A Semantic Wiki for Software Development

Dimitris Panagiotou and Gregoris Mentzas

National Technical University of Athens, Athens, Greece,
dpana@mail.ntua.gr, gmentzas@mail.ntua.gr

Abstract
Modern software development is highly knowledge intensive; it requires that software developers create and share new knowledge during their daily work. However, current software development environments are “syntactic”, i.e. they do not facilitate understanding the semantics of software artifacts and hence cannot fully support the knowledge-driven activities of developers. In this paper we present DevWiki, a semantic wiki environment which focuses on the software development domain and strives to address these problems. DevWiki aims at providing software developers such a tool to ease their daily work and facilitate the articulation and visualization of software artifacts, concept-based source code documentation and related problem solving. Building a knowledge base with software artifacts by using the DevWiki system can then be exploited by semantic search engines or P2P metadata infrastructures in order to foster the dissemination of software development knowledge and facilitate cooperation among software developers.

Keywords: Semantic Wiki, Software Development

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. Although most software developers use modern state of the art tools, they still struggle with the use of technologies that are “syntactic”, i.e. they use tools that do not facilitate the understanding of the concepts of the software artifacts they are managing (e.g. source code).

Flexible ways of solving problems are necessary when a developer is frustrated investigating source code that he has never seen before (i.e. when extending a third party’s software system) and is not capable of understanding its rationale. Additionally, there are many situations that find a developer seeking source code that is already developed by others. He might not be aware of its existence, or even if he is, he is not able to find it effectively [Bauer and Roser (2006)], [Thaddeus and Kasmir (2006)].

Recent trends in data integration and combination have led to the semantic web vision [Berners-Lee et. Al. (2001)], [Fensel et. Al. (2002)]. The semantic web strives to
provide integration among information scattered across different locations in a uniform fashion. It defines a “common language” between these information resources.

Additionally, there is a growing interest in the last few years on exploiting semantic web technologies in software engineering. Since 2005 the Semantic Web Enabled Software Engineering (SWESE) conference takes place every year with promising results. The Semantic Web Best Practice and Deployment Working Group (SWBPD) in W3C included a Software Engineering Task Force (SETF) to investigate potential benefits of applying semantics in software engineering processes. As noted by SETF [Knublauch et. Al. (2006)], advantages of applying semantic web technologies to software engineering include reusability and extensibility of data models, improvements in data quality, enhanced discovery, and automated execution of workflows.

In this paper we propose a new system – code-named DevWiki for Semantic Wiki for Software Development – which is the result of exploiting semantic web technologies in the context of a well known software development IDE (Integrated Development Environment) such as Eclipse (http://www.eclipse.org/), in order to provide the basis for better collaboration in the software development domain and strive against the aforementioned problems. DevWiki aims at providing software developers such a tool to ease their daily work and facilitate source code documentation and problem solving.

The rest of the paper is organized as follows. In the next section, we describe what motivated us in developing the DevWiki system. Afterwards, we describe its functionality and a usage scenario. Thereafter, we give the results of a scenario-based evaluation we conducted. Finally, we discuss the benefits of using the system and conclude the paper.

2. Motivation

Modern IDEs such as Eclipse provide means for documenting and annotating source code, e.g. JavaDoc and Java annotations. However, this kind of documentation is unstructured and the means to retrieve it are limited.

In most cases, software developers when in need of specific functionalities find themselves seeking in forums, web sites or web search engines using keyword search for an API capable of delivering these functionalities. Additionally, software developers often use dedicated forums and mailing lists to report on bugs and issues they have with specific source code fragments and have to wait for answers from experts. This switching of environments while developing code is obtrusive and requires a great amount of effort in order to maintain scattered resources of software development related knowledge [Gall and Reif (2008)].
We argue that a more formal description of the aforementioned knowledge could be beneficiary for the software developers and the means to articulate and visualize this knowledge should be unobtrusive i.e. in the IDE itself in order to provide unified access to it. On top of that, knowledge about source code or software components should be described in a common way in order to avoid ambiguities when this knowledge is retrieved and used by others.

