Using a Topic Grid and Semantic Wikis for Ontology-Based Distributed Knowledge Management in Enterprise Software Development Processes

Axel Korthaus, Martin Schader
University of Mannheim, Schloss, D-68131 Mannheim, Germany
{korthaus|mscha}@uni-mannheim.de

Abstract

Enterprise software development is a very knowledge-intensive process. Modern component-based application development organizations often require collaboration of several stakeholders distributed between different sites. To support developers in performing their tasks, suitable knowledge management tools are required that are able to leverage semantics. Recent developments in the context of the Semantic Web and ontologies including technologies such as Topic Maps or RDF/OWL show potential to address those issues.

In this paper, we present an architecture for a distributed knowledge management information system that uses Topic Maps in the background and Semantic Wikis as front end for developers to document and access relevant software development knowledge. The backbone is a Topic Map application infrastructure (called “Topic Grid” in the following) allowing transparent, node-spanning access to different Topic Maps distributed in a network.

1. Introduction

While knowledge management and software engineering existed as separate communities for a long time, the management of software engineering knowledge has become a trend in software engineering recently [4]. The development of Enterprise Application Systems is a human-based, highly knowledge-intensive process. Knowledge and experience of the developers involved in software engineering projects are key success factors to those projects. Although proponents of agile software development methodologies have pointed out that a company’s organizational and project memory should be shifted from external to tacit knowledge, i.e. informal face-to-face communication among team members should be estimated higher than written documentation, the absence of explicit knowledge documentation leads to several problems [16]. For example, given frequent personnel turnovers, implicit knowledge residing only in the heads of the developers is often lost for an organization. Also, experts in larger teams will have to answer the same questions again and again, team members encounter problems that they knew they have solved before but cannot remember the solution, or direct knowledge exchange is difficult if the members of the development teams are not co-located or belong to different organizations.

Thus, the need to employ knowledge management approaches to the elicitation, packaging, management and reuse of knowledge in order to make knowledge explicit and reusable remains valid even in the context of agile approaches. In software engineering, by the way, all kinds of artifacts including models, patterns, standards, lessons learned, requirements, code etc. can be seen as representing explicit software engineering knowledge that can and should be stored in knowledge bases.

Many problems connected with the design, the population and the effective use of knowledge bases have been identified in empirical case studies in the literature. For example, forcing developers to document their knowledge (e.g. in the form of lessons learned) for storage in a knowledge base on a regular basis will probably lead to acceptance problems as long as this task is perceived as being on top of and additional to the actual productive development tasks. This will especially be the case in agile software development processes where rigid and formalized process steps are largely rejected. Thus, suitable incentives must be established to increase the developers’ motivation, and, even more important, the knowledge management approach must be non-invasive and seamless with respect to the actual development process. To achieve the latter, lightweight tool approaches relying on the use of wikis are currently viewed as a promising possibility. The most recent variant of wikis, semantic wikis, appear to be well-suited to bridge the gap between a lightweight, informal tool for knowledge
Another problem category results from the fact that enterprise software development today is most often organized in a distributed way, involving different sites and organizations in a potentially globally distributed “software eco system”. Generally, it will be impossible to standardize a comprehensive, overall ontology on which knowledge management is based for all players and collaboration partners in the software eco system. On the contrary, local ontologies will have to be used in loosely coupled systems which, however, must be matched in order to reuse knowledge across single sites. Here, Topic Maps offer a powerful means for information management, representation of ontologies and merging of isolated ontologies.

To bring these ideas together, this paper will present an architecture that demonstrates how a Topic Map-based system can be used to organize local software engineering knowledge bases that can be populated by using semantic wikis, among other things. The architecture furthermore describes a distributed system infrastructure, called Topic Grid, which allows software developers to query the Topic Map-based knowledge bases not only locally, but across all participating sites, giving them the impression of querying one big, virtual Topic Map.

The remainder of the paper is structured as follows. Before we describe the use of Topic Maps for knowledge representation in Section 3 and present our Topic Grid infrastructure in Section 4, we motivate the design by some considerations about decentralized knowledge management in the following Section 2. Sections 5 and 6 are then dedicated to the explanation of how wikis and semantic wikis in particular can be used for documenting software engineering knowledge. In Section 7, we the propose the use of semantic wikis and front ends to the Topic Grid, list some related work in Section 8 and conclude the paper with some final remarks in Section 9.

