Making Wikis productive as the glue code of project information (Project WAVES)

Tim Romberg, Peter Szulman {romberg|szulman}@fzi.de

FZI Research Centre for Information Technologies at the University of Karlsruhe

Abstract. Today's software engineering projects require a knowledge management infrastructure that allows integration of existing information sources (e.g. issue trackers) and the exchange across organizational boundaries. In Project WAVES, we implement a solution based on a semantic, virtual knowledge base and a Rich Wiki Client.

1 Trends and Knowledge Management scenarios in software engineering today

Software engineering today is faced with the following trends:

- A greater proportion of effort is spent on *maintaining, refactoring and reintegrating* existing software, rather than writing it from scratch. This is because hardware platforms have been more stable and scalable in recent years than in the earlier computing history, when the entire application stack had to be created from scratch periodically.
- Typical projects can involve *several programming languages and frameworks*. It is not uncommon, for example, to find an application server component written in Java accessing a library written in C++; the Java component in turn serving both a Web interface written in PHP with some Flash elements and a Microsoft .NET Rich Client. Common Internet standards such as SOAP or XML make it possible to plug together best-of-breed frameworks for each task, rather than having to choose one.
- Projects are *distributed both geographically and over several organizations*. A full offshoring of all programming activities to a low-cost country is only one possibility of many.
- The *growing library of Open Source software* provides opportunities to jump-start development on new products by integrating or extending existing code, or to reach a large audience of existing users with add-on products and services.

This leads to Knowledge Management scenarios which cross organizational borders and require integration of existing knowledge, such as:

- A small interface design consultancy is hired, together with a number of other specialized service providers, into a software development project at a large telecommunications company. The question is how to set up a knowledge repository for this project where partners are provided selective access to each other's existing knowledge (such as a style guide), and it is ensured that any contributions by a contractor’s employees is still accessible to at least this contractor after the project ends.

- A company (e.g. Apple) publishes a software component (e.g. Safari WebKit) as Open Source. While the Open Source project has a public site providing version management, issue tracking, continuous integration, etc., the company still needs to manage private information with respect to this component (e.g. issues which would reveal coming features of its competitive offerings). The relationships of this information with the public information need to be kept in sync.

2 The role of Wikis

It is a frequent point of contention how *Knowledge* Management is different from *Information* Management, where the one starts and the other ends. No matter where one draws the line, the members of a software engineering team (encompassing programmers and testers as well as designers, business analysts and customer contacts), clearly need access to both knowledge and information, for example:

- the concepts and methods that define their respective qualification (such as the concepts of object-oriented programming or lambda calculus for a programmer, the concept of perspective for a graphic designer);
- knowing how to use the specific tools and raw material in their job (programming languages, libraries and environments, graphics software etc.);
- background knowledge of the problem domain their activities address (e.g. new legislation that will have to be implemented by the software product, users’ background);
- specific infrastructure and processes in their work environment (how to log onto the staging server, whom to notify when changes are made);
- and finally, knowledge concerning the very project at hand – what the responsibility of certain modules are, why certain design decisions were made, which limitations and workarounds exist etc.

Today, mature tools and integrated Application Lifecycle Management (ALM) suites exist for managing standard processes and most aspects of the current project information. In many of the other areas, wherever a team needs to capture evolving knowledge, Wikis have established themselves as a versatile tool for managing knowledge whose structure is not well defined beforehand (as opposed to project information that can be modelled in databases). In fact, the first Wiki (Ward Cunningham’s WikiWiki) was created to share software engineering patterns – an example for a semi-structured type of knowledge.

3 Integration, sharing and distribution requirements

However, today's Wiki engines constitute an island of their own with respect to existing information sources and the outside world. The goal of the Waves project is to address the mentioned challenges of Knowledge Management across organizational borders and integrating existing information sources. While similar challenges exist in many problem domains, we consider Software Engineering to be especially relevant, since it involves electronic artefacts which to a certain degree are vehicles of knowledge themselves, and since the dependencies between projects across organizations are long-term and require ongoing collaboration.

