An Empirical Study on Cross-language Clone Bugs

Honghao Chen, Ye Tang, and Hao Zhong
Department of Computer Science and Engineering, Shanghai Jiao Tong University, China
{chenhonghao,tangye_22,zhonghao}@sjtu.edu.cn

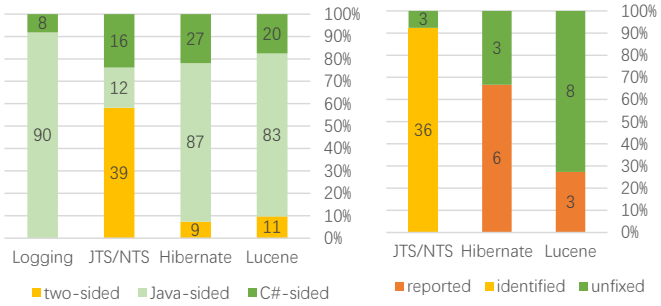
ABSTRACT

Many applications have implementations in different languages. Although their languages are different, they can implement similar or even identical functionalities. If an implementation has a bug, the other implementations can have corresponding bugs. In this paper, we call them cross-language clone bugs, or mirror bugs for short. Mirror bugs are important since many applications release implementations in different languages. From mirror bugs, it can be feasible to learn more bug patterns, and thus detect more types of bugs. Although researchers have conducted empirical studies to analyze the bugs in clones, to the best of our knowledge, no study has ever explored mirror bugs. As a result, many research questions are still open. For example, are there any mirror bugs in real projects? Are bug fixes in a language useful to detect and repair bugs in other languