The Journal of Systems and Software 000 (2016) 1–17

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Construction and utilization of problem-solving knowledge in open source software environments

Hyung-Min Koo*, In-Young Ko

School of Computing, Korea Advanced Institute of Science and Technology (KAIST), 291 Daehakro, Yuseong-gu, Daejeon, 305-701, Republic of Korea

ARTICLE INFO

Article history:
Received 15 June 2015
Revised 1 February 2016
Accepted 23 June 2016
Available online xxx

Keywords:
Software reuse
Open source software
Knowledge-based software reuse
Bayesian network

ABSTRACT

Open Source Software (OSS) has become an important environment where developers can share reusable software assets in a collaborative manner. Although developers can find useful software assets to reuse in the OSS environment, they may face difficulties in finding solutions to problems that occur while integrating the assets with their own software. In OSS, sharing the experiences of solving similar problems among developers usually plays an important role in reducing problem-solving efforts. We analyzed how developers interact with each other to solve problems in OSS, and found that there is a common pattern of exchanging information about symptoms and causes of a problem. In particular, we found that many problems involve multiple symptoms and causes and it is critical to identify those symptoms and causes early to solve the problems more efficiently. We developed a Bayesian network based approach to semi-automatically construct a knowledge base for dealing with problems, and to recommend potential causes of a problem based on multiple symptoms reported in OSS. Our experiments showed that the approach is effective to recommend the core causes of a problem, and contributes to solving the problem in an efficient manner.

1. Introduction

Software reuse is the process of creating software systems from existing software assets rather than building them from scratch (Ali et al., 1999; Kruger, 1992). The main goal of software reuse is to improve software productivity, quality, maintainability, and reliability while reducing the cost, time, and complexity of software development (Boehm, 1999; Rothenberger et al., 2003). Reusable software assets can be classified into two types: tangible software assets and intangible software assets. *Tangible software assets* are software artifacts that are explicitly documented in forms such as source code, software components, and requirement specifications. *Intangible software assets* are implicit and tacit knowledge that are not documented, such as skills and experiences (Lee, 1993; McCarey et al., 2008; Rus and Lindvall, 2002). As the complexity of software is increased, reuse of intangible software assets is essential to reduce duplicated efforts of developing software, and to solve problems that occur while reusing tangible software assets in an effective manner.

Although the importance of utilizing the knowledge of software reuse has been stressed for a long time, there has not been a great deal of study done for constructing and reusing problem-solving knowledge in software reuse (Frakes, 1994; Haeiger et al., 2006). Software reuse environments such as SmartAPI, the NASA Reuse portal, and Software Reuse System (SRS) (Gerard et al., 2007; Eberhart and Agarwal, 2004; Antunes et al., 2007) provide developers with a Web-based reuse environment that facilitates easy and efficient access to software assets. However, they focus on managing and reusing tangible software assets.

Recently, Open Source Software (OSS) has become an important software reuse environment in various software development domains (Godfrey, 2000). OSS environments such as Sourceforge.net[1] and GitHub[2] enable developers to share and reuse software assets in a collaborative manner (Haefliger et al., 2008; Lakhani and Hippel, 2003). Although developers can find useful software assets to reuse from OSS, they often face problems when they try to integrate the software assets with their own software project. Developers usually use a discussion forum in an OSS environment to solve the problems by utilizing other developers' experience and knowledge of solving problems (Georg et al., 2005). However, the forum messages are written in a textual form, and it is usually difficult for developers to find appropriate messages that are helpful to solve their problems.

* Corresponding author.
E-mail addresses: hmkoo@kaist.ac.kr (H.-M. Koo), iko@kaist.ac.kr (I.-Y. Ko).

[1] http://www.sourceforge.net.
[2] https://www.github.com.

http://dx.doi.org/10.1016/j.jss.2016.06.062
0164-1212/

In our previous study, we investigated how developers interact with each other to solve a problem in OSS. Developers who face a problem in reusing a software asset usually post descriptions on a discussion forum about specific symptoms that they experienced. Other developers who experienced similar symptoms then join and post descriptions about additional symptoms on the discussion forum. After some period of time, possible causes of the symptoms are discussed among developers. Finally, solutions to the problem start appearing in the discussion forum. We defined these interactions between developers as *an interaction pattern for solving problems* in OSS (Koo and Ko, 2015).

In this paper, we propose an approach to construction and utilization of problem-solving knowledge in OSS. In our previous work, we analyzed and considered only the direct correlations between symptoms and causes of problems in OSS (Koo and Ko, 2015). However, we found that often there are multiple symptoms and causes that are related to a problem. We speculated that a problem that has multiple symptoms and causes is more difficult to solve, and requires more interactions between developers. To deal with these multiple symptoms and causes in an effective way, it is necessary to consider relative and probabilistic relationships based on co-occurrences of various combinations of symptoms and causes. A Bayesian Network (BN) is widely used for representing relative and probabilistic relationships between entities (Heckerman et al., 1995; Starr and Shi, 2004). Therefore, we applied a BN for constructing a knowledge base and recommending potential causes of problems in OSS. We have developed an approach of automatically building a BN from ontologies that are used to represent essential symptoms and causes in a software domain. We also defined a metric to calculate conditional probabilities of each cause or symptom in the BN.