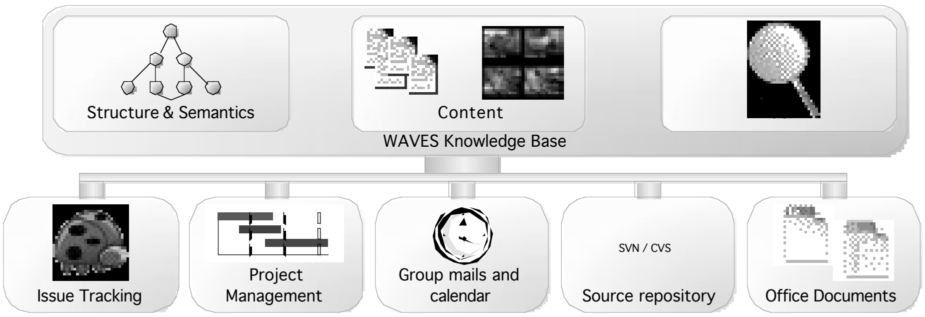
In this paper, we want to especially address the integration, sharing and distribution requirements, such as:

- being able to easily reference any existing information item (such as a bug or a release) when authoring Wiki pages;
- annotating any existing information item (e.g. using a Wiki page);
- integrating different visibility spaces (public, company-wide, team-wide, personal), by
  - being able to capture private (less visible) comments on public (more visible) knowledge resources;
  - being able to easily publish currently private (less visible) knowledge into a public (more visible) space while keeping a maximum of outgoing and incoming references intact.

4 Solution architecture

Our solution architecture consists of an integrated virtual knowledge base which allows querying knowledge (integrated search) and authoring knowledge (Rich Wiki client).

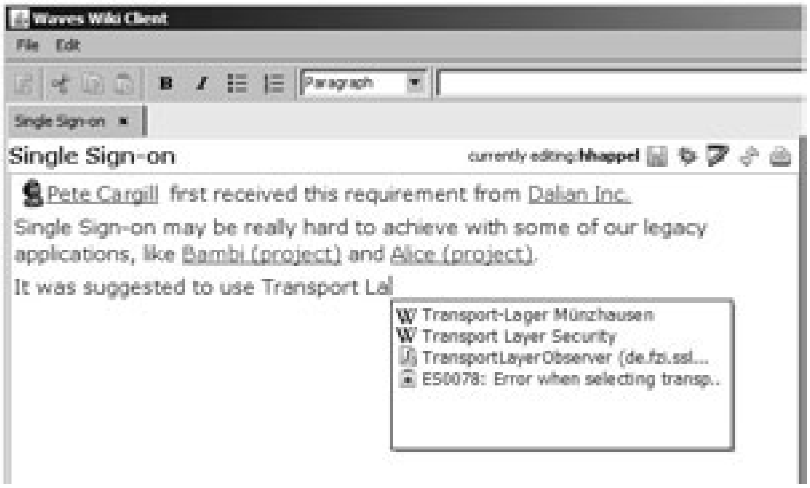
4.1 The integrated virtual knowledge base



Waves' virtual knowledge base integrates the various existing information sources, as depicted in the diagram, through a flexible adapter interface. The sources then become accessible through the client interface of the knowledge base (so-called SWECR – semantic web content repository, [Vö07]). Authored knowledge (Wiki pages including ad-hoc form data) is stored directly in the knowledge base. The knowledge base is implemented by combining a semantic metadata store (currently Sesame) with a binary store (for content) and a full-text index (Lucene). Both replication and virtual access is supported: Replication is usually chosen for data which is highly relevant for search criteria, low volume, and stable. Virtual access is preferred for seldom accessed, high volume, and volatile data.

4.2 Rich Wiki client

Waves provides a Rich Wiki client based on Java Swing. This enables a user-friendly, productive WYSIWYG interface. The client automatically analyzes the currently entered text to suggest hyperlinks to existing resources and Wiki pages (autocomplete). The user can also explicitly ask for link suggestions in a given context and analyze text pasted into the Wiki page from another application.



Single Sign-on

Pete Cargill first received this requirement from Dalian Inc.

Single Sign-on may be really hard to achieve with some of our legacy applications, like Bambi (project) and Alice (project).

It was suggested to use Transport La

Transport-Lager Münzhausen
Transport Layer Security
TransportLayerObserver (de.fzi.ssl...
ES0078: Error when selecting transp..

Figure 1. Automatic link suggestion in the Rich Wiki client

4.3 Sharing across organizational spaces

Sharing across organizational spaces involves the physical distribution of content and metadata on the one hand, and their conceptual organization in different visibility spaces.

4.3.1 Physical distribution

The SWECR interface (client interface of the virtual knowledge base) implements concepts borrowed from distributed relational databases, and thereby allows chaining together virtual knowledge bases and stream query responses through them, to form a comprehensive virtual knowledge base as seen by the user. For writing (transactions), there is always one authoritative source. For reading, local server instances can replicate more remote instances to provide more throughput and reliability. A replica of any remote content that has been locally annotated is also kept for future reference, since the external resource may become unavailable. With this distribution model we hope to solve the Knowledge Management scenarios in our introduction: When an organization or individual joins a new project, they can explicitly define certain areas within their knowledge base which are allowed to be access and replicated by their partners.

