Towards Agile Ontology Maintenance

Markus Luczak-Rösch

Freie Universität Berlin, Institute of Computer Science,
Corporate Semantic Web Workgroup, Berlin D-14195, Germany
markus.luczak-roesch@fu-berlin.de

Abstract. Ontologies are an appropriate means to represent knowledge on the Web. Research on ontology engineering reached practices for an integrative lifecycle support. However, a broader success of ontologies in Web-based information systems remains unreached while the more lightweight semantic approaches are rather successful. We assume, paired with the emerging trend of services and microservices on the Web, new dynamic scenarios gain momentum in which a shared knowledge base is made available to several dynamically changing services with disparate requirements. Our work envisions a step towards such a dynamic scenario in which an ontology adapts to the requirements of the accessing services and applications as well as the user's needs in an agile way and reduces the experts' involvement in ontology maintenance processes.

1 Introduction

Ontologies are an appropriate means to represent knowledge on the Web. Research on ontology engineering methodologies has come from describing the scratch development of ontologies and reached practices for an integrative lifecycle support. The ontology engineering discipline has changed from an individual art towards a collaborative and distributed process with disparate skilled users develop consensual models and distributed networks of ontologies. However, a broader success of ontologies in Web-based information systems remains unreached. They gained momentum in some characteristic and closed domains, such as health care and life sciences. On the every-day Web the more lightweight semantic approaches are rather successful which are based upon small vocabularies, e.g. the emerging linked data initiative. But also this lightweight semantic cannot deploy its full potential. The Web 2.0 resulted huge so called data silos. By use of wrappers or crawlers huge RDF datasets are derived from the relational databases of such silos. Consolidating and integrating the whole data of a specific application-dependent purpose or a specific individual remains a cumbersome task. Not to mention the control of the unintegratedly evolving knowledge in the silos.

As a next logical step one should await that, against the trend of the data silos, the user holds and controls her data on her own. Paired with the emerging trend of services and microservices on the Web [3] this results in a dynamic scenario in which a shared knowledge base is made available to several dynamically

A. Bernstein et al. (Eds.): ISWC 2009, LNCS 5823, pp. 965–972, 2009.

changing services with disparate requirements. This work envisions a step towards such a dynamic scenario in which an ontology adapts to the requirements of the accessing services and applications in an agile way. Therefore I design an innovative approach for agile ontology maintenance.

This paper starts with a presentation of the motivation for and the concrete problem statement of our work in Section 2. From our best knowledge we derived related approaches in the field of the research topic which we introduce briefly in the same section. Section 3 outlines the artifacts which we will contribute as the results of this work. The followed research methodology as well as the aimed evaluation are part of Section 4 before this paper ends with a summary of the goals, initial results, and the work in the nearest future in Section 5.

2 Motivation, Problem Statement and Related Work

The general and personal motivation for this work consists of three core parts. The first part is based upon our studies of the existing ontology engineering methodologies. It represents the fundamental direction of this work. As a second part, we derive from personal interviews with small and mid-sized enterprise (SME) partners of the project Corporate Semantic Web, that they look for a lightweight and dynamic process for ontology maintenance which minimizes the need for ontology experts to be present. We explicitly focus this scenario, however, we respect that there are enterprise settings as well which need and deal with heavyweight ontology engineering processes. On the whole, our idea meets the gap, which we identified as the result of the study of ontology engineering approaches and which has been also identified by others, such as [11]. That means concretely that research regards human-centered feedback as elementary part of the ontology lifecycle and ontology maintenance is more or less treated as the loop back to the beginning of the development process. Thus, ontology maintenance results as the specific direction of this work. The third part of the motivation is our personal vision of the next logical step of the Web from a social Web 2.0/3.0 towards a Web of services and alternative access devices. That means, that the next generation of the Web will be less driven by direct human access to contents and services by use of conventional client tools (e.g. Web browsers) but more by mobile devices and services. As a result of that the concepts of human-centered ontology engineering, such as argumentation to concepts and relations to reach the ontology consensus will loose impact.

