Abstract

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. To support adaptive, semantic collaboration between domain experts and knowledge engineers, a new, agile knowledge engineering methodology, called RapidOWL is proposed. This methodology is based on the idea of iterative refinement, annotation and structuring of a knowledge base. A central paradigm for the RapidOWL methodology is the concentration on smallest possible information chunks. 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.

1 Introduction

RapidOWL is an adaptive, light-weight methodology for collaborative Knowledge Engineering. The major aim of RapidOWL is to make the elicitation, structuring and processing of knowledge and thus the cooperation of domain experts and knowledge engineers more efficient. The RapidOWL methodology is based on the idea of iterative refinement, annotation and structuring of a knowledge base. Central to the paradigm for the RapidOWL methodology is the attention given to the smallest possible information chunks (i.e. RDF statements). The collaborative aspect comes into it’s own by allowing those information chunks to be selectively added, removed, or annotated with comments and/or ratings. Design rationales for the RapidOWL methodology are to be light-weight, easy-to-implement, and supportive spatially distributed and highly collaborative scenarios.

RapidOWL is, on the one hand, inspired by the XP.K methodology (eXtreme Programming of Knowledge-based systems, [16]), which extends Extreme Programming to an agile methodology for the development of knowledge-based systems. On the other hand, RapidOWL is influenced by the Wiki idea [17], which established agile practices for collaborative text editing. However, contrary to XP.K the RapidOWL methodology stresses the generic nature of a knowledge base and thus focuses on development of knowledge bases, whose final usage scenario is either not a priori known or a single usage scenario is not easily definable. This is usually the case for conceptualizations targeting at information integration as well as for shared classification systems and vocabularies. Different from the Wiki idea on the other side RapidOWL’s artifacts are structured information and knowledge represented in statements rather than the Wiki’s unstructured text documents.

In this paper we will first clarify the notion of a knowledge engineering methodology (Section 2), then we present briefly approaches related to RapidOWL (Section 3). We present RapidOWL by exhibiting underlying paradigms and its single process (Section 4) and finally discuss RapidOWL in the light of criteria for analyzing knowledge engineering methodologies (Section 5).

2 Knowledge Engineering Methodology

Many definitions of methodology can be found in the literature (see [16, Page 9]). In this document we will (consistent with [16]) adopt the view of Alistair Cockburn [9], who defines a methodology as “an agreement of how multiple people will work together. It spells out what roles they play, what decisions they must reach, how and what they will communicate”.

Definition 1 Knowledge Engineering Methodology A knowledge engineering methodology is an agreement of how multiple people will work together. It defines a process in which domain experts and knowledge engineers will build a knowledge base. This knowledge base is
represented in a knowledge representation language with suitable tools. Processes, languages and tools are based on knowledge representation paradigms.

The RapidOWL methodology is presented in this document following other agile methodologies. Figure 1 summarizes the important ingredients, i.e. people, paradigms, processes, models and tools. RapidOWL is grounded on paradigms, which influence the process (see Section 4), they lay the foundation for the models, and have to be internalized by people. Last but not least, tools implement the collaboration processes between people on the basis of the methodologies models. This document primarily aims at sketching the agile Knowledge Engineering process of RapidOWL (for a more detailed description see [2]). In addition, a framework supporting RapidOWL tool development exists [1], with OntoWiki [2, Chapter 7] a domain independent tool supporting RapidOWL was developed and RapidOWL has also been applied in practice [5].

3 Existing Approaches

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 to changed prerequisites, becomes increasingly important, agile methodologies recently also appeared in other areas than Software Engineering.

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 [19]) are based on the ontology paradigm [15]. 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. For these reasons, methods from Knowledge Engineering are often too expensive to apply and rarely used in practice (cf. also [16, page 59]). However, for ontology construction the need for methodologies supporting ontology evolution and the basis of interactive collaboration between actors has been tackled by few approaches (e.g. [18]).

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 (e.g. [6, 10, 20]). 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’ [17] for Content Management, Rapid Prototyping [14] for Industrial Engineering. Also, the Lean Management method was used to some extent in the business management domain.

4 Process

Conventional methodologies distinguish different phases within the life-cycle of either software or knowledge. Agile methodologies give the importance of applying a change
a much higher value than being located in a certain stage of the life cycle. Consequently agile methodologies do not provide a phase model. Instead, they propose values from which (on the basis of paradigms) principles are derived for the engineering process in general, as well as practices for establishing those principles in daily life. We will describe RapidOWL along these dimensions in the following subsections, and then give an overview of how they can be combined and applied in practice.

Paradigms  The basic paradigms of conventional knowledge engineering methodologies are the generic architecture of knowledge-based systems, ontologies and problem-solving methods. We argue that the paradigms of an agile knowledge engineering methodology, with a focus on semantic-web knowledge representation standards, must both reflect the distributed interlinked nature of the Web and recognize statements as being the smallest building blocks of semantic-web knowledge bases. Hence, the vague knowledge representation paradigm of ontologies is replaced by knowledge representation grounded on the semantic-web data model, i.e. RDF statement paradigm and the use of web technologies is established as an additional paradigm.

