Studying Redundant Functionality Through Concolic Analysis

Daniel E. Krutz
Software Engineering Department

October 17, 2013

1 Overview

Redundant functionality, or code clones, are extremely widespread in software development. It has been estimated that clones typically comprise between 5-30% of an application's source code [1] [7]. Code clones may adversely affect the software development process for several reasons. They often raise the maintenance costs of a software project since alterations may need to be done several times [5] and unintentionally making inconsistent bug fixes to cloned code across a software system will likely cause further system faults [2].

There are four types of code clones. Type-1 clones are the simplest and represent identical code except for differences in whitespace, comments and layout. Type-2 clones are syntactically similar except for variations in identifiers and types. Type-3 clones are two segments which differ due to altered or removed statements. Type-4 clones are the most difficult to detect and represent two code segments which significantly differ syntactically, but produce identical results when executed [3].

Although there have been a substantial number of proposed clone detection techniques [10] [9], to our knowledge only two known tools are able to reliably discover type-4 clones. These are Memory Comparision-based Clone Detector (MeCC) [6] and our tool, Concolic Code Clone Detection (CCCD) [10]. Most existing clone detection systems have struggled at finding type-4 clones due to an over-reliance on the semantic or syntactic properties of the source code. The ability to detect type-4 clones is critical since these clones not only comprise a significant portion of the duplicate functionality of an application, but may be the most problematic as well [12].

An area which has recently gathered significant attention is software vulnerabilities. New vulnerabilities are discovered everyday, and often have adverse results on software systems and the people who rely upon them. Vulnerability classification and categorization may be used to generate helpful insights. Some of which include how it was created, why it was created, the proper repair method, and what vulnerabilities are yet to be discovered [11].

The ability to reliably and effectively detect functional redundancy in an application has the ability to significantly assist in vulnerability detection and prevention. We propose using concolic analysis to first identify functionally similar portions of an application, and when a vulnerability is discovered in the system, similar functional segments of the source code will be examined for the same vulnerabilities. While we do not envision that all redundant vulnerabilities may be detected in this manner, we feel that we will be able to discover a significant portion using this process. Information about cloned vulnerabilities may then be aggregated together to provide more information about the vulnerability. We are unaware of any previous techniques which have attempted this, so the knowledge learned will be valuable for future research.

Problem Statement: Detecting duplicate functionality is important in several areas of computing, such as code clone detection and vulnerability remediation. Unfortunately, the vast majority of current techniques suffer due to their inability to find the more complicated types of duplicate functionality. The goal of this work is to clearly and concisely define and refine a process of finding duplicate functionality using concolic analysis.

1.1 Concolic Analysis for Clone Detection

Concolic analysis combines concrete and symbolic values in order to traverse all possible paths of an application (up to a given length). Concolic analysis has been traditionally used in software testing to find application faults [8]. Since we have already demonstrated that concolic analysis is capable of discovering clones on a small scale [10] [9], we will next evaluate concolic analysis using a more robust set of test data. Concolic analysis will be run against several open source applications to determine what types of clones it is capable of finding on a much larger scale. Current candidates include Apache, Firefox and Thunderbird. Next, CloneDR[1], Simian[2], and MeCC[3] will be executed against

[1] http://www.semdesigns.com/products/clone/
[2] http://www.harukizaemon.com/simian/
[3] http://ropas.snu.ac.kr/mecc/

these applications. Each clone detection tool will have its analysis time recorded along with clones correctly identified (true positives), clones mistakenly identified (false positives), clones missed (false negatives) and items correctly not identified as clones (true negatives). These results will be recorded in Table 1.

Tool	Application	Analysis Time	True Positives	False Positives	False Negatives	True Negatives
MeCC	Apache					
	Chromium					
	Android					
CCCD	Apache					
	Chromium					
	Android					
Simian	Apache					
	Chromium					
	Android					
CloneDR	Apache					
	Chromium					
	Android					

Table 1: Clone Tool Comparison Results

1.2 Redundant functionality and Vulnerabilities

Concolic analysis will then be used to assist with the detection and analysis of software vulnerabilities. The first step will be to select several open source applications for analysis which all have well documented vulnerabilities. Current candidates include Apache, Chromium and Android. A previous version of each application's source code will be analyzed for similarities using concolic analysis. We will then examine the reported vulnerabilities on this version to determine:

1. If a vulnerability exists in one cloned segment, does it exist in another?
2. Have vulnerabilities been detected and fixed, only to have the same vulnerability to later be found in a cloned segment?
3. What other useful information which concolic analysis could have provided may have been useful in resolving and preventing vulnerabilities across clone portions of code?

1.3 Empirical Analysis

The next phase will be to analyze how clones affect software development. An empirical analysis will be conducted on the version control systems of several open source applications to determine the effect all types of code clones have on the the software development process. This will be accomplished by using the previously identified clones in the target applications to perform a qualitative and quantitative analysis to answer questions about how clones affect software development. In order to examine existing repositories and answer these questions, a tool will be created which will gather information about each identified cloned and non-cloned items including when the clones were created, what developers modified the functions, the recorded bugs in the functions and the steps taken to fix each issue. This will be used to answer the questions of why the clones were created, if more bugs exist in these clones, do bugs occur consistently across the clones, do vulnerabilities exist across clones, and if vulnerabilities are linked to clones. While there has been work in determining how type-1 and type-2 clones affect the software development process [5] [4], to our knowledge no work has been conducted on how more complicated type-3 and type-4 clones affect software development.

