What Clone Coverage Can Tell

Nils Göde, Benjamin Hummel, Elmar Juergens
CQSE GmbH, Germany
{*goede, hummel, juergens*}@*cqse.eu*

***Abstract*—In research and practice there is a desire to express the amount of cloning in a system in a compact form and compare different systems with each other. A popular choice for doing so is to use the clone coverage—the percentage of source code being part of at least one clone. However, the clone coverage is strongly influenced by the parameters used for clone detection and the peculiarities of each system. In this paper, we summarize certain pitfalls and argue that clone coverage values should be interpreted with care.**

***Keywords*-Software quality, code clones, clone detection**

I. Introduction

When it comes to analyzing code clones in a system, we often want to summarize the cloning situation in a very compressed form. The easiest way to compare the severity of cloning in different systems is to use a single value that expresses the overall extent of cloning for a given system. A popular choice to boil down the cloning situation to a single number is the *clone coverage*—the percentage of the source code that is part of at least one clone. It can be interpreted as the probability that a randomly selected source code statement is cloned.

The clone coverage has been used since the early days of clone research [1] and is now established as one of the standard cloning metrics. It is tempting to use the clone coverage to compare cloning in different systems and interpret higher clone coverage as "the system has more problems with cloning". However, we observed two major problems regarding the clone coverage. First, it strongly depends on the clone detection parameters. Small changes in the parameters may already have a strong influence on the clone coverage. Consequently, comparing systems requires the same choice of clone detection parameters for both systems. Second, even if the parameters are identical, each system has a unique structure, history, and development process. These peculiarities may also have a significant influence on the clone coverage. In this paper we demonstrate how clone detection parameters and systems with different characteristics influence the clone coverage.

II. Clone Detection Parameters

In this section we demonstrate that the parameters used for clone detection have a strong influence on the clone coverage. We selected three prominent parameters which are the minimum length of a clone, the exclusion of generated code (which can be regarded as a boolean parameter), and whether identifiers and literals are normalized or not. Figure 1 illustrates how the clone coverage changes for different combinations of these parameters when detecting clones within an industrial C/C++ system with 1400 KLOC (600 KLOC when generated code is excluded).

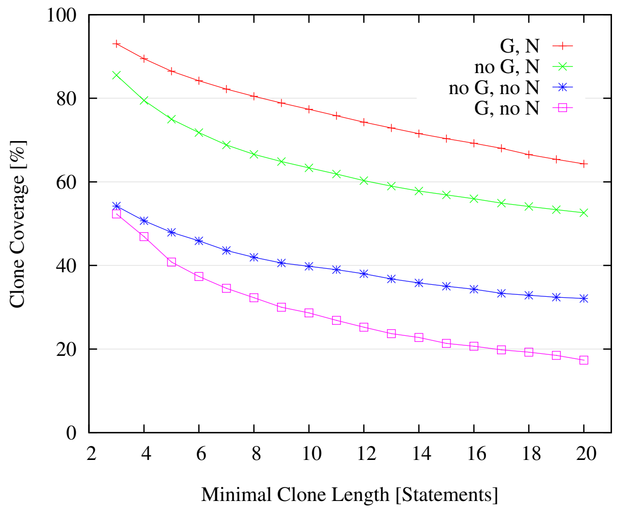

Figure 1. Clone coverage using different parameters (G = generated code is included, N = identifiers and literals are normalized)

The numbers show that we can achieve a clone coverage anywhere between 19% and 92% depending on which parameters we choose. Even if the minimum clone length is fixed, changing the other two parameters can cause the clone coverage to change by up to 50%.

In some cases, the effect of changing a parameter is not immediately obvious. In our example, exclusion of generated code decreases the clone coverage when normalization is applied but increases the clone coverage when there is no normalization. The reason is that, while the generated code is highly stereotypical, the exact identifiers used differ. Without normalization all the generated code looks different, reducing the overall clone coverage when including it.

