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 Rapid-OWL.

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