2.1 A Running Example

To clarify this motivation we will briefly come up with a simple running example for the problem which we want to solve. Consider a company's knowledge base which includes information about the employees. In the beginning only personal information have been collected conforming the friend of a friend (FOAF) vocabulary. One service uses the knowledge base which generates lists of employees with certain interests. Each time a new service is bound to the

knowledge base, such as a service for displaying absent employees or information about the income (e.g. for the accounting), the maintainer of the knowledge base has to find out which facts, in the sense of the T-box of the ontology, are missing and how she can easily adopt the current T-box and possibly the A-box as well to these new application requirements. It is also possible that separate services require the same information represented in different vocabularies (e.g. foaf:name vs. myvocabulary:name) which yields the conflict for the maintainer whether to replace the present representation or to model a mapping between both. The decision for either the first or the latter depends on several criteria, such as the computability of the ontology for complex reasoning or obsolete and unused information.

This yields the central research questions of our work:

1. *How does a methodology for ontology maintenance in an agile environment look like?* We search for a process which puts less emphasize on the initial development of an ontology but more on the ontology usage and evolution. That covers technical aspects of the modeling as well as aspects of the release management.
2. *Can we reduce the necessary influence of human experts in the ontology maintenance process by tracking feedback about ontology usage?* In this case our work searches for a formal model that allows the analysis of ontology usage for ontology evolution purposes.

2.2 Related Work

The methodologies and architectures for ontology engineering purposes mostly differ in details regarding to the composition of ontology engineering and application development, the range of users interacting in ontology engineering tasks, and the degree of lifecycle support. We studied the broad range of engineering methodologies, such as [6,15,7,9] beside others. While theses methodologies are from the perspective of the ontology engineers our approach changes this to the perspective of the ontology adopter which is a person with much less expertise in knowledge modeling and ontologies. We also border our approach from the recent one called *RapidOWL* [1], since RapidOWL introduces an idea of agile knowledge engineering. In contrast to us, Auer proposes a paradigm-based approach without any phase model and which is application-independent. The actually running NeOn project contributes the NeOn methodology for ontology engineering and the NeOn architecture for lifecycle support[16]. The NeOn methodology is well researched and adequate for projects with long iterations which are less dynamic than those which we address. The NeOn architecture proposes a feedback loop as part of the ontology lifecycle and adopts principles from service oriented architectures, thus it is an interesting base for our maintenance management framework and possibly reusable. However, it differs in detail since the feedback which it tracks is intended as explicit human feedback and an observation of informal background domain knowledge [18] while we want to

adopt to the requirements of dynamically changing applications which use the knowledge base.

Ontology evolution has been researched under two scopes, that are (1) ontology versioning and change management as in [8,17] and (2) identification of informational extension, reduction, or reorganization as in [4,14,18]. Mostly, the representatives of both scopes end up in an expert-oriented perspective with focus to the initial ontology as the input and an evolved ontology as the output of the evolution process. The deployment of the new ontology including side effects to systems which apply it, such as updates of distributed stored instances because of schema changes, is not explicitly treated and the communication of the impact of ontology change from the ontology engineer to the ontology user are out of scope as well. We are also aware of the usage-oriented approaches presented in [12] and [13]. Compared to these we go one step further, since we focus both, user-oriented as well as application-oriented needs to an ontology.

As it is more or less usual for the ontology engineering discipline the whole field of software engineering methodologies is relevant work from which well-researched principles may be adopted. In this special case this is focused to maintenance management including bug and feature tracking [5] for software engineering processes. For sure, the field of database maintenance is related work, too. Here we focus on research on the incorporation of feedback into schema evolution and multi-tenant databases [2].

3 Contributions

We plan a multi layered result of this research activity. Altogether, from the bottom of theories to the top of method and tool support, we are working on (1) an ontology maintenance management methodology and (2) an ontology feedback tracking mechanism which both will be part of (3) an integrative ontology maintenance management framework.