Documentation of software, bug fixing and software related knowledge transfer between employees of a large software house can be facilitated by the use of ontologies to sufficiently capture and describe software artifacts in a coherent way. Furthermore, ontologies are ideal for interlinking software artifacts and the resulting semantic relationships can be exploited in order to maximize productivity when seeking related knowledge of a specific software artifact.

Towards this end we have been motivated to design and implement the DevWiki system that offers the opportunity to articulate and visualize software artifacts driven by an ontology backbone. The created artifacts formulating an organization’s software development knowledge base can be exploited by semantic search engines and P2P metadata infrastructures to enable knowledge sharing and reuse in order to boost the collaboration among the employees and speed up time-consuming procedures when facing problems with code or searching for APIs or components to be reused.

3. DevWiki

DevWiki is designed and implemented in order to assist software development work inside the Eclipse IDE exploiting the power of semantic-based technologies. The semantic wiki can be used for articulating and visualizing formal descriptions of software development related knowledge in a flexible and lightweight manner. This knowledge can be then retrieved and used in a productive manner by integrating DevWiki with a semantic search engine and a P2P metadata infrastructure such as GridVine [Mauroux et. Al. (2007)]. This integration will assist the collaboration and the formation of teams of software developers who can benefit from each others’ knowledge about specific problems or the way to use specific source code while developing software systems. We have already integrated and tested DevWiki with a semantic search engine and GridVine in the context of the TEAM project [TEAM (2008)]. A demo of the TEAM system which includes DevWiki is available online - http://www.team-project.eu/demo.php.

DevWiki is the result of the improvement of an older version which is SoWiSE [Panagiotou and Mentzas (2008)] and provides common functionalities with a conventional wiki with semantic capabilities and software knowledge articulation support with the usage of ready to use templates. DevWiki uses a lightweight and flexible editor instead with auto-completion and popup support to eliminate the need
of templates which sometimes are cumbersome to use. Browsing through knowledge is done like surfing through a conventional wiki using the semantic links between different knowledge artifacts. This browser is available inside the Eclipse IDE so that the software developer does not have to switch to another external browser.

3.1 Ontologies

We have deployed a set of software development ontologies to the DevWiki system in order to describe and capture knowledge related to software artifacts. The system architecture allows for the extension or even the use of different software development ontologies. The set of the deployed ontologies constitutes of three separate ontologies.

The knowledge artifact ontology describes different types of knowledge artifacts such as the structure of the project source code, reusable components, software documentation, knowledge already existing in some tools, etc. These knowledge artifacts typically contain knowledge that is rich in both structural and semantic information. Providing a uniform ontological representation for various knowledge artifacts enables us to utilize semantic information conveyed by these artifacts and to establish their traceability links at the semantic level.

The problem/solution ontology models the problems occurring during the software development process as well as how these problems can be resolved. This ontology is essential to the DevWiki system, as source code can be documented when a certain problem is met and the respective solution to it is described.

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 knowledge artifacts’. We distinguish two different types of annotations: (a) domain annotation – software providers in different domains should classify their software using a common vocabulary for each domain. Common vocabulary is important because it makes both the users and providers to talk to each other in the same language and (b) software engineering annotation – general knowledge about the software domain including software development methodologies, design patterns, programming languages, etc.

3.2 Functionality overview

An overview of the DevWiki functionality is given below.

Validation of the user input when incorrect values for selected concepts, properties, related instances or property values are introduced. The semantic wiki informs the user about the wrong input value when a wiki page is saved.
Auto-complete facilities provide assistance to the user for selecting the right concept, properties and related instances of an object property as a value.

Intuitiveness – the software developer is able to understand what is required as input using the auto-complete facilities of the Semantic Wiki.

Syntax colouring – the software developer is assisted with syntax colouring to easily determine what kind of information is recognized by the wiki engine in order to avoid mistyping.

