Stability of COBOL Clones

Jan Harder, Nils Göde
University of Bremen
Bremen, Germany
{harder, nils}@informatik.uni-bremen.de

Marcus Rausch
Debeka-Group
Koblenz, Germany
Marcus.Rausch@debeka.de

Abstract

Code clones are said to threaten the maintainability of software systems. Changes to one cloned fragment must be propagated to the other fragments, which might increase change effort. Hence, different studies have investigated the stability of cloned code and compared it to the stability of the non-cloned code. However, only open-source systems have been regarded so far. We conducted a study on two industrial COBOL systems, measuring the clone stability. In this paper, we present the results and compare them to the observations of the previous studies.

1 Introduction

Copy&Paste is an indispensable strategy in programming. Its use, however, leads to duplicated passages of source code—clones—that might decrease the maintainability of the code base. Changes applied to a copied passage likely need to be propagated to the other occurrences, increasing the change effort.

Different studies compared the stability of cloned and non-cloned code, that is, the likelihood of these parts of the code to change. If cloned code were less stable than non-cloned code, then the costs of maintaining it would be higher. With his study on five open-source systems, Jens Krinke was the first to go into the matter [3]. He concludes that—contrary to the initial assumption—clones are even more stable in general. However, if only deletions of code are regarded, clones are less stable.

We varied and extended his study, using a more accurate token-based method and a more fine grained stepping between the versions analyzed [1]. Our results support Krinke's findings. We also investigated the impact of different definitions for a clone (changing the similarity threshold and size of the clones) and found smaller clones to be more stable than larger ones. Furthermore, we observed that exact clones are less stable than clones with slight differences.

The impact of four different clone detection tools was measured in a third study [2]. The authors come to the same conclusion that cloned code is more stable in general. Nevertheless, they found that the relation of the stabilities may be converse in early development stages.

So far, all studies were limited to open-source systems, which are developed in an open process and exposed to permanent public review and discussion. It has not been investigated, if the properties found in the open-source systems also apply to industrial ones, which are developed under different circumstances.

2 Case Study

In this paper we extend our previous study with the analysis of two productive COBOL systems.

2.1 Subject Systems

The two industrial systems are developed by the Debeka-Group, one of the top-ten companies in Germany's insurance and home-savings business. Debeka offers a variety of insurance and financial services. The systems we have analyzed are written in COBOL and have been in use for more than ten years. System *lv-la* (385 KSLOC) is dedicated to life insurance whereas *zw-wb* (585 KSLOC) provides functionality for commission calculation, which is a common concern to different services. For each system we analyzed all versions that went productive within a time period of at least two years—in both cases more than 500 versions.

2.2 Clone Stability

The stability of source code can be measured by its inverse: the *instability*. To measure instability, a lexical analysis is performed. For each token in each version of the analyzed system, we can tell whether it is part of a clone or not, applying clone detection. We can furthermore detect whether and how it was changed from the previous version using a token-based diff. The instability is now defined as the number of tokens changed divided by the total number of tokens. In all versions, this measurement can be applied to the cloned part of the code as well as to the non-cloned part. Both values can be compared to determine which part exposes higher instability.

We measure the instability separately for additions, deletions and modifications of tokens. This way we calculate six instability values—one for each change type in cloned and, respectively, non-cloned code. As an example, ι_{AC} denotes the instability (ι) of cloned code (C) in respect to additions (A). Analogously, we define ι_{AN}, ι_{DC}, ι_{DN}, ι_{MC}, ι_{MN}, where N stands for non-cloned code. For an extensive description of our method, please refer to our previous publication [1].

2.3 Study Procedure

Using our incremental clone detector *iclones*, we identify clones in all versions. The detection of changes is embedded in this process. Two preprocessing steps are applied to the COBOL code prior to the clone detection: First, generated code is removed. Second, to avoid that reported clones cross syntactic boundaries, special delimiters are inserted between procedures.

