

## Managing Ontology Lifecycles in Corporate Settings

**Markus Luczak-Rösch and Ralf Heese**

(Freie Universität Berlin)

Institut für Informatik, AG Netzbasierte Informationssysteme

Takustr. 9, D-14195 Berlin, Germany

{luczak,heese}@inf.fu-berlin.de)

**Abstract:** Weaving the Semantic Web the research community is working on publishing publicly available data sources as RDF data on the Web. Well-known cost- and process-oriented problems of ontology engineering hinder the employment of ontologies as a flexible, scalable, and cost-effective means for integrating data in corporate contexts. We propose an innovative ontology lifecycle, examine existing tools towards the functional requirements of the lifecycle phases and propose a versioning approach supporting them integratively.

**Key Words:** knowledge management, knowledge engineering methodologies, knowledge life cycles, knowledge maintenance

**Category:** M.1, M.2, M.3

### 1 Introduction and Related Work

Within the past years the Semantic Web community has developed a comprehensive set of standards and data formats to annotate semantically all kinds of resources. Currently, a main focus lies on integrating publicly available data sources and publishing them as RDF on the Web. In contrast, many corporate IT areas are just starting to engage in Semantic Web technologies. Early adopters are in the areas of enterprise information integration, content management, life sciences and government. Applying Semantic Web technologies to corporate content is known as *Corporate Semantic Web*. To employ ontologies as a flexible, scalable, and cost-effective means for integrating data in corporate contexts, *corporate ontology engineering* has to tackle cost- and process-oriented problems [Simplerl 06].

In Section 2 we present a set of requirements characterizing corporate settings for ontology-based information systems. We use these requirements in Section 3 to conclude the need of a new lifecycle model, which we introduce afterwards. The lifecycle raises new functional requirements, which we use for a comparison of accepted tools for ontology engineering (Section 5) and a conclusion about their applicability.

Research reached a wide range of ontology engineering methodologies which mostly differ in details referring to the composition of ontology engineering and application development, the range of users interacting in ontology engineering

tasks, and the degree of lifecycle support. Some methodologies assume the users to be ontology experts only or at least to be knowledge workers with little technical experience while others also address users with no experience with ontologies at all.

*METHONTOLOGY* [Fernandez 97] transfers standards for software engineering to the task of ontology engineering and is a concept-oriented approach to build ontologies from scratch, reuse existing ontologies or re-engineer knowledge. The lifecycle model of *METHONTOLOGY* does not respect any usage-oriented aspects. The On-To-Knowledge methodology (*OTK*) [Sure 02] is less concept-oriented because it has an application-dependent focus on ontology engineering. It integrates participants which are not very familiar with ontologies in early phases of the process for identification of the use cases and competency questions. *OTK* assumes a centralized and a distributed strategy for ontology maintenance but neither presents a detailed description or evaluation of both strategies nor addresses ontology usage. The methodologies *HCOME* [Kotis 06] and *DILIGENT* [Pinto 06] address the problem of ontology engineering from the point of view that reaching an ontology consensus is highly depending on people with disparate skill level. Both methodologies assume a distributed setting. Every individual is free in adapting the central ontology consensus locally. The evolution of the consensual model is depending on these local adoptions. Thus, *HCOME* and *DILIGENT* propose a human-centered approach, but they do not provide any application-dependent point of view. Recently, the well-thought approach of agile engineering has come into focus of research in ontology engineering. In [Auer 06] *RapidOWL* is introduced as an idea of agile ontology engineering. Auer proposes a paradigm-based approach without any phase model. *RapidOWL* is designed to enable the contribution of a knowledge base by domain experts even in absence of experienced knowledge engineers. However, the view on ontology usage is limited to the rapid-feedback, which is nonspecific referring to the stakeholder who gives it and how it is given. As a recent result of the NeOn project the NeOn methodology for ontology engineering and the NeOn architecture for lifecycle support have been developed [Tran 07]. Again, both lack the agility of knowledge lifecycles referring to knowledge evolution by usage of ontologies in an enterprise.

## 2 Corporate Ontology Engineering Settings

*Corporate Semantic Web* assumes a corporate environment for the application of Semantic Web technologies. Two major goals aim at their beneficial application in the core areas *Corporate Semantic Search* and *Corporate Semantic Collaboration*. We consider ontologies, which appear highly *application-dependent* in this scenarios, to characterize a corporal competency.

