Managing Open Source Software Development Projects With Semantic Web Technology

Wikan Danar Sunindyo[1] Thomas Moser[1] Stefan Biffl[1]

[1] Institute of Software Technology and Interactive Systems
Vienna University of Technology
Vienna, Austria
{wikan, tmoser, biffl}@ifs.tuwien.ac.at

Abstract. In open source software (OSS) project development typically tools and models originating from heterogeneous background are used. Project managers want to analyze the state of project across these artifacts but often face the challenge of linking semi-structured information across artifacts, e.g., source code versions, mailing list entries, and bug reports. Manual analysis based on finding related information between data sources is costly, error-prone, and often brings results too late for decision making. In this paper we propose linking OSS artifacts with a semantic web technology approach: the engineering knowledge base (EKB). The EKB consists of two types of ontology layers: 1. the common domain knowledge layer and 2. local tool knowledge layer. Mappings between local and domain ontology layers allow querying the local knowledge using the more stable domain ontology syntax. This EKB foundation enables the design of applications, such as a project monitoring system. We empirically evaluate the feasibility, effort, and complexity of an EKB-based project monitoring system based on real-world data.

Keywords: open source software development, engineering knowledge base, ontology, semantic web

1 Introduction

The stakeholders of open source software (OSS) development projects from heterogeneous backgrounds usually use different tools and models for developing and managing the software. For example, the developers may use SVN for managing source code version, mailing list for communication between developers and bug reports for managing the defects from the software.

In the OSS development projects, the stakeholders come from distributed areas and may work in different time zones. Therefore, the project manager needs to be able to manage and monitor the status of the project and how the people work, e.g., by monitoring the communication level between the stakeholders.

Currently, the project managers use a quantitative measurement approach for managing and monitoring the status of the projects, i.e., measure the numbers of commits in SVN, numbers of mailing lists posts or the number of issues in bug report in certain periods. Further analysis to find related information between data sources gets done manually, which is costly, error-prone, and often takes too much time for the analysis results to be useful for decision making.

In this paper we propose the concept of an engineering knowledge base (EKB), a semantic web technology approach, which explicitly links data model elements based on their semantic descriptions. The EKB consists of two types of ontology layers; the first layer contains the common domain knowledge, while the second layer contains the local tool knowledge. To bridge both ontology layers, there are mappings between the local and domain ontologies that allow people to access and make query to the local knowledge by using the common domain ontology syntax.

The software development data from heterogeneous data sources will be modeled and stored in local ontologies. We populate the ontologies by using fetcher tool that will collect the data from SVN, mailing list and bug report. The rule assigned to the ontologies during the population makes the completeness and correctness of the stored data can be checked. The data collected from heterogeneous data sources can show the communication relationships between the stakeholders and then show the relationships between different data sources that can be seen by using communication metric.

We will check the feasibility of querying the ontologies in EKB written in SPARQL[1] format. The idea is by giving query to the common domain ontology, it is possible to get related information from the local ontologies by transforming the query from the common domain ontology to the related local ontologies. Then we will compare the complexity of query between using SPARQL and SQL formats. The major result is the easier definition of queries on project data originating from heterogeneous background. We will evaluate the proposed approach by comparing the using of SPARQL and SQL queries on the EKB of Apache Tomcat[2] open source project.

This remainder of this paper is organized as follows. The second section is about related work support this paper. The third section contains research issues. The fourth section introduces use cases related to the research issues. The fifth section reports on the evaluation of the use cases. The sixth section will discuss the advantage and limitations of the EKB approach in the evaluation context. The last section concludes and identifies further research work.

2 Related Work

This work is based on previous works on project monitoring and reporting for open source software development project. The conventional approach to use manual reporting is fit well on tightly coupled form of organization, but it is hard to be applied in loosely coupled form of organization, like open source software (OSS) development projects [12].

---

[1] http://www.w3.org/TR/rdf-sparql-query/
[2] http://tomcat.apache.org/

Some initiatives to apply the semantic web technologies in (open source) software development projects have been done by some researchers. We will discuss the related works in this chapter.

2.1 Project Monitoring and Reporting in Software Development Projects

In [12] Wahyudin et al. discussed that the project monitoring traditionally is focusing on human reporting, which is good for tightly-coupled organization to ensure the quality of project reporting. In the loosely-coupled organization like open source software development projects, this approach can not be applied because the stakeholders do their job voluntarily and flexibly. One way to measure the performance of the project is correlating and analyzing processes event data from the OSS community.

von Krogh and von Hippel [11] made a policy research in OSS development and found that there are differences between monitoring proprietary software development and OSS development. In commercial software development, the project manager can applied tight management of processes and take precautions to prevent employees from leaking software-related trade secrets and information to competitors, while in the OSS development the software architecture and functionality are governed by a community consisting of developers who can commit code to the authorized version of the software.

