RapidOWL — An Agile Knowledge Engineering Methodology

Sören Auer and Heinrich Herre

University of Leipzig, 04109 Leipzig, Germany
auer@informatik.uni-leipzig.de
http://www.informatik.uni-leipzig.de/~auer/

1 Problem

The analysis of the application of the existing knowledge engineering methodologies and tools shows that they are up to now virtually not used in practice (see [13, page 16]). This stands in contrast to the often proclaimed necessity for knowledge engineering. What can be the reason for this discrepancy? Most of the existing knowledge engineering methodologies adopt techniques and apply process models from software engineering. However, in many scenarios required knowledge engineering tasks reveal specific characteristics, which an knowledge engineering methodology should be aware of. In the following, we describe briefly some specific characteristics of Knowledge Engineering important for RapidOWL.

Knowledge Engineering is not a Business in itself. There is no market for Knowledge Engineering as there is for Software Development. This is not because Knowledge Engineering is less important in the economic sphere, but due to the fact that the flow of knowledge in most cases accompanies the development of products and services, rather than being an economic asset itself. Hence, Knowledge Engineering services are often required when spatially distributed users have to collaborate on a semantic level. For example, this is the case when a common terminology has to be established, dispersed information must be integrated, or when shared classification systems and taxonomies have to be developed. This type of semantic cooperation is for example often required for Virtual Organizations [1], scientific communities or standardization boards, or intra-organizational use.

Lack of a Unique Knowledge Serialization. Agile methodologies rely heavily on sophisticated versioning and evolution strategies due to their focus on small incremental changes. However, agile methodologies, as well as their respective versioning and evolutions strategies within software development, do not seem to be reasonably applicable to knowledge engineering. For example, contrary to software development paradigms, most knowledge representation paradigms do not provide unique serializations. In other words, the ordering of statements or axioms in a knowledge base is irrelevant, while the ordering of source-code lines in software is fixed. Consequently, the use of existing software versioning strategies (e.g. delta method) and their respective implementations (e.g. CVS, Subversion) would not be efficiently suitable.

Spatial Separation of Parties. Most agile Software Development methodologies assume a small team of programmers working closely (especially spatially) with domain experts. This is a reasonable assumption for commercial software development, where a client requests software developers to implement a certain functionality. But when the involved parties are spatially separated, the use of a formal, tool-supported Knowledge Engineering methodology becomes particularly important. Furthermore, the knowledge engineering tasks of establishing common classification systems, shared vocabularies and conceptualizations are especially important in distributed settings. When teams are co-located implicit knowledge representation in the form of text documents in conjunction with verbal communication turns out to be more efficient and for a long time established.

Involvement of a Large Number of Parties. The growing together of the world by Internet and Web technologies enabled completely new mechanisms of collaboration. Open source software projects as for example the Linux kernel or collaborative content authoring projects as Wikipedia demonstrate this power of scalable collaboration impressively. However, Knowledge Engineering is especially challenging when a large number of domain experts have to be integrated into the knowledge-engineering process. Agile software development methodologies claim to be best suited for small to medium sized development scenarios. This is mainly due to the accent on and need for instant communication. On the other hand, the interlinking of people and tools using internet technologies facilitates scaling of agile cooperation scenarios. Knowledge Engineering scenarios in most cases differ from software development scenarios: it is usually not optional, but crucial to integrate a large number of domain experts, knowledge engineers and finally users of the knowledge bases.

