New Applications for Wikis in Software Engineering

Michael Geisser[1], Hans-Jörg Happel[2], Tobias Hildenbrand[1],
Axel Korthaus[3], and Stefan Seedorf[3]

[1]: University of Mannheim, Department of Information Systems I {geis-ser|hildenbrand}@wi.uni-mannheim.de
[2]: Research Center for Information Technologies, Information Process Engineering
happel@fzi.de
[3]: University of Mannheim, Department of Information Systems III
{korthaus|seedorf}@uni-mannheim.de

Abstract: Within software development, wikis are currently mainly used for brainstorming and documentation purposes or error management and project coordination. This article describes four advanced application scenarios for wiki support in software development processes: Requirements Engineering, Traceability and Rationale Management, Architectural Knowledge Sharing, and Lessons Learned Management in a distributed knowledge infrastructure. Finally, we will give a conclusion by summarizing the main advantages and drawbacks of the presented innovative uses of wikis in software engineering.

1 Introduction

Wikis are easy to use but powerful tools for collaborative knowledge formation and knowledge sharing. Due to their widespread adoption in Web communities, wikis are being increasingly employed in enterprise settings [MaWY06], especially in software development projects. Their current uses in software development range from brainstorming and documentation to error management and project coordination. However, there is still potential for a more extensive support of software processes, which has not been fully tapped yet. The question is therefore how other core and supporting activities in Software Engineering (SE) can benefit from innovative uses of wikis.

This article describes new applications of wikis to support various SE activities. First, we give an outline of wikis in general and the more recent "semantic wikis", which enhance traditional wikis to allow machine-interpretable content marking and contribute to an integrated process support. After a short overview of wiki applications in SE, we describe four concrete application scenarios and corresponding prototype implementations: Requirements Engineering (RE), Traceability and Rationale Management (TRM), Architectural Knowledge Sharing, and Lessons Learned Management in a distributed knowledge infrastructure.

The process of RE, the initial phase of a software development project, comprises requirements elicitation, analysis, specification, and validation for the planned system. For this purpose, wikis are a lightweight and agile alternative to many commercial and partially very complex solutions. In RE and the following phases of software design, interdependencies of requirements and between requirements and the resulting artifacts, as well as the rationale they are based on, should be made accessible (TRM). For the collection and cross-linking of such information, a systematic TRM-process is necessary and Wiki-systems can be appropriate implementation tools due to their usability and hypertext capabilities. They can also serve for architectural knowledge sharing by enabling multi-perspective, collaborative documentation and exchange of different views of software architectures. By leveraging the semantic power of semantic wikis, additional value can be achieved, e.g. with regard to advanced browsing, querying, and searching of the knowledge base. The Ontobrowse semantic wiki, which is described in section 3.4, is an example of a semantic wiki tool supporting this application scenario. Finally, we present ToMaWiki, a semantic wiki that can be used as a front end to a distributed, topic map-based knowledge infrastructure and can be employed, for example, to implement a Lessons Learned Management System (LLMS) for storing, synthesizing, and reusing software project knowledge according to the Experience Factory organizational concept [BaCR94]. We conclude by summarizing the main advantages and drawbacks of the presented innovative uses of wikis in software engineering.

The methods and tools introduced in this paper have mainly been developed in the context of the CollaBaWue research project[1]. The first two examples are based on the wiki system of CodeBeamer[2], a collaborative software development platform from Intland, but can be transferred to most other wiki systems. The Ontobrowse semantic wiki has been developed from scratch, and ToMaWiki uses the open source semantic wiki Ikewiki[3] as its foundation.

2 Wikis

Wiki (Hawaiian for “fast”) is a term used to denote software programs that provide an easy method for multi-user web-based cooperation. It is important to distinguish between the software required to operate a wiki (“wiki engine”) and a wiki instance (“wiki”, “wiki installation”). The well-known online encyclopedia Wikipedia, for example, uses the MediaWiki software as its wiki-engine. Although most wiki engines are available as open source, the number of commercial products and hosting/ASP vendors in the market (e.g. SocialText, JotSpot) has been rising lately.

[1] http://www.collabawue.de
[2] http://www.codebeamer.com
[3] http://sourceforge.net/projects/ikewiki/

A wiki is basically made up of several distinct pages, which are interconnected via hyperlinks – comparable to a “small version” of the World Wide Web, which is also made up of linked documents. As in web content management systems (WCMS), wiki pages can be directly created and modified using a Web browser. However, they do not have fixed content categories or limited user rights. Generally, all users can edit the contents of a wiki by using a simple wiki language or a WYSIWYG editor. When a user edits a page, a revision history is saved so that previous versions can always be restored. Thus, social control replaces complicated rights management in WCMS [EbGl05].