COLM – An Agile Ontology Maintenance Methodology. The Corporate Ontology Lifecycle Methodology (COLM) reflects the agility of knowledge engineering processes and brings application dependency through the concrete definition of the application environment. We define it as an agile ontology maintenance methodology since it is focused on continuously evolving ontologies in an application-dependent context. To clarify which process steps are more expert-oriented and thus need higher human involvement and those which need less, COLM consists of two different cycles, namely the engineering cycle (high involvement) and the usage cycle (less involvement). The overall goal is to use an intuitive reporting of tracked usage information which indicates the necessity of change.

As depicted in Figure 1, the process starts at the *selection / development / integration* phase. The result of this phase is an ontology, which is *validated* within an application-dependent context. If it is approved that the ontology suites the requirements it is *deployed* to be in use and it is *populated*. Throughout the

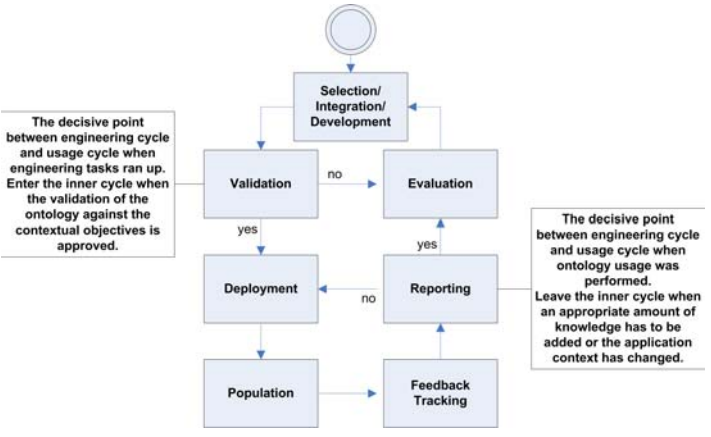

Fig. 1. The Corporate Ontology Lifecycle Methodology COLM

whole *feedback tracking* phase, formal statements about users' feedback and behavior are recorded and finally a *reporting* of this information is performed. The usage cycle is left if any necessary change has been detected and the knowledge engineers *evaluate* the weaknesses of the current ontology.

Ontology Change Recommendation by Feedback Tracking. Inspired by the approaches of argumentation-based ontology engineering and motivated by our viewpoint, that the human feedback within ontology engineering processes has to be reduced our integrative feedback tracking respects implicit feedback hidden in behaviors of users and applications. We winnow two categories of feedback. First, we regard the application-oriented feedback as the feedback which helps to fulfill the dynamically changing application requirements. Second, we regard the user-oriented feedback which helps to follow the evolution of dynamically changing and informal needs of the ontology user. In detail the feedback mechanisms which we design are *query observation* for the application-oriented feedback and *annotation behavior observation* for the user-oriented feedback.

The overall goal is to keep track of relevant information which help to create an ontology which properly describes a domain of interest that easy and rather small for better computation, so that the requirements of all accessing applications and the user's information needs are fulfilled.

Putting the Things Together. As we described it before we propose an evolution of ontologies by respecting their application-dependent context. Integrating new knowledge and evolving the existing by tracking the usage and distributing coexisting ontology versions to applications by intelligent version control are its central concepts. Our first draft proposal of this architecture is shown in Figure 2.

