Self-organized Reuse of Software Engineering Knowledge Supported by Semantic Wikis

Björn Decker[1], Eric Ras[1], Jörg Rech[1], Bertin Klein[2], Christian Hoecht[3]

[1] Fraunhofer Institute for Experimental Software Engineering (IESE)
67663 Kaiserslautern, Germany
{bjoern.decker, eric.ras, joerg.rech}@iese.fraunhofer.de
[2] German Research Institute for Artificial Intelligence (DFKI)
67663 Kaiserslautern, Germany
{bertin.klein }@dfki.de
[3] Chair of Pedagogics, Technical University of Kaiserslautern
67663 Kaiserslautern, Germany
{hoecht }@ rhrk.uni-kl.de

Abstract. Self-organized reuse of artifacts from software and system development, using the lightweight Wiki-Technology, promises a sustainable preservation and availability of business-critical information. However, due to the organic, sometimes chaotic growth of content inside a Wiki, additional support for structuring the knowledge and finding interrelated useful content is needed. The enhancement of Wiki content with ontologies – named semantic Wikis - can solve these problems. The application of such semantic Wikis and the development of reasoning mechanisms for software engineering is subject of the project RISE (Reuse in Software Engineering).

1 Introduction

The successful, methodic reuse of software artifacts was first motivated by Dough McIllroy on the 1968 NATO conference of Software Engineering. Despite this long tradition, systematic reuse is still facing several challenges. These challenges are caused by insufficient support for the reuse steps [2] search, evaluation, and adaptation. Concerning *search*, people do not find existing artifacts or do not even start to search due to the effort related with it. *Evaluation* challenges are mostly caused by either lengthy or insufficient documentation of the artifact found. This leads often to bad understanding and thus difficulties in evaluating the retrieved artifacts. Finally, *adaptation* refers to the (perceived) effort to understand and adapt the artifact in contrast to the effort to its first creation. Key to support all three steps is the identification of potential reuse candidates according to their similarity of the artifact to be created [26]. Determining this similarity is a complex task, since multiple criteria (e.g., project characteristics, technologies, desired non-functional requirements) need to be taken into account.

Reuse requires that the reused artifacts are "fit for reuse", i.e., that the developed artifacts are represented in an understandable way and that they are useful to other people who did not create these artifacts. Therefore, reuse implies an intra-organizational, cross-project coordination of development activities. Most of the reuse approaches ([1], [2]) address this issue by suggesting certain dedicated organizational structures. In order to reduce the effort for such dedicated organizational structures, self organization of the community could be done by lightweight communication platforms and the usage of agile development approaches. *Self-organized reuse* means that the community does not only provide the artifacts to be reused, but also cares about how to organize them. However, this self-organized reuse implies to distribute the effort for the "development of artifacts *for* reuse" within the community. Otherwise, it would just replicate the dedicated structures to the communication platforms. Therefore, to support self-organized reuse, the effort for annotating artifacts needs to be done as a work that is directly beneficial to the developing projects and their creators.

This paper addresses self-organized reuse in the domain of software development and presents an approach where artifacts are captured and reused by means of semantic Wikis. The objectives of this paper are:

- to show the necessity and usefulness of semantics in Wikis in general,
- to introduce a new paradigm of *Wikitology*, i.e., where the Wiki itself act as the ontology,
- and to present an implementation of a Wikitology in the domain of software engineering.

Before semantic Wikis are introduced and motivated for self-organized reuse of software artifacts, the next section elaborates shortly the advantages of Wikis and provides examples of Wikis in software engineering. Section 3 describes the lack of semantics in Wikis and motivates how ontologies could help to address this problem. Section 4 introduces the new paradigm of Wikitology and provides a concrete application example for gathering and exchanging artifacts during the requirement phase of software development.