Yamauchi et al. [13] stated that in the traditional perspective, the managing and leading of OSS development seem to be impossible, because no formal quality control program exist and no authoritative leaders monitor the development. It is surprising that the OSS development achieves smooth coordination, consistency in design and continuous innovation while relying heavily on electronic environment as face-to-face supplementary. The paper discussed how OSS development avoids limitations of dispersed collaboration and addressed the sources of innovation in OSS development. However, further research is needed to reveal the software quality improvement by using the OSS project data.

In [10], Sharma et al. divided the OSS development framework into three aspects, namely structure, processes and culture. In the OSS processes, the stakeholders can have governance mechanisms, for example by applying membership management, rules and institutions, monitoring and sanctions, and reputation as one of the prime motivators for the OSS developers. Even though the membership of OSS is open to anyone, the OSS communities manage membership in conjunction with rule and institutions, and monitoring and sanctions. The sanction is given if they misbehave or disrupt the progress of the project, the form of sanctions are flaming, spamming or shunning. This paper is quite good to illustrate how the OSS can be monitored via social interaction and sanction from the communities. However the relationships between the project data produced by the stakeholders, the activities of the stakeholders, and the quality measurement of OSS are not illustrated here. We can continue this research by making correlation analysis from heterogeneous data sources from the OSS projects for software monitoring, for instance.

2.2 Semantic Web Technology in Software Development

The application of ontology as semantic web technology for managing knowledge in many specific domains is inevitable. Noy and Guinness [9] noted five reasons to develop an ontology, namely to share common understanding of the structure of information among people or software agents, to enable reuse of domain knowledge, to make domain assumptions explicit, to separate domain knowledge from the operational knowledge, and to analyze domain knowledge.

In [8], Moser et al. has introduced an Engineering Knowledge Base (EKB) as a semantic web technology approach for solving data heterogeneity problems in production automation domain. Furthermore, in [4] Biffl et al. used the approach for solving similar problems but in different domain, i.e. frequent-release software projects. The software projects domain has richer issues and frequent changing data than production automation domain. However these works are still on initial stage and need further research.

Happel and Seedorf summarized the using of ontology approach throughout the phases of software engineering lifecycle in [5]. They compared the advances in software engineering and knowledge engineering, and found that the software engineering community now is more focusing on software modeling, while the knowledge engineering community is eager to promote several modeling approaches to realize the vision of the semantic web. Hence, both disciplines are closely related and the communities from both areas can contribute each other. The paper is also categorizing ontologies in software engineering into four areas based on time dimension (development-time vs. run-time) and types of knowledge (infrastructure vs. software), namely ontology-driven development, ontology-enabled development, ontology-based architectures, and ontology-ruled architecture. The authors also mentioned the advantages of using ontology in software engineering, among others are the ability to make logic-based formalisms for software engineering in the context of the semantic web efforts. The flexibility of ontologies makes the combination from various sources of software engineering data easier to be done. However this paper is just a preliminary step towards a better understanding of possible benefits of ontologies in software engineering.

The other using of ontology in software engineering for example for supporting software architecture decision approach suggested by Akerman and Tyree [1] and for software development knowledge reuse suggested by Antunes et al [2]. The open source software development method has a little bit differences with the conventional software development method that could be interesting for application of ontology as well.

2.3 Open Source Software Development

Open source software development is a knowledge-intensive domain that could be improved by using semantic web technology. Sharma et.al. [10] stated that the open source software development models are considered to be successful and meet the software quality requirement. Even tough some OSS is not quite successful, but in general can be used as a model for proprietary software development as well. In this paper [10], the authors gave the framework for creating hybrid-open source software communities, that consists of structure, processes and culture. The different kind of stakeholders involved, the processes that should be done and the heterogeneous culture can lead to the semantic gaps between the stakeholders that can be solved by using the ontology.

Mockus et.al. [6, 7] report on the analysis on two open source software, namely the Apache web server and Mozilla browser. In this paper, the authors wanted to quantify the aspects of developer participation, core team size, code ownership, productivity, defect density and problem resolution from both open source software. To do that the authors collect and analyze the data from different sources, for example developers' mailing list, concurrent version archive (CVS) and problem reporting database (BUGDB). However, these papers focus on using of separate historical artifacts to measure the quality of software. In this research, we will integrate historical and current artifacts to get more information by using ontology.