2 Aim of RapidOWL

The aim of this paper is to help make the development and use of knowledge bases more efficient. For that purpose, a new, agile knowledge engineering methodology, called RapidOWL is proposed. RapidOWL is founded on the observation that knowledge must necessarily be modeled evolutionary, in a close collaboration between users, domain experts and knowledge engineers. We argue that existing heavy-weight development methodologies from Software Engineering and Knowledge Engineering are inefficient for certain application scenarios, because they make changes in knowledge models too expensive. Most existing Knowledge Engineering methodologies (e.g. Uschold [19], Grüninger and Fox [10], Methontology [7]) take a task as the starting point, i.e. they suggest performing ontology construction with the ontology's usage scenarios in mind. This requires significant initial effort and makes changes to and reuse of the resulting ontologies inherently hard (cf. [13]). The starting point of RapidOWL is the hypothesis that light-weight (or agile) development processes can be suitable for knowledge engineering, because they stress the importance of supporting frequently changing requirements and models and rely on a minimum of representation artifacts. The application of RapidOWL is supported by concepts for knowledge base versioning and evolution (see [4]) as well as rapid querying and view generation. Both are accompanied by a framework supporting the development of semantic-web applications on the basis of the RDF statement paradigm (see [2,3]). Applications and examples are presented how efficient tool support for RapidOWL can then be implemented on top of the framework in either generic or domain specific way (e.g. [5]).

3 Related Work

Related approaches can be roughly classified into two groups. Accompanied by the formation of knowledge engineering as an independent field of research several Knowledge Engineering methodologies were developed. Most of them are much inspired by Software Engineering methodologies. In the Software Engineering domain, in the 90's several Agile Software Engineering methodologies emerged. Triggered by the fact that flexibility, in particular fast and efficient reactions on changed prerequisites, becomes increasingly important, agile methodologies recently also appeared in other areas than Software Engineering. The most prominent representatives from each of these directions are briefly presented in the following.

Knowledge Engineering. The main goal of Knowledge Engineering is to structure the development and use of knowledge bases. For that purpose, the most widely known Knowledge Engineering approaches (such as CommonKADS [17]) are based on the ontology paradigm (i.e. knowledge should be represented in formal and explicit specifications [12]). Ontologies capture the semantics of the knowledge in a format that is designed to be on the one hand easy to maintain and on the other hand efficient to process by reasoning algorithms. The development of both ontologies and adequate reasoning algorithms is supported by various methodologies, the phases and models of which resemble traditional Software Engineering approaches. These Knowledge Engineering methodologies now also reveal similar problems to traditional Software Engineering approaches. Significant initial efforts are needed to make the purpose of the final ontology explicit and to deduce an appropriate model. It is often hard to estimate the required level of detail for the knowledge structuring a priori. Changes to the knowledge structuring are difficult and costly. Finally, only a few knowledge modeling tools allow easy collaboration between domain experts and knowledge engineers. For these reasons, methods from Knowledge Engineering are often too expensive to apply and rarely used in practice.

Agile Methodologies. Agile methodologies have recently gained growing success in many economic and technical spheres. This is due to the fact that flexibility, in particular fast and efficient reactions to changed prerequisites, is becoming increasingly important in the information society. This development started in Software Engineering after the realization in the mid 1990's that the traditional 'heavy' methodologies do not work well in settings where requirements are uncertain and change frequently. Several adaptive or agile Software Engineering methodologies subsequently evolved (see [6,8,18,15,16]). Agile methodologies are especially suited for small co-located teams and for the development of non life-threatening applications. Since the problem of uncertain, changing requirements is not limited to the Software Engineering domain, the idea of establishing adaptive methodologies, which can react to changing prerequisites, was also adopted by other domains than Software Engineering. These include 'The Wiki Way' [14] for Content Management, Rapid Prototyping [11] for Industrial Engineering. Also, the Lean Management method was used to some extent in the business management domain.