As the result of personal interviews with industrial partners of the project Corporate Semantic Web, we collected key-points about applying ontology-based information systems in corporate contexts. We raise the following main requirements as the results of the interviews: (1) application-dependence of the ontologies, (2) central allowance and control of the conceptualization, (3) individual views to the knowledge domain (e.g. units or hierarchies), (4) lack of time for manual population and/or annotation, (5) unexperienced users working with the ontologies and (6) need for estimation of development costs and potential improvement. We also derived two use-cases from the interviews which proof the correctness of these requirements: (a) a **semantic trouble-ticketing system** where the terminology of internal employees and external customers have to be matched against the fixed terms of the released product and (b) a **corporate semantic intranet** application which integrates data from various conventional sources and should transfer views and restrictions of the sources into the central semantically enriched knowledge base.

Based on this set of requirements of corporate ontology engineering settings, we derive a new point of view on ontology engineering processes. The widely accepted methodologies METHONTOLOGY, OTK, HCOME, and DILIGENT regard ontology engineering loose from ontology usage. However, they agree that ontologies are undergoing lifecycles with engineering phase and usage phase, but they do not consider ontology engineering as a combination of both. From our perspective the evolution of knowledge is the basal entity of an adequate ontology lifecycle and that it is strongly depending on the usage by unexperienced people with lack of time to note, annotate, or feedback explicitly.

### 3 A Corporate Ontology Lifecycle

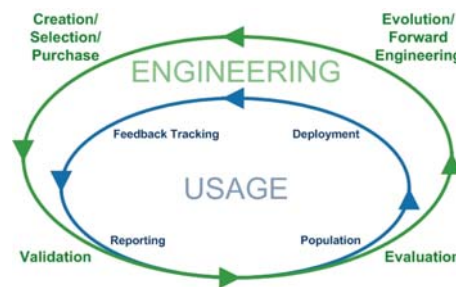
From our assumptions mentioned in Section 2, we conclude a need of a new ontology lifecycle model for ontologies in corporate contexts. The model should allow an intuitive understanding of raising costs per iteration. Because of the change in the environment complexity from Web-scale to corporate-scale, we assume that it is possible to converge ideas of agile software engineering and ontology engineering. But we think it is necessary to change the definition of what is assumed as being agile.

Recent approaches such as RapidOWL focus the agile paradigms *value*, *principle*, and *practice* as a development philosophy. That accompanies agile software engineering as it is intended in the *Agile Manifesto*<sup>1</sup>. But, again, this focus is limited to engineering tasks, while usage is factored out. It comes clear, that changing requirements over time are only one agile aspect influencing ontology prototype evolution. Another is the dynamic of knowledge referring to the

<sup>1</sup> <http://agilemanifesto.org/>

evolving dimensions of data and user activities depending on processes.

The corporate setting needs an ontology engineering model which respects these agile aspects and allows an intuitive way of estimating costs for evolution steps. Both points play a key-role for our approach towards a generic corporate ontology lifecycle which is depicted in Figure 1. Eight phases of our two-part cycle are marked, which refer either to the outer cycle as creation/selection, validation, evaluation, evolution/forward engineering or to the inner circle as population, deployment, feedback tracking, and synchronization. The outer cycle represents pure engineering, which is an expert-oriented environmental process. The inner constitutes the ontology usage, which is a human-centered concurrent process.



**Figure 1:** The Corporate Ontology Lifecycle

Starting the process at *creation/selection* means to start the knowledge acquisition and conceptualization, to re-use or re-engineer existing ontologies, or to commission a contractor to develop an ontology. The result of this phase is an ontology, which is *validated* against the objectives. At the intersection point between the engineering and the usage cycles the ontology engineers and the domain experts decide whether the ontology suites the requirements or not. If this is approved the ontology is *populated*, which means that a process for instance generation from structured, semi-structured and unstructured data runs up. The ontology is *deployed* and in use by applications. Throughout the whole *feedback tracking* phase, formal statements about users feedback and behavior are recorded. A *reporting* of this feedback log is performed at the end of the usage cycle. That means that all feedback information, which were collected until a decisive point, are analyzed respecting internal inconsistencies and their effects to the currently used ontology version. The usage cycle is left and the knowledge engineers *evaluate* the weaknesses of the current ontology with respect to the feedback log. This point may also be reached, when the validation shows that the

new ontology is inappropriate to the specification. The lifecycle closes with the *evolution/forward engineering* of the ontology by engineers and domain experts.