2. Decentralized Knowledge Management

Corpor ate knowledge management systems today are typically based on centralized architectures with centralized knowledge bases containing the entire explicit knowledge managed in the company. A company-wide ontology often serves as the foundation for structuring that knowledge base. Using this design, a homogenization of different sources of data and information is enforced in order to allow for or improve interchange of knowledge on a corporation-wide level. Enterprise Knowledge Portals typically serve as an entry point to access the knowledge base from the intranet.

In software development settings, however, we typically have distributed organizations with stakeholders from different locations and even different companies involved. In such environments, we have to recognize that knowledge emerges in distributed ways, too (cf. Schwotzer ([27], [28]). If possible at all, forcing all local groups into the shared ontology of a centralized knowledge management system will certainly lead to user-side acceptance problems. Sigel [30] emphasizes this fact by stating that “different information needs require different views on contents”. Having the Semantic Web in mind, Cayzer states that in all of the conventional systems “… the ontology lacks semantics and is both centralized and universal. The Semantic Web may be better served by precise, local, domain-specific vocabularies that are loosely coupled, rather than by a one-size-fits-all central ontology, no matter how collaborative” [7].

As a consequence, each local group or team should be able to use their own ontology in order to avoid those problems. In the design of our distributed knowledge management system for enterprise software development we thus had to consider this requirement. What we had in mind was a grid-like infrastructure consisting of interconnected but independently acting knowledge nodes each having its own knowledge base. In that scenario, the overall knowledge captured in the system is given by the union of all the knowledge stored on the individual nodes.

According to Bonifacio [6], such systems must at the same time provide autonomy and coordination. The first requirement refers to the goal that knowledge nodes in the system must be able to act autonomously and independently of each other. In each node, knowledge can be created and stored locally. Particularly, each node is allowed to organize this knowledge on the basis of an individual ontology in order to guarantee what is called “semantic autonomy”.

On the other hand, in order to be able to reuse knowledge in a location-spanning way, it must be guaranteed that all nodes are capable of exchanging knowledge. A common ontology is not needed as the basis for this. Rather, there must be concepts that allow intermingling and linking of different ontologies. This ability is denoted as “semantic interoperation” or “coordination”. By specifying sophisticated identity and merging mechanisms, the Topic Map standard offers the basic concepts on which such solutions can be built and which are missing in similar metadata standards such as RDF (for a detailed comparison of RDF and Topic Maps see, for example, [12]).
3. Using Topic Maps for Knowledge Representation

Similar to RDF and OWL, Topic Maps ([17], [18]) belong to the set of semantic technologies constituting a new paradigm shift that can be observed in the IT domain recently, e.g. along the road towards the Semantic Web, Semantic Web Services and Semantic Social Software, or for the implementation of knowledge management applications. Topic Maps are a relatively new formal ISO standard (ISO/IEC 13250:2003) and are conceptually similar to semantic networks and concept maps. Essentially, they serve for the representation and interchange of knowledge. As a standard for managing metadata, they can be used for many different application purposes, e.g. to generate navigation for a website, to integrate heterogeneous information resources and for lots of other metadata tasks.

A Topic Map can express knowledge using topics representing any subject or concept, associations representing the relationships between them, and occurrences, which represent relationships between topics and information resources relevant to them. Topic Maps offer a powerful foundation for knowledge representation and the implementation of knowledge management applications. Using ontologies to model knowledge structures, they offer concepts to link these knowledge structures with unstructured data stored in files, external documents etc. This is achieved by strictly separating the knowledge structures and the associated potentially unstructured information resources. Thus, semantic queries on the heterogeneous information resources become possible, which outperform conventional index-based or full-text searches by far. The business problem to be addressed by Topic Maps is thus the “disconnectedness of information and knowledge” in order to achieve what is sometimes referred to as “knowledge integration”, “seamless knowledge” or “global knowledge federation” [26].