2 Wikis for Software Engineering

In order to realize the self-organized reuse in software engineering – in particular when performed under the agile software development paradigm [14] – Wikis [4] can be seen as a lightweight platform for exchanging reusable artifacts between and within software projects. From our perspective a Wiki system can be considered as lightweight *Organizational Memory* [7] or *Experience Factory* [11]. Wikis facilitate the communication by a basic set of features and delegate the actual way of coordination to the people who are using the Wiki. From the authors' point of view, these basic features are: *one place publishing*, meaning that there is only one version of a document available that is regarded as the current version; *simple and safe collaboration* refers to versioning and locking mechanisms that most Wikis provide; *easy linking* means that documents within a Wiki can be linked by their title using a

simple markup; *description on demand* means that links can be defined to pages that are not created yet, but might be filled with content in the future.

In summary, using a Wiki is like working in an "open source" knowledge repository, where content can be edited collectively using a web browser. In particular, this allows us to set up software development infrastructures that have a low technical barrier for usage.

Wikis were initially used in a software engineering setting, namely the portland pattern repository [4]. Furthermore, they are often used to support software development, in particular in the area of Open Source Software. The Wikis of the Apache Foundation [27] are a prominent example of this application scenario. Some examples of Wikis offer specific functionality for software engineering:

- *Trac* [16] is a Wiki written in python that integrates an issue tracker, allowing relate Wikis pages to issues and vice versa. Furthermore, the python code of a project can be integrated as read-only documents.
- *MASE* [17][18] is an extension to the JSP Wiki that offers plug-ins for agile software development, in particular for iteration planning and integration of automated measurement results.
- *SnipSnap* [21][41] is implemented in Java and allows a read only integration of code documentation. Furthermore, it offers support for the integration of Wiki entries into the integrated development environment eclipse.
- *Subwiki* [28] is a Wiki implementation that uses the versioning system subversion as a data repository. Since subversion allows to attach metadata to files, the resulting Wiki is supposed to have the same features. However, this project has not released a stable version yet.
- *WikiDoc* [24] is a conceptual work that supports to add java code documentation via a Wiki interface. This allows non-programmers to participate in the creation of code documentation.

All those examples show that Wikis are increasingly be used as a platform for software development. However, "regular" Wikis (see [19] for an overview of most of the Wikis currently available) as well as software engineering-oriented Wikis build upon the fact that the relation between documents and further metadata are maintained by the users of those Wikis. This lack of explicit semantic information is addressed by an extension to the regular Wiki functionality that is developed in the RISE project. These so-called *semantic Wikis* are elaborated in the next section.

3 Semantic Wikis

One way to organize the organic growth of Wiki content is to add structure by enriching Wiki-pages with additional metadata. But current standard Wikis are lacking machine-understandable semantics. By this lack of semantics, structuring the Wiki content by machine reasoning is very difficult. Furthermore, when a common semantic is missing, the sensible integration of similar content from other Wikis needs human intervention– which is often not done due to time constraints. However, the

emergent standards in the area of ontologies (like RDF and OWL) provide the techniques to express the semantics in a machine-understandable format.

First, for the practitioner, it is often enough to conceive the proposed "semantic approach" as: extend your Wiki with RDF (i.e., features that support to enter metadata according to the RDF standard). Second, to provide a brief idea of the term semantics in the current context, one can say that it is about the different meaning of a word or statements to one person, to another person, or to a computer program. Only humans are able to read and understand the texts contained in the Wiki encyclopedia – for machines, without sophisticated processing the only thing visible of the knowledge contained within the Wiki is a large number of text pages which link to each other. Third, as a rule of thumb, an "absence of semantics" is close to "an absence of structure", and typically leads to what people call information overload when they are searching for specific information due to the lack of filtering based on these semantics.

The next section motivates ontologies for enhancing Wikis with semantics and illustrates the usefulness of the semantic dimension of the pyramid of ontology use. Examples of semantic Wikis are provided.