From these results we conclude that comparing the clone coverage of different systems requires the same set of parameters to be used for detecting clones in both systems. Small differences may already make the clone coverage incomparable. Unfortunately, many studies do not describe their choice of parameters in sufficient detail to allow a

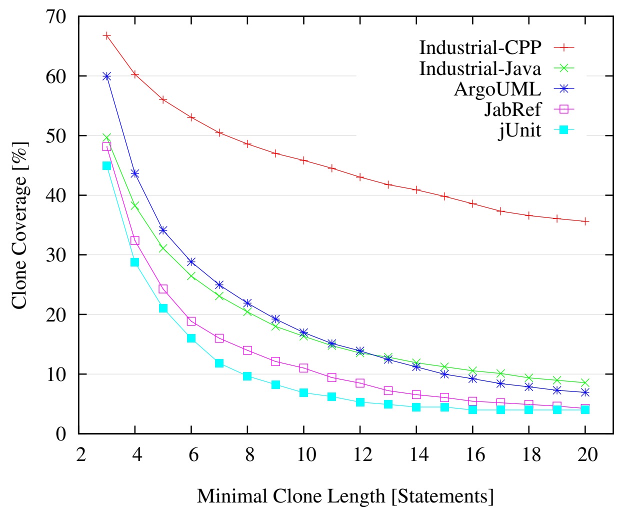

Figure 2. Clone coverage of different systems

comparison of the results. A precise description of the parameters would certainly increase the comparability of results from different studies.

III. System Characteristics

Assuming that the parameters used for clone detection are the same, systems may still not be comparable by their clone coverage due to their different structure and design. For example, if one system contains large amounts of generated code while the second does not, we might observe the first one as "better" or "worse" depending on whether we (consistently) exclude generated code or not. Similarly, other parameters used for clone detection might influence which system we perceive as "better". To illustrate this, Figure 2 compares the clone coverage for different systems and varying minimum length. All other parameters are fixed.

Not surprisingly, we can observe that the clone coverage of all systems decreases with increasing minimum clone length. We can also observe that, in most cases, the relative ordering of systems with respect to their clone coverage is stable. For example, jUnit always has a lower coverage than ArgoUML, independent of the clone length chosen. However, there is also the case where the relative order changes. ArgoUML has a higher clone coverage than the industrial Java system up to a minimum clone length of 12 statements. For a length of 13 and above, the clone coverage of the industrial system is higher. Hence, we conclude that using the clone coverage for comparing systems should be taken with care, as the outcome of the comparison heavily depends on the exact parameters used.

IV. Alternatives

Although comparing systems based on a single number will always be problematic, it might be reasonable to think about an alternative measure to quantify the extent of cloning. Some of these measures are built on the notion of a *redundancy-free size (RFS)*, which measures the hypothetical size of the system after perfectly removing from each clone all instances but one. The RFS allows to differentiate situations where there are few clones with many instances or many clones with few instances. For systems with identical clone coverage, the first situation would result in a smaller redundancy-free size than the second one.

Although the RFS allows more differentiation, it is to some degree related to the clone coverage (CC). Let S be the *size* of the system and $RS = S - RFS$ the *redundant size*. There are two extreme situations: either all redundancy is caused by many instances of a single small clone, then the RS will consist of nearly the entire region covered by clones ($RS = CC \cdot S$), or the redundancy is caused by neatly aligned clone pairs, which makes the RS consist of only half of this region ($RS = 0.5 \cdot CC \cdot S$). Hence,

$$0.5 \cdot CC \cdot S \leq RS \leq CC \cdot S,$$

which is equivalent to

$$0.5 \cdot CC \leq 1 - (RFS/S) \leq CC.$$

This inequality implies that the redundancy-free size is determined within certain bounds by the clone coverage. As a consequence, the extreme volatility of the clone coverage carries—at least to some degree—over to the redundancy-free size and derived metrics, such as the *clone overhead* defined as $(S/RFS) - 1$ [2], [3].

V. Conclusion

We conclude that despite its straightforward calculation and intuitive meaning, the clone coverage should always be interpreted with care. When comparing different systems based on the clone coverage, the same parameters should be used for clone detection. A comprehensible description of the parameters used would ease the comparison of results of different studies. In addition, the structure and design of both systems should be analyzed and considered in the comparison. In summary, we believe that comparing the cloning situation of different system requires an in-depth analysis of both systems and can hardly be done based on a single number.

References

[1] B. S. Baker, "On finding duplication and near-duplication in large software systems," in *Proceedings of the 2nd Working Conference on Reverse Engineering*. IEEE Computer Society, 1995, pp. 86–95.

[2] E. Juergens, "Why and how to control cloning in software artifacts," Ph.D. dissertation, Technische Universität München, 2011.

[3] E. Juergens and F. Deissenboeck, "How much is a clone?" in *Proceedings of the 4th International Workshop on Software Quality and Maintainability*, 2010.