4.3.2 Conceptual organization

Content and metadata is organized into discrete visibility spaces. Larger spaces are created by the administrators of a server instance. But there is also the possibility to create ad-hoc spaces for small groups by users themselves. When a piece of knowledge is published, it is actually transferred into the more visible space (which may involve a physical transfer to a different server instance). The transfer process then involves updating the incoming and outgoing references, and issuing warnings about any inconsistencies arising from this process.

5 Evaluation plans

The project takes an iterative approach, with major releases about every 6 months which are evaluated by 4 industry partners. The first release concentrated on the integration of the most important information sources (about a dozen applications in total, including several issue trackers and "legacy" Wikis, as well as file servers, code repositories etc.), and their joint querying. Evaluation of the second release, which includes the Wiki, is currently underway.

6 Related work

One of the earliest Wiki engines addressing specific software engineering activities and structured information types was SnipSnap, developed at Fraunhofer FIRST.[1] For example, SnipSnap allows collaborative editing of UML diagrams within a Wiki page using special syntax. The RISE project and the subsequent SOP project at Fraunhofer IESE have elicited the potential of combining Wikis and ontologies for the purposes of distributed requirements engineering ([DRR05], [DRR07]): Using the same Wiki principles, users can not only collect requirements, but also redefine the concepts and relationships used in the requirements model. This flexible and powerful model is especially appropriate when diverse stakeholders are involved. The SOP project is implemented on top of the fairly popular Semantic Mediawiki extension ([KVV06]). Structured information is freely mixed with free-form text.

[1] http://snipsnap.org

The use of collaborative ontologies for requirements engineering is also the focus of the OntoWiki ([ADR06]) and SoftWiki ([AF06]) projects at the University of Leipzig, with a stronger accent on structured information types and browsing and editing of RDF data. Similarly to the Waves project, SoftWiki tries to address usability and productivity issues with an intuitive interface, and provides Social Networking functions such as ratings and comments.

The Waves project is conceptually compatible and complementary with these efforts, as it shares the vision of a semantic collaborative knowledge base and its access through Wiki principles. However, it focuses more on the integration of existing information sources into this knowledge base through a service-oriented architecture, and therefore its Wiki serves as a complement rather than an alternative to commercial, structured requirements engineering tools and ALM suites. It also takes a more radical, rich client approach with respect to usability and uniquely addresses challenges of knowledge sharing across organizational spaces.

7 Conclusion

The proposed technology enables the scenarios of integrating various existing information sources, and of sharing knowledge across organizational boundaries. At the current implementation stage, it is possible to search through a great variety of relevant sources in Software Engineering, and author knowledge using the described Rich Wiki Client. The distribution architecture will be implemented in future releases.

References

[ADR06] Auer, S.; Dietzold, S.; Riechert, T.: OntoWiki – a tool for social, semantic collaboration. In Cruz et al. (Eds.): ISWC 2006. LNCS 4273 pp. 935--942. Springer

[AF06] Auer, S.; Fähnrich, K.-P.: SoftWiki – Agiles Requirements-Engineering für Softwareprojekte mit einer großen Anzahl verteilter Stakeholder. Statuskonferenz Forschungsoffensive „Software Engineering 2006“, 26.-28. Juni 2006, Leipzig. http://www.informatik.uni-leipzig.de/~auer/publication/softwiki.pdf (7.2.2008). Project web page: http://softwiki.de

[DRR05] Decker, B.; Ras, E.; Rech, J.; 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), ISWC 2005

[DRR07] Decker, B.; Ras, E.; Rech, J.; Jaubert, P.; Rieth, M.: Wiki-based stakeholder participation in requirements engineering. IEEE Software March/April 2007. SOP project web page: http://www.sop-world.org

[KVV06] Krötzsch, M.; Vrandecic, D.; Völkel, M.: Semantic MediaWiki. In Cruz et al. (Eds.): ISWC 2006. LNCS 4273 pp. 935--942. Springer. http://korrekt.org/papers/KroetzschVrandecicVoelkel_ISWC2006.pdf (7.2.2008). Project web page: http://semantic-mediawiki.org

[Vö07] Völkel, M.: A semantic web content model and repository, Proceedings of the 3rd International Conference on Semantic Technologies, SEP 2007.