3.1 Usage of Ontologies

From the viewpoint of Artificial Intelligence, knowledge engineering, ontologies, and semantic web technology are very closely related, if not identical. From the viewpoint of software engineering, the corresponding modeling approaches are also closely related. All aim to capture the meaningful aspects of real world aspects and express it with a varying degree of formality. Recently a survey of ontology use and its further potential has been published [25]. In particular, they present a framework to characterize and structure the field of ontology applicability. From this framework, we adopt the semantic dimension (what the semantics stored with ontologies can be used for). As depicted below, the three levels of the semantic framework are: Communication, Integration, and Reasoning.

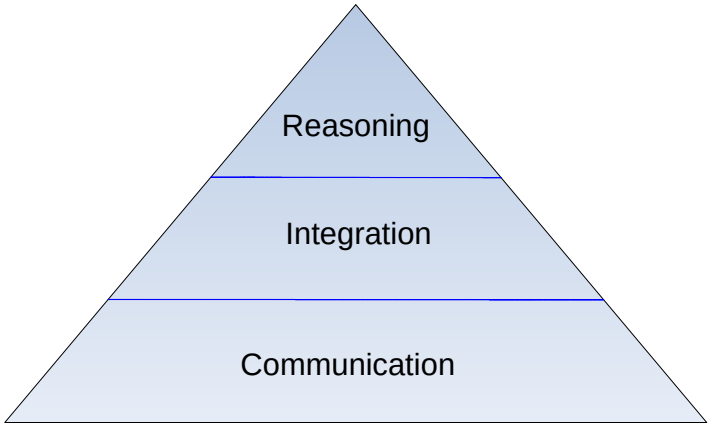

Figure 1: Pyramid of Ontology Use (Semantic Dimension) According to [25]

Communication: Since ontologies represent an explicit statement of relevant concepts in a certain domain, they allow to discuss those concepts and in order to reach a shared understanding of these concepts. Furthermore, by referring to concepts in an ontology, used concepts do not need to be explained during regular communication and ambiguities are reduced.

Integration: The next level in the semantic framework is to relate the concepts to each other. These relations allow to navigate and to search through the domain. This improves understanding.

Reasoning: According to the degree of formalization, inference about concepts and relations can be performed. Inference builds the most advanced application for ontologies. As denoted by the name of this level, this use of ontologies allows to reason why concepts are related and what the implications of those relations are.

For analyzing ontologies, this framework offers two opportunities. First, it acts as a maturity model for the actual use of an ontology or parts of it. Second, it can be used as an analysis tool for determining the potential of an ontology.

Furthermore, the maturity model implied by the framework provides a strategy to build up an ontology. First, the understanding of the concepts within a community needs to be clarified. Second, the interrelations of these concepts need to be discussed. Finally, reasoning about the ontology supports the coordination of the community. These three steps do not need to be performed in sequence: Extensions to the ontology that are demanded by the community could be added iteratively.

3.2 Why using a semantic Wiki?

Annotating the Wiki content with additional information can be beneficial to support reuse across projects. The shared ontology allows locating relevant information from further projects. However, providing this information requires additional effort from the content's creator. In most cases, adding this additional information provides no additional benefits to the creator. In addition, the creator carries the additional burden to classify the content. Therefore, it is naive way to motivate this additional effort by that this information can be found easier, and thus, that the contribution of the creator is visible to a broader audience. While SE Wikis should also support this meritocratic approach, the additional ontology should provide a direct benefit to the creator. Within RISE, two main direct benefits are identified to motivate the addition of ontological information to Wiki content: First, by providing an ontological classification, further relevant Wiki content an be identified that can support the work of the creator. Second, automatic reasoning support the creator to find inconsistencies regarding the ontology.

Since the benefits are "just" an application of reasoning mechanism to the ontological information, they are crucial to motivate users to actually provide this information. An example of how semantic Wikis could be applied to support self-organized reuse is presented in the following section.

3.3 Examples of Semantic Wikis

In this section, we present further examples of semantic Wikis. Most of these examples are taken from the overviews presented in [33] and [34]. Those examples show that a) even "regular" Wikis offer some support for structuring their content and b) that semantic, RDF-based Wikis can be implemented. Most of those examples are general purpose Wikis that do not focus on Software Engineering in particular.

