An Empirical Analysis of Software Changes on Statement Entity in Java Open Source Projects

Xiaoyan Zhu, Department of Computer Science and Technology, Xi’an Jiaotong University, Xi’an, P. R. China
Qinbao Song, Department of Computer Science and Technology, Xi’an Jiaotong University, Xi’an, P. R. China
Zhongbin Sun, Department of Computer Science and Technology, Xi’an Jiaotong University, Xi’an, P. R. China

ABSTRACT

Software projects keep changing all the time. Understanding the nature of the changes can help build higher quality projects. In this paper, the authors studied software changes on a new entity, statement. They found some types of statements are more likely to change than others. Furthermore, the authors studied software changes to fix bugs and also found some types of statements are more likely to change than others to fix bugs. These statements are more likely to cause bugs, which should be paid more attention to.

Keywords: Change Hunk, Co-Change Statements, Patch, Software Change, Statement

INTRODUCTION

Software development can be seen as a sequence of changes (Nagappan, Zeller, Zimmermann, Herzig, & Murphy, 2010). For the purposes like adding new features, improving efficiency, adapting to new environment, fixing bugs or just beautifying the programs, the codes of the software projects keep changing all the time. So the study of software changes has caught the attention of many researchers (Elish & Al-Rahman Al-Khiaty, 2012; Mens, Fernandez-Ramil, & Degrandart, 2008; Sharafat & Tahvildari, 2008; Tsantalis, Chatzigeorgiou, & Stephanides, 2005).

Configuration management systems, and more specifically, version control systems keep records of the modification history of a software project (German, 2004). The logs from these systems track who modified what, when and what the changes were. Such logs provide rich

DOI: 10.4018/jossp.2013040102
information about the software changes and enable us to implement study on these changes.

Understanding the nature of fine-grained source code changes can help comprehending software evolution (Fluri & Gall, 2006). Previous researchers have studied software changes on different entities. Some researchers implemented their study on file entity (Kagdi, Yusuf, & Maletic, 2006; Ying, Murphy, Ng, & Chu-Carroll, 2004; Van Rysselberghe & Demeyer, 2004) while some on class entity (Elish & Al-Rahman Al-Khiaty, 2012; Vasa, Schneider, & Nierstrasz, 2007). Changes on function entity have also been studied broadly (Wang, Wang, & Liu, 2009; Malik & Shakshuki, 2010). To obtain more specific knowledge on software changes, some researchers paid attention to study on a better-grained entity (Kim, Whitehead, & Bevan, 2006; Pan, Kim, & Whitehead, 2009; Whitehead, 2008). Some others studied changes on several different entities at the same time (Fluri & Gall, 2006).

Statement is an important unit of a programming language. However, none of the previous studies have explored software source code changes on statement entity. In this paper, we will study the changes of software projects on a new entity, statement. Such study has many potential uses.

When designing a language, the designers have to consider which types of statements to be used and what the grammar of each type of statements should be. What is more, they have to consider how to compile and run a specific type of statements on computer for a compiled language like Java or C++. So how a type of statements changed when maintaining the project is an important problem they have to consider. If a type of statements changed very frequently, this type of statements might be designed improperly. So developers have problems to use this type of statements.

For maintainers of a project, the change of different types of statements is also a key problem for them to consider. Knowing which types of statements changed very frequently in the past, they can consider if the use of such types of statements in the project is proper or if the developers of the project have problem with certain types of statements to finish some functions of the project.

When starting a new project, developers can obtain experience from the statement change history of other projects. So they can write a more stable project that needs fewer maintenance efforts in the future.

It is important to understand the nature of software defects so that we can have better estimation of software quality and better control of software process (Zhang, 2008). Exploring how bugs were fixed can help us to understand the nature of the bugs. Statements changed frequently to fix bugs may be partially or entirely the cause of the bug, so programmers should code them carefully and test them extensively. In this paper, statement changes committed to fix bugs are another important problem for us to study.

Since its creation in 1991, Java has gained popular use all around the world. Many projects are developed using Java language. Some of these projects have been developed and maintained for a long time. We will focus our study on the statement changes of projects written in Java.

In this paper, we seek to find answers to the following research questions.

**RQ1:** Which types of statements changed frequently in Java open source projects?

We hypothesize there are some similarities in the types of statements that changed frequently across different projects.

**RQ2:** Which types of statements changed frequently in Java open source projects to fix bugs?

We hypothesize there are some similarities in the types of statements that changed frequently to fix bugs across different projects.

Exploring the above research questions, the rest of the paper is arranged as follows. We introduce the related work first. Statement change extraction and analysis methods are presented afterwards. Next, we devoted to
Morality and Pragmatism in Free Software and Open Source
www.igi-global.com/chapter/morality-pragmatism-free-software-open/21176?camid=4v1a