Since the basic functions of wikis are as simple as those of common web-based email services, they allow an easy start. Although this leads to fast results and user satisfaction in editing text-based knowledge, it provides only little help in structuring this knowledge. Thus, with increasing use of the wikis, they tend towards complexity and content sprawling [MaWY06]. This leads us to the main drawback of wikis: Although the content of a wiki page might provide structure and meaning to a human reader, it does not possess any machine-interpretable semantics. Advanced knowledge management features such as semantic search and metadata-based filtering are thus not available in traditional wikis. However, a site like Wikipedia could heavily benefit from structuring content with additional metadata, which can be used to derive a knowledge model [VöKr06]. In that way, explicit but informal knowledge embedded in a page could be transferred into machine-processible knowledge, which can then be used for semantic queries.

This extension towards so-called “semantic wikis” has been realized in several projects, either by implementing a completely new wiki engine or by extending an existing one. Although the core idea of all semantic wikis is to provide a machine-processible knowledge model described in the wiki pages, they vastly differ in terms of required user experience and knowledge representation languages. For example, the Semantic MediaWiki adds some extra syntax for the semantic annotations to the wiki markup language [VöKr06]. It therefore realizes a very open approach where a user can optionally add semantic markup. Our Ontobrowse semantic wiki, on the other hand, interprets every page as an entity, which may be a concept, object property, datatype property, or individual object [HaSe07]. Thus, its semantics are defined in a more rigorous style, which enables us to acquire semantic data from external knowledge bases. In sections 3.4 and 3.5, two applications are described and the added value of semantic wikis is demonstrated.

3 Wikis Supporting the Software Development Process

The wiki technology was invented and originally used by software developers. The first wiki, initiated by Ward Cunningham[4] in 1995, has served as a knowledge repository for design patterns. Since then, software development has been remaining one of the most important application areas of the wiki technology [Lour06]. Frequently, a small group of users sets up a wiki, which is subsequently used by an ever-growing number of em-

[4] http://www.c2.com/

ployees. The simple availability and installation of wiki engines is especially helpful in this respect.

Aside from open source development communities such as the Apache Foundation and smaller enterprises, large software enterprises such as SAP, Novell or Yahoo have adopted wikis as well (cp. [HGN06]). General applications in SE comprise knowledge transfer, technical documentation, quality and process management, release planning, and error tracing [BaMe05, MaWY06, TWIK06]. However, for specific, well-structured content, traditional wikis often reach their limits with their core functionality. Thus, add-ons with specific functionalities are already available for problems such as source code documentation [AgDa05, JoSW05] or error tracing [Edge06].

We will now describe four concrete application domains of wikis in software development processes: Requirements Engineering, Traceability and Rationale Management, Architectural Knowledge Sharing, and Lessons Learned Management in a distributed knowledge infrastructure. Among these, the last two examples make special reference to the concept of semantic wikis.

3.1 Requirements Engineering

RE, the systematic elicitation, analysis, specification, validation, and management of software requirements, is usually being performed either with heavyweight RE tools or even with common office software. Traditional RE tools like RequisitePro and DOORS as well as office products are widely spread, but were originally not designed for an internet-based environment. This causes, for instance, performance problems and inefficient processes (cp. [IHGH07]). For larger software projects, distributed RE via office software can only be regarded as a makeshift solution. Documents often are only distributed via email but do not get assigned centralized version numbers.

There are no integrated methods for distributed RE so far. Therefore, we present DisIRE (Distributed Internet Based Requirements Engineering), a wiki-supported method for distributed, internet-based RE with specific tool support, which can be integrated in existing development platforms. DisIRE combines successful RE approaches and extends them to create a solid theoretical and empirically valid methodical framework[5]. Moreover, DisIRE offers the first integrated RE method for a distributed environment where requirements are explicitly taken into account. This method allows a system vendor to perform a largely distributed and at the same time systematical RE process. A moderator accompanies all parties through the process as described below.

3.1.1 Requirement Elicitation and Analysis

DisIRE follows immediately after the feasibility study. At this point, aside from a vision of the planned system, there is also the certainty that the project has realistic goals. Usu-

[5] Such approaches are: EasyWinWin [Grün03], QuantitativeWinWin [RuEP03] and the Cost-Value-Approach [KaRy97]. A detailed description of the DisIRE method can be found in [GHHR07].

ally the first step includes the requirements elicitation and analysis, which is described in this section.

As face-to-face meetings of all parties involved lead to better results in distributed work contexts as well (compare [GeHi06]), an initial meeting takes place with as many participants as possible, but no less than one representative from each location and organizational unit involved. The goal is to make sure that all attendants get a uniform picture of the planned system.

After the initial meeting, the requirements are elicited asynchronously and then analyzed. An overview of the different roles with their corresponding tasks is shown in Table 1. When all requirements are commented, consolidated and verified, and it seems like no further comments or requirements will be communicated, the moderator can freeze these requirements.