Multipage editor consisting of the semantic wiki editor and the semantic wiki browser which facilitates navigation of the knowledge base in an HTML-fashion inside the Eclipse environment.

Navigation through the semantic wiki is enabled via following an instance’s semantic links to other related instances.

On the fly synchronization with the knowledge base. DevWiki keeps no files anywhere and all information is persisted directly to the knowledge base.

As noted above the problem/solution ontology is essential to DevWiki. This is due to the fact that it models problems in everyday’s coding activities of the software developer and the respective solutions to these problems.

The developer is able to instantiate these concepts easily using the wiki. He/she can describe a problem by using the popup support of DevWiki in order to choose properties that he/she wishes to instantiate and then save the wiki page resulting in a new problem instance in the knowledge base. An example of using the problem/solution ontologies is given in Section 3.3.

3.3 Usage scenario

Dave has a problem with SWT applications which embed Fedora’s version of Firefox. He decides to document his problem in the wiki editor (see Figure 1).

John who is connected to the P2P network while browsing the wiki notices Dave’s problem and documents a workaround about the issue using the wiki.

At a later point of time, Dave is able to perform a structured query in the P2P network and define that he is searching for a solution which is about “Mozilla” (Figure 2).

He retrieves the result as in Figure 3 and can click on the link provided to open the workaround description in the browser mode of the wiki (Figure 4). He can further browse related knowledge by clicking on the semantic links provided by the wiki, such as “SWT” or “Mozilla” in order to see further descriptions.
Figure 1. DevWiki wiki editor view describing an issue with SWT applications that embed Fedora’s version of Firefox.

Figure 2. Structured P2P query for solutions that are related to “Mozilla”.

Figure 3. Search results of the query sent as in Figure 2.

Figure 4. DevWiki wiki browser view describing how to overcome the crash in the SWT web browser when visiting a page with password field.
4. Scenario-based evaluation

We have conducted a scenario-based evaluation of DevWiki in small teams of software developers in a total of 9 persons in the following organizations: (1) a Brussels based company specializing in the field of Information and Communication Technology (ICT) services (Intrasoft International S.A. – 2 developers), (2) a leading hungarian association dealing with open source software at corporate level (Linux Industrial Association – 2 developers), (3) an italian company which operates in the Information Technology market, focusing on business applications (TXT e-Solutions – 2 developers) and (4) the corporate research laboratory of the Thales group – a global electronics company serving Aerospace, Defence, and Information Technology markets worldwide (Thales Research & Technology – 3 developers).

4.1 Method

Scenario-based evaluation is drawn from the well-known technique of scenario based design [Carroll (1995)], [Malin et. Al. (2001)]. Briefly, a scenario describes sequences of actions taken by a user with a specific goal in mind. Scenarios are paper based descriptions of key processes. Comments on these by users tell designers:

- what users want from a system;
- how they intend to use it;
- how the system could be structured to facilitate that use.

It establishes a structured dialogue with representatives of user groups, to enable the developers to use this data as a source for formative design. According to this method, end-users will write scenarios, based on how they would use the system according to a specific user role, and for a specific task. For the application of this method in the evaluation of the DevWiki system we adopted a slightly different configuration to capture more aspects of the system in a more structured way. Our approach also involved end users but instead of writing scenarios they had to perform specific tasks using the system and then to document their experience using predefined questionnaires (see Table 1 for an example). The process took place using an online collaborative platform (CENTRA - http://www.korimvos.gr/en/products/centra.htm).

4.2 Execution

The execution of the scenario-based evaluation involved a considerable effort in preparing the evaluation material, installing the DevWiki system, coaching the end-users, and organizing and performing the sessions with the end users.