3 Research Issues

The rapid growth in OSS development nowadays needs more control from the project managers and quality managers to manage the process of software production successfully and efficiently. New approaches in organizing the development of high quality open source software are introduced, for example the Apache Software Foundation (ASF)[3] that provides organizational, legal, and financial support for a broad range of open source software projects. Formerly known as the Apache Group, the Foundation has been incorporated as a membership-based, not-for-profit corporation in order to ensure that the Apache projects continue to exist beyond the participation of individual volunteers.

Currently there are about 70 OSS projects under ASF. The Apache projects are characterized by a collaborative, consensus based development process, an open and pragmatic software license, and a desire to create high quality software that leads the way in its field.

To manage the project and the quality of open source software development, the managers could use Goal Question Metrics (GQM) approach from Basili [3]. GQM is a method to do empirical experiment and measurement on software development, which consists of conceptual level (goal), operational level (question) and quantitative level (metric). By using GQM method, the managers should define the goals, set questions to fulfill the goals and collect the data to answers the questions.

In this paper, the aim of the project managers is to improve the efficiency of project reporting and monitoring. The questions can be raised are what is the current situation of project reporting and monitoring in OSS domain, and how the efficiency can be improved. For answering these questions the project managers should collect communication data from development project in the form of communication metric for further analysis purposes, like project reporting and monitoring or decision making on the OSS development.

[3] http://www.apache.org

Collecting the related information from heterogeneous data sources for project reporting and monitoring is not an easy task for project managers, because each data source has different format and structure. In the current approach, the project managers should collect the data independently from the data sources and then find the relationships between the different data sources. The making of tools to integrate the different data sources can help the data integration process, but as long as the data become more complex and the new data sources are added, this approach is not sufficient anymore. More flexibility is needed to handle the complexity of project data.

The semantic web technology approach can help to solve the semantic heterogeneity problem among different data sources, by supporting the knowledge representation and storage for all project development data. The utilization of rule and reasoning can help the user to check the correctness of the related data from heterogeneous data sources and find the relationships (e.g., synonym, generality-specialty) between the data.

One approach of using ontology as a semantic web technology is introduced by Moser et al. [8] in the form of Engineering Knowledge Base (EKB) concept. EKB is a repository that reserve all knowledge, e.g., data model, user information, from different kinds of engineering fields, in this case we focus on the engineering fields that used by the OSS development projects domain. EKB consists of two types of ontology layers. The first layer contains the common domain knowledge, while the second layer describes the local tool knowledge. Mapping between local and domain ontologies allow querying the local knowledge by using the domain ontology syntax.

For the application of EKB in the OSS development projects domain, we can define the research issues of this paper as follows.

RI-1: the feasibility of the EKB approach. We will investigate the feasibility to define OSS reporting queries on a domain level based on the data collection mechanism from the EKB with real-world use cases.

RI-2: the complexity of the EKB approach. We will measure the complexity of a database and a semantic integration query approach.

We will use one of Apache project, namely Apache Tomcat[4] as a test bed to answer these research issues. Apache Tomcat is a servlet container that implements the Java Servlet and the JavaServer Pages (JSP) specifications from Sun Microsystems. We choose Apache Tomcat project, because it has long story of development, quite stable, has several versions and a lot of development data that can be analyzed.

The communication metric will be an application to support project reporting and monitoring. Consisting of communication artifacts of OSS development projects, like SVN, developer's mailing list and bug reports, together with the semantic meaning and their relationship, the communication metric can provide significant insight into the development process that produced them. In the use case chapter, we will show the project communication cockpit as an application of EKB utilization for OSS development projects domain that is easier to be defined by using EKB than former conventional approach.

[4] http://tomcat.apache.org/

4 Use Case

This section describe the implementation of EKB in OSS development projects especially in making query from the domain level ontology and getting the result from the local level ontologies, and using communication metrics for supporting project reporting and monitoring for project managers. For doing this, we will collect the data from Apache Tomcat project.

Apache Tomcat[5] is an open source software implementation of the Java Servlet and JavaServer Pages technologies. The Java Servlet and JavaServer Pages specifications are developed under the Java Community Process.

There are several versions of Apache Tomcat, for example Tomcat 5.5 and Tomcat 6.x . Tomcat 5.5 implements the Servlet 2.4 and JSP 2.0 specifications. It reduced garbage collection, improved performance and scalability, and used native Windows and Unix wrappers for platform integration and has faster JSP parsing.