The innovative approach towards agile ontology engineering allows an evolution of rapidly released ontology prototypes. We expect from our model to allow an intuitive view to ontology engineering processes and facilitate a cost estimation in the run-up of cost-intensive evolution steps. We reach these improvements by a convergence of ontology engineering and ontology usage controlled by an innovative versioning approach.

#### 4 Functional Requirements of Corporate Ontology Engineering

Since we developed this corporate ontology lifecycle<sup>2</sup> it is possible to derive a set of functional requirements, which enables a holistic support for it. Thus, we examined each phase for needed tools and list the requirements as follows:

**Creation:** Access to global repositories of standard ontologies or available contractor for initial ontology development.

**Validation:** Discussion support and support for collaborative decision making for experts and non-experts.

**Population:** Tools for automatic knowledge acquisition

**Deployment:** System for supplying the appropriate ontology version to applications.

**Feedback tracking:** System for integration of lightweight extended communication platforms, e.g., forums or feedback forms and automatic recovery of user behavior into a feedback log.

**Synchronization:** System, which exports a snapshot of the log at a dedicated point of time.

**Evaluation:** Validation and reasoning tools which enable an evaluation of the log snapshot referring to the actual working ontology version.

**Evolution:** System which allows the evolution of ontologies (e.g. creation of views, coexisting branches or just new versions).

As a result of this it is necessary to examine existing tools along the new functional requirements raised. We aim at finding the appropriate tool(s), which suite the process integrative.

---

<sup>2</sup> <http://www.corporate-semantic-web.de/colm-lifecycle-methodology.html>

## 5 Comparison of Tools

In this section we give a brief overview of some accepted tools for ontology engineering tasks compared to the support of the different phases of our corporate ontology lifecycle. The desktop-applications Protégé and SWOOP as well as the web-applications OntoWiki and Ikwiki are in focus. These tools are representatives for the currently most accepted approaches to ontology engineering under the requirements of the methodologies we introduced.

**Protégé** is the most accepted tool for ontology building. Its appearance is similar to software development environments. Protégé is rich in function and language support and very scalable in cause of its extensibility. Since Protégé contains collaborative components it is possible to develop consensual ontologies in a distributed fashion using lightweight access to the process by discussion and decision making about proposed changes. This feature does not respect any roles or permissions. Versioning control is enabled on ontology level, but not on conceptual level, enriched by the annotations from the structured argumentations. Any abstraction from technical terms is missing. To sum up, Protégé is a very useful tool for engineering ontologies in a team of experts with a lack of lifecycle support in a usage-oriented architecture.

**SWOOP** is a desktop environment for ontology engineering, which is a bit straightforward at the expense of functionality. The representation of the concepts allows a web-browser-like navigation and is a bit intuitive for non-experts. A search form supports quick searches on the recently used ontology or at least all ontologies stored. Quick reasoning support is implemented in the same fashion. However, there is no abstraction from technical primitives enabled. By definition of remote repositories, it is possible to commit versions of ontologies.

Altogether, SWOOP is a tool for ontology engineering tasks for experts and well-experienced users. It has its strengths in quick and intuitive navigation in and search on ontologies but lacks functional flexibility and lifecycle support.

**OntoWiki** is a php-based wiki-like tool for viewing and editing ontologies. It is setting up on pOWL which makes use of the RAP API<sup>3</sup>. OntoWiki<sup>4</sup> allows administration of multiple ontologies (called knowledge bases) and provides in-line editing as well as view-based editing. As an abstraction from conceptual terms OntoWiki includes an alternative visualization for geodata (Google Maps) and calendars auto-generated from the semantic statements stored. However, a general abstraction from technical primitives (e.g. class,

<sup>3</sup> <http://www4.wiwiss.fu-berlin.de/bizer/rdfapi/>

<sup>4</sup> <http://aksw.org/Projects/OntoWiki>

subclass, SPARQL, etc.) in the user front-end is missing. Altogether, it allows only one single view for all users and does not respect any roles or permissions. Changes to the conceptualized knowledge have to be done manually. The ontology history is concept-oriented not ontology-oriented and implemented as known from wiki systems.

We subsume that OntoWiki is an ontology engineering tool and a knowledge base for experienced users with an academic background and that it does not support lifecycle management.