This depiction describes the various systems which access an ontology, respectively an ontology repository, for certain tasks, e.g. the ontology editors for

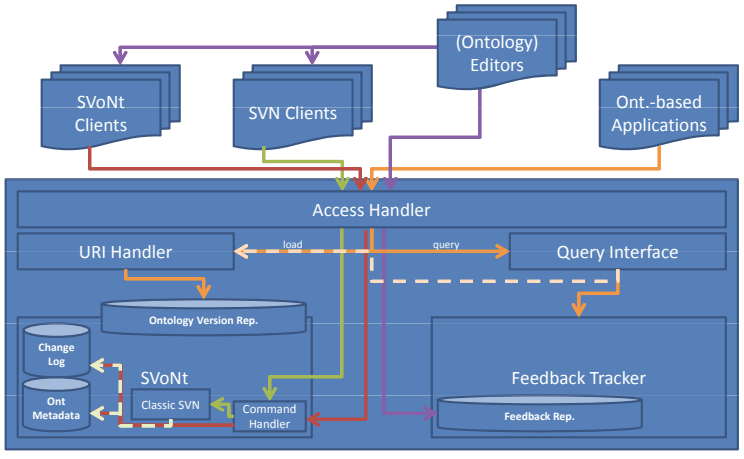

Fig. 2. High-level architecture of a proposed ontology maintenance management framework

manipulating an ontology or ontology-based applications for querying them. A central component is the feedback tracker which observes actions performed on the ontologies and supports the detection of new knowledge or potentially weak parts of the model.

The underlying technical essential of evolution in an agile environment is a flexible version management. At the moment we plan to integrate SVN for ontologies (SVoNt) into our maintenance management framework, which sets up on top of the well-known version control system SVN. To facilitate ontology versioning the text-based approach of SVN has to be extended to act on semantic structures of OWL ontologies. Internally, we add two major components which facilitate this additional functionality, namely consistency checks and generation of the ontology diff. From the set of differences we calculate a set of atomic change operations and store them into a log. In extend of SVoNt we also work on a process for an ontology release management which allows push and pull scenarios for the deployment of coexisting ontology versions. The model will provide a mechanism for the asynchronous evolution of a T-box and various central as well as distributed A-boxes which conform this schema.

4 Research Methodology and Aimed Evaluation

As we mentioned it in Section 2 the initial point for our work is based upon personal studies of the foundations of ontology engineering processes as well as a set of face to face interviews with representatives of small and mid-sized enterprises in Germany about the applicability of semantic technologies and ontologies in their corporate contexts.

Coming from theses requirements we aim at a multi layered contribution of this work, which touches fundamental research, instrumentalist research, and applied research. Thus it is necessary that we evaluate it multi-perspectively. We will choose methodologies which reflect the domain and the amount of the approach, range the results with reference to related work, and at least proof the applicability of the concepts in practice.

By today we see the chance to run and test COLM within two use cases. First, at the Ontonym GmbH which is a company that provides semantic search services supported by self-constructed and self-maintained ontologies in the background. Second, in cooperation with the DBpedia project, which supports a giant linked dataset based on s self-constructed and self-maintained ontology and crawled information from Wikipedia. Especially the latter use case seems to be valuable since our study will run in parallel to the development of a community-driven approach for an evolution of the DBpedia ontology. To set the approach in relation to other approaches and to measure its quality we will apply the ONTOCOM cost-estimation model for ontology engineering on it as it has been done for other methodologies, e.g. DILIGENT [10], as well.

5 Initial Results, Outlook and Conclusions

In this paper we presented our work towards an integrative ontology maintenance management for agile application-dependent scenarios. Based on the COLM methodology the SVoNt system for SVN-based version control of OWL ontologies and a feedback tracking mechanism will be combined into a framework application that supports the whole ontology lifecycle from the viewpoint of the ontology user.

The COLM methodology has been developed, published, and iteratively refined based on several valuable comments by experts. It is in a mature state right now. Based on the theories of COLM we proposed a high level architecture of an ontology maintenance management framework which will integrate the SVoNt server and the feedback tracking mechanism. The latter two components are in a preliminary design state right now. We envision to finish the fundamental work on these partial components until the end of 2009. The evaluation will be performed after the prototype implementation of our proposed framework is finished. We plan to finish this work until the beginning of 2011.

Altogether, our approach towards agile ontology maintenance should promote ontology engineering from the user's and the usage's perspective. In combination with several research trends, such as end-user generated microservices and the service Web 3.0, this is a promising step ahead to bring ontologies to broader success.