- The most common way to categorize within Wikis is the usage of the backlink function of a Wiki [32]. Basically, a page is created that represents a certain category. Pages belonging to this category have a reference to this page. The backlink function lists all of these references. However, this approach has a major disadvantage: Pages that are used to navigate to the category entry (and thus are not semantically belonging to this category) are also included in the backlink list.
- Some Wikis offer additional support for structuring content. For example *TikiWiki* [35] allows assigning pages to structures (table of contents) and categories (taxonomy style classification). XWiki [36] offers forms (templates) that contain metadata, which are instantiated in documents derived from this form.
- From the area of SE-specific Wikis, *TRAC* [16] offers a labeling feature for pages (smart tags) that could be used for a facetted presentation of the pages annotated with those tags. *SnipSnap* allows to determine the template of a document and offers RDF export.
- *Platypus* [15], *SHAWN* [32], and *Wiki.Ont* [20] allow to add RDF annotations for each page. Pages within Platypus represent a concept. While viewing a page, the RDF-triples are displayed that have the current page as object (in particular pages that reference this page) and as subject (in particular, metadata about a page). SHAWN also offers navigation support based on the ontology information added to a page. Wiki.Ont is still in preliminary version.
- *Rhizome* [37] and *RDF-Wiki* [38] are Wikis that provide their content in RDF, thus allowing to reason about their context.

Only the Wikis mentioned under the last two bullets can be seen as "real" semantic Wikis, since they allow to adapt their content to a RDF ontology. However, all of them – at least in their current state – do not integrate their ontology into the Wiki, e.g., they neither provide metadata-templates to be filled in based on an ontology nor they check whether metadata entered is consistent with an ontology. Therefore, when related to the semantic web layer cake [31], all of these semantic Wikis implement the RDF and RDFS Layer. The vocabulary layer of these applications is not domain specific, and thus does not allow to infer about domain specific relations.

4 Application Example of a Wikitology in Software Engineering

Ontologies are used to improve work with corpora of information. As the definition of ontologies in computer science "an ontology is a specification of a conceptualization" is very general, we use the term as describing the combination of concepts, instances

and relationships (between concepts). More precisely, with a focus on Riki - the Wiki variant created in the RISE project - we label this as: an *ontology* is the frame to ”package“, to ”label“ and to “structure” the pages’ content. Throughout the last decade, the notion of ontology has gained very practical importance for systems that are supposed to host knowledge [7][8]. Other areas refer to such instrument sometimes individually different, e.g., as a metadata schema in the Digital Library or the e-learning areas [9]. Typically, ontologies are used for collections of content to induce a structure over them. Such a content structure is a means for humans or intelligent software, to retrieve wanted information out of the content, e.g., by mediating problems to solutions.

4.1 The new Paradigm of Wikitology

The acquisition of the right ontology for a set of content is a current scientific issue [8]. Further, due to the ontology being the skeleton for the knowledge assets that reside in the system to be retrieved, the ontology is a major crystallization point in the process of continually adapting the system, i.e., “its lifecycle” [10]. This leads to the following new idea: the concepts that are important during daily work of the users, belong to dedicated pages, and their meaningful relations will be links in the Riki.

With this new paradigm, *Wikitology* [39][40], it is possible to smartly and semi-automatically derive an ontology needed for information retrieval from the pages in the Riki itself. Such a Wikitology automatically updates “itself”, i.e., it reflects the changes of contained knowledge, changed views of the users accounts, new projects, customers, rules, trends, and ideas inside the Wiki. By considering the Riki as the ontology, the ontology will be always collectively edited up-to-date and will automatically reflect all changes.