Tomcat 6.x implements the Servlet 2.5 and JSP 2.1 specifications. It supports for Unified Expression Language 2.1 and designed to run on Java SE 5.0 and later. It support for Comet through the CometProcessor interface, but does not include an administration console, unlike past release. The development data will be collected by using project data fetcher tool and analyzed using project monitoring cockpit that will be explained below.

4.1 Project Data Fetcher Tool

The project data fetcher tool is a special application running on OSS development project to collect the development data from heterogeneous data sources of the project and then store into a semantically-enabled data warehouse. The semantically-enabled data warehouse then can be used as a basis for advanced data analysis and support for decision making.

The project data fetcher tool prototype is implemented in the Java programming language to support platform independence and to access its rich libraries from its big community. The tool is using Jena[6] framework to access the ontology. The Jena framework provides an OWL API for programmatic acess to OWL ontologies using Java and a tool called “schemagen” which creates a java class file, containing an instance of the ontology model as well as the elements of the input ontology as static fields.

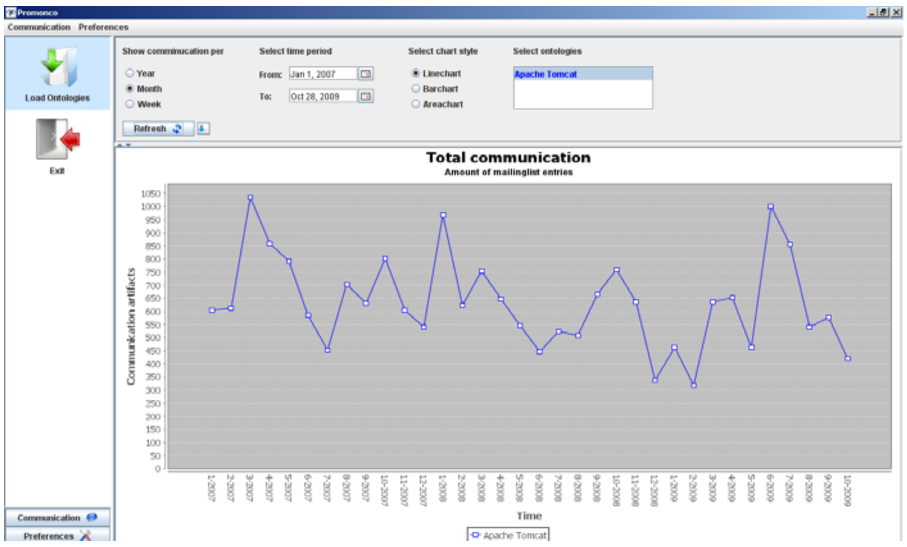
The tool is also using OntModel class for creating the ontology model and Commons Configuration for setting the configuration of the input. The tool will take the data from SVN, mailing list and bug tracker as input and transform into development data ontology as output for further purpose. Further information about this tool can be seen in Biffl [4].

[5] http://tomcat.apache.org/
[6] http://jena.sourceforge.net

4.2 Project Monitoring Cockpit

The project monitoring cockpit tool is an application that will be used for monitoring the status of project based on the communication artifacts from the project. By using the cockpit tool, the project manager can see the total communication of the project stakeholders during some periods of time from the number of mailing list entries. The relationships of the stakeholders also can be seen by using this tool, for example to see the user coupling on some related topics. If a stakeholder communicates with other stakeholder or work in the same module/file, then there is a relationship between the stakeholders.

The cockpit tool is developed using Java programming language. It takes ontology resulted from project data fetcher as input and produced visualization of communication metrics as outputs. One of the communication metrics graph can be seen in the figure 1. This figure shows the amount of mailing list entries of Apache Tomcat development per month from January 1st, 2007 to October 28th, 2009.



Show communication per: Year [✓ Month] Week

Select time period: From: Jan 1, 2007 To: Oct 28, 2009

Select chart style: [✓ Linechart] Barchart Areachart

Select ontologies: Apache Tomcat

Fig. 1. Total Communication of the Apache Tomcat Project Graph display using Project Monitoring Cockpit Tool

5 Evaluation

In this section, we evaluate the use case described in section 4 based on the research issue in section 3, to show the capability of EKB approach in solving semantic heterogeneity problem in OSS development projects domain comparing to the traditional database approach.