Roles	Task
Representative of the customers	• Transmission of requirements • Commenting on requirements • Identifying unclear technical terminology • Definition of unclear technical terminology • Extending and specifying vision
Moderator	• Force specification • Consolidation and categorization of requirements • Identification of unclear technical terminology • Extending and specifying vision
Software Engineer	• Commenting on requirements • Feasibility testing • Identification of unclear technical terminology

Table 1: Roles with their corresponding tasks in requirement elicitation and analysis with DisIRE

A wiki is especially suitable for the process described above: for each initial requirement a distinct wiki page is created where related comments follow that are consolidated directly in the description of the requirement. An integrated versioning system guarantees that all changes can be traced on each page. Furthermore, articles or particular domains are locked during editing to avoid conflicts. Another advantage of wiki systems is the possibility to use an integrative glossary: Technical terminology, which is used for the description of the requirements and which needs further explanation, can be marked through wiki-links and be defined in a glossary. All other asynchronous activities can also be thoroughly supported by wiki technology. CodeBeamer, a collaborative software development platform from Intland Software[6] [RBGH07], has an integrated wiki module and was used within the CollaBaWue project for an integral support of the DisIRE process [GHHR07].

[6] http://www.intland.com/

3.1.2 Requirements Selection

Requirements elicitation and analysis produces a consolidated list of requirements, whose full implementation, however, is not always economically rational. The list may contain requirements that cause an elaborate implementation but bring little gain. Hence, those requirements should be selected whose implementation appears to be rational under economic criteria. Since no wiki technology is used in this step within DisIRE, this activity will not be presented here (see [GHHR07]).

3.1.3 Requirements Specification and Validation

Subsequent to requirements selection, in order to achieve a certain degree of specification for the later steps, the selected requirements have to be specified. If, as in DisIRE, stakeholders on the customers' side are asked directly about their requirements, generally functional requirements are concerned. Use cases are particularly suitable for the specification of this type of requirements. Hence, for an internet-based specification, templates for such use cases are made available directly in CodeBeamer's requirements tracker. As a result, software engineers can use these templates on every workstation connected to the Internet to specify the requirements. Finally, the specified requirements have to be validated by the stakeholders on the customers' side to make sure that during the specification no information has been lost and no misunderstandings have occurred. This is done with the help oft an integrated comments function of CodeBeamer’s Requirements Tracker.

To enable versioning of specifications and respective comments as well as parallel editing, a wiki was directly integrated into CodeBeamer’s requirements tracker and its comments function. In this way, wiki content and other tracker items and artifacts from CodeBeamer projects can be referenced very easily and quickly.

3.2 Traceability and Rationale Management

In the next phases of software design, interdependencies of requirements and between requirements and the resulting artifacts, as well as the rationale they are based on, should be made available. This entails, above all, the administration of different interdependencies and the exact reasons and alternatives for each decision, such as modifications, in every step of the process [DMMP06]. The more distributed stakeholders are, the more complicated this task becomes [RaJa01]. Thus, based on the requirements specifications in RE, “Wiki-TRaM” will now be introduced, a Wiki-based TRM-process for information acquisition, management, visualization, and analysis. Wiki-TRaM is made up of two disciplines, “Collection and Management” and “Visualization and Analysis“, at which we will take a closer look below.

3.2.1 Collection and Management of Traceability and Rationale Information

Although DisIRE allows systematic requirements elicitation, analysis, and selection, the specification is in most cases not yet complete and stable. Therefore, an equally systematic change management is also necessary. That is only possible if traceability of the development process of the requirements (source traceability), of the dependencies of requirements among each other (requirements traceability) and of the resulting artifacts (design traceability) is given [Somm04]. The dependencies of pre-specification artifacts are vitally important for the requirements and change management within the framework of RE. These consist of the requirements descriptions and comments, which are linked with the specified requirements from an issue tracker through a corresponding wiki page. Issue trackers are used for managing task descriptions and status. These in turn are deposited together with the dependencies among the specified requirements, the cost and value ratios, as well as the reasons for the selection of the decisions in CodeBeamer’s issue tracker. Thus, apart from the traceability and rationale information, the implementation status of the individual requirements can also be managed and the respective artifacts versioned.

In the next step of the software project, the post-specification phase, relationships between the specified requirements and the resulting artifacts, e.g. design models, source code, and test cases, as well as all essential decisions and their rationale (design rationale) are recorded. For this, wikis offer the necessary functionality for content linking and commenting.

This allows for easier customization, maintenance, and reuse of the system and faster training of new project staff. To this end, different trackers for each task field (RE, architecture design, development, etc.) are used on the CodeBeamer platform and the respective tasks are linked through their association mechanism or through wiki links. Standardized relationships between tracker items themselves, artifacts, and external URLs are recorded using associations. Besides CodeBeamer’s own document management system, artifacts, e.g. source code, from external repositories like Subversion can also be referenced. All changes to an artifact require the posting of a comment, or in the case of SVN a commented “commit”. Due to the use of wiki comments, relationships to all CodeBeamer artifacts and external resources can be established. With the help of these mechanisms (associations and wiki links), traceability and rationale information can be documented and linked during the entire project.

