Fault-Prone Module Prediction Approaches Using Identifiers in Source Code

Osamu Mizuno, Graduate School of Science and Technology, Kyoto Institute of Technology, Kyoto, Japan
Naoki Kawashima, Graduate School of Science and Technology, Kyoto Institute of Technology, Kyoto, Japan
Kimiaki Kawamoto, Graduate School of Science and Technology, Kyoto Institute of Technology, Kyoto, Japan

ABSTRACT

Prediction of fault-prone modules is an important area of software engineering. The authors assumed that the occurrence of faults is related to the semantics in the source code modules. Semantics in a software module can be extracted from identifiers in the module. Identifiers such as variable names and function names in source code are thus essential information to understand code. The naming for identifiers affects on code understandability; thus, the authors expect that they affect software quality. In this study, the authors examine the relationship between the length of identifiers and existence of software faults in a software module. Furthermore, the authors analyze the relationship between occurrence of “words” in identifiers and the existence of faults. From the experiments using the data from open source software, the authors modeled the relationship between the fault occurrence and the length of identifiers, and the relationship between the fault occurrence and the word in identifiers by the random forest technique. The result of the experiment showed that the length of identifiers can predict the fault-proneness of the software modules. Also, the result showed that the word occurrence model is as good a measure as traditional CK and LOC metrics models.

Keywords: Fault-Prone Module Prediction, Random Forest, Software Engineering, Source Code Identifier, Word Occurrence Model

1. INTRODUCTION

Faults in software are not evenly distributed to the modules but are concentrated to specific modules (Hata, Mizuno, & Kikuno, 2012). Many studies pointed out that the 20% of modules include 80% of faults (Fenton & Neil, 1996, 1999). We call such fault-injected modules as fault-prone modules. This fact indicates that if we can predict fault-prone modules correctly, the efficiency of testing improves more. For example, Khoshogoftaar said that if we can

DOI: 10.4018/ijsi.2015010103

Copyright © 2015, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.
predict fault-prone module, we can reduce cost on software testing by half (Khoshgoftaar & Seliya, 2004).

As for the measures for fault-prone module detection, various metrics have been proposed so far (Hata et al., 2012). For example, CK metrics in Object-oriented code (Briand, Melo, & Wust, 2002; Gyimóthy, Ferenc, & Siket, 2005), process metrics (Ostrand, Weyuker, & Bell, 2005), software structures (Graves, Karr, Marron, & Siy, 2000), the metrics from static analysis (Nagappan & Ball, 2005; Zheng et al., 2006), and other metrics.

Faults are injected in software for various reasons. For predicting faults, it is necessary to make an assumption on the relationship between the cause of faults and the existence of faults. In this study, we made an assumption that the semantics in the software module has an effect to the fault existence. Semantics in a software module can be extracted from identifiers in the module.

Identifiers used in source code, such as names of class, method, function, and variable, can provide important information for readability and understandability with meanings of words in identifiers (Deissenboeck & Pizka, 2006). The length of identifiers, the way of naming, and the way of abbreviations of identifiers thus affect to the quality and understandability of source code (Butler, Wermelinger, Yu, & Sharp, 2009; Lawrie, Morrell, Feild, & Binkley, 2006).

Yamamoto et al. adopted the number of variable names as predictor variables of fault density prediction models (Yamamoto, Kamei, Matsumoto, Monden, & Matsumoto, 2009). Their approach used identifiers but did not use words in identifiers. It can be said that identifiers are one of the most important elements of the source code, and thus useful information can be obtained by analyzing the identifiers.

To investigate, we built a model to determine faulty modules using a machine learning technique from the number of occurrences of the identifiers. Using the fault-prone module detection model, we can investigate the relationship between the length of identifiers and software faults in a software module. For further investigation, we analyze the relationship between occurrence of “words” in identifiers and the existence of faults.

The rest of this paper is organized as follows: In Section 2, we describe an importance of identifier. In Section 3, we state the research questions in this study. In Section 4, we describe the first experiment to investigate the relationship between the length of identifiers and fault-proneness. We then describe the second experiment to investigate the relationship between word occurrence and fault-proneness in Section 5. In Section 6, we discuss the result of experiments and find the answer to the research questions. Finally, we conclude this paper in section 7.

2. IMPORTANCE OF IDENTIFIERS

It is said that nearly 70% of the source code of a software system consists identifiers (Deissenboeck & Pizka, 2006). Hence, we expect that the identifiers are key aspects in the source code analysis. Other research such as (Chen, Thomas, Nagappan, & Hassan, 2012) also focuses on the identifiers of source code.

Identifiers are a string of a name of function or variable except reserved words and literals. Usually, programmers often generate identifiers by a combination of words, such as “getPassword”, “doCalculationProcess”, and so on. A function name by a combination of words that adequately describe the contents of the function can be understood without reading the contents of the code and annotation (comments). When we look at the function call in the code, it is possible to understand what the function does without looking at the body of the function if the function name is acceptable. In cases of variables, with a variable name by combining the words that describe the contents of the stored value and type information, developers can easily understand the meaning and role of variable. In this way, identifiers have useful information to understand the contents of source code as well as the comment lines.
Web Services Description and Discovery for Mobile Crowdsensing: Survey and Future Guidelines
www.igi-global.com/article/web-services-description-and-discovery-for-mobile-crowdsensing/178563?camid=4v1a

Construction of Situational Information Systems Management Methods
www.igi-global.com/article/construction-situational-information-systems-management/70926?camid=4v1a