5.1 UC-1: Feasibility of EKB Approach

The problem describe here is related with the project monitoring tasks, to find the relationships of the different data sources, for example between mailing list and SVN. In this case, the project manager wants to know whether a certain issue in mailing list has been resolved by some developers by implementing the code in the source code repository (SVN) or not. For doing this task, the project manager has to collect information from SVN and mailing list before getting the answer of the query.

Database approach. By using the database approach, all data from different sources should be stored in the common repository. Then the project manager will make a join query to summary the information from different sources and the common repository. The illustration of the query making can be seen in figure 2b. The weaknesses of this approach are the different structures from heterogeneous data sources should be reconciled before getting into the common repository, the updates and synchronization between the data sources and the common repository should be done each time the new data entries update the local data source, and some error-prone steps could be taken during the reconfiguration of the data sources, e.g., adding new data sources to the monitoring system.

EKB approach. By using the EKB approach, the project manager can make an integrated query to obtain the information from the local ontologies via the common domain ontology. The steps can be explained as follows, referring to the listing 1 that used a simplified OWL[7] syntax. The project manager will find the SVN entry that related with the *build.xml* by making query to the domain ontology. This query from the domain ontology is then continued and transformed to the mailing list ontology, resulting affected artifact named *build.xml_890256* that also has relationship with the SVN ontology. The next step is finding the affected artifact in SVN, and here we get the result SVN_890256_build.xml, which then will be transferred to the common domain ontology, and giving the right answer to the project manager.

5.2 UC-2: Complexity of EKB Approach

The complexity of the approaches here is determined by measuring the needed effort to do the approach for completing the query and getting the answer from the

[7] http://www.w3.org/TR/owl-semantics/

repository. The illustration of the comparison between the approaches can be seen in figure 2.

```
SELECT (?a) WHERE {domain:build.xml domain:relatedWith ?a}

mailinglist:build.xml owl:equalTo domain:build.xml

SELECT (?b) WHERE
{mailinglist:build.xml mailinglist:hasAffectedArtifact ?b}
Result: b = mailinglist:build.xml_890256

mailinglist:build.xml_890256 owl:equalTo domain:build.xml_890256
SVN:build.xml_890256 owl:equalTo domain:build.xml_890256

SELECT (?c) WHERE
{?c SVN:hasAffectedArtifact SVN:build.xml_890256}
Result: c = SVN:SVN_890256_build.xml

SVN:SVN_890256_build.xml owl:equalTo domain:SVN_890256_build.xml

Result: a = domain:SVN_890256.xml
```

Listing 1. Feasibility of EKB approach – query example to find related SVN entry from the mailing list issue.

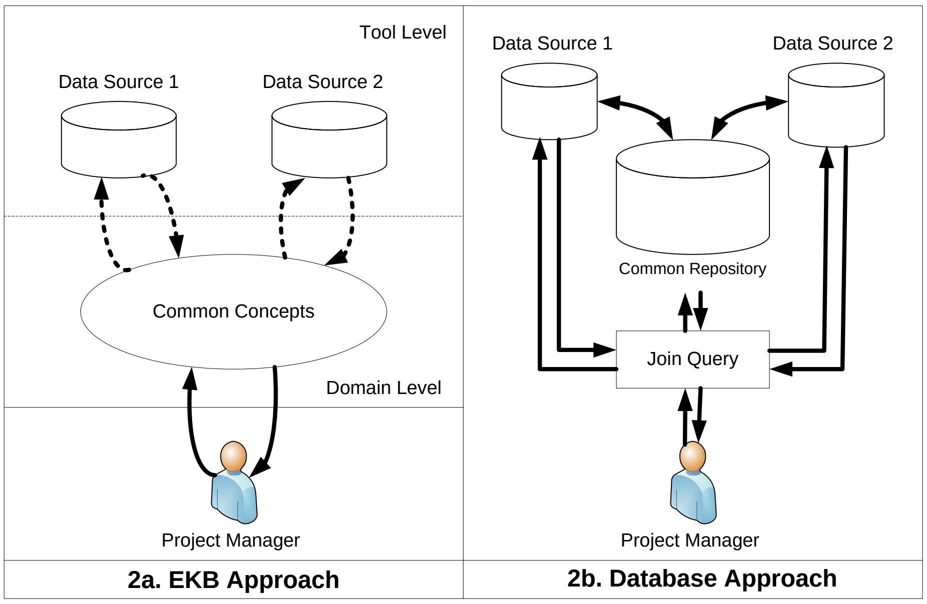


Fig. 2. Comparison of the efforts needed for querying between using EKB approach and database approach

Database Approach. In the database approach, the project manager should put all data from the heterogeneous data sources to the common repository. Then he makes the join query from the local data sources and the common repository to get the needed information as an answer. The extra efforts should be done by the project manager to make a join query than a normal query, since he has to see the structure of each data source and common repository to be able to make a right join query. The transformation and data updating from the data sources to the common repository make the complexity of this approach quite high and error-prone, because it used more manual efforts.