**Ikewiki** implements the semantic wiki-idea and focuses annotation of wiki-pages and multimedia content. It is possible to generate an alternative graph visualization for the context of each annotated page. However, Ikewiki does not support any abstraction from technical primitives for users with less experience in the field of ontologies. Restricted views referring to roles or permissions are not provided. The ontology history is concept-oriented not ontology-oriented and implemented as known from wiki systems.

We summarize about Ikewiki, that this tool addresses familiar wiki users with technical experience which do not need any control of the conceptualization and lifecycle support.

Our experience includes that there exist a strong distance between the recently accepted approaches and the needs of our ontology lifecycle. The tools either have an engineering-oriented perspective, which deals with the ontology application- and user-independent, or they reckoning the conceptualization on an application level for knowledge management without respecting unfamiliar users. The latter is emphasized if we note that the barriers of wiki-syntax for users without any technical background are underestimated. Thus, we subsume the support per phase of our model as follows:

Phase	Protégé	SWOOP	OntoWiki	Ikewiki
Creation/Selection	+	+	+	+
Validation	+	-	-	+
Population	-	-	-	-
Deployment	-	-	-	-
Feedback Tracking	-	-	-	-
Reporting	-	-	-	-
Evaluation	-	-	-	-
Evolution/Forward Engineering	+	+	+	+

Finally, we now conclude that no tool exists, which currently supports our lifecycle model. This is because available tools handle engineering tasks and ontology usage separately. Some tools work with ontologies as the central artifact on an engineering level while others support the application level only. Searching

for an adequate architecture or tool for integrative lifecycle support means to start from the perspective of the evolution by usage of knowledge. A smart versioning control is needed as the central component to enable this.

## 6 Conclusion and Outlook

In this paper we introduced our approach towards an innovative ontology lifecycle for corporate settings. From this model we derived functional requirements for an integrative tool support and compared four ontology development tools with reference to these requirements. We concluded that there is yet a lack of methodological foundations as well as tool support for the agile engineering of ontologies which is strongly needed in corporate contexts. We aim at an extension of this approach towards an innovative architecture for ontology lifecycle management in corporate contexts.

### Acknowledgement

This work has been partially supported by the "InnoProfile-Corporate Semantic Web" project funded by the German Federal Ministry of Education and Research (BMBF).

### References

- [Auer 06] Auer, S., Herre, H.: RapidOWL - An Agile Knowledge Engineering Methodology. In Irina Virbitskaite Andrei Voronkov, editeurs, Ershov Memorial Conference, volume 4378 of Lecture Notes in Computer Science, pages 424-430. Springer, 2006.
- [Abrahamsson 03] Abrahamsson, P., Warsta, J., Siponen, M. T., Ronkainen, J.: New Directions on Agile Methods: A Comparative Analysis. ICSE, pages 244-254. IEEE Computer Society, 2003.
- [Fernandez 97] Fernández-López, M., Gómez-Pérez, A., Juristo, N.: METHONTOLOGY: From Ontological Art Towards Ontological Engineering. AAAI-97 Spring Symposium on Ontological Engineering: Stanford, AAAI Press, 1997.
- [Kotis 06] Kotis, K., Vouros, A.: Human-centered ontology engineering: The HCOME methodology. *Knowl. Inf. Syst.*10(1), pages 109-131, 2006.
- [Lindvall 02] Lindvall, M. et al.: Empirical Findings in Agile Methods. Second XP Universe and First Agile Universe Conference, LNCS 2418, pages 197-207. Chicago, IL, USA, August 2002.
- [Noy 01] Noy, N. F., McGuinness, D. L.: *Ontology Development 101: A Guide to Creating Your First Ontology*. Stanford University, 2001.
- [Pinto 06] Pinto, H. S., Tempich, C., Staab, S., Sure, Y.: *Distributed Engineering of Ontologies (DILIGENT)*. Semantic Web and Peer-to-Peer, Springer, 2005.
- [Simperl 06] Simperl, E., Tempich, C.: *Ontology Engineering: A Reality Check*. OTM Conferences (1), volume 4275 of LNCS, pages 836-854. Springer, 2006.
- [Sure 02] Sure, Y., Studer, R.: *On-To-Knowledge Methodology — Expanded Version*. On-To-Knowledge deliverable, 17. Institute AIFB, University of Karlsruhe, 2002.
- [Tran 07] Tran, T. et al.: *Lifecycle-Support in Architectures for Ontology-Based Information Systems*. ISWC/ASWC, volume 4825 of LNCS, pages 508-522. Springer, 2007.