The practical implementation of this Wikitology is as follows: The ontology used by the community is edited via the Wiki itself. A set of naming conventions is used to determine automatically the actual ontology from the Wiki content. For example, document templates may start with “DocType”. This naming convention allows a user to add new document templates just by creating a new document starting with DocType (an alternative to this convention would be to use namespaces). Since those templates contain a reference to themselves, an instance of this template is automatically linked to the template. In addition, domain-ontology independent information like the document type or the date of creation is derived from the Wiki and included into the ontology. These conventions and automatic determination of meta-data can be seen as an easily remindable, implicit meta-ontology.

Therefore, the main use of ontologies in RIKI is to support reuse. However, the ontology itself is subject to reuse as a shared understanding of the concepts relevant to their using community.

One implementation of RIKI is based on the Sekt Integration Platform (SIP). This platform implements an RDF-based access to the ontology and supports the integration of further reasoning tools. For the actual determination of similarity to support the self-organized reuse, we use two techniques: Latent Semantic Analysis [29] and Case Based Reasoning [30].

4.2 Application Example for Requirement Engineering

This section presents an application example of new paradigm presented above in scope of the RISE project. The core of the software engineering documents considered in this example are adapted from the *use case* approach of Cockburn [22] and the *ready set* template [23]. These use cases represent a textual representation of the behavior of the system to be developed (these documents should not be confused with the graphical oriented approach suggested by the UML). We augmented those use cases with additional documents that capture further software engineering knowledge. In Fig. 2, the document ontology (i.e., the document types and their relationships) is presented. Besides this ontology of the documents, each document possesses a set of metadata. A more detailed presentation of the metadata set would be out of scope of this paper.

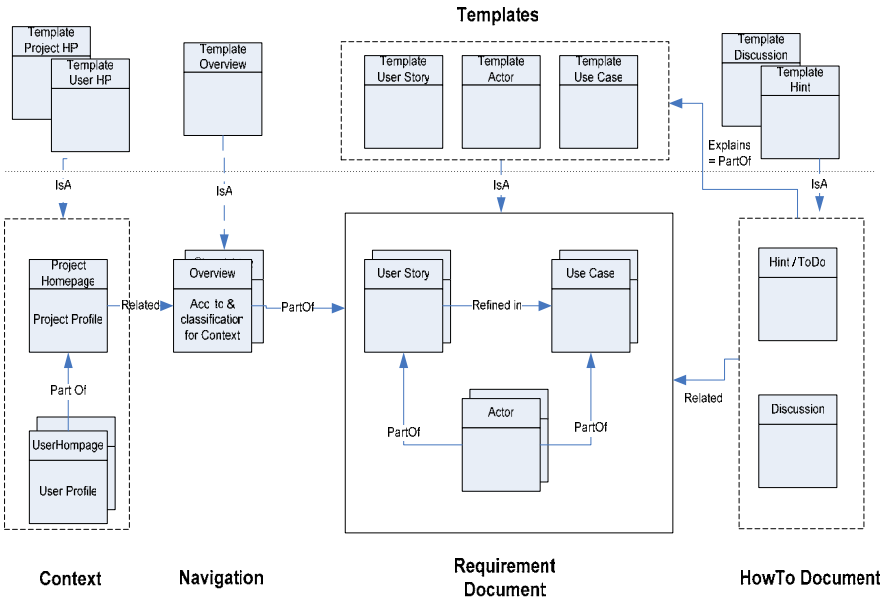

Figure 2: Example of a Document Ontology

To facilitate the representation, we focus on five different types of documents: 1) The main part of this ontology refers to *requirement documents*: instances of *Actor* contain descriptions of persons or entities interacting with the system like users or other system components, instances of *User Story* contain narrative descriptions of interactions between system and Actor, and instances of *Use Case* contain semi-formal representations of these interactions; 2) *Templates* define the structure, content, and allowed interrelations of these documents; 3) *How To* documents give information on how to create and use requirement documents. By relating them to a certain template, instances of How To documents could be created based on the related template; 4) support for structuring the whole set of documents is provided by *Navigation* documents, i.e., overviews that group the requirements documents

according to the document ontology or the metadata contained in the document; 5) Finally, *Context* documents contain information about the project or the authors of documents (like personal skills).

