Subjectivity in Clone Judgment: Can We Ever Agree?

Cory Kapser[1], Paul Anderson[2], Michael Godfrey[1], Rainer Koschke[3], Matthias Rieger[4], Filip van Rysselberghe[4], and Peter Weißgerber[5]

[1] University of Waterloo, David R. Cheriton School of Computer Science,
Waterloo, Ontario, N2L 3G1, Canada
{cjkapser,migod}@uwaterloo.ca

[2] GrammaTech, Inc.
317 North Aurora Street, Ithaca, N.Y., U.S.A. 14850
paul@grammatech.com

[3] Universität Bremen, Fachbereich 3 - Mathematik und Informatik,
Postfach: 330 440, 28334 Bremen, Germany
koschke@informatik.uni-bremen.de

[4] Universiteit Antwerpen, Departement of Mathematics and Computer Science,
Campus Middelheim, Building G, Middelheimlaan 1, B-2020 Antwerpen, Belgium
{Filip.VanRysselberghe,matthias.rieger}@ua.ac.be

[5] Universität Trier, Fachbereich IV - Informatik,
54286 Trier, Germany
cs@p.-weissgerber.de

Abstract. An objective definition of what a code clone is currently eludes the field. A small study was performed at an international workshop to elicit judgments and discussions from world experts regarding what characteristics define a code clone. Less than half of the clone candidates judged had 80% agreement amongst the judges. Judges appeared to differ primarily in their criteria for judgment rather than their interpretation of the clone candidates. In subsequent open discussion the judges provided several reasons for their judgments. The study casts additional doubt on the reliability of experimental results in the field when the full criterion for clone judgment is not spelled out.

Keywords. code clone, study, inter-rater agreement, ill-defined problem

1 Introduction

An ever-present problem in the analysis of "code clones" in software is the identification, or selection, of *false positives*—i.e., meaningless clones—while correctly matching clones in a set of detected results. A major problem with this task is that it is inherently (or at least presently) subjective. The lack of a formal definition of what constitutes a code clone—or at least a loosely agreed-upon on—there are major threats to the validity of the results we report in our research. This much was made plain during the Dagstuhl seminar on Duplication, Redundancy and Similarity of Software (DRASIS) [1]. In the meetings and talks

Dagstuhl Seminar Proceedings 06301
Duplication, Redundancy, and Similarity in Software
http://drops.dagstuhl.de/opus/volltexte/2007/970

that occurred throughout the workshop, it became clear that no such description exists, or can currently be agreed upon [2].

Given the uncertainty in a foundational definition of the area, we must ask an important question: without an agreed upon description of what constitutes a code clone, will we agree when classifying detected candidate code clones as a clone or not a clone? The answer to this question has far reaching consequences to the validity and reproducibility of clone detection and analysis results reported in the literature. A previous study has already indicated that clone detector experts can show worrying levels of disagreement, particularly when their judgment tasks are poorly defined [3].

This paper adds another data point regarding the disagreements that may exist within the community in the definition of software clones. It presents the results of a small study of expert judgments about code clones. The DRASIS seminar provided a unique opportunity to collect this information. During a working session at the seminar, eight experts (code clone researchers from around the world) provided their judgments on clone candidates. The study is described, results are summarized, and conclusions are drawn.

2 Study

The purpose of this working session was to measure the degree of agreement experts would exhibit when classifying segments of code as clones or not clones. A second aim was to elicit data about what judgments the experts formed and their reasoning behind their judgments.

2.1 Subjects, Apparatus, and Protocol

A convenience sample of 20 candidate clones was (semi-randomly) selected by the session leader from the clone detection results reported by the tool CCFinder [4]. The 20 candidate clones were selected from the Postgresql source code. These clones were detected by CCFinder using a minimum acceptable match length of 30 tokens. CCFinder uses a parameterized string matching method. This allows for a high degree of variability of the identifiers amongst the clone pairs. The candidates were displayed using CLICS, which simply highlights the regions of code that are said to be matched and does not annotate the results in any other way.

All eight attendees of the discussion group privately recorded their judgments. The experts were asked to independently judge whether each candidate should be considered as a clone according to their own definition of what a clone is, as this study was intended to measure how well researchers agree independently. The experts were instructed to simply note whether or not a candidate was a clone or not a clone, and were to told to optionally record a brief reason why they made this decision. Experts did not collaborate on their decisions. Care was taken not to bias the experts by discussing the definition of code clone before the study. As all participants were considered experts in the field, they should already have had a working definition they have used in their studies or evaluation of others work.

After the initial judgment round, each clone candidate was revisited and, as a group, we discussed the attributes that made us decide whether it was a clone or not.