Values. RapidOWL adopts the values of eXtreme Programming, namely Communication (to enable collaborative ontology development), Feedback (to enable evolution), Simplicity (to increase knowledge base maintainability) and Courage (to be able to escape modeling dead-ends). However, RapidOWL combines the values of Communication and Feedback in it’s Community value. This includes the social constructs that underly the communication and subsumes feedback as a special form of communication. In addition to XP’s values, RapidOWL includes the value of Transparency. These values are explained below.

Principles. Based on the four values which represent long-term goals, the RapidOWL development process is guided in the mid-term by various principles. They are partly inspired by Ward Cunningham’s design goals for the first Wiki system [11]. The principles include an open-world assumption, promotion of incremental changes, uniform authoring methods for both modeling and instance acquisition, observable development and rapid feedback (see [2] for details). The principles describe the single RapidOWL process axiomatically in the sense that they define characteristics the RapidOWL process should possess. Concrete practices are derived from the principles aiming at achieving these desired characteristics in daily routine without prescribing a rigid process.

4.1 Practices.

The practices of RapidOWL are inspired by the practices of eXtreme Programming (XP) and from the modifications for software / knowledge base co-design in XP.K [16]. Due to the specific characteristics of knowledge engineering (e.g. in contrast to software development it is usually not a business model) not all practices from XP have an equivalent in RapidOWL and inversely. Other than in software development, where the team of full-time programmers can be easily instructed to put the values and principles of XP into daily routine, RapidOWL aims to turn domain experts into part-time knowledge engineers by keeping practices as simple as possible and by proposing strategies to support them with tools. In the following we highlight the practices specific to knowledge engineering, namely 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).

Joint Ontology Design. Based on the simple subject-predicate-object statement paradigm, domain experts initially only define and describe concepts in a quite spontaneous manner. For this, they make use of worldwide unique concept identifiers (URI references) as subjects, predicates and objects within statements (i.e. RDF triples). After the information is thus represented, it can be enriched by more experienced domain experts or knowledge engineers with more detailed categorizations, classifications and logical descriptions (e.g. OWL axioms or SWRL rules). Domain experts on the other hand should be able to attach comments to the statements and vote for correctness and usefulness of the statements. In order to technically achieve this, RDF reifications may come into effect. To observe changes on higher conceptual levels than that of additions and deletions of statements, multiple added or deleted statements can be put into change sets. This is comparable to the way that patches subsume multiple atomic changes in software source-code versioning. Hierarchically organizing changes even allows change reviews on different levels of detail (see also the practice of Ontology Evolution).

Information Integration. The awareness of the need for developing a knowledge base often arises when a multiplicity of information sources exist and their interrelations are hard to maintain. Hence, it is crucial for the success of a knowledge engineering project to integrate existing information sources. Two approaches are possible to achieve
this goal: Importing information into the common knowledge base - all further maintenance of the information has to be done within the knowledge base. This method can be easily applied for simple structured information sources as for example Excel sheets containing tabular information (Powl [1] e.g. contains such functionality). The other possibility is to interlink existing data sources with the knowledge base. This method will be the preferred way if information sources are highly structured (such as e.g. relational databases or LDAP directories), existing applications rely on access to the information sources in conventional ways and the maintenance of the information should be performed in established processes. For relational databases the problem is tackled by D2RQ [8]. LDAP2OWL [12] is an approach to make information from LDAP directories accessible to knowledge bases.

**View Generation.** An aim of classical knowledge engineering is to realize problem-solving capabilities comparable to a domain expert ([22]). Additionally, knowledge engineering can be seen as an support strategy helping users and experts to transfer knowledge between individuals or to gain new insights by presenting knowledge in unforeseen ways. Generating different views on a knowledge base will support greatly the latter mentioned aspects of knowledge engineering. Views select and combine certain parts of the knowledge base for human users. On type of views are user generated views, where the user or viewer is identical to the generator to the view. Such ‘self service’ views can be realized in a user friendly manner by provision of full-text searches in combination with filtering and sorting functionality (see [2, Section 7.4] for details of an actual implementation). However, for certain investigations more sophisticated selection strategies than filtering and sorting might be needed or the presentation for humans should be tuned. In this case view generation will require either a more experienced user, able to cope with query languages and aware of the knowledge structures or knowledge engineers assisting in generating the view. Technically, this can be achieved by employing templates or declarative ways to encode presentation knowledge (e.g. Fresnel [7]).

**Ontology Evolution** The more spontaneous new modelings and information is integrated into the knowledge base the higher is the demand later to adopt modelings and migrate instance data accordingly. Many approaches to ontology evolution emerged recently (in [21] an overview is given), most of them, however, presuppose a well-defined and fixed set of possible knowledge base changes. To be able to review changes on different levels of detail (e.g. statement, ontology, domain level) we developed a hierarchical versioning strategy (see [4]) for the usage with RapidOWL.