4. A Topic Grid Infrastructure

The backbone infrastructure component of our distributed software development knowledge management system is what we call a “Topic Grid”. It provides the functionality required to be able to exchange knowledge structures between several different applications.

In [20], the basic idea of the Topic Grid is described as being a network of nodes each of which provides its own knowledge base in the form of one or more Topic Maps (see Figure 1). These Topic Maps are made available for queries from other nodes by means of a standardized protocol. In the Topic Grid, a client is capable of querying all knowledge bases belonging to a certain group in parallel, so that a single, transparent view on the knowledge bases and knowledge structures is created. The Topic Grid aims at providing applications with a homogeneous view of distributed Topic Maps pretending that the user works with one big, connected Topic Map. This is evocative of a typical notion of grid computing, namely the virtualization of a multitude of physically separated computer systems.

Figure 1: Abstract sketch of the Topic Grid

In [19], we describe the design and implementation of Java-based Topic Grid prototype using an access protocol stack for distributed Topic Maps in a network to realize the idea of the Topic Grid. In order to enable queries on groups of distributed knowledge bases, higher protocols need to be built on a suitable technology for distributed systems development, like web services, CORBA etc., and require group or multicast communication facilities on the implementation level. Since we already implemented these features in the form of an Object Group- and Join Service (OGS) [2] in CORBA in an earlier project, we decided to reuse this CORBA-based solution for the prototypical Java implementation of our Topic Grid.

The Topic Grid architecture was designed as a multi-protocol layered model in order to enhance its reusability. For example, instead of using our OGS as the technical communication base of this architecture, several other possible technologies such as HTTP, OGSI, or UDP could be employed. On top of this basic layer lies the Semantic Network API (SNAPI) [32], which implements a generic protocol for the exchange of knowledge structures. The next layer is formed by our Topic Map Fragment Remote Exchange Protocol (TMFREP) implementation which provides the functionality required to query Topic Maps in a network. Being able to handle fragments of Topic Maps, TMFREP represents a core functional component for the realization of a Topic Grid. On the top layer, we
implemented a framework that simplifies the access to the Topic Grid for client applications. The design of this framework is geared to the Model-View-Controller design pattern (MVC) [33]. As opposed to the lower layers, the framework on the uppermost layer is stateful and uses memory space to store previously performed queries.

As already mentioned, the modular architecture facilitates potential replacement of the basic transport layer without affecting any of the upper layers. This can be beneficial if a Topic Grid is required that has to be extended with real grid computing features. To this end, only changes in the SNAPI layer implementation are necessary, while all the upper layers can remain unchanged since they only rely on the unmodified interface provided by SNAPI.

5. Software Engineering Wikis

The Topic Grid is of no use if the knowledge bases are not filled with valuable and reusable information. A relatively new and increasingly popular way to integrate documentation and authoring tasks into everyday work processes is the use of so-called wikis. Before we analyze how wikis need to be adjusted to serve as a front-end to our Topic Grid, we first describe the basics of the wiki concept.

“Wikis” (from the Hawaiian “wiki wiki” for “quick”, “fast”) are a relatively new type of software application offering a lightweight approach to web-based content management that provides ease of interaction and operation for collaborative writing and editing of web documents (titled wiki pages) by multiple users. The ease with which such titled wiki pages can be edited and updated using a very simple markup language is characteristic of this collaborative technology. The body of wiki pages, which are usually highly interconnected via hyperlinks, is similar to a “WWW in the small” and represents a very basic user-maintained database for searching information. Usually, wikis provide at least a title search, and sometimes a full-text search. Links between wiki pages can be created using a specific syntax.

Due to these characteristics, wikis have become very popular for different application purposes in private, academic and commercial environments. In enterprises, they are often used as internal documentation of application systems, for example. In order to describe the knowledge contribution, storage and exchange function of wikis, the word “wiki” has sometimes even been viewed as a backronym for “what I know is”.