3.2.2 Visualization and Analysis of Traceability and Rationale Information

Based on detailed recording of the information above, a heterogeneous “trace network” emerges, which contains relationships between process steps (i.e. tracker items), artifacts, and persons involved. For a clearer visualization and a more effective analysis of this traceability information, the tool “TraVis” was developed (TraVis = Trace Visualization). It is a Java-application, which extracts the respective basic information using CodeBeamer’s interface and displays it according to role-based filters. These filters allow different views for developers, consultants, project managers, etc. The displayed artifact categories and relationships can vary. Additionally, individual artifacts, tracker items, or users with their direct and indirect traceability network can be extracted. Further details of this tool will be left out here, since it only processes data from a wiki.

3.3 Architectural Knowledge Sharing

The bridge between requirements engineering and concrete design is software architecture. Many stakeholders are involved in the development and maintenance of an architecture. Hence, integrated tool support is difficult, because various knowledge needs have to be catered for. On the one hand, developers want technical support and guidance for their implementation task at hand. On the other hand, architects and business analysts demand tools for analysis and documentation. Thus, the actual representation of an architecture instance is usually split up into several "views", such as functional, physical or logical [Kruc95]. Most of these views can be assigned a certain purpose, such as quality, communication, analysis, or reuse [BaCK03, Bosc00]. Software architecture documentation should include all these views. However, existing tools are often not flexible enough to support the requirements of both groups appropriately, which leads to scattering of architectural knowledge into different information spaces.

What is sought after is a solution that offers both flexibility in documentation and collaboration and a formal basis for leveraging machine-interpretable semantics. Although this requirement sounds contradictory at first glance, semantic wikis are a promising candidate for solving this trade-off. From our point of view, semantic wikis are well-suited to bridge the gap between technical and business documentation. First, they encourage collaborative documentation and information exchange. Second, they provide the means for processing machine-interpretable knowledge, which is required for handling technical descriptions. Within the CollaBaWue project, the Ontobrowse semantic wiki has been specifically developed for the sharing of architectural knowledge. It provides the following main features [HaSe07]:

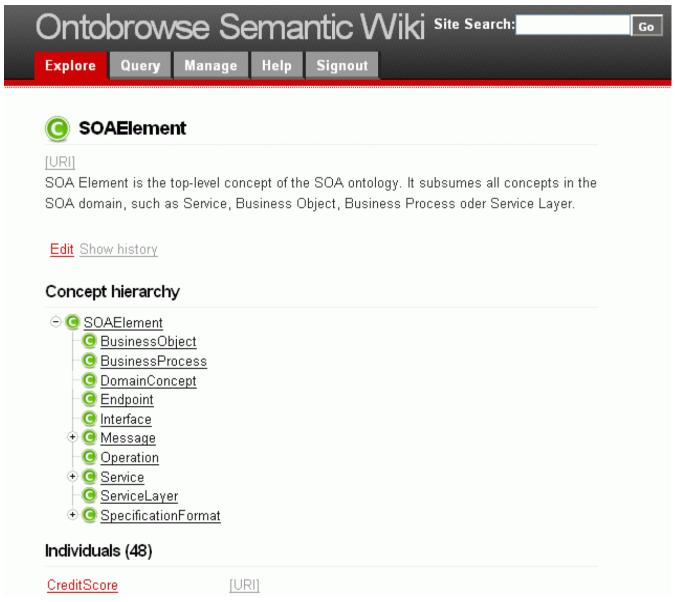
- Defining a knowledge structure using ontologies
- Browsing, querying and searching of a knowledge base
- Combining informal and formal documentation
- Integrating asserted knowledge from external sources
- Consistency checking with rules

The semantic wiki has been implemented employing the Web Ontology Language (OWL) as its knowledge representation format. The knowledge base can be configured to use either Jena or Kaon2 as reasoner. There is also experimental support for enforcing architectural rules using the Semantic Web Rule Language (SWRL).

An application scenario is the documentation of a service-oriented architecture (SOA). SOA is an architectural style, which propagates the orchestration of business processes from independent, loosely-coupled services. Service-orientation leads to a rising level of alignment between business processes and IT implementation. Therefore, it becomes more important to monitor and guide the development of the services landscape

[HaSe07]. Because the black-box specification of a service encourages a higher decoupling of software systems, responsibilities are shared by different service providers – together with the associated business and technical knowledge.

In order to integrate architectural descriptions of a SOA into the wiki, one has to perform two distinct steps: First, the users of the wiki need to agree upon a unifying *SOA ontology*. The ontology defines the terminology of the architecture together with relations and constraints. The terminology can usually be displayed as a concept hierarchy, with a generic concept such as “SOAElement” subsuming more specific concepts such as “Service” and “BusinessObject” (see Figure 1).



SOAElement

[URI]

SOA Element is the top-level concept of the SOA ontology. It subsumes all concepts in the SOA domain, such as Service, Business Object, Business Process oder Service Layer.

