Electronic Communications of the EASST
Volume 63 (2014)

Proceedings of the Eighth International Workshop on Software Clones (IWSC 2014)

Towards an Analysis of Who Creates Clone and Who Reuses it

— Position Paper —

Takuya Moriwaki, Yuki Yamanaka, Hiroshi Igaki, Norihiro Yoshida, Shinji Kusumoto and Katsuro Inoue

5 pages

Guest Editors: Nils Göde, Yoshiki Higo, Rainer Koschke
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Towards an Analysis of Who Creates Clone and Who Reuses it

Takuya Moriwaki[1], Yuki Yamanaka[1], Hiroshi Igaki[1], Norihiro Yoshida[2], Shinji Kusumoto[1] and Katsuro Inoue[1]

[1] m-takuya@ist.osaka-u.ac.jp
[2] y-yuuki@ist.osaka-u.ac.jp
[3] igaki@ist.osaka-u.ac.jp
[5] kusumoto@ist.osaka-u.ac.jp
[6] inoue@ist.osaka-u.ac.jp
Graduate School of Information Science and Technology
Osaka University, Japan

[4] yoshida@is.naist.jp
Graduate School of Information Science
Nara Institute of Science and Technology, Japan

Abstract: Code clone analysis is valuable because it can reveal reuse behaviours efficiently from software repositories. Recently, some code reuse analyses using clone genealogies and code clones over multiple projects were conducted. However, most of the conventional analyses do not consider the developers' individual difference to reuse behaviors. In this paper, we propose a method for code reuse analysis which takes particular note of the differences among individuals. Our analysis method clarifies who reused whose source code across multiple repositories. We believe the result might provide us with constructive perceptions such as characteristics of reused code itself by multiple developers, and developers who implement reusable code.

Keywords: source code reuse, code clone, clone evolution, code reuse analysis

1 Introduction

In the software industry and OSS(Open Source Software) projects, it is said that adequate code reuse could improve productivity and reliability of software development, and reduce development time[Lim94]. Though adhoc source code reuse might cause deterioration of software quality, in some cases, source code reuse seems to be a reasonable decision[KG08]. In order to encourage such source code reuse, it is meaningful to analyze developers' reuse behaviors in existing projects.

Code clone detection is widely regarded as the promising approach to identify source code reuse[HDG+11]. Meanwhile, there exist few analyses in consideration of the developers' individual differences to reuse behaviors. Sojer et al.[SH10] indicated the importance of code reuse, and heterogeneity in developers' reuse behavior through a survey in OSS projects.

Balint et al.[BGM06]and Harder[Har13] analyzed the authorship of the code clones to extract code reuse behaviors of each developer within a project. Especially, Harder[Har13] reports

differences between single-author and multi-author clones.

Conventional research implies there might exist differences between developers in the reuse behaviors. Therefore, in this paper, we propose a source code reuse analysis for each developer over two or more projects in the viewpoint of specifying individual differences in ways of reusing source code.

Our analysis method might promote interaction between developers over reused code, and the result enables us to recommend reusable code using source code reuse behaviors. Furthermore, we hope our research motivates the code clone community to think twice about an individual developer.

2 Source Code Reuse Behaviors

2.1 Definition

We define the source code reuse behaviors utilizing the authorship of the clone set.

We call one of the code fragments which is implemented first in a clone set “original code” of the clone set, and call the developer of the original code “clone set authorship”. Furthermore, we call developers who implemented the code fragments other than the original code in the clone set as “clone set users”. Generally, the authorship of source code fragments can be identified using the blame command in VCS(Version Control System).

The source code reuse behavior is defined as the relation among the authorship and the users of each clone set. In our definition, we regard authorship-user relationships in clone sets as create-reuse relationships in the source code reuse behaviors.