This function is it what makes wikis interesting in the software engineering context. As Decker et al. [9] put it, “wikis can be seen as a lightweight platform for exchanging reusable artifacts between and within software projects”. They consider wiki systems as lightweight Organizational Memories or Experience Factories and emphasize the following basic features of wikis:

- one place publishing: at any time, there is only one version of a document available that is regarded as the current version
- simple and safe collaboration: most wikis provide versioning and locking mechanisms
- easy linking: documents within the wiki can be linked by their title using a simple markup
- description on demand: links can be defined to pages that are not yet created but might be filled with content in the future.

According to the authors, by offering a way to edit content collaboratively using a web browser, wikis provide a kind of basic open source knowledge repository with a very low technical barrier for usage. Decker et al. provide a nice overview of wikis with specific functionality for software engineering. Among others, they mention the Portland Pattern Repository [22], the wikis of the Apache Foundation [3] and more specific wikis such as Trac [10], MASE [23], SnipSnap [11], SubWiki [8], and WikiDoc [25], which all have special application areas such as issue tracking, iteration planning for agile software development projects or the creation of code documentation, to name just a few.

In their RISE project [9], the authors attempt to add value to the use of wikis as platforms for software development by augmenting the wiki concept with semantic features. In RISE, this concept is called “Wikitology”, since it combines the idea of a wiki with a background ontology for semantic linking and annotations of wiki pages. The semantic wiki in the RISE project is used to support lightweight software requirements engineering. Since we plan to implement a semantic wiki as one of several front-ends to our Topic Grid infrastructure, the next section provides some more details of the concept.

6. Semantic Wikis

If used as a simple knowledge repository, regular wikis have the disadvantage that the information about a topic on a wiki page might provide structure and meaning to a human reader but does not possess any machine-understandable semantics. Similarly, the hyperlinks between different wiki pages are untyped. For
these reasons, advanced features required for effective knowledge management, like enhanced semantic searching and browsing based on metadata, are not available.

As Völkel [35] points out, sites like Wikipedia would greatly benefit from adding ontology-based metadata, since their pages contain a lot of possibly structured information, which could be leveraged for several purposes if they contained metadata in a machine-interpretable format. In that way, the internal knowledge could be made more explicit and more formal, so that it can be queried in better ways than just with full-text searches.

For these reasons, several projects were initialized to implement semantic enhancements to the wiki approach. The basic idea of semantic wikis is to provide an underlying formal and (partially) machine-processable model of the knowledge described in their pages, which allows users to capture or identify metadata about the pages and their relations. Based on this knowledge model, new facts, e.g. implied relations between pages, can be inferred from existing facts in the knowledge model.

The level of ontology support offered by different semantic wiki systems differs. For example, the approach of the Semantic Wikipedia implementation is to add some extra syntax for semantic annotations to the wiki markup language [35]. In this approach, wiki pages can be interpreted as semantic concepts, but do not have to be. For our software development support system, we are envisioning a semantic wiki that is fully based on an ontology. All wiki pages represent concepts or instances and contain semantic relations to other entities as well as textual descriptions.

7. Semantic Wiki-Based Access to the Topic Grid

By combining the Topic Grid and a Semantic Wiki that uses the Topic Map technology in the background to provide an ontology that helps to annotate wiki pages and links with machine-interpretable metadata, we are able to meet the requirements stated in the introduction. First, by sticking to the wiki technology, we provide a very lightweight approach to knowledge provisioning that does not significantly hinder the core software development process. By using wikis with semantic enhancements, we achieve the benefits implied by ontology-based semantic technologies, like improved semantic searches. Because of the Topic Grid infrastructure in the background, we can allow distributed teams at different sites to use different ontologies while remaining to be able to search the complete Topic Map information space.

As a first application scenario, we think of developers documenting their current development tasks and lessons learned using the semantic wiki. A suitable ontology for annotating their information serves as an entry point into the Topic Grid, which can be seen as a superimposed semantic network over a multitude of heterogeneous electronic documents and information resources (maintained inside and outside the company) providing relevant information to the developers. New pages edited in the wiki can thus automatically be added in a semantically structured way to the local Topic Map managed at the developer’s site and thus become part of the body of information. This body of information, however, will also be increased by many other Topic Map-aware applications. We have already implemented several prototypical applications in that sense, e.g. a tool to generate a Topic Map semi-automatically from a conventional document index provided by the Lucene tool [21] or a Topic Map-based web application modeling knowledge about refactoring tasks.