**Short Releases.** A long-lasting develop-release cycle as in traditional software engineering would not adequately reflect the continuous information integration and knowledge structure evolution, which is characteristic for shared classification systems, vocabularies and conceptualizations. Instead we propose to make the development process as transparent as possible, by publishing changes immediately and releasing stable versions frequently. The sooner an ontology is released to the public, the easier it is to discuss changes and the earlier it is that potential problems become visible. Short releases will be facilitated if the consequences of changes to the ontology are known, i.e. if data adhering to the ontology has to be migrated or if queries about and views on the data have to be adopted.

![Figure 2. The building blocks of RapidOWL: Values, Principles, Practices.](Image)

**4.2 Putting it all together.**

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 analysis and design activities. Instead, it follows the philosophy of agile methodologies, in which agility in the face of changing requirements and knowledge models is a major goal. The individual tasks and contributions of prospectively involved parties in building a knowledge base on the basis of the RapidOWL principles are presented in the next paragraphs.

RapidOWL encourages domain experts to initially express all facts they assume as true and worth being, represented by means of statements. This can be simply the adding of a statement which attaches an rdfs:label or rdfs:documentation to a URI reference, to give that URI reference an informal meaning. Or, instance data can be gathered by importing existing documents (spreadsheets, listings, text documents) into the knowledge base. A spreadsheet containing structured information in tabular form, for example, can be interpreted as a class, where columns indicate properties and rows usually represent instances. This activity promotes a shallow learning curve, since domain experts can instantly participate. As soon as an expert starts working on such a knowledge representa-
tion, other experts can observe every single step. They can add comments, vote about the usefulness of certain representations, add their own knowledge fragments and/or delete other ones.

More experienced domain experts assist in restructuring, interlinking and consolidating the gathered data. Importing information from legacy documents often results in duplicates, because, for example, columns in different spreadsheet documents representing the same information are not labeled in a uniform manner. Such duplications have to be detected and eliminated (e.g. by merging the respective properties into one). To reduce the costs of such changes, RapidOWL will rely on implemented wizards assisting in the detection of frequent modeling errors and by providing (semi-)automatic resolution to support evolution and migration. It might also be necessary to convert literal data, as, for example, two properties to be merged can represent the same information differently (e.g. names like “Auer, Sören” or “Sören Auer”). This kind of consolidation activity also includes the establishing of relationships which are not yet represented.

Knowledge engineers can support such a community of domain experts with advice for reasonable representation methods and by providing ontology evolution and data migration strategies. Knowledge engineers can further enrich the knowledge base with logical ontology axioms. This includes property characteristics (e.g. transitivity) and restrictions (e.g. cardinality restrictions). Class descriptions are refined with set operators (owl:intersectionOf, owl:unionOf, owl:complementOf). The knowledge engineer can also extract distinct parts or ‘slices’ of the knowledge base adhering to the OWL species, DL and lite to perform consistency checks by means of Description Logic reasoner. Since RapidOWL does not restrict domain experts in their usage of RDF it is likely that the knowledge base does not fall into the OWL DL or OWL lite categories. The species validation of an OWL reasoner, however, can give hints about how the knowledge base has to be modified. Another task is the testing of existing queries and views on the knowledge base after changes have been incorporated.

RapidOWL focuses primarily on establishing conceptualizations for information integration as well as the establishing of shared classification systems and vocabularies. Hence, tools supporting RapidOWL will have a rather generic than domain specific nature. However, software developers participate in the collaboration by developing domain specific applications providing specific views onto the knowledge base, or by assisting domain experts and knowledge engineers in formulation more complex queries to the knowledge base.

RapidOWL does not enforce a distinct succession, neither does it require all the just mentioned tasks and activities to be accomplished by the respective parties. The quality of the knowledge base, however, is determined by carefully performing activities related to consolidation, restructuring and modeling as well as consistency checking. Tools on the other hand, can highly automatize and integrate these activities into the Gathering activity.

5 Conclusion

The purpose of RapidOWL is to bring about a stable state of the knowledge base through small incremental changes from a multiplicity of contributors. To achieve that, RapidOWL applies various techniques and practices with the explicit goal of reducing the cost of change. Although each of these practices have weaknesses when applied individually, their benefits greatly outweigh their weaknesses when they are used as a combined approach. In other words, the practices of RapidOWL support one and other. This analogous to other agile methodologies (cf. [16, 6]) and due to the ‘axiomatic’ description of their single process. An example for individual weaknesses and mutual compensation of such is the interplay of the practices of Short Releases and Ontology Evolution. Short Releases of the ontologies may result in instabilities of the resulting knowledge-based systems. However, Ontology Evolution supports the early detection of prospective problems and enables the revoke of individual problematic changes in a simple way.

In [13] 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 querying and 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 using elements of RapidOWL containing approximately 20,000 concepts and serving 3,000 parties (cf. the case study in [5]).

**References**