Tracking Code Clone for Software Traceability and Quality

Shinji Kawaguchi
Nara Institute of Science and Technology, Takayama-cho 8916–5, Ikoma-shi, Nara, Japan
{kawaguti@is,iida@itc}.naist.jp

Abstract

As one of the major problems in software maintenance, code clone, a pair of duplicated code in a source code, is well known. For software improvement, it is valuable to track code clone which has potential risk. This is because to determine how to treat code clones needs deep insight for the code clones. However, such code clone tracking is hard task especially for fresh developer. To support tracking code clones, we have proposed code clone history detection method. Code clone history reflects various manipulations to code clones including introduction, deletion, or just leaving unchanged. It also contains information about creator or modifier of code clones, and time of each manipulation. This means that tracking code clone history reveals many results of developer’s decisions about code clones. Thus, we assume various clues to improve software quality can be retrieved from tracking information of code clones.

1 Introduction

One of the major problems in maintenance phase is duplicated similar codes in program. Such duplicated code is called code clone (hereinafter referred to as just “clone”). Spread of clone makes software maintenance costly because if the defect was found in the code which has many clones, all of the clones must be inspected one by one and fixed when needed. Such operation is incredibly difficult for a large-scale software system. To address the problem, many methods are proposed to detect clone automatically and some of them are turned into actual utilization [1]. These clone detection system enables to automatically detect clones from huge source code.

However, there are various kinds of clones and all of them are not considered harmful. Some clones are duplicated business logic that should be consolidated into one function. Some clones are just algorithmic idioms and not harmful. Kapsar et al. [2] categorize clones into eight patterns and argued their pros and cons. Thus, when someone judges clones should be consolidated into one function or not for software quality improvement, he need to check why they are created and whether the reason is still reasonable. It is not very easy for fresh developer. Especially, in globally distributed environment, the clones may be created in another company. In such situation, investigating the clones is big hurdle. As a result, many harmful clones remain and they affect bad influence for software quality.

To investigate past state of clones, we proposed clone history retrieval method which analyzed source codes stored in configuration management system and specified where the clones were at any time in the past [3]. Our method enables tracking individual clones and analyzing when the clones were created and how the situation at the time was.

Moreover, clone history, the result of clone tracking, consists actual edit record made by developers. In other word, clone history reflects actual decisions of developer such as which clone was deleted and which one was left unchanged. Analyzing clone history enables tracking evolutions of clones individually. Such information is important clue for detecting potentially risky code and useful for improving software quality.

2 Code Clone History

Fig.1 shows the analysis model of clone history. At first, we retrieve source code in the past with certain intervals and apply a clone detection method for them. Then we analyze clone history relationship. At first, for each clone detected at time \( t \), we analyze where it was at time \( t - 1 \). If the correspondence clone is found, we consider they have a clone history relationship. In the same way, we analyze clones at time \( t - 2, t - 3, \ldots \) iteratively. Resulting chains of the relationships represent code clone history. Detailed definition is described in [3].

Clone history relationship is essential for tracking clone changes. It reveals when clones were copied, who copied, what files were changed at the same time, and so on. Such information would help fresh developer to investigate clones. In addition, summary of these results answers
such questions that who mostly copied the clones and when clones mostly increased.

3 Research Questions

Although we think tracking clone history is important, there are some questions to utilize them for improving software quality.

- How do we quantify and evaluate the code clone history? Especially, what kind of metrics does software quality relate?

- Can we assume whole history is the results of well-considered and right decision? If not so, we may need to distinguish right operations from the other operations somehow.

- What kinds of operations can we classify? We observed that new clones are not always created by copy and paste. Some clones are created by source code unification. Unification means assorting different style but similar logic source codes into similar style.

- What developers’ attributes do affect for clone handling?

- Is developer consistent during development? If not, how does the difference affect for clone handling? How to consider such effect in analyzing clone history?

4 Conclusion

In this paper, we mentioned about importance and usefulness of tracking clone history and briefly stated our clone history relationship model. In addition, we enumerated some research questions for enforce our research. However there are some obstacles, we think traceability of code clones is promising for improving software quality.

Acknowledgement

This work is supported by the Comprehensive Development of e-Society Foundation Software program of MEXT and Stage Project, the Development of Next Generation IT Infrastructure, supported by MEXT. And this work is supported by JSPS KAKENHI(18800024).

References

