF structures , which are used to convey conceptual structures (e.g., the definition of what a conference is) as well as corresponding data (e.g., data about I-Know-2004). For structured queries as well as for keyword queries, SWAPSTER uses SeRQL, an SQL-like query language that allows for queries combining the conceptual and the data level and for returning newly constructed RDF-structures.

3 DILIGENT process overview

As we have described before, decentralized cases of knowledge sharing, like our example of a virtual organization, require an ontology engineering process that reflects this particular organizational setting [8].[2] Therefore, we have drafted the template of such a process — we cannot claim that it is a full-fledged methodology yet. The result, which we call DILIGENT, is described in the following. In particular, we elaborate on the high-level process, the dominating roles and the functions of DILIGENT, before we give the concrete case in Section 4 as an indicator for the validity of our ontology engineering process design.

Key roles: In DILIGENT there are several experts, with different and complementary skills, involved in collaboratively building the same ontology. In a virtual organization they often belong to competing organizations and are geographically dispersed. Ontology builders may or may not use the ontology. Vice versa, most ontology users will typically not build or modify the given ontology.

Overall process: An initial ontology is made available and users are free to use it and modify it locally for their own purposes. There is a central board that maintains and assures the quality of the shared core ontology. This central board is also responsible for deciding to do updates to the core ontology. However, updates are mostly based on changes re-occurring at and requests by *decentral*ly working users. Therefore the board only *loosely controls* the process. Due to the changes introduced by the users over time and the on-going integration of changes by the board, the ontology *evolves*. Let us now survey the DILIGENT process at the next finer level of granularity. DILIGENT comprises five main steps: (1) **build**, (2) **local adaptation**, (3) **analysis**, (4) **revision**, (5) **local update** (*cf.* Figure 1).

Build. The process starts by having *domain experts*, *users*, *knowledge engineers* and *ontology engineers* **build** an initial ontology. In contrast to existing ontology engineering methodologies (cf. [5, 10]), we do not require completeness of the initial shared

[2] In fact, we conjecture that the majority of knowledge sharing cases falls into this category.

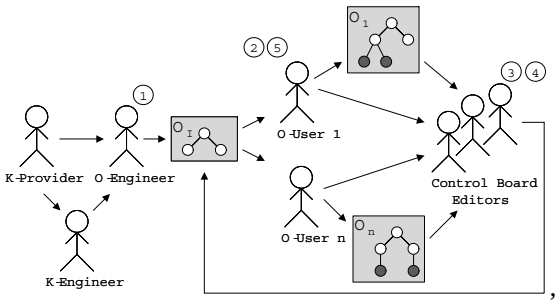

Figure 1: Roles and functions in distributed ontology engineering

ontology with respect to the domain. The team involved in building the initial ontology should be relatively small, in order to more easily find a small and consensual first version of the shared ontology.

Local adaptation. Once the core ontology is available, users work with it and, in particular, adapt it to their local needs. Typically, they will have their own business requirements and correspondingly evolve their local ontologies (including the common core). In their local environment, they are also free to change the reused core ontology. However, they are not allowed to directly change the core ontology from which other users copy to their local repository. Logging local adaptations (either permanently or at control points), the control board collects change requests to the shared ontology.

Analysis. The board **analyzes** the local ontologies and the requests and tries to identify similarities in users' ontologies. Since not all of the changes introduced or requested by the users will be introduced to the shared core ontology,[3] a crucial activity of the board is deciding which changes are going to be introduced in the next version of the shared ontology. The input from users provides the necessary arguments to underline change requests. A balanced decision that takes into account the different needs of the users and meets user's evolving requirements[4] has to be found.

Revise. The board should regularly **revise** the shared ontology, so that local ontologies do not diverge too far from the shared ontology. Therefore, the board should have a well-balanced and representative participation of the different kinds of participants involved in the process: knowledge providers, domain experts, ontology engineers and users. In this case, users are involved in ontology development, at least through their requests and re-occurring improvements and by evaluating it, mostly from an usability point of view. Knowledge providers in the board are responsible for evaluating the ontology, mostly from a technical and domain point of view. Ontology engineers are one of the major players in the analysis of arguments and in balancing them from a technical point of view. Another possible task for the controlling board, that may not always be

[3] The idea in this kind of development is not to merge all user ontologies.

[4] This is actually one of the trends in modern software engineering methodologies (see Rational Unified Process).

a requirement, is to assure some compatibility with previous versions. Revision can be regarded as a kind of ontology development guided by a carefully balanced subset of evolving user driven requirements. Ontology engineers are responsible for updating the ontology, based on the decisions of the board. Revision of the shared ontology entails its evolution.