Edit Show history

Concept hierarchy

- SOAElement
  - BusinessObject
  - BusinessProcess
  - DomainConcept
  - Endpoint
  - Interface
  - Message
  - Operation
  - Service
  - ServiceLayer
  - SpecificationFormat

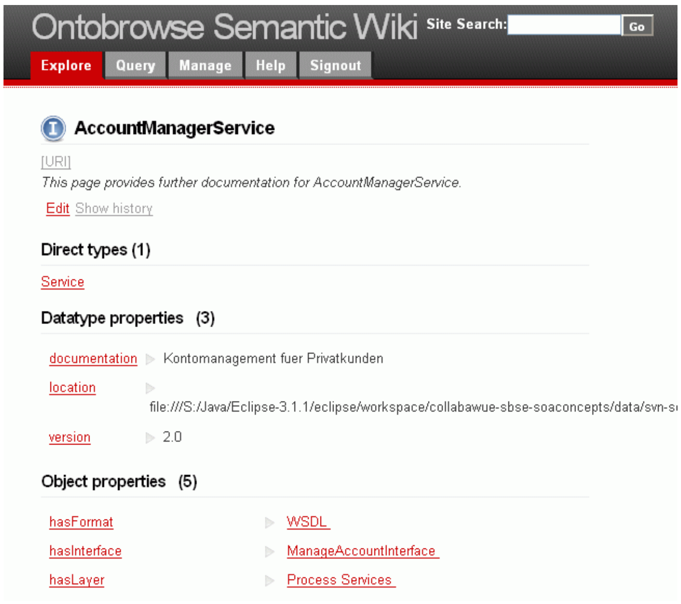
Individuals (48)

CreditScore [URI]

Figure 1: A generic concept in Ontobrowse

The ontology reduces conceptual ambiguity and enables information integration. Second, a plug-in has to be defined, which performs a mapping of architectural knowledge in a given format to the SOA ontology. The plugin can then be configured to crawl for matching specification files in one or more directories. As a result, a service specification mapped from a WSDL file is displayed on a page together with a property description (see Figure 2). Once an entity has been imported into the knowledge base, it can be augmented with textual descriptions and additional metadata. The same pattern can be repeated for other types of knowledge, such as requirements or business process specifications.

Ontobrowse addresses key issues in the documentation and maintenance of software architectures. Previously, the different views of an architectural instance were maintained in separate information spaces. The semantic wiki enables the integration of both business-oriented and technical knowledge, thus serving as a single point of information for all stakeholders. Due to the underlying formal representation based on ontologies, searching and querying can be significantly improved. At the same time, the plugin infrastructure makes it possible to integrate knowledge from external sources. Architectural descriptions such as service specifications in two different formats can be mapped to the same ontology. Finally, the conceptual structure is modular so that additional ontologies can be added at any time, e.g. to describe the organizational structure. These extension features are particularly important in distributed development settings (cf. [Cock96]), where the participants have to share their knowledge with other developers.



AccountManagerService

[URI]

This page provides further documentation for AccountManagerService.

Edit Show history

Direct types (1)

Service

Datatype properties (3)

documentation	Kontomanagement fuer Privatkunden
location	file:///S:/Java/Eclipse-3.1.1/eclipse/workspace/collabawue-sbse-soaconcepts/data/svn-s
version	2.0

Object properties (5)

hasFormat	WSDL
hasInterface	ManageAccountInterface
hasLayer	Process Services

Figure 2: An individual object in Ontobrowse

3.4 Software Engineering Lessons Learned Management

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 impracticable 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. To support this scenario, we have implemented the backbone infrastructure component for a distributed software development knowledge management system, which we call a “Topic Grid”. It provides the functionality required to be able to exchange knowledge structures between various applications.

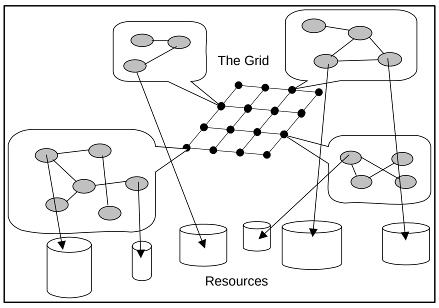


Figure 3: Abstract Sketch of the Topic Grid

In [KoHi03], 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 3). It 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 software developers. The 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. In [KAHS06] and [KoAH08], we describe the design and implementation of a 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.

To use the Topic Grid infrastructure in a real-world context, we have implemented a knowledge management system based on this infrastructure, which will be used to support software developers collaboratively working together at different locations to perform distributed software engineering activities. As Topic Map-aware applications to be integrated via the Topic Grid in that setting, we envision, among others, semantic wikis. A first prototype of a semantic wiki ("ToMaWiki") as front end to the Topic Grid has already been built. ToMaWiki not only provides typical semantic wiki functionality but also a graph-based navigation feature for displaying fragments of the complete Topic Grid (cf. Figure 4; [KoSc06]). With the help of this feature, users can quickly jump to relevant knowledge topics whose details are then displayed as regular wiki pages. Software developers can use this kind of Topic Grid-aware semantic wikis in an ad hoc way to document software engineering knowledge or project experience on the fly, which can subsequently be accessed from other wikis or applications on the Topic Grid. Software developers can use the wikis, for example, to exchange ideas on domain and technical issues, to store decisions made and their rationales, to share social information, to identify and locate experts, to self-coordinate collaborative work tasks, and to track progresses on project tasks (cf. [ChMa05]).