References

1. Auer, S., Herre, H.: RapidOWL — An Agile Knowledge Engineering Methodology. In: Virbitskaite, I., Voronkov, A. (eds.) PSI 2006. LNCS, vol. 4378, pp. 424–430. Springer, Heidelberg (2007)

2. Aulbach, S., Jacobs, D., Kemper, A., Seibold, M.: A comparison of flexible schemas for software as a service. In: SIGMOD 2009: Proceedings of the 35th SIGMOD international conference on Management of data, pp. 881–888. ACM, New York (2009)
3. Davies, M.: Towards a Semantic Infrastructure for User Generated Mobile Services. In: ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 924–928. Springer, Heidelberg (2009)
4. Djedidi, R., Aufaure, M.-A.: Ontological knowledge maintenance methodology. In: Lovrek, I., Howlett, R.J., Jain, L.C. (eds.) KES 2008, Part I. LNCS (LNAI), vol. 5177, pp. 557–564. Springer, Heidelberg (2008)
5. Erdil, K., Finn, E., Keating, K., Meattle, J., Park, S., Yoon, D.: Software Maintenance As Part of the Software Life Cycle. Comp180: Software Engineering Project, December 16 (2003)
6. Fernndez-Lpez, M., Gmez-Prez, A., Juristo, N.: METHONTOLOGY: From Ontological Art Towards Ontological Engineering. In: AAAI 1997 Spring Symposium on Ontological Engineering. AAAI Press, Menlo Park (1997)
7. Kotis, K., Vouros, A.: Human-centered ontology engineering: The HCOME methodology. Knowl. Inf. Syst. 10(1), 109–131 (2006)
8. Noy, N.F., Kunnatur, S., Klein, M., Musen, M.: Tracking Changes During Ontology Evolution. In: Proceedings of the Third International Semantic Web Conference, pp. 259–273. Springer, Berlin (2004)
9. Pinto, H.S., Tempich, C., Staab, S., Sure, Y.: Distributed Engineering of Ontologies (DILIGENT). Semantic Web and Peer-to-Peer. Springer, Heidelberg (2005)
10. Simperl, E., Tempich, C.: How Much Does It Cost? Applying ONTOCOM to DILIGENT. Technical Report, FU Berlin (2005)
11. Simperl, E., Tempich, C.: Ontology Engineering: A Reality Check. In: ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 836–854. Springer, Heidelberg (2006)
12. Stojanovic, N., Stojanovic, L.: Usage-Oriented Evolution of Ontology-Based Knowledge Management Systems. In: Meersman, R., Tari, Z., et al. (eds.) CoopIS 2002, DOA 2002, and ODBASE 2002. LNCS, vol. 2519. Springer, Heidelberg (2002)
13. Stojanovic, L., Stojanovic, N., Gonzalez, J., Studer, R.: OntoManager - A System for the Usage-Based Ontology Management. In: CoopIS/DOA/ODBASE, vol. 2888. Springer, Heidelberg (2003)
14. Stojanovic, L.: Methods and Tools for Ontology Evolution. PhD Thesis, University of Karlsruhe, Germany (2004)
15. Sure, Y., Studer, R.: On-To-Knowledge Methodology — Expanded Version. In: On-To-Knowledge deliverable, vol. 17, Institute AIFB, University of Karlsruhe (2002)
16. Tran, T., Haase, P., Lewen, H., Muñoz-García, Ó., Gómez-Pérez, A., Studer, R.: Lifecycle-Support in Architectures for Ontology-Based Information Systems. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 508–522. Springer, Heidelberg (2007)
17. Völkel, M., Groza, T.: SemVersion: RDF-based Ontology Versioning System. In: Proceedings of the IADIS International Conference WWW / Internet 2006 (ICWI 2006), Murcia, Spain (2006)
18. Zablith, F.: Dynamic Ontology Evolution. In: International Semantic Web Conference (ISWC) Doctoral Consortium, Karlsruhe, Germany