This ontology can be extended according to the needs of the project or the policies of the whole organization. An example of such an extension could be a document template describing certain states of the system (e.g., user logged in) that are linked by different Use Cases.

In the remainder of the section, we describe how this ontology is used in different scenarios to support the management and reuse of software engineering documents.

Offering relevant documents: Since the allowed links between different document types are defined in the templates, suggestions for links to instances of those related document types can be offered. For example, while creating a Use Case, the titles that are instances of the template Actor are presented together with a summary of their content. A user can directly integrate these titles (and thus, the link to this document) in the current Use Case.

Consistency checks: The semantic annotation allows also to define and execute consistency checks. For example, the part-of relationships imply that each Actor should be part in at least one Use Case. The "refined-in" relation defines a temporal order. Therefore, if the User Story has a more recent date than the refined Use Cases, these refined Use Cases need to be checked at least whether they are consistent with the updated User Story.

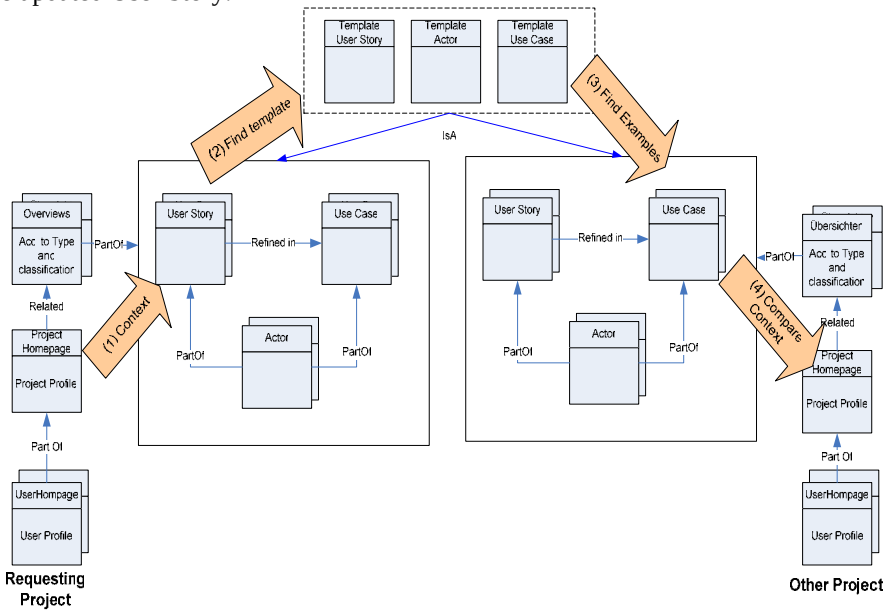

Figure 3: Application Example: Cross-project Offering of Similar Documents

Cross-project offering of similar documents: Fig. 3 illustrates how a user is creating a new user story. Via (indirect) linking to the project and user's homepage, context information can be derived (e.g., the expertise level of the user and type of the system to be developed). Furthermore, by linking to the document template, the type

of the document can be derived. This allows to find documents of the same type created within other projects. However, the set of those potential candidates can be rather large. By comparing the context information derived in the first step, this initial set can be restricted to documents from similar context, which offer a probable higher reuse potential. In RISE, we use Case-based reasoning for the retrieval of similar documents.

5 Summary and Outlook

This paper has presented an approach to support self-organized reuse by using augmented semantic Wikis. Augmenting Wikis with domain-specific ontologies (e.g., for requirements) provides a means to manage the organic growth implied by the Wiki approach. The introduction of a new paradigm called Wikitology enable us to embed semantics into the Wiki itself. This way of adding semantics supports the communication of used concepts within the Wiki and by performing reasoning, inconsistencies within pages and amongst different pages of the same document types can be found. Besides the empirical evaluation of the project results, the next steps of the RISE project focus on integrating engineering information beyond the documents captured in the Wiki. In particular, code developed by the projects should be integrated into the Wiki to allow traceability of requirements and support reuse also on code level. In the long run, the ontological information represented in a semantic Wiki could be used for model driven software development.