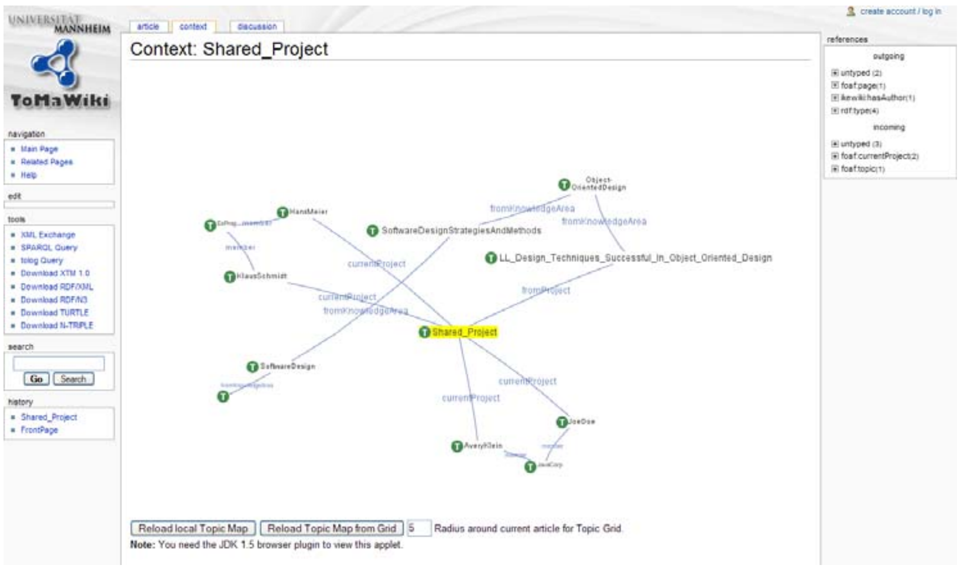


Figure 4: Screenshot of the Graphical Navigation Feature of the Prototypical Semantic Wiki Front End to the Topic Grid

By combining the Topic Grid and the ToMaWiki semantic wiki using Topic Map technology in the background to provide an ontology that helps to annotate wiki pages and links with machine-interpretable metadata, we were able to achieve some important goals. First, by sticking to the well-known 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 leveraged by ontology-based semantic technologies, like improved semantic searches. Based on the Topic Grid infrastructure in the background and the Topic Map technology's concept of merging and identity, which allows automated integration of topic maps from diverse sources into a coherent new topic map, we enable distributed teams at different sites to use at least partially different ontologies while remaining able to search the complete Topic Map information space.

As a first concrete application scenario, we think of developers documenting their current development tasks and "lessons learned" using the semantic wiki to implement a kind of "self-organized experience factory unit" (cf. [ChMa05]). A suitable ontology for annotating their information serves as an entry point into the Topic Grid. New pages edited in the wiki can then 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 in the Topic Grid. This body of information, however, will also be increased by many other Topic Map-aware applications. We have already implemented several prototypical applications for this purpose, e.g. a tool to generate a Topic Map semi-automatically from a conventional document index provided by the Lucene tool

[KoKS04] or a Topic Map-based web application modeling knowledge about refactoring tasks.

The semantic search functionality offered by the semantic wiki is not limited to local wiki contents, but accesses the whole Topic Grid, hence tapping 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.

4 Conclusion and Outlook

New Web-based collaboration technologies, such as wikis, which let users easily publish and share content, are thriving recently not only in the context of Web 2.0 and the Social Web, but also in various business contexts, and especially in the domain of software development. Although they lack sophistication, the strength of traditional wikis as a platform for collaborative authoring and sharing of contents lies in their simplicity and efficiency. As has been demonstrated in this paper, the “social software” wiki (cf. [Bäch06]) can be of great use for software engineering, not only in traditional application areas such as distributed documentation. With the continuing evolution of wiki technology and the development of new methodological approaches, other usage scenarios in software engineering can be opened up. We have presented Requirements Engineering as well as Traceability and Rationale Management as two concrete examples of software development activities that can benefit from the methodical employment of wiki technology.

Both application cases and their wiki-based implementations make it clear that wikis are in no way limited to open-source development, but are also an interesting and flexible approach for the support of enterprise processes. In the open-source domain, wikis have so far been employed as an “agile” tool for knowledge management and asynchronous collaboration. Aside from RE and TRM, other application areas of software development should be entered in the future, since—with the increasing specialization of roles in the development process—an ever growing amount of information requires distributed storing and automatic processing.

