KNOWBENCH
A Semantic User Interface for Managing Knowledge in Software Development

Dimitris Panagiotou and Gregoris Mentzas
School of Electrical and Computer Engineering, National Technical University of Athens
9 Iroon Polytechniou Street, Athens, Greece

Keywords: Semantic User Interface, Software Development, Knowledge Workbench, Knowledge Management in Software Development, KnowBench.

Abstract: Modern software development consists of typical knowledge intensive tasks, in the sense that it requires that software developers create and share new knowledge during their daily work. In this paper we propose KnowBench a knowledge management system integrated inside Eclipse IDE that supports developers during the software development process to produce better quality software. The goal of KnowBench is to support the whole knowledge management process when developers design and implement software by supporting identification, acquisition, development, distribution, preservation, and use of knowledge – the building blocks of a knowledge management system.

1 INTRODUCTION

Modern software development consists of typical knowledge intensive tasks, in the sense that it requires that software developers create and share new knowledge during their daily work. Since software development is a highly collaborative task, developers are in need of simple and easy-to-use tools that also enable collaborative work.

Knowledge has to be collected, organized, stored, and easily retrieved when it needs to be applied. Probst (1997) has well described the building blocks of KM systems: identification, acquisition, development, distribution, preservation, and use of knowledge. Many knowledge problems occur because organizations neglect one or more of these building blocks.

In this paper we propose KnowBench (knowledge workbench) which strives to provide an intelligent, semantic user interface environment – a knowledge management system integrated inside a widely used software development environment namely Eclipse IDE (http://www.eclipse.org/) that supports developers during the software development process to produce better quality software. The goal of KnowBench is to collect knowledge the developers gain during software development in order to avoid mistakes and leverage successes in future projects.

KnowBench mainly supports software developers in resolving error handling and component reuse problems by providing an ontology-based knowledge management system that is build upon the KM systems' building blocks.

The rest of the paper is organized as follows. In the next section, we depict KnowBench's architecture and core components. Thereafter, we give the results of its evaluation which draws the guidelines for further optimization and improvements of the system. Finally, we conclude the paper.

2 KnowBench

The KnowBench system can be used to capture the knowledge and experience generated during the software development process. Although every software development project is unique in some sense, sharing similar experiences can assist developers to perform their activities in a better way. Moreover, reusing knowledge can prevent the repetition of past failures and guide the solution of recurrent problems.

KnowBench provides functionality that can be used for articulating and visualizing formal descriptions of software development related knowledge in a flexible and lightweight manner.

Panagiotou D. and Mentzas G. (2010).
KNOWBENCH - A Semantic User Interface for Managing Knowledge in Software Development.
In *Proceedings of the 5th International Conference on Software and Data Technologies*, pages 508-511
DOI: 10.5220/0003006105080511

This knowledge is then retrieved and used in a productive manner by a semantic search engine and a P2P metadata infrastructure – namely GridVine (Mauroux, 2007). Thus, the collaboration of dispersed software developers is achieved who can benefit from each others' knowledge about specific problems or the way to use specific source code while developing software systems.

2.1 Ontologies

We have deployed software development ontologies (Georgousopoulos, 2007) inside the KnowBench system in order to describe and capture knowledge related to software artefacts. The system architecture allows for the extension or the use of different software development ontologies. The set of the deployed ontologies includes three main ontologies (artefact, problem/solution and annotation) which interact among each other and are all under a common parent ontology (KnowBench ontology) which binds them.

The artefact ontology describes different types of knowledge artefacts such as the structure of the project source code, reusable components, software documentation, knowledge already existing in some tools, etc. The problem/solution ontology models the problems occurring during the software development process as well as how these problems can be resolved. The annotation ontology describes general software development terminology as well as domain specific knowledge. This ontology provides a unified vocabulary that ensures unambiguous communication within a heterogeneous community. This vocabulary can be used for the annotation of the artefacts.

2.2 Architecture

Figure 1 depicts the overall KnowBench architecture. The core components delivering KnowBench's functionality are: semantic search, global metadata store, software development semantic wiki (DevWiki) and (manual/semi-automatic) semantic annotation of source code. In the rest of this section we give more details about their actual interaction.

The above components are based on these APIs: semantic annotation API, shallow NLP and IE (information extraction) API (wrapping up and customizing Text2Onto (Cimiano, 2004)), the semantic search API and the global metadata store.

The global metadata store API provides an abstract layer on top of the local and P2P repositories through customized APIs (Jena and GridVine-based APIs (McBride, 2002, Mauroux, 2007).