Acknowledgements

Our work is part of the project RISE (Reuse in Software Engineering), funded by the German Ministry of education and science (BMBF) grant number 01ISC13D.

Furthermore, we would like to thank our project partners Volker Haas from Brainbot and Ralf Traphöner from empolis (the “inventor” of the concept of Wikitology).

References

1. Mili, H.; Mili, A.; Yacoub, S.; Addy, E.: *Reuse-Based Software Engineering – Techniques, Organization and Measurement*, John Wiley and Sons (1995)
2. Karlsson, E.-A.(Ed.): *Software Reuse – a holistic approach*, John Wiley and Sons (1995)
3. Zand, M.; Arango, G.; Davis, M.; Johnson, R.; Poulin, J.;Watson, A.: *Reuse R&D: Is it on the right Track*. In: Proceedings of the 1997 Symposium on Software Reusability, Boston, USA, Mai 1997, S. 212-216
4. Leuf, B.; Cunningham, W.: *The Wiki Way*, Addison-Wesley Professional, (2001)
5. Arnold, P.; Smith, J.: *Adding connectivity and losing context with ICT: contrasting learning situations from a community of practice perspective*. In: Huysman, M.; Wenger, E.; Wulf, V.

(Eds.): Proceedings of the International Conference on Communities and Technologies (C&T 2003), Kluwer, Dordrecht 2003. (2003)
6. Clark, H. H.; Brennan, S. E.: *Grounding in communication.* In : Resnick, L.; Levine, J. M.; Teasly, S. D. (Eds.): Perspectives on socially shared cognition. Washington D.C.: American Psychological Association, P. 127-149. (1991)
7. Abecker, A..; Bernardi, A. ; Hinkelmann, K.; Kühn, O.; Sintek, M.: *Towards a Technology for Organizational Memories.* IEEE Intelligent Systems, 13(3), May/June 1998. (1998)
8. Maeche, A. Staab, S.: *Ontology Learning for the Semantic Web*. IEEE Intelligent Systems, 13, (2001)
9. Apostolou, D., Georgolios, P.-P., Klein, B., Franz, J., Maass, W., Abecker, A., Kafentzis, K., Mentzas, G.: *Towards provision of knowledge-intensive products and services over the Web***.** In*:* 22nd IASTED International Multi-Conference on Applied Informatics, February 2004, Innsbruck, Austria.
10. Klein, B., Traphöner, R..: *A Practical Application of Ontologies for Knowledge Sharing and Trading*, In: FGWM 2004 - Workshop on Knowledge and Experience Management, October 2004.
11. Basili, V. R.,; Caldiera, G.; Rombach, H. D., *Experience Factory* in Encyclopedia of Software Engineering, vol. 1, J. J. Marciniak, Ed. New York: John Wiley & Sons, 2001, pp. 511-518.
12. Ras E., Weibelzahl S.,: *Embedding Experiences in Micro-didactical Arrangements*, in *Proc. of 6th International Workshop on Advances in Learning Software Organisations (LSO 2004),* Springer LNCS, Banff, Canada, 2004, pp. 55-66
13. Wenger, E.: *Communities of practice. Learning, meaning, and identity*. Cambridge University Press.
14. *Agile Manifesto*: www.agilemanifesto.org, last visited 29.9.05
15. *Platypus Wiki Homepage*: http://platypuswiki.sourceforge.net/, last visited 29.9.05
16. *Trac Wiki*: http://www.edgewall.com/trac/, last visited 29.9.05
17. *MASE Homepage*: http://ebe.cpsc.ucalgary.ca/ebe/Wiki.jsp?page=Root.DeployMASEOnProduction, , last visited 29.9.05
18. Maurer, F.: *Supporting Distributed Extreme Programming*, XP Agile Universe 2002 http://ebe.cpsc.ucalgary.ca/ebe/attach?page=Root.PublicationList%2FMaurer2002.pdf, , last visited 29.9.05
19. *List of Wiki Clones*: c2.com/w4/wikibase/?LongListOfWikiClones, last visited 17.5.2005
20. *WikiOnt*: http://boole.cs.iastate.edu:9090/wikiont/, last visited 17.5.2005
21. *SnipSnap*: snipSnap.org, last visited 17.5.2005
22. Cockburn, A.: *Writing effective Use Cases*, Addison Wesley Professional (2000)
23. *Readyset Requirements Engineering Template*: readyset.tigris.org,, last visited 29.9.05
24. Oezbek, C.:*WikiDoc*, http://www.inf.fu-berlin.de/~oezbek/, last visited 29.9.05
25. Mika, P. Akkermans, H.. *Towards a new synthesis of ontology technology and knowledge management*. Technical Report IR-BI-001, Free University of Amsterdam, March 2004.
26. Basili V. R., and Rombach H. D., *Support for comprehensive reuse.* Journral of Software Engineering, Vol 6, Issue 5, (1991)
27. *Apache Foundation Wiki*: http://wiki.apache.org/general/, last visited 29.9.05
28. *Subwiki Homepage:* http://subwiki.tigris.org, last visited 29.9.05
29. *Latent Semantic Analysis*: http://en.wikipedia.org/w/index.php?title=Latent_semantic_analysis&oldid=24255335, last visited 29.9.05
30. Aamodt, A. Plaza, E..: *CBR Foundational Issues, Methodological, Variations and System Approaches*. AI Communications Vol. 7, No. 1, 39-59 (1994)
31. Berners-Lee, Tim, *Semantic Web Layer Cake*, Semantic Web - XML2000, http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html, last visited 29.9.05
32. Aumüller, D.: *SHAWN: Structure Helps A Wiki Navigate*, http://the.navigable.info/2005/aumueller05shawn.pdf, last visited 29.9.05