To this end, however, an alleviation of the weak points of traditional wikis regarding machine-interpretable structuring of contained knowledge is necessary. In traditional wikis, a metadata infrastructure is absent, and information is usually handled in an ad hoc fashion. Semantic wikis represent an innovation that aims at expressing content in machine-processable forms that enhance search precision and logical reasoning. Based on two examples of prototypical semantic wiki implementations for architectural knowledge sharing and software lessons learned management, we have demonstrated the potential advantageousness of semantic wiki technology for distributed software development scenarios. Although the added value that can be gained from semantic approaches in software development is not generally contested, their long-term success will depend considerably on the question how seamlessly they can be integrated in the core business processes without being perceived as causing too much undesired work overhead.

References

[AgDa05] Aguiar, A.; David, G.: WikiWiki weaving heterogeneous software artifact. In: Proceedings of the 2005 International Symposium on Wikis, San Diego, CA, 2005, S. 67-74.

[BaMe05] Bachmann, F.; Merson, P.: Experience Using the Web-Based Tool Wiki for Architecture Documentation. Technical Note CMU/SEI-2005-TN-041. September 2005.

[Bäch06] Bächle, M.: Social Software. In: Informatik Spektrum, 2006, 29, S. 121-124.

[BaCR94] Basili, V.R.; Caldiera, G.; Rombach, D.: Experience Factory. In: Marciniak, J.J. (ed.), Encyclopedia of Software Engineering, vol. 1, 1994, John Wiley & Sons, S. 469-476.

[BaCK03] Bass, L.; Clements, P.; Kazman, R.: Software Architecture in Practice. 2. Addison Wesley, 2003.

[BeHL01] Berners-Lee, T.; Hendler J.; Lassila, O.: The Semantic Web. In: Scientific American, 284, 5, 2001.

[Bosc00] Bosch, J.: Design and use of software architectures: adopting and evolving a product-line approach. ACM Press/Addison-Wesley Publishing Co., 2000.

[ChMa05] Chau, T. and Maurer, F. (2005) 'A Case Study of Wiki-based Experience Repository at a Medium-sized Software Company', Proc. of 3rd Int. Conf. on Knowledge Capture (K-CAP '05), Oct. 2-5, 2005, Banff, Alberta, Canada, pp. 185-186.

[Cock96] Cockburn, A.: The interaction of social issues and software architecture. In: Commun. ACM 39, October, Nr. 10, 1996, pp. 40-46.

[DeRe05] Decker, B.; Rech, J.; Ras, E.; Klein, B.; Hoecht, C.: Self-organized Reuse of Software Engineering Knowledge supported by Semantic Wikis. In: Proceedings of the Workshop on Semantic Web Enabled Software Engineering (SWESE). November 2005.

[DMMP06] Dutoit, A.H.; McCall, R.; Mistrik, I.; Paech, B. (Hrsg.) Rationale Management in Software Engineering, Springer Verlag, 2006.

[EbGl05] Ebersbach, A.; Glaser, M.: Wiki. In: Informatik Spektrum, (28), 2005, S. 131-135

[Edge06] Edgewell.org: The Trac User and Administration Guide, URL: http://trac.edgewall.org/wiki/TracGuide (26.09.2006), 2006.

[GeHi06] Geisser, M.; Hildenbrand, T.: A Method for Collaborative Requirements Elicitation and Decision-Supported Requirements Analysis. In: Ochoa, S.F. und Roman, G.-C. (eds): IFIP International Federation for Information Processing, 2006 (219), Advanced Software Engineering: Expanding the Frontiers of Software Technology, S. 108-122, Springer, Boston.

[GHHR07] Geisser, M.; Heinzl, A.; Hildenbrand, T.; Rothlauf, F.: Verteiltes, internetbasiertes Requirements-Engineering. In: WIRTSCHAFTISINFORMATIK, Volume 49 (3), 2007, pp 199-207.

[Grün03] Grünbacher, P.: EasyWinWin: Eine groupware-unterstützte Methode zur Erhebung und Verhandlung von Anforderungen. In: Softwaretechnik-Trends der Gesellschaft für Informatik, 23, 2003.

[HaSe07] Happel, H.-J. and Seedorf, S.: Ontobrowse: A Semantic Wiki for Sharing Knowledge about Software Architectures. In: Proc. of the 19th Int. Conf. on Software Engineering and Knowledge Engineering (SEKE), Boston, USA, July 9-11, 2007, S. 506-512.