We used the following parameters to the clone detector: The minimum clone length is set to 50, 100, and 150 tokens. The type parameter has two variations so that only exact clones respectively clones with different variable names or literals or gaps are detected. The three length settings and the two type settings multiply to six combinations per system.

2.4 Results

As in our previous study, we first investigate the stability for one clone detector setting in detail. Then we evaluate the impact of different parameter settings.

2.5 Stability

The instability values for exact clones with a minimum token length of 100 are shown in Figure 1. Each bar denotes the percentage of tokens that were affected by the respective change type in the cloned or non-cloned part of the code. The leftmost bar, for example, indicates that 0.00578% of the cloned tokens were added during the analyzed period. The instability of the non-cloned code is always higher for the COBOL systems, resembling previous findings.

ArgoUML is a system, we analyzed in our previous study. Its stability values are provided for comparison to the industrial systems. There are two notable differences: First, the distance between the instabilities of cloned and non-cloned code is bigger for the COBOL systems. That is, COBOL clones are much more stable compared to non-cloned code as in the open-source systems. The difference is caused by the high instability of non-cloned COBOL code, while the instability values for the clones do not differ significantly among the three systems. The higher relative stability of COBOL clones could be caused by the industrial environment and the programming language. Developers reported to us that they tend to refrain from changing code that is known to work. Instead it might be preferred to create a copy if the same code is needed for another program. The second difference in the results is, that regarding deletions, clones in the COBOL systems are also more stable than non-cloned code. This contrasts to the properties of all open-source systems that have been analyzed so far.

2.6 Parameter impact

In the open-source systems we observed that the parameter settings have an impact on the stability values. This is also true for the COBOL programs, but only to a lower extent. In the industrial code, we also found the tendency that exact clones expose less stability than clones with differences. Nevertheless, some properties that are identical among the open-source systems differ among the COBOL ones. In *lv-la*, smaller clones are more stable than larger ones; but in *zw-wb*, the opposite is the case. These differences between the systems are only small, though.

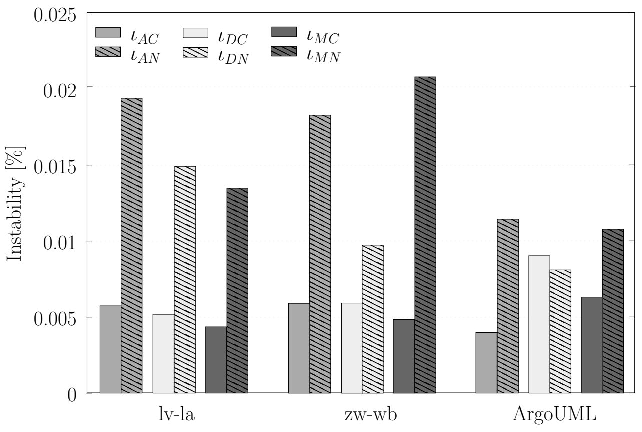

Figure 1: Instability of source code

3 Conclusion

Our results show that the general conclusions of previous studies also apply to industrial COBOL systems. Most important, cloned code is more stable than non-cloned code. The stability of cloned code, however, is more distinctive than in the open-source systems—the COBOL clones are even more stable. Reasons for this could be the maturity of the code and the special development process which leads to duplicates that are not meant to change in the future. The exceptional instability to deletions, that we observed for clones in open-source systems, could not be found in the industrial programs. Despite these slight differences, the conclusion remains that the presence of clones does not decrease stability in general. Nevertheless, clones still may decrease maintainability by other means, like increased comprehension costs or inconsistent changes that introduce bugs.

References

[1] N. Göde and J. Harder. Clone Stability. In *Proc. of the CSMR '11*, pages 57–66. IEEE.

[2] K. Hotta, Y. Sano, Y. Higo, and S. Kusumoto. Is duplicate code more frequently modified than non-duplicate code in software evolution?: an empirical study on open source software. In *Proc. of the IWPSE-EVOL '10*, pages 73–82. ACM.

[3] J. Krinke. Is Cloned Code more stable than Non-Cloned Code? In *Proc. of the SCAM '08*, pages 65–74. IEEE.