4 Results

The RapidOWL methodology is based on the idea of iterative refinement, annotation and structuring of a knowledge base. Its aim is to bring about a stable state of the knowledge base through small incremental changes from a multiplicity of contributors. A central paradigm for the RapidOWL methodology is the concentration on smallest possible information chunks (i.e. RDF statements). The collaborative aspect comes into play, when those information chunks can be selectively added, removed, annotated with comments or ratings. Design rationales for the RapidOWL methodology are to be light-weight, easy-to-implement, and support of spatially distributed and highly collaborative scenarios. The RapidOWL methodology is presented following other agile methodologies. It is grounded on the paradigms of the generic architecture of knowledge-based systems, knowledge representation for the semantic-web on the basis of the RDF statement paradigm and web technologies. The RapidOWL process then is characterized by values from which (on the basis of the paradigms) principles are derived for the engineering process in general, as well as practices for establishing those principles in daily life (cf. Fig. 1). The values of RapidOWL are Community (to enable collaborative knowledge base development and evolution), Simplicity (to increase knowledge base maintainability), Courage (to be able to escape representation dead-ends) and Transparency (to promote early detection of modeling errors). The practices envisioned to organize the Knowledge Engineering process in daily life include among others: Joint Ontology Design (to ease collaboration between knowledge engineers, domain experts and users), Information Integration (to ground the knowledge elicitation on existing information), View Generation (to provide domain specific views for human users and software systems) and Ontology Evolution (enabling the smooth adoption of modelings and corresponding instance data migration). These values, principles, and practices are the major ingredients of the RapidOWL methodology. In contrast to systematic engineering methodologies, RapidOWL does not prescribe a sequence of modeling activities that should be precisely followed. Furthermore, RapidOWL does not waste resources on comprehensive initial analysis and design activities. RapidOWL is primarily suited for establishing conceptualizations for information integration as well as the establishing of shared classification systems and vocabularies. The idea of applying agile paradigms to Knowledge Engineering with respect to the specific characteristics of Knowledge Representation is the major innovation of the presented approach.

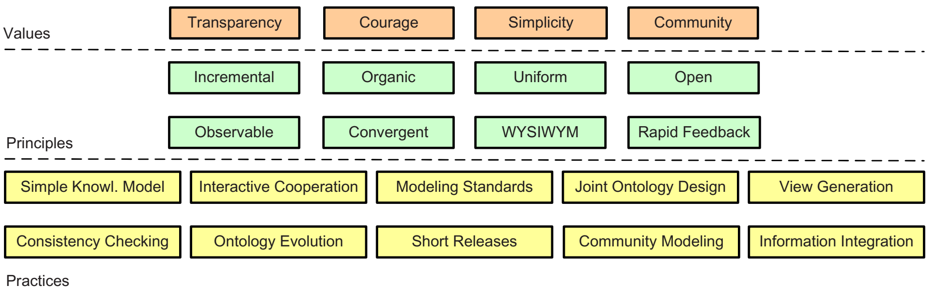

Fig. 1. The building blocks of RapidOWL: Values, Principles, Practices

In [9] a number of criteria for analyzing methodologies was proposed. In the following we discuss RapidOWL in the light of these criteria.

Detail of the methodology. RapidOWL is a rather lightweight methodology. This is primarily due to the recognition that knowledge engineering is usually not a business in itself and thus significant resources for evaluating the methodology and later controlling the compliance of the processes with the methodology are not available. RapidOWL rather banks on tools supporting it than on exhaustive documentation.

Recommendations for knowledge formalization. RapidOWL bases on representation of all knowledge in the form of triples, i.e. RDF statements. A concrete degree of formalization is not prescribed. However, RapidOWL proposes to justify the degree of formalization according to the required reasoning capabilities of the resulting knowledge base.

Strategy for building ontologies. Regarding this criteria it is questioned whether the strategy to develop ontologies is (a) application-dependent, (b) application-semidependent, or (c) application-independent. RapidOWL focuses on the development of rather application-independent ontologies. However, RapidOWL is primarily suited for information integration tasks and tasks related to the establishing of shared classification systems, vocabularies and conceptualizations.

Strategy for identifying concepts. RapidOWL here follows a middle-out strategy, i.e. from the most relevant to the most abstract and most concrete. By stressing the collecting of example or instance data RapidOWL tries to abolish knowledge elicitation by means of face-to-face communication between domain experts and knowledge engineers.

Recommended life cycle. Due to its adaptive nature RapidOWL does not explicitly propose a rigid life cycle. However, many aspects of stages in the life cycle of conventional methodologies can be discovered in RapidOWL's single process.

Differences between the methodology and IEEE 1074-1995. This criteria is related to the conviction that knowledge engineering processes should be similar to conventional software development processes. In this regard RapidOWL is different in two ways: Firstly it stresses the need to react on changed prerequisites, i.e. being agile. Secondly it assumes knowledge engineering to be fundamentally different from software engineering in certain scenarios.

