

On the Need for Human-based Empirical Validation of Techniques and Tools for Code Clone Analysis

Jeffrey Carver
University of Alabama
Tuscaloosa, AL 35487-0290
+1-205-348-9829
carver@cs.ua.edu

Debarshi Chatterji
University of Alabama
Tuscaloosa, AL 35487-0290
+1-205-348-6363
dchatterji@ua.edu

Nicholas A. Kraft
University of Alabama
Tuscaloosa, AL 35487-0290
+1-205-348-4740
nkraft@cs.ua.edu

ABSTRACT

Code clone analysis techniques and tools are popular topics among the software engineering research community. Many studies draw conclusions solely based on an analytical analysis. These claims focus primarily on tool performance in terms of portability, scalability, robustness, precision, and recall. However, these types of analytical studies cannot adequately evaluate the behavior of the developers while using the tools. Human-based empirical studies are complementary to studies based on analytical data because they provide direct insight into developer behavior. In this paper we argue for the need for more human-based empirical studies in the area of code clone analysis techniques and tools.

Categories and Subject Descriptors

D.2.7 [Software Engineering]: Distribution, Maintenance, and Enhancement

General Terms

Measurement, Experimentation, Human Factors

Keywords

Human-based empirical studies, code clones, clone management

1. INTRODUCTION

A primary goal of code clone research is to provide techniques and tools to help developers correctly and efficiently maintain source code that contains clones. Working toward this goal, researchers have developed techniques and tools through which they have studied topics such as clone detection and clone management. For example, Bellon et al. [1] use eight open source software systems to quantitatively compare and evaluate six clone detectors, and Roy et al. [11] use a unified conceptual framework to qualitatively compare and evaluate over 40 clone detectors. Moreover, Duala-Ekoko and Robillard [6] and Hou et al. [7] each provide a clone management system for tracking clones throughout the evolution of a software system.

There are a large number of extensive tool evaluation studies in the literature, some of which are discussed in Sections 2 and 3. While these studies can rightfully make claims about properties of

techniques and tools, they also tend to make claims about the behavior of the developer using the tool. Inferring developer behavior or intent solely from results obtained via mining software repositories is problematic. Software repositories capture a wealth of data that can be used to infer information about software process and software evolution. However, studies that rely solely on such data have a threat to construct validity. Specifically, the repositories contain only snapshots of data, across unevenly and inconsistently distributed intervals. A researcher can only obtain the state of artifacts before and after that interval. There is no way to confidently determine the behavior of the developer which transformed the artifact between the two versions.

In this paper, we argue that to truly understand human behavior and to validate the claims about it, researchers should add human-based empirical studies to their validation toolbox. For example, to truly understand and improve the usefulness of tool output for completing a development task, humans must be observed while using the tool. Another example in which human studies could provide insight is the question of whether clones are helpful or harmful. While early studies indicated that clones were harmful from a maintenance perspective, more recent studies have suggested that clones may actually improve productivity. An in-depth study of the actual effects of clones on developer behavior could provide important insight to this question.

2. PRIOR HUMAN-BASED EMPIRICAL STUDIES

Empirical evidence indicates that developers create, edit, and remove clones when maintaining source code. An understanding of developer behavior with regard to these operations is needed to produce techniques and tools that effectively address the clone-related issues which developers encounter. Further, the manner in which developers engage these techniques and tools should influence their design. However, to our knowledge, there exist only three studies of code clones in which developers are observed directly [4, 5, 9]. In only one of these studies [5] are the developers observed while using a clone-aware tool.

3. CLAIMS NEEDING VALIDATION

Recently, a team including the third author conducted a systematic literature review on clone evolution [10]. Of the 30 primary studies in the review, 27 include empirical studies that involve software repository mining and source code analysis. The authors of these 27 studies focus primarily on analytical findings regarding tool efficacy, the characteristics of a subject software system, or the characteristics of a collection of subject software systems. However, some papers also include claims about developer behavior and intent. Two examples follow.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

IWSC'11, May 23, 2011, Waikiki, Honolulu, HI, USA

Copyright 2011 ACM 978-1-4503-0588-4/11/05...\$10.00.

First, Bettenburg et al. [2] report the results of an empirical study of the effect of inconsistent changes to clones on software quality. They track clones across the releases of *Apache Mina* and *jEdit*, and use manual inspection to identify inconsistent changes to clones. Their results indicate that the majority of the long-lived clones in the two systems are instances of the “replicate and specialize” pattern, in which code is cloned and then customized to introduce a new feature. Their results also indicate that of the clone-related defects exhibiting this pattern, most were not introduced through inconsistent changes. Based on this finding, Bettenburg et al. infer that developers of both *Mina* and *jEdit* are aware of those long-lived clones. They conclude that the developers are able to manage the independent evolution of those clones effectively. While we have no reason to doubt the veracity of these claims, we do believe that claims regarding developer intent should be validated via human-based empirical studies.