[HGN06] Hildenbrand, T.; Geisser, M.; Nospers, M. (2006): "Die Übertragbarkeit der Open Source-Entwicklungsmethodik in die Unternehmenspraxis". In: Softwaretechnik-Trends, Volume 26 (1), pp 37-42

[IHGH07] Illes-Seifert, T.; Herrmann, A.; Geisser, M.; Hildenbrand, T. (2007): "The Challenges of Distributed Software Engineering and Requirements Engineering: Results of an Online Survey". In: Proceedings of the 1st International Global Requirements Engineering Workshop (GREW´07), pp 55-65, Munich, Germany

[JoSW05] John, M.; Jugel, M.; Schmidt, S.; Wloka, J.: Wikis in der Softwareentwicklung helfen. In: Java Magazin, 7, 2005, S. 88-91.

[KaRy97] Karlsson, J.; Ryan, K.: A Cost-Value Approach for Prioritizing Requirements. In: IEEE Software, 14, 5, 1997, S. 67-74.

[KoAH08] Korthaus, A., Aleksy, M., and Henke, S.: A Distributed Knowledge Management Infrastructure Based on a Topic Map Grid. In: International Journal of High Performance Computing and Networking (IJHPCN), Inderscience Publishers, vol. 6, issue 2/3, 2008 (to appear).

[KAHS06] Korthaus, A., Aleksy, M., Henke, S., and Schader, M., "A Distributed Topic Map Architecture for Enterprise Knowledge Management". Proc. of the 1st IEEE/ACIS Workshop on Component-Based Software Engineering, Software Architecture and Reuse (COMSAR '06), July 10-12, Honolulu, Hawaii, USA, 2006.

[KoHi03] Korthaus, A., Hildenbrand, T., "Creating a Java- and CORBA-Based Enterprise Knowledge Grid Using Topic Maps", in: Cheung, W.K., and Ye, Y. (eds.), Proc. Workshop on Knowledge Grid and Grid Intelligence", Oct. 13, 2003, Halifax, Canada, 2003, pp. 207-218.

[KoKS04] Korthaus, A., Köhler, C. and Schader, M. (2004) '(Semi-) Automatic Topic Map Generation from a Conventional Document Index', *Proc. IASTED Int. Conf. on Knowledge Sharing and Collaborative Engineering (KSCE 2004)*, St. Thomas, US Virgin Islands, Nov. 22-24, pp. 101-108.

[KoSc06] Korthaus, A. and Schader, M. (2006) 'Using a Topic Grid and Semantic Wikis for Ontology-Based Distributed Knowledge Management in Enterprise Software Development Processes', *Proc. of the IEEE Int. Vocabularies, Ontologies and Rules for the Enterprise Workshop (VORTE 2006) / Tenth IEEE International EDOC Conference - The Enterprise Computing Conference,* 16. October, Hong Kong, China, IEEE Computer Society.

[Kruc95] Kruchten, P.: The 4+1 View Model of Architecture. In: IEEE Softw. 12 November, Nr. 6, 1995, pp. 42-50.

[Lour06] Louridas, P.: Using Wikis in Software Development. In: IEEE Software, 23, 2, 2006, S. 88-91.

[MaWY06] Majchrzak, A.; Wagner, C.; Yates, D. 2006. Corporate wiki users: results of a survey. In: Proceedings of the 2006 International Symposium on Wikis, Odense, Denmark, ACM Press, New York, NY, 2006, S. 99-104.

[Onto06] Ontoworld.org: Semantic Wikis. URL http://ontoworld.org/wiki/Semantic_wiki (16.09.2006), 2006.

[RBGH07] Rodriguez, Felix and Berkling, Kay and Geisser, Michael and Hildenbrand, Tobias, "Evaluating Collaboration Platforms for Offshore Software Development Scenarios". In: Meyer, Bertrand (eds): Proceedings of the First International Conference on Software Engineering Approaches For Offshore and Outsourced Development (SEAFOOD'07), pp. 96-108, Springer Lecture Notes in Computer Science (LNCS).

[RaJa01] Ramesh, B.; Jarke, M.: Toward Reference Models for Requirements Traceability. In: IEEE Transactions on Software Engineering, IEEE Press, 2001, 27, S. 58-93.

[RuEP03] Ruhe, G.; Eberlein, A.; Pfahl, D.: Trade-Off Analysis For Requirements Selection. In: Int. Journal of Software Engineering and Knowledge Engineering, 13, 2003, S. 345-366.

[SaVa01] Saaty, T.L.; Vargas, L.G.: Models, methods, concepts & applications of the analytic hierarchy process. Kluwer, 2001.

[Somm04] Sommerville, I.: Software Engineering. Addison-Wesley, 2004.

[TWIK06] TWiki.org: TWiki Success Stories. URL: http://twiki.org/cgi-bin/view/Main/TWikiSuccessStories (26.09.2006), 2006.

[VöKr06] Völkel, M.; Krötzsch M.; Vrandecic D.; Haller, H.; Studer, R.: Semantic Wikipedia. In: Proc. of the 15th Int. Conf. on World Wide Web, Edinburgh, Scotland, May 23-26, 2006.

[W3C04] W3C 2004: The Web Ontology Language (OWL) Specification. URL: http://www.w3.org/TR/owl-features/ (16.09.2006), 2004.