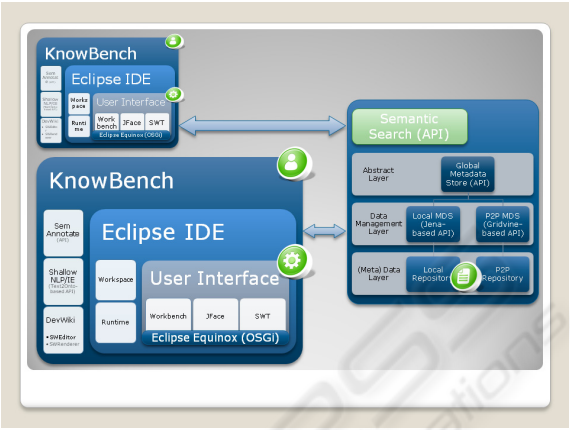

Figure 1: KnowBench architecture.

2.3 Core Components

Below we outline the core components which constitute the KnowBench system.

2.3.1 Semantic Search

KnowBench supports advanced methods for knowledge search through a semantic search engine (Giesbrecht, 2008) by taking into account three different types of search, namely keyword, structured and semantic search.

2.3.2 Global Metadata Store

The global metadata store consists of two components (APIs) – the LocalMDS and P2PMDS and two managers – KeyStore Manager and Policy Manager. It provides an abstract layer for handling these and its purpose is to manage knowledge stored either locally or in the P2P network.

The P2PMDS is based on the GridVine/P-Grid system (Aberer, 2005) and is customized for KnowBench in order to realize an access control aware P2P metadata store using the services of KeyStore and Policy managers.

2.3.3 Software Development Semantic Wiki (DevWiki)

KnowBench utilizes the DevWiki system (Panagiotou, 2009) in order to assist software developers in the articulation and navigation of software development related knowledge. DevWiki uses a lightweight and flexible editor with auto-

completion and popup support. Browsing through knowledge is done like surfing through a conventional wiki using the semantic links between different knowledge artefacts. This browser is available inside the Eclipse IDE so that the software developer does not have to switch to another external browser.

2.3.4 Manual/Semi-automatic Semantic Annotation of Source Code

An important aspect of the KnowBench is the ability to annotate semantically source code (Panagiotou, 2008). We have extended the standard Eclipse JDT editor to add this possibility. The software developer is able to annotate source code with semantic annotation tags that are available or define new tags and extend the used annotation ontology.

The manual semantic annotation of source code provides granularity regarding the respective source code fragment to be annotated. This granularity level is restricted by the underlying Eclipse platform itself as the IJavaElement interface is exploited to map between source code fragments and metadata.

On the other hand, the shallow natural language processing (NLP) and information extraction (IE) API is used to semi-automatically annotate source code corpora. The API is built on top of the Text2Onto API and provides customizations suitable for source code (e.g. java keywords are ignored such as import, package for, while, etc.).

In order to achieve this goal in KnowBench we exploited ontology learning (Maedche, 2004) and information extraction techniques (Cunningham, 2002). Ontology learning is needed in order to extract ontology concepts from source code corpora that can be used for annotating it. On the other hand, information extraction is needed in order to instantiate the annotation ontology with individuals (like in manual semantic annotation). A framework exploiting both technologies is Text2Onto (Cimiano, 2004) which we adopted and extended in order to implement the desired functionality.

3 KM SUPPORT WITH KNOWBENCH

3.1 Evaluation According to the KM Building Blocks

We have conducted detailed evaluation of KnowBench in small groups of software developers in the following organizations: (1) Intrasoft International S.A. – 2 developers, (2) Linux Industrial Association – 2 developers, (3) TXT e-Solutions – 2 developers and (4) Thales Research & Technology – 3 developers. See (Samiotis, 2009) for a comprehensive report on the KnowBench evaluation.

We grouped the evaluation results according to the KM building blocks depicted in section 2. Please note that the knowledge identification block is not described below as the KnowBench ontologies serve this purpose. (Samiotis, 2009) provides details on the KnowBench support of the knowledge identification block.

3.2 Knowledge Acquisition

According to the respondents, KnowBench achieves a good score (73% are positive) for its support in acquiring existing knowledge. Regarding the supported types of knowledge sources, 85% of the respondents were satisfied with the support; 23% found that additional types of knowledge sources relevant for coding should be supported. As far as the system's response time, although 64% of the respondents found it quite fast – some further optimization of the system would be useful.