Second, Hou et al. [7] describe CnP, a tool for proactive management of clones. CnP tracks clones from creation and provides features meant to facilitate clone evolution, including visualization, consistent renaming, and simultaneous editing. However, Hou et al. do not present a human-based empirical study of CnP. The (seemingly reasonable) assumption that the tool is usable and helpful for developers should be validated with such a study.

4. HUMAN-BASED EMPIRICAL STUDIES

Claims about human behavior cannot be validated without the use of human-based empirical studies. Such studies have been commonly used to understand developer behavior with regards to other aspects of software engineering. For example, software inspections have been thoroughly studied by the first author and others (e.g. [3]). There is an entire discipline of empirical software engineering, which has a conference (*Empirical Software Engineering and Measurement*), a journal (*Empirical Software Engineering*) and a number of handbooks (e.g. [8]).

Many types of human-based empirical studies have proved to be useful in different settings and for addressing different types of research questions. These types of studies range from very controlled laboratory-type studies to case studies which allow for naturalistic observation of behavior in practice, with many options and combinations in between.

For example, we take the claim made by Bettenburg et al. [2] discussed in the previous section. To understand whether developers are aware of long-lived clones and can actually manage independent clone evolution, we would need to design a study in which these claims could be measured and evaluated. Such a study would likely involve surveys and/or interviews with developers to determine their awareness of the clones. In addition, it would require the observation of maintenance tasks to identify exactly how developers address the evolution of clones. Such a study would provide insight both to researchers trying to understand behavior as well as to tool builders who seek to better facilitate that behavior.

As another example, the practical usefulness of the CnP tool proposed by Hou et al. [7] needs to be validated with human-based studies. In this case, we could design a study in which some developers use only Eclipse and others use CnP (an Eclipse plugin) for performing a task. Observation of the developers as performed in previous studies [4, 9] can provide valuable data based on the qualities of interest. This data can be gathered and analyzed to judge whether CnP met its claims. In addition,

qualitative data from surveys and observers’ notes could be elicited from the study which would be invaluable in helping Hou et al. improve CnP.

5. CONCLUSION

The code clone literature exhibits a dearth of human-based empirical studies [9]. There are a number of studies which draw conclusions based on analytical data. These studies often use the analytical data to speculate about human behavior. Due to a number of confounding factors that affect human behavior, such claims need to be validated more thoroughly. Human-based empirical studies focused on understanding how developers create, edit and maintain clones can open new opportunities to understand the primary needs of clone analysis and ultimately help the end user, the software developer. We acknowledge that planning and executing human-based studies require a lot of effort. However we believe that such efforts will pay high dividends.

6. ACKNOWLEDGEMENTS

We acknowledge support from NSF grant CCF-0915559.

7. REFERENCES

- [1] Bellon, S., Koschke, R., Antoniol, G., Krinke, J. and Merlo, E., "Comparison and Evaluation of Clone Detection Tools," *IEEE Trans. on Software Engineering*, 33(9): 577-591. 2007.
- [2] Bettenburg, N., Weyi, S., Ibrahim, W., Adams, B., Ying, Z. and Hassan, A. E. "An empirical study on inconsistent changes to code clones at release level." In *Proc. of WCSE 2009*. pp. 85-94.
- [3] Carver, J. C., Nagappan, N. and Page, A., "The Impact of Educational Background on the Effectiveness of Requirements Inspections: An Empirical Study," *IEEE Trans. on Software Engineering*, 34(6): 800-812. Nov. 2008.
- [4] Chatterji, D., Massengill, B., Oslin, J., Carver, J. and Kraft, N. "Measuring the efficacy of code clone information: An empirical study." In *PLATEAU 2010*. Oct. 18, 2010.
- [5] de Wit, M., Zaidman, A. and van Deursen, A. "Managing code clones using dynamic change tracking and resolution." In *Proc. of ICSM 2009*. 2009. pp. 169-178.
- [6] Duala-Ekoko, E. and Robillard, M. P., "Clone region descriptors: Representing and tracking duplication in source code," *ACM Trans. Softw. Eng. Methodol.*, 20(1): 3:1-3:31. July. 2010.
- [7] Hou, D., Jablonski, P. and Jacob, F. "CnP: Towards an environment for the proactive management of copy-and-paste programming." In *Proc. of ICPC 2009*. pp. 238-242.
- [8] Juristo, N. and Moreno, A., *Lecture Notes on Empirical Software Engineering*. Singapore: World Scientific, 2003.
- [9] Kim, M., Bergman, L., Lau, T. and Notkin, D. "An ethnographic study of copy and paste programming practices in OOPL." In *Proc. of ISESE 2004*. pp. 83-92.
- [10] Pate, J. R., Tairas, R. and Kraft, N. A., "Clone evolution: A systematic review," Department of Computer Science, University of Alabama, Tech. Rep. SERG-2010-01, Dec. 2010. <http://software.eng.ua.edu/reports/SERG-2010-01>
- [11] Roy, C. K., Cordy, J. R. and Koschke, R., "Comparison and evaluation of code clone detection techniques and tools: A qualitative approach," *Science of Computer Programming*, 74(7): 470-495. 5/1. 2009.