The semantic search functionality offered by the semantic wiki will thereby not be limited to the local wiki contents, but will access the whole Topic Grid, hence accessing a very comprehensive knowledge base in order to satisfy the developer’s information needs that had arisen for him to successfully perform his current development activity.

For example, questions like “What issues are involved when combining EJBs and JDBC?” can lead to the system providing links to information resources containing the relevant information or even contact information of experienced colleagues, lesson learned stories, relevant design patterns, newsgroup postings etc.

8. Related Work

The use of Topic Maps-based approaches for implementing different kinds of distributed systems has been studied by several research groups recently. While the prototypical implementation of our proposal relies on a group communication-based architecture, TMShare [1], for example, is a proprietary approach based on a peer-to-peer architecture realized with the open source framework JXTA. The focus of this research was on how the exchange of subsets of Topic Map information between peers can be used as the basis for information sharing in a distributed environment. The notion of Topic Map fragments goes back to this work. Similarly, the kPeer project [31] represents a
P2P-based Topic Map system approach. Ontopia pushes the rather ambitious idea of providing “seamless knowledge” [26] based on the worldwide use of Topic Maps in the Internet. To this end, Ontopia tries to establish two standardized protocols for the communication with and between Topic Map engines: the Topic Map Remote Access Protocol (TMRAP) [14] and the Topic Map Query Language (TMQL) [15]. However, both specifications are still unstable and are not yet supported by prominent Topic Map engines like TM4J. Contrary to our approach, TMRAP is designed for Topic Map-based interoperability between web portals and thus provides a HTTP binding. However, it strongly influenced our design of the TMFREP protocol, which is realized as a Java interface and supports distributed fragment extraction. We used tolog [13] instead of TMQL, because of the support for this query language in the TM4J engine. Many other related approaches entirely depend on the yet unstable TMQL, like the REST-based TMIP [5] and [34], require Topic Maps to be stored in specific SQL databases for TMRQL queries, like the SOAP- and WSDL-based Topic Maps Service [24], or they address different design aspects, like the Shark architecture [29] for mobile handheld units.

For a god overview on different semantic wiki implementations, which are not specific to the software engineering domain, see [9]. The RISE project with its concept of “Wikitology”, however, is an example of the use of a semantic wiki in the software development context to support requirements engineering tasks. As opposed to our approach, it does not consider the potential of integrating the semantic wiki with a distributed Topic Grid in the background.

9. Conclusions

Topic Maps are a relatively new way of organizing explicit knowledge and providing a means to search or navigate in information resources based on semantic net structures, thus exceeding the possibilities of conventional index-based or full text-based searches. In this paper, we have presented an architecture and a prototypical implementation for a distributed knowledge management system infrastructure that allows a client to query any number of Topic Maps residing on distributed nodes in a network in a transparent manner, as if the client would locally access one big unified Topic Map. The architecture uses a clearly structured layered protocol stack in order to facilitate the replacement of parts of the implementation. For example, in order to provide for the required group communication, we use our own CORBA-based Object Group and Join Service as the basic communication layer, but different protocols could be used as well without affecting higher layers of the architecture. Thus, the effort of modifying the Topic Grid implementation in order to use a Grid computing infrastructure like the Globus Toolkit as the foundation should be limited, because it only affects the next layer, the implementation of the SNAP protocol.

Our prototypical implementation in Java showed very promising results, so that we currently plan to implement a knowledge management system based on the Topic Grid architecture, which will be used to support software developers collaboratively working together in different locations in performing their distributed software engineering activities. Topic Map-aware applications that will be integrated via the Topic Grid in that setting will include semantic wikis used to document software engineering knowledge.

In this paper, we could just give a first impression of the ideas we are exploring and were required to neglect many crucial aspects, starting from technological details like security considerations to the point of methodological and organizational questions, e.g. regarding suitable incentive systems or consequences of the potential need for new organizational roles like knowledge engineers. These aspects will be the focus of later publications, after first experiences with the semantic wiki prototype as a front end to the Topic Grid have been made.

10. References