2.2 Individual Differences

We conducted a miniature case study about reuse behaviors in a part of an OSS Java project (eclipse.platform.text). We generated a set of change evolutionary history (clone genealogy) from a part of the project repository based on the model of genealogy of code clones[KSNM05]. The repository contains 826 rev. and 16 developers. We adopted CCFinder[KKI02] to generate the clone genealogy of the repository.

In this case study, we detected 443 clone sets. Out of them, we analyzed the authorships of 34 clone sets which are classified into the evolutionary pattern ”ADD” (i.e., the clone sets include code clones which are reused in different revision after creation). Detail steps for identifying the authorship of the clone sets are described in the Section 3.

Figure 1 indicates the number of times which developer A∼F committed, created, and reused in the project. Developers who committed below 10 times are omitted. There is little relationships among #commit, #creation, and #reuse as evidenced by the figure. One developer reuses actively and is reused frequently, and another never reuses the code. In this figure, though #commits by developer A is more than twice than B, #create and #reuse by A are fewer than half the number of them by B. The result shows that there might exist individual differences in reuse behaviors among developers.

Adhoc reuse of source code is usually described to cause quality reduction of software. On the other hand, in some situations, code duplication seems to be a reasonable design option [KG08].

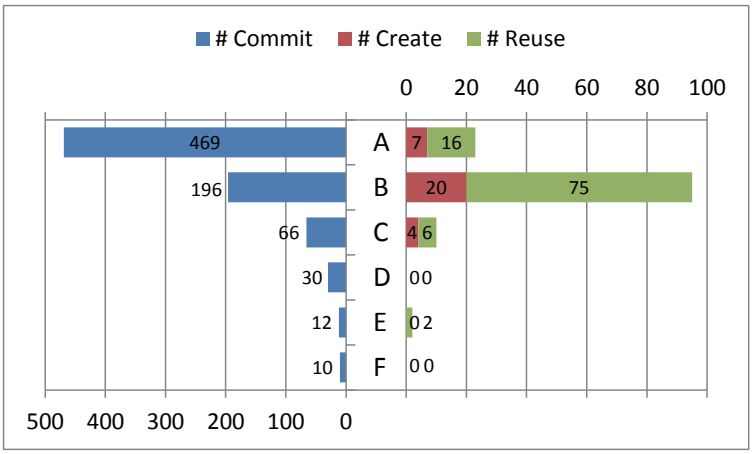

Figure 1: Individual Differences in Source Code Reuse Behavior

For example, developers reuse more existing code the less mature their projects [SH10]. In the survey by Sojer et al.[SH10], the perception of each developer to reuse behavior seems to affect his/her actual reuse.

We propose an analysis method for identifying reuse behaviors by each developer over multiple repositories. Feedback of the analysis results might motivate each developer to decrease harmful reuse and increase beneficial reuse.

3 Source Code Reuse Analysis over Multiple Repositories

Our analysis method takes repositories of multiple projects as an input and provides source code reuse behaviors for each clone set as an output. Detailed analysis steps are as follows.

Step1: Merges commit logs of multiple repositories

In VCS, each repository has a set of commits which indicates activities in configuration management by developers. In this step, commit logs collected from multiple repositories are concatenated into one commit log, and the commits in the concatenated log are sorted by date committed. The concatenated commit log represents development histories of the composite repository containing multiple repositories virtually.

Step2: Classifies Clone Sets

In this step, the clone genealogy of the composite repository is generated based on the model by Kim et al.[KSNM05]. Concretely, every revision of the composite repository is checked out based on the concatenated commit log first. Next, we adopted the Clone Notifier[YCY+13] to generate the clone genealogy. The generated clone genealogy of the composite repository classifies evolution patterns of all clone sets in each revision according to the model of the genealogy

such as “Same” and “ADD”. Here, “Same” means all code snippets in a clone set did not change between adjacent revisions, and “ADD” means at least one code snippet was newly added to the clone set in the latter revision[KSNM05].