2.2 Results of Independent Classification

Of the 20 candidates, only 50% had a classification that was agreed upon by more than 80% of the experts (meaning 7 or more persons agreed). If the standard for agreement was lowered to 75% of the experts agreeing (6 experts), then 65% of clones were agreed upon.

2.3 Discussion of Independent Classification Results

Many of the results reported in the current literature depend, to some extent, on subjective filtering of the clone candidates. The present result does little to dispel the question of whether such subjective clone judgment can be considered accurate and trustworthy, and puts in question the reproducibility of currently existing case studies.

3 Results of Open Discussion Session

Following the independent classification, the group discussed the decisions individual experts had made. Attention was focused on what attributes of the candidate clone contributed to the decisions, and what attributes caused the experts to disagree. From this discussion, it became apparent that all experts “saw” the same attributes of the clone, but interpreted their significance or meaning differently. The same features, such as the size or types of differences, lead to opposite conclusions. For example, idiomatic expressions caused some experts to consider a candidate to be a clone, yet caused other experts to consider the segments not a clone.

A short list of reasons was compiled as to why two segments of code may not be a clone (the reasons are listed as they were noted):

- the code is idiomatic
- the design force[sic] repeated solutions
- the expert would not have explicitly copied and pasted the code
- the candidate was too short
- types were changed
- there was no way to refactor the code
- semantics of procedures (represents different operator)
- main portion of the code is literals

- the code was BORING!!!

A similar list was compiled as to why two segments of code may be considered a clone:

- the code is idiomatic
- the code segments were very similar
- lots of changes but identifiers are very similar
- the expert would have copied and pasted the code
- the code is generatable
- context of the code segment made it a clone
- the structure was the same

The apparent lack of agreement amongst experts leads us to an important question: was rating code as a code clone or not a code clone an appropriate question? Would there be an observable difference if the question were to be phrased using terms such as duplicated code, redundant code, similar code, or code clone? One English definition of clone implies one segment of code is copied from another: does this unjustly bias the individual definitions used by the experts? Redundant code implies that the code is unnecessary, i.e., more than what is needed for the software system. But some code that is considered a code clone is necessary due to variety of constraints such as design, language or corporate divisions; this counters the definition of redundant. Duplicated code can lead us to think about merely copying and pasting, while similar code leaves us with a very ambiguous definition regarding what is similar. This led us to a very good discussion during the session summary, much of which is summarized in a separate paper [2]

3.1 Threats to Validity

Forcing a binary answer may exaggerate the difference in opinions. In a future study, a multi-point scale may be one method to be used in measuring how close experts are to agreement. Also, the visualization may have some effect on the result. Some experts may have noticed small details that others did not. In the follow up to this study, a more explicit visualization may be warranted to ensure that judges are basing decisions on the same information.

Another threat to the validity is the use of clone pairs rather than clone groups (more than two segments of code that have been matched). Experts opinions may have changed when made aware of the total number of segments that were actually reported to match a give segment of code.

4 Conclusions and Future Work

With no clear definition of what a code clone is, it is clear that experts cannot agree when subjectively identifying segments of code as code clones. This finding has a major impact on the reproducibility of results, and highlights the importance of clear documentation of the criteria for selection of clones when reporting studies. Without clear documentation of the criteria used, there is little chance that a reviewer can be sure of the true meaning or even magnitude of reported results, leaving the evaluation of the quality of work far less accurate than would be hoped. This also makes the comparison of previous works difficult.

A larger study needs to be performed, and the threats to validity listed above need to be addressed. During the wrap-up session a web based experiment was suggested and will hopefully be prepared and carried out the following summer.

References

1. Walenstein, A., Koschke, R., Merlo, E.: Duplication, redundancy, and similarity in software: Summary of Dagstuhl seminar 06301. [5] ISSN 1682–4405.
2. Walenstein, A.: Code clones: Reconsidering terminology. [5] ISSN 1682–4405.
3. Walenstein, A., Jyoti, N., Li, J., Yang, Y., Lakhotia, A.: Problems creating task-relevant clone detection reference data. In van Deursen, A., Stroulia, E., Storey, M.A.D., eds.: WCRE, IEEE Computer Society (2003) 285–295
4. Kamiya, T., Kusumoto, S., Inoue, K.: CCFinder: a multilinguistic token-based code clone detection system for large scale source code. IEEE Transactions on Software Engineering **28** (2002) 654–670
5. Proceedings of Dagstuhl Seminar 06301: Duplication, Redundancy, and Similarity in Software, Dagstuhl, Germany, Dagstuhl (2006) ISSN 1682–4405.