1.4 Relevant Experience

The proposed project is closely related to my research expertise and my PhD thesis. For my dissertation [9] I proposed and demonstrated the effectiveness of concolic analysis for clone detection on a fairly small scale. In a recent publication [10], we used this technique to develop a tool for discovering code clones, Concolic Code Clone Detection (CCCD) [10]. This research demonstrated the ability of CCCD to effectively discover type-4 code clones, something that only one other tool (MeCC) is able to do. Based upon my dissertation work, our tool development and subsequent research, I believe that we have the necessary qualifications to successfully carry out the proposed project.

2 Proposal Goals

The findings of this project are anticipated to produce innovative and highly visible results and are anticipated to produce publications in venues such as FSE, ICSE, MSR or FASE. Several companies have expressed interest in such a robust concolic analysis process for clone, and possibly malware detection. Additionally, they are enthusiastic about a technique which would be able to analyze the amount of similarity between the source code of multiple applications to determine if functionality has been illegally replicated or the amount of new functionality a company would be achieving with the acquisition of a another software organization or package. The goal of this project lay a foundation for future research and demonstrate results to support a larger project which will be proposed to the NSF and industry for funding.

3 Budget

I am requesting a total of $6,750 for this project. The project will be completed with the assistance of an undergraduate or graduate student over the course of both the Spring and Summer semesters. The requested budget is based on having the student work 10 hours per week for 15 weeks @ $15/hr (totaling $2,250) during the Spring term and 20 hours per week for 15 weeks during the Summer semester @$15/hr (totaling $4,500).

The student will assist in comparing concolic analysis against leading clone detection tools and by performing an empirical analysis on the discovered type-3 and type-4 clones to further analyze how they affect the software development process. The student will also conduct research in determining how the vulnerability detection and remediation process may be enhanced using information gathered from concolic analysis. The final results will then be cleaned and properly formatting before being placed on the project website.

References

[1] Ira D. Baxter, Andrew Yahin, Leonardo Moura, Marcelo Sant'Anna, and Lorraine Bier. Clone detection using abstract syntax trees. In *Proceedings of the International Conference on Software Maintenance*, ICSM '98, pages 368–, Washington, DC, USA, 1998. IEEE Computer Society.

[2] Florian Deissenboeck, Benjamin Hummel, and Elmar Juergens. Code clone detection in practice. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2*, ICSE '10, pages 499–500, New York, NY, USA, 2010. ACM.

[3] Nicolas Gold, Jens Krinke, Mark Harman, and David Binkley. Issues in clone classification for dataflow languages. In *Proceedings of the 4th International Workshop on Software Clones*, IWSC '10, pages 83–84, New York, NY, USA, 2010. ACM.

[4] Stan Jarzabek and Yinxing Xue. Are clones harmful for maintenance? In *Proceedings of the 4th International Workshop on Software Clones*, IWSC '10, pages 73–74, New York, NY, USA, 2010. ACM.

[5] Elmar Juergens, Florian Deissenboeck, Benjamin Hummel, and Stefan Wagner. Do code clones matter? In *Proceedings of the 31st International Conference on Software Engineering*, ICSE '09, pages 485–495, Washington, DC, USA, 2009. IEEE Computer Society.

[6] Heejung Kim, Yungbum Jung, Sunghun Kim, and Kwankeun Yi. Mecc: memory comparison-based clone detector. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 301–310, New York, NY, USA, 2011. ACM.

[7] Miryung Kim, Vibha Sazawal, David Notkin, and Gail Murphy. An empirical study of code clone genealogies. *SIGSOFT Softw. Eng. Notes*, 30(5):187–196, September 2005.

[8] Yunho Kim, Moonzoo Kim, YoungJoo Kim, and Yoonkyu Jang. Industrial application of concolic testing approach: a case study on libexif by using crest-bv and klee. In *Proceedings of the 2012 International Conference on Software Engineering*, ICSE 2012, pages 1143–1152, Piscataway, NJ, USA, 2012. IEEE Press.

[9] Daniel E. Krutz. *Concolic Code Clone Detection*. PhD thesis, Nova Southeastern University, 2012.

[10] Daniel E. Krutz and Emad Shihab. Cccd: Concolic code clone detection. In *Reverse Engineering (WCRE), 2013 20th Working Conference on*, 2013.

[11] Yan Wu, Harvey Siy, and Robin Gandhi. Empirical results on the study of software vulnerabilities (nier track). In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 964–967, New York, NY, USA, 2011. ACM.

[12] Yang Yuan and Yao Guo. Cmcd: Count matrix based code clone detection. In *Proceedings of the 2011 18th Asia-Pacific Software Engineering Conference*, APSEC '11, pages 250–257, Washington, DC, USA, 2011. IEEE Computer Society.