3.3 Knowledge Development

All respondents found the knowledge development support in KnowBench clear and easy to follow and agree that the system provides support at an adequate level (76%). The meaning of knowledge items is understandable for 86% of the respondents. Regarding annotations, their meaning and purpose were clear for 86% of the respondents. 79% of the respondents found the granularity level of source code that can be annotated sufficient. 92% of the respondents found that KnowBench provides friendly and easy-to-use forms for creating annotations and only 36% of them considered the manual annotation as effortful activity. On the other hand, in semi-automatic annotation, 42% of proposed annotations were chosen, thus 82% of the respondents were satisfied with the suggestions.

3.4 Knowledge Sharing

Knowledge sharing in KnowBench meets the expectation of 69% of respondents. Even though all aspects of knowledge sharing in KnowBench are above the threshold, only the level of details to be specified in order to share knowledge and the

usefulness of shared information received high marks (greater than 90%).

3.5 Knowledge Usage

The search functionality received an average score (57%). The respondents were satisfied with the quantity and quality of search results. As far as quantity of search results is concerned, 91% of the respondents found the number of results optimal. For 73% of the respondents the list of result did not contain any irrelevant result. As regards the quality of search results, 62% of the respondents confirmed that the search results satisfy their information needs more than average.

3.6 Knowledge Preservation

The lifecycle of the knowledge items i.e. creation, update, deletion in KnowBench seems to be supported well (expectation of 67% of the respondents). Modification of knowledge is not a time consuming function for 75% of the respondents and can be done very easily by 73% of the developers.

4 CONCLUSIONS

In this paper we presented the KnowBench system – an intelligent, semantic user interface environment for software developers which is integrated in the Eclipse IDE. Semantic web technologies provide the driving force to better manage knowledge in software development activities inside KnowBench. KnowBench offers an easy to use environment to facilitate knowledge articulation and visualization pertinent to software development. Additionally, it provides means to annotate manually or semi-automatically this kind of knowledge in order to foster easier knowledge acquisition and sharing by exploiting a semantic search engine and a P2P metadata infrastructure. Thus, better and more flexible collaboration among software developers scattered across the globe is facilitated.

ACKNOWLEDGEMENTS

This work was partly supported by the TEAM project, which is funded by the EU-IST program under grant FP6-35111. The authors are responsible for the content of this publication.

REFERENCES

Aberer, K. Datta, A. Hauswirth, M. Schmidt, R. 2005. Indexing data-oriented overlay networks, 31st *International Conference on Very Large Databases* (VLDB), Trondheim, 30 Aug - 2 Sep, 2.

Cimiano, P., Pivk, A., Schmidt, L.T., Staab, S., 2004. Learning taxonomic relations from heterogeneous sources, In Proc. *ECAI 2004 Ontology Learning and Population Workshop.*

Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V., 2002. GATE: A framework and graphical development environment for robust NLP tools and applications, *In Proc. 40th Annual Meeting of the ACL.*

Georgousopoulos C., Happel H-J., Khan O., Maalej W., Narendula R., Ntioudis S., Panagiotou D., Stojanovic L., 2007. *TEAM Project Deliverable D9*: User Requirements and Conceptual Architecture.

Giesbrecht, E., Stojanovic, L., Tran, T., 2008. TEAM Project Deliverable D29: Second iteration prototype of Semantic Search.

Maedche, A., Staab, S., 2004. Ontology learning, In Proc. Staab, S., Studer, R. (eds.), Handbook on Ontologies, pp. 173-189. *Springer.*

Mauroux, P.C., Agarwal, S., Aberer, K., 2007. GridVine: An Infrastructure for Peer Information Managment, *IEEE Internet Computing*, vol. 11, no. 5, pp. 36-44

McBride, B., 2002. Jena: A Semantic Web Toolkit*, IEEE Educational Activities Department*, Piscataway, NJ, USA, pp. 55-59.

Panagiotou, D., Mentzas, G., 2009. A Semantic Wiki for Software Development. In *Proc. 13th Panhellenic Conference on Informatics*, 10 - 12 September 2009, Corfu, Greece.

Panagiotou, D., Paraskevopoulos, F., Ntioudis, S., 2008. *TEAM Project Deliverable D31*: 2nd iteration prototype of Knowledge Desktop.

Probst, G.B., 1997. Practical knowledge management: a model that works, Prism, No. second quarter, pp.17-33.

Samiotis, K., Stojanovic, L., 2009. *TEAM Project Deliverable D39*: Summative Evaluation Report.

SETF, 2006. A Semantic Web Primer for Object-Oriented Software Developers, http://www.w3.org/TR/sw-oosd-primer/.