Step3: Identifies Authorship and Users of Clone Sets

First, code authorship of every code snippet in the classified clone sets is identified by using the blame command in VCS. Here, the code authorship means who created the code snippet. Next, for each clone set, our analysis method specifies the first code snippet which was implemented. In our definition, the first code snippet is the original code of the clone set. We regard the code authorship of the first snippet as the clone set authorship. The code authorship of the snippets other than first one indicates the clone set users.

The above mentioned steps clarify the authorship and users for each clone set.

4 Discussion and Conclusion

We have proposed source code reuse analysis over multiple projects taking into consideration difference among developers. In our assumption, our analysis method promotes inter-developer relationships, in the viewpoint of the source code reuse. For example, developers within a team using our analysis know who reused whose code snippets mutually. Such mutual understanding about reuse behaviors might motivate developers to look back on their source code reuse.

On the other hand, our method contains the following limitations on threats to validity.

Multiple account names are used by the same developer

Developer might use different account names for different projects. Blame command in VCS adopts account names to identify code authorship. If a developer uses multiple account names, create-reuse relationships are identified incorrectly. However, we assume our analysis is used for projects within the specific community whose members know their account names each other. As a consequence, multiple account names by one developer might not cause severe problems.

Multiple developers implement one original code

Blame command represents code authorship for each line. If there exist code snippets as the original code which were implemented by multiple developers, our analysis method provides multiple developers as the clone set authorship. Currently, we admit only one developer as the clone set authorship who developed most SLOC(Source Lines of Code) in the original code.

Developers reuse code fragment from some code base that is not part of the analysis

Developers might reuse code fragment from some code base which does not belong to their projects. In such a case, our analysis method cannot identify the original code fragment. However, if the code fragment has value to multiple developers, we expect the fragment to be reused by developer two or more times in the project.

For future research, we are planning to evaluate the validity of our analysis method based on some large-scale experiments. Moreover, by using the result of our analysis, we try to develop a source code reuse promoting system which motivates interaction between a creator and a user of each clone set.

Acknowledgment

This work was supported by MEXT/JSPS KAKENHI 24700030, 25220003.

Bibliography

[BGM06] M. Balint, T. Girba, R. Marinescu. How Developers Copy. In *Proc. of 14th IEEE International Conference on Program Comprehension(ICPC) 2006*. Pp. 56–68. 2006.

[Har13] J. Harder. How multiple developers affect the evolution of code clones. In *29th International Conference on Software Maintenance*. 2013.

[HDG+11] L. Heinemann, F. Deissenboeck, M. Gleirscher, B. Hummel, M. Irlbeck. On the Extent and Nature of Software Reuse in Open Source Java Projects. *Top Productivity through Software Reuse*, pp. 207–222, 2011.

[KG08] C. Kapser, M. W. Godfrey. "Cloning considered harmful" considered harmful: patterns of cloning in software. *Empirical Software Engineering* 13(6):645–692, December 2008.

[KKI02] T. Kamiya, S. Kusumoto, K. Inoue. CCFinder: A Multi-Linguistic Token-based Code Clone Detection System for Large Scale Source Code. *IEEE Transactions on Software Engineering* 28(7):654–670, 2002.

[KSNM05] M. Kim, V. Sazawal, D. Notkin, G. Murphy. An Empirical Study of Code Clone Genealogies. In *Proceedings of the 10th European Software Engineering Conference Held Jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ESEC/FSE-13, pp. 187–196. 2005.

[Lim94] W. Lim. Effects of reuse on quality, productivity, and economics. *IEEE Software* 11(5):23–30, 1994.

[SH10] M. Sojer, J. Henkel. Code Reuse in Open Source Software Development: Quantitative Evidence, Drivers, and Impediments. *Journal of the Association for Information Systems* 11(12):868–901, 2010.

[YCY+13] Y. Yamanaka, E. Choi, N. Yoshida, K. Inoue, T. Sano. Applying clone change notification system into an industrial development process. In *Proc. of 2013 IEEE 21st International Conference on Program Comprehension (ICPC)*. Pp. 199–206. May 2013.