33. Dahl, I.; Eisenbach M.: *Anwendung: Semantic Wikis*, Seminal Thesis AIFB Karlsruhe, http://www.aifb.uni-karlsruhe.de/Lehre/Sommer2005/SemTech/stuff/semwiki.pdf, last visited 29.9.05
34. *Semantic Wiki Overview*: http://c2.com/cgi/wiki?SemanticWikiWikiWeb, last visited 29.9.05
35. *Tiki Wiki Homepage*: http://tikiwiki.org/, last visited 29.9.05
36. *XWiki Homepage*: http://www.xwiki.org/xwiki/bin/view/Main/WebHome, last visited 29.9.05
37. *Rhizome Homepage*: http://rx4rdf.liminalzone.org/, last visited 29.9.05
38. *RDF Wiki*: http://infomesh.net/2001/05/sw/#rdfwiki, last visited 29..05
39. Decker, B.; Ras, E.; Rech, J.; Klein, B.; Reuschling, C.; Höcht, C.; Kilian, L.; Traphöner, R.; Haas, V.: *A Framework for Agile Reuse in Software Engineering using Wiki Technology*, Workshop on Knowledge Management for distributes agile Processes KMDAP, WM2005. Conference Professional Knowledge Management - Experiences and Visions. Proceedings, p 411-414 (2005)
40. Klein, B.; Höcht, C.; Decker, B.: B*eyond Capturing and Maintaining Software Engineering Knowledge - "Wikitology" as Shared Semantics*, Workshop on Knowledge Engineering and Software Engineering, at conference of Artificial Intelligence 2005, Koblenz (2005)
41. John, M. *Linking the customer's needs to developer tasks*, in Knowledge-based Software Engineering, vol. 108, Frontiers in Artificial Intelligence and Applications, V. S. a. K. Kaijiri, Ed., 1st Edition ed: IOS Press, 2004, pp. 49-58. (2004)