Local update. Once a new version of the shared ontology is released, users can **update** their own **local** ontologies to better use the knowledge represented in the new version. Even if the differences are small, users may rather reuse *e.g.* the new concepts instead of using their previously locally defined concepts that correspond to the new concepts represented in the new version.

4 Case study

We are now going to describe how DILIGENT ontology engineering is taking place in the IBIT case study.

This case study took place in one organization with seven peers and it lasted for two weeks. The case study will be extended in the future to four organizations corresponding to 21 peers and it is expected that the total number of organizations will grow to 7 corresponding to 28 peers.

Building. In the IBIT case study two knowledge engineers were involved in building the first version of the shared ontology with the help of two ontology engineers. In this case, the knowledge engineers were also knowledge providers. Moreover, they received additional training such that, they are able to act as ontology engineers on the board. This they did already during this case study — together with two experts from the domain area.

The ontology engineering process started by identifying the main concepts to be represented in the ontology through the analysis of competency questions and their answers. The most frequent queries and answers exchanged by the participants were analyzed. The identified concepts were divided into three main modules: "Sustainable Development Indicators", "New Technologies" and "Quality&Hospitality Management". From the competency questions we quickly created a first ontology with 22 concepts and 7 relations for the "Sustainable Development Indicator" module, which was the domain of the then participating organization. This ontology was defined during one workshop lasting for three hours. The other modules will be further elaborated in future efforts.

Based on previous experience of IBIT with the participants we could expect that users would mainly specialize the modules of the shared ontology corresponding to their domain of expertise and work. Thus, it was decided by the ontology engineers and knowledge providers involved in building the initial version that the shared ontology should only evolve by addition of new concepts, and not from other more sophisticated operations, such as restructuring or deletion of concepts.

Local Adaptation. The developed core ontology for "Sustainable Development Indicator" was distributed among the users and they were asked to extend it with their local structures. With assistance of the developers they extracted on average 14 folders. The users mainly created sub concepts of concepts in the core ontology from the folder names. In other cases they created their own concept hierarchy from their folder structure and aligned it with the core ontology. They did not create new relations. Instance assignment took place, but was not significant.

Analyzing. The members of the board gathered the evolving structures and analyzed them. The following observations were made:

Concepts matched A third of the extracted folder names was directly aligned with the core ontology. A further tenth of them was used to extend existing concepts.

Folder names indicate relations In the core ontology a relation inYear between the concepts Indicator and Temporal was defined. This kind of relation is often encoded in one folder name. *e.g.* the folder name "SustInd2002" matches the concepts Sustainable Indicator and Year[5]. It also points to a modelling problem, since Sustainable Indicator is a concept while "2002" is an instance of concept Year.

Missing top level concepts The concept project was introduced by more than half of the participants, but was not part of the initial shared ontology.

Refinement of concepts The top level concept Indicator was extended by more than half of the participants, while other concepts were not extended.

Concepts were not used Some of the originally defined concepts were never used. Concepts are identified as used, when users created instances, aligned documents with them, or created sub concepts.

Folder names represent instances The users who defined the concept project used some of their folder names to create instances of that concept *e.g.* "Sustainable indicators project".

Different labels The originally introduced concept Natural spaces was often aligned with a newly created concept Natural environments and never used itself.

Ontology did not fit One user did create his own hierarchy and could use only one of the predefined concepts. Indeed his working area was forgotten in the first ontology building workshop.

The DILIGENT methodology is supported by an Ontoedit plug-in[11], which is an implementation of the *Edit* component in the SWAP system. The plug-in supports the board mainly in recognizing changes and extensions by different users to the core ontology. It also supports the user in performing these changes.

From the discussions with the domain experts we have the impression that the local extensions are a good indicator for the evolution direction of the core ontology. However, since the users made use of the possibility to extend the core ontology with their

[5] Year is sub class of class Temporal

folder names, as we expected, the resulting local ontologies represent the subjects of the organized documents. Therefore, a knowledge engineer is still needed to extend the core ontology, but the basis of his work is being improved and eased significantly. From our point of view there is only a limited potential to automate this process.

Revision. The board extended the core ontology where it was necessary and performed some renaming. More specifically the board introduced (1) one top level concept (Project) and (2) four sub concepts of the top level concept Indicator and one for the concept Document. The users were further pointed to the possibility to create instances of the introduced concepts.

Local update. The extensions to the core ontology were distributed to the users. The feedback of the users was in general positive. However, due to the early development stage of SWAPSTER a prolonged evaluation of the user behavior and second cycle in the ontology engineering process has not yet been performed.