EKB Approach. In the EKB approach, the project manager is only facing the common concepts in the domain level. The collection of data from heterogeneous data sources is done automatically via the project fetcher data tool. The query from the common concepts to the local data sources is done using the ontology mechanism. In this approach, the complexity of using the manual efforts is reduced by using the ontology.

6 Discussion

This section discuss the lesson learned of using the engineering knowledge base approach as a part of semantic web technology to solve the semantic heterogeneity problem in the OSS development project domain.

This approach is feasible to be done and will be beneficial in tackling the heterogeneity problem as the scale of the OSS development project is growing larger and larger. The using of ontology in the OSS development make the project monitoring easier, since the coordination in the OSS development is loose relative to the commercial software development, while it still needed to keep the software quality high.

The query design and implementation to find needed information from the ontology are easier and less complex than by using the traditional query language, like SQL. However, the extra efforts to design and implement the general ontology and the supportive tool are still needed, but it’s worthwhile for further purpose.

7 Conclusion and Further Work

In this paper, we have explained the EKB approach as a semantic web technology approach to solve the semantic heterogeneity problem in the OSS development projects domain. By separating the level of ontologies in EKB, the complexity of querying can be reduced, while still makes the approach feasible to be done.

The future work includes make empirical measurement on the efforts needed, especially for adding new data sources to the project monitoring systems. This could be useful for making better decision on the status of the project, since more development data sources could be obtained from the project.

References

1. Akerman, A., Tyree, J.: Using ontology to support development of software architectures. IBM Syst. J. 45 (2006) 813-825
2. Antunes, B., Seco, N., Gomes, P.: Using Ontologies for Software Development Knowledge Reuse. Progress in Artificial Intelligence (2007) 357-368
3. Basili, V., Caldiera, G., Rombach, H.: The goal question metric approach. Encyclopedia of Software Engineering. John Wiley & Sons, Inc (1994)
4. Biffl, S., Sunindyo, W.D., Moser, T.: Semantic Integration of Heterogeneous Data Sources for Monitoring Frequent-Release Software Projects. 2010 International Conference on Complex, Intelligent and Software Intensive Systems (to be published). IEEE Computer Society, Krakow, Poland (2010) 8
5. Happel, H.-J., Seedorf, S.: Applications of Ontologies in Software Engineering. 2nd International Workshop on Semantic Web Enabled Software Engineering (SWESE 2006) held at the 5th International Semantic Web Conference (ISWC 2006), Athens, GA, USA (2006)
6. Mockus, A., Fielding, R.T., Herbsleb, J.: A case study of open source software development: the Apache server. Proceedings of the 22nd international conference on Software engineering. ACM, Limerick, Ireland (2000)
7. Mockus, A., Fielding, R.T., Herbsleb, J.D.: Two case studies of open source software development: Apache and Mozilla. ACM Trans. Softw. Eng. Methodol. 11 (2002) 309-346
8. Moser, T., Biffl, S., Sunindyo, W.D., Winkler, D.: Integrating Production Automation Expert Knowledge Across Engineering Stakeholder Domains. 2010 International Conference on Complex, Intelligent and Software Intensive Systems (to be published). IEEE Computer Society, Krakow, Poland (2010) 8
9. Noy, N.F., McGuinness, D.: Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory (2001)
10. Sharma, S., Sugumaran, V., Rajagopalan, B.: A framework for creating hybrid-open source software communities. Information Systems Journal 12 (2002) 7-25
11. von Krogh, G., von Hippel, E.: Special issue on open source software development. Research Policy 32 (2003) 1149-1157
12. Wahyudin, D., Tjoa, A.M.: Event-Based Monitoring of Open Source Software Projects. Proceedings of the The Second International Conference on Availability, Reliability and Security. IEEE Computer Society (2007)
13. Yamauchi, Y., Yokozawa, M., Shinohara, T., Ishida, T.: Collaboration with Lean Media: how open-source software succeeds. Proceedings of the 2000 ACM conference on Computer supported cooperative work. ACM, Philadelphia, Pennsylvania, United States (2000)