Using the forum data sets that are stored in the 'User Interfaces' and 'Algorithms' categories from Sourceforge.net, we performed experiments to prove the effectiveness of using a BN to construct problem-solving knowledge, and to find appropriate causes of problems in OSS. The results of the experiments show that the precision of finding potential causes of problems can be improved by up to 10% by using the BN to consider multiple symptoms and causes of the problems. In addition, we could reduce the time spent to solve problems in OSS by up to 40% by recommending potential causes of the problems early in the problem-solving process.

We explain related work on utilizing knowledge for software reuse in Section 2. In Section 3, we summarize the results of our previous study to find interaction patterns of developers in solving problems in OSS. We explain the approach of creating an ontology-based BN and the metric to calculate conditional probabilities for the BN in Section 4. The results of the experiments that we conducted to prove the effectiveness of using the ontology-based BN for recommending the causes of problems in OSS are described and analyzed in Section 5. Finally, we discuss the contributions of our work and future research directions in Section 6.

2. Related work

BORE, Conceptually Oriented Design Environment (CODE), and NASA Experience Factory are the systems that were developed to enable developers to utilize knowledge about the project management such as tools, design, people, process, cost, and development methods for their software developments (Basii et al., 1992; Henniger, 1997; Skuce, 1995). These works focus on building and managing general knowledge for effective management of software development projects rather than constructing and utilizing problem-solving knowledge in software reuse.

A wide body of research has been carried out on building knowledge bases about software reuse and facilitating knowledge for developers in software reuse activities. Source code ECOsystem Linked Data (SeCold) provides a software reuse environment where developers can find necessary source code with various granularity (project, method blocks, and lines of source code) along with relevant knowledge to effectively reuse them (Iman et al., 2012). SeCold has an ontological model for extracting source code data and representing the semantic inter-linking between them by using Linked Data,[3] a Web-based infrastructure for connecting and sharing Semantic Web data. By using this source code related knowledge, SeCold provides developers with a set of capabilities to find information about implementing source code and handling errors and bugs. However, in SeCold, developers must interlink their source code with the concepts in ontologies for constructing a knowledge base, which is usually a difficult task, especially when the size of the source code is large. In addition, the utilization of the knowledge is focused mostly on checking the similarity between source-code blocks. In addition, the scope of the recommended knowledge is restricted to fixing bugs or errors in source code.

Linked Data Driven Software Development (LD2SD) provides a development environment that allows developers to search open source code and relevant software artifacts from a knowledge base built on Linked Data (Aftab and Hausenblas, 2013). LD2SD provides an Integrated Development Environment (IDE) that is connected to the knowledge base. All source code and their execution results in the IDE are stored and managed in the knowledge base. Developers can reuse not only the source code but also the execution results that other developers already experienced by using their own IDE. This work differs from ours in terms of the scope of knowledge and the way of constructing the knowledge base. The main approach used in this work for construction of knowledge is to make interlinks between Linked Data and existing software artifacts such as bug reports, version information, and source code. We focus on extracting problem-solving knowledge from the interactions (message exchanges) between developers.

Evolutionary Ontology (EvoOnt) is a repository system that stores information about errors and bugs that are related to source code (Kiefer et al., 2007). Errors and bugs that are posted in OSS are collected and stored by using ontologies. EvoOnt provides three ontologies: the software ontology, version ontology, and bug ontology. Based on these ontologies, the experiences of solving the errors and bugs are collected and stored by using the imprecise SPARQL (iSPARQL)[4] search engine. EvoOnt continuously detects bad-smell code, and orphan methods that cause errors and bugs, and finds their fixes to build the knowledge base. However, in this work, the collection and evolution of knowledge is done at the source code level rather than general problem-solving knowledge from actual interactions between developers.

KnowBench is a knowledge management system that manages the essential knowledge for handling errors and problems in reusing software components that developers experienced in past projects (Dimitris and Gregoris, 2011). The main goal of KnowBench is to provide an efficient way of dealing with the errors that developers face at the development time by allowing them to search relevant knowledge or documents. However, KnowBench focuses on building a static knowledge base at the source code level, and the main source of knowledge is the set of documents from past software development projects.

General Architecture for Text Engineering (GATE) defines a process for constructing a knowledge base by extracting data from software-related documents such as source code and user manu-

[3] http://www.w3.org/standards/semanticweb/data.
[4] https://files.ifi.uzh.ch/ddis/oldweb/ddis/research/completed-projects/semweb/isparql/.

Please cite this article as: H.-M. Koo, I.-Y. Ko, Construction and utilization of problem-solving knowledge in open source software environments, The Journal of Systems and Software (2016), http://dx.doi.org/10.1016/j.jss.2016.06.062