5 Lessons learned

The case study helped us to better comprehend the use of ontologies in a peer-to-peer environment. First of all our users did understand the ontology mainly as a classification hierarchy for their documents. Hence, they did not create instances of the defined concepts. However, our expectation that folder structures can serve as a good input for an ontology engineer to build an ontology was met.

Currently we doubt that our manual approach to analyze local structures will scale to cases with many more users. Therefore, we are looking into technical support to recognize similarities in user behavior. Furthermore, local update will be a problem when changes happen more often. Last, but not least, we have so far only addressed the ontology creation task itself – we have not yet measured if users get better and faster answers with the help of DILIGENT-engineered ontologies. All this remains work to be done in future.

In spite of the technical challenges, user feedback was very positive since (i) the upfront ontology engineering effort was low, thus the system could be used quickly (ii) they are integrated into the ontology development.

6 Discussion

It is now widely agreed that ontologies are a core enabler for sophisticate knowledge management systems [3]. The development of ontologies in centralized settings is well studied and established methodologies exist (*cf.* [5]). However, current experiences from projects suggest, that ontology engineering should be subject to continuous improvement rather than a one time action and that ontologies promise the most benefits in decentralized rather than centralized systems. Hence, a methodology for distributed, loosely-controlled and dynamic ontology engineering settings is needed. In [6] a methodology for collaborative ontology engineering is proposed. The aim of their

work is to support the creation of a static ontology in a collaborative ontology engineering setting. With DILIGENT we define a process which takes into account that requirements on a knowledge management system change over time. Furthermore, we allow a quick introduction phase with later refinement. Obviously, such a process needs tool support from ontology engineering environments. There exist already some which allow for remote and collaborative ontology engineering (*cf.* [7]). However, none exists which could support the complete cycle. We have an implementation, which is a first step towards such a tool.

DILIGENT will eventually result in a methodology with tool support which supports ontology engineers to build ontologies in a decentralized environment yet systematically.

Acknowledgements.
Research reported in this paper has been partially financed by EU in the IST project SWAP (IST-2001-34103), the IST thematic network OntoWeb (IST-2000-29243), the IST project SEKT (IST-2003-506826) and Fundação Calouste Gulbenkian (21-63057-B). In particular we want to thank Immaculada Salamanca and Esteve Lladó Martí from IBIT for the fruitful discussions and the other people in the SWAP team for their collaboration towards SWAPSTER.

References

1. M. Bonifacio et al. Peer-mediated distributed knowldege management. In L. van Elst et al., editors, *Proceedings of the AAAI Spring Symposium "Agent-Mediated Knowledge Management (AMKM-2003)"*, Stanford, CA, USA, 2003.
2. Luis M. Camarinha-Matos and Hamideh Afsarmanesh, editors. *Processes and Foundations for Virtual Organizations*, volume 262 of *IFIP INTERNATIONAL FEDERATION FOR INFORMATION PROCESSIN*. Kluwer Academic Publishers, 2003.
3. D. OLeary. Using AI in knowledge management: Knowledge bases and ontologies. *IEEE Intelligent Systems*, 13(3):34–39, May/June 1998.
4. M. Ehrig et al. The swap data and metadata model for semantics-based peer-to-peer systems. In *Proceedings of MATES-2003. First German Conference on Multiagent Technologies*, LNAI, Erfurt, Germany, September 22-25 2003. Springer.
5. A. Gómez-Pérez et al. *Ontological Engineering*. Advanced Information and Knowlege Processing. Springer, 2003.
6. Clyde W. Holsapple and K. D. Joshi. A collaborative approach to ontology design. *Commun. ACM*, 45(2):42–47, 2002.
7. Riichiro Mizoguchi. Ontology engineering environments. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, chapter 14, pages 275–298. Springer, 2004.
8. H. Sofia Pinto and J.P. Martins. Evolving Ontologies in Distributed and Dynamic Settings. In D. Fensel et al., editors, *Proc. of the 8th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR2002)*, pages 365–374, San Francisco, 2002. Morgan Kaufmann.
9. G. Schreiber et al. *Knowledge Engineering and Management — The CommonKADS Methodology*. The MIT Press, Cambridge, Massachusetts; London, England, 1999.
10. S. Staab, H.-P. Schnurr, R. Studer, and Y. Sure. Knowledge processes and ontologies. *IEEE Intelligent Systems*, 16(1), January/Febrary 2001. Special Issue on Knowledge Management.
11. Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke. OntoEdit: Collaborative ontology development for the semantic web. In I. Horrocks and J.A. Hendler, editors, *Proc. of the 1st Int. Semantic Web Conf. (ISWC 2002)*, volume 2342 of *LNCS*, pages 221–235, Sardinia, IT, 2002. Springer.