Changes to Code Clones in Evolving Software

Jens Krinke
FernUniversität in Hagen, Germany
krinke@acm.org

1 Introduction

Although cut-copy-paste (-and-adapt) techniques are considered bad practice, every programmer is using them. Because such practices not only involve duplication but also modification, they are called *code cloning* and the duplicated code is called *code clone*. A *clone group* consists of the code clones that are clones of each other. Code cloning is easy and cheap during software development, but it makes software maintenance more complicated, because errors may have been duplicated together with the cloned code and modifications of the original code often must also be applied to the cloned code.

There has been little empirical work that checks whether the above mentioned problems are relevant in practice. Kim et al. [2] investigated the evolution of code clones and provided a classification for evolving code clones. Their work already showed that during the evolution of the code clones, common changes to the code clones of a group are fewer than anticipated. Geiger et al. [1] studied the relation of code clone groups and change couplings (files which are committed at the same time, by the same author, and with the same modification description), but could not find a (strong) relation. Therefore, this work will present a study that (in)validates the following hypothesis:

> *During the evolution of a system, code clones of a clone group are changed commonly.*

Of course, a system may contain bugs like that a change has been applied to some code clones, but has been forgotten for other code clones of the clone group. For stable systems it can be assumed that such bugs will be resolved at a later time. This results in a second hypothesis:

> *During the evolution of a system, if code clones of a clone group are not changed commonly, the missing changes will appear in a later version.*

This work will validate the two hypotheses by studying the changes that are applied to code clones during 200 weeks of evolution of two open source software systems.

2 Experiment Setup

For the study the version histories of two open source systems have been retrieved: The first system is `jhotdraw`. It is a drawing application and framework written in Java. Its goal is the demonstration of good use of design patterns. Its version archive is available via CVS and for the study, the `jhotdraw6` CVS-module has been used. The second system is a subsystem of `eclipse`. Its version archive is also available via CVS and for the study, the `org.eclipse.jdt.core` CVS-module has been used.

For both systems, the sources have been retrieved based on their status on 200 different dates, such that each version is exactly one week later or earlier than the next or previous version. For both systems, only the Java source files have been analyzed. Also, the source files have been transformed to eliminate spurious changes between versions: Comments have been removed from the sources and afterward, the source files have been reformatted with the pretty printer *Artistic Style*. The transformed sources are saved to a repository.

The changes between the versions of the system have been identified by the standard *diff* tool. For each version v of the analyzed system, the changes between version v and $v + 1$ (the version of the next week) have been identifed. Also, the changes between a version in week v and and in five weeks later ($v + 5$) have been identified. For each of the 200 versions, the clone groups have been identified by the use of the clone detection tool *Simian* from RedHill Consulting Pty. Ltd.

The analysis has been done on 195 version because the versions of the last five weeks were used only for the generation of the changes between the versions. For each week w, $0 \leq w < 195$, the tool has generated the list of clone groups with common and non common changes, based on the clone groups of the analyzed system in week w and the changes from week w to week $w + 1$ (and to week $w + 5$).

3 Results

This section will present the results of the study as described in the previous section. It will first describe the results for `eclipse` and `jhotdraw` independently and will then present common results together with the effects of comment removal. Also, the part of the study that validates the second hypothesis is presented.

3.1 Results for `eclipse`

Most of the times when changes occur to clone groups, only one or two clone groups have non common changes. It is also rarely the case that during one week more than three clone groups are affected by any kind of change. Table 1 shows the accumulated numbers for all 198 weeks.

	eclipse		jhotdraw	
	1 week	5 weeks	1 week	5 weeks
clone groups with non common changes	105	188	30	41
clone groups with common changes	79	135	19	20

Table 1: Comparison for non common changes during one and five weeks

This reveals that in the `org.eclipse.jdt.core` system, clone groups have more often non common changes than common changes. This suggests that if a code clone is changed, it is more often adapted in the environment it is used in, than it is modified in the same way like the other code clones in its group.

3.2 Results for `jhotdraw`

The development of `jhotdraw` was much less active and happened in bursts. For only 21 versions occur changes to the clone groups, half of them due to irrelevant changes. The three bursts occur in week 76, 141, and 182. The burst in week 76 shows an interesting behavior: Although 16 clone groups are affected by changes, only two are affected by non common changes. A manual inspection of the changes revealed that during that week, a massive code cleanup has been applied to the source code with a lot of small refactorings. Again, non common changes to more than two clone groups are rare.

The numbers in Table 1 are similar to the numbers from `eclipse`. Again, clone groups have more often non common changes than common changes. This supports the assumption that if a code clone is changed, it is more often adapted in the environment it is used in, than it is modified in the same way like the other code clones in its group.

3.3 Evolution of Changed Clone Groups

Up to now, the presented results only gave evidence for the validation of the first hypothesis. The second hypothesis "*During the evolution of a system, if code clones of a clone group are not changed commonly, the missing changes will appear in a later version*" has also been validated within this study. If this hypothesis is true, changes to a clone group that are not applied to all code clones of a group will appear in a later version. The study only considered the changes to a system that appear within one week. If the hypothesis is true, there have to be more common changes if a longer time period is considered. To validate this, the study has been repeated with a time distance of five weeks instead of one as it is assumed that missed changes to a clone will be detected and fixed within four weeks.

The study generated the changes that appear within the next five weeks for the state of the software system at 195 weeks (0 to 194). Table 1 shows the numbers of changed clone groups within one week and the number of changed clone groups within five weeks. However, for the five weeks comparison, only the weeks were non common changes occurred have been used, i.e. the weeks where no (relevant) changes or common changes occurred within one week were ignored for the five week comparison.

The numbers show that during the five week period much more changes occur than within only the first week. Moreover, the non common changes increase much more than the common changes. This indicates that the second hypothesis is not valid. If it would be valid, the common changes would increase much more. However, there are exceptions to this general observation as manual inspection revealed: For example, in week 125 in `eclipse` six clone groups are changed in a non common way within the following week. Within the next four weeks, four of the of the six clone groups receive additional changes such that they are now changed in a common way during the five week period. In weeks 19, 59, 118, and 144 the changes are similar: two (for week 144, three) clone groups that have non common changes within one week have only common changes within five weeks. For `jhotdraw`, manual inspection did not reveal such behavior.

These observations suggest that the second hypothesis occurs in practice, but not very often. Moreover, during a longer period of observed time, many more non common changes to clone groups appear, such that occurrences of changes that change non common changes to common changes are hard to identify.

4 Conclusions

The study showed that the hypotheses are not valid generally. The study showed that clone groups more often have non common changes than common changes, invalidating the first hypothesis. The study also showed that the second hypothesis is only valid partially, because the non common changes appear much more often. However, a small amount of such changes that turn non common changes to a clone group to common changes in a later version has been observed.

References

[1] Reto Geiger, Beat Fluri, Harald C. Gall, and Martin Pinzger. Relation of code clones and change couplings. In *Proceedings of the 9th International Conference of Funtamental Approaches to Software Engineering (FASE)*, pages 411–425, March 2006.

[2] M. Kim, V. Sazawal, and D. Notkin. 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)*, pages 187–196, 2005.