Recommended techniques. RapidOWL stresses the importance of providing concrete techniques for performing the different practices of which the methodology is composed. However, in the description of RapidOWL's practices within this document only starting points on how to put them into effect are mentioned.

Usage and Application. Due to the fact that RapidOWL is rather new and significant resources had not been at our disposal for a broad evaluation the number of successfully realized RapidOWL projects is still small. However, ontologies and applications have been build on the basis of RapidOWL containing approximately 20,000 concepts and serving 3,000 parties (see e.g. [5]).

References

1. Wolfgang Ph. Appel and Rainer Behr. Towards the theory of virtual organizations: A description of their formation and figure. Arbeitspapiere wirtschaftsinformatik, Justus-Liebig-Universität Gießen Fachbereich Wirtschaftswissenschaften, 12 1996.
2. Sören Auer. Powl: A web based platform for collaborative semantic web development. In Sören Auer, Chris Bizer, and Libby Miller, editors, *Proceedings of the Workshop Scripting for the Semantic Web*, number 135 in CEUR Workshop Proceedings, Heraklion, Greece, 05 2005.
3. Sören Auer, Sebastian Dietzold, and Thomas Riechert. Ontowiki - a tool for social, semantic collaboration. In I. Cruz et al., editor, *Proc. of 5th International Semantic Web Conference, Athens, GA, USA, Nov 5th-9th*, number 4273 in LNCS, pages 736–749. Springer-Verlag Berlin Heidelberg, 2006.
4. Sören Auer and Heinrich Herre. A versioning and evolution framework for rdf knowledge bases. In *Proc. of Sixth International Conference Perspectives of System Informatics, Novosibirsk, Russia, Sep 27-30*, 2006.
5. Sören Auer and Bart Pieterse. "vernetzte kirche": Building a semantic web. In *Proceedings of ISWC Workshop Semantic Web Case Studies and Best Practices for eBusiness (SWCASE05)*, 2005.
6. Kent Beck and Cynthia Andres. *Extreme Programming Explained: Embrace Change, Second Edition.* Addison Wesley Professional, 2004.
7. Mercedes Blázquez, Mariano Fernández, Juan Manuel Garcia-Pinar, and Asunción Gómez-Pérez. Building ontologies at the knowledge level using the ontology design environment. In *Proceedings of the Eleventh Workshop on Knowledge Acquisition, Modeling and Management (KAW-98), Banff, Canada*, 1998.
8. Alistair Cockburn. *Crystal Clear.* Addison-Wesley Professional, 2004.
9. Mariano Fernández-López. Overview of methodologies for building ontologies. In *IJCAI99 Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends*, 1999.

10. Mark S. Fox and Michael Gruninger. Methodology for the design and evaluation of ontologies. In *Proceedings of the IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing, Menlo Park, USA*. AAAI Press, 1995.
11. Andreas Gebhardt. *Rapid Prototyping*. Hanser Gardner Pubns, 2003.
12. Tom R. Gruber. A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2):199–220, June 1993.
13. Holger Knublauch. *An Agile Development Methodology for Knowledge-Based Systems*. PhD thesis, University of Ulm, 2002.
14. Bo Leuf and Ward Cunningham. *The Wiki Way: Collaboration and Sharing on the Internet*. Addison-Wesley Professional, 2001.
15. Ken Orr and James A. Highsmith. *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*. Dorset House Publishing Co, 2000.
16. Stephen R. Palmer and John M. Felsing. *A Practical Guide to the Feature-Driven Development*. Prentice Hall PTR, 2002.
17. Guus Schreiber, Hans Akkermans, Anjo Anjewierden, Robert de Hoog, N. Shadbolt, Walter Van de Velde, and Bob J. Wielinga. *Knowledge Engineering and Management: The CommonKADS Methodology.* MITpress, 2000.
18. Ken Schwaber and Mike Beedle. *Agile Software Development with Scrum*. Prentice Hall, 1st edition, Oct 2001.
19. Mike Uschold. Building ontologies: Towards a unified methodology. In *16th Annual Conf. of the British Computer Society Specialist Group on Expert Systems*, Cambridge, UK, 1996.