The adopted process had four distinct steps: (a) Plan: identification of user classes of the DevWiki system for the user sites (i.e. developer, knowledge provider, manager, etc.) and selection and recruitment of user participants according to the user classes,
(b) Conduct evaluation session: briefing and training (if necessary) required to undertake the task (i.e. in the aims/functions of the system) and feedback elicitation using questionnaires, (c) Analyze and document results: consolidation and analysis of scenario-based evaluation results and (d) Feedback results to designers: report for the DevWiki designers who in turn report back to evaluators with adjustments of the system.

4.3 Example of a Scenario-based Questionnaire

The purpose of this questionnaire (see Table 1) was to understand how developers perceived the task of creating a new semantic wiki page for a run-time error, describe it and specify knowledge artifacts for it.

**Table 1. Scenario-based questionnaire example**

<table>
<thead>
<tr>
<th>Question</th>
<th>Percentage of affirmative answers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can the developer easily understand what to do in order to create a new wiki page?</td>
<td>100% (9/9)</td>
</tr>
<tr>
<td>Is the wiki syntax clear and easy to follow?</td>
<td>78% (7/9)</td>
</tr>
<tr>
<td>Can the developer easily locate a property he/she wants to instantiate using the wiki support?</td>
<td>100% (9/9)</td>
</tr>
<tr>
<td>Can the developer easily locate an individual he/she wants to associate with the currently-created wiki page?</td>
<td>78% (7/9)</td>
</tr>
<tr>
<td>Can the developer easily understand what is required as input data?</td>
<td>78% (7/9)</td>
</tr>
<tr>
<td>Does the DevWiki system provides the developer with the friendly and easy-to use pop-up menus for creating wiki pages?</td>
<td>89% (8/9)</td>
</tr>
</tbody>
</table>

5. Discussion

There are emerging efforts in research towards exploiting semantic web technologies in software engineering. For example, SemIDE [Bauer and Roser (2006)] is a framework which is based on meta-models in order to describe semantically software engineering processes. OSEE [Thaddeus and Kasmir (2006)] is a tool which aids the construction of a knowledge base which captures software related artefacts. SeRiDA [Athanasiadis et. Al. (2007)] combines object-oriented programming and relational databases, with semantic models.

With DevWiki integrated into an IDE, it is easier for software developers to create new knowledge. The advantage of recording the solution to difficult or frequently asked questions/problems is that developers are recording their explicit knowledge about particular problems, as well as tacit knowledge that they have internalized,
while creating software development knowledge artifacts that can be organized, managed, and reused.

This seamless integration with the Eclipse IDE gives software developers the incentives to develop or explore knowledge without bothering to open an external tool or web browser to accomplish this task which has the advantage of leaving the user in his/her current work context and not distracting him/her from the rest of his/her tasks. For example, he/she can develop source code and at the same time document problems that he/she encounters or provide guidelines and workarounds for a specific problem in source code that is commonly known to his/her organization by simply choosing to open the DevWiki editor next to the source code editor.

Wikis can significantly help developers fill this knowledge management need. DevWiki provides this kind of wiki-based support for knowledge provision and offers innovative features especially in software development but is also a contemporary semantic wiki in the general sense since it provides many features common in most of the state-of-the-art semantic wikis [Maalej et al. (2008)].

6. Conclusion

In this paper we presented the DevWiki system, a Semantic Wiki for Software Development which is integrated in the Eclipse IDE, in order to assist software developers in articulating and visualizing software development knowledge. The resulting knowledge base can be exploited by semantic search engines or P2P metadata infrastructures in order to foster better and more flexible collaboration among software developers scattered across the globe.

We aim at providing support to the software developer for documenting code, finding the best solution to a problem or the way to move forward in critical decisions in order to save time during the heavy task of software development.

The introduction of semantics and the formal description of software development knowledge has the advantage of allowing for a more efficient organization and management of this knowledge which can be reused afterwards using semantic search and structured queries. Additionally, semantic relationships between similar concepts can be exploited in order to reveal additional knowledge (e.g. between similar solutions for a given problem) as opposed to forums where the developer can only follow a single thread for a particular topic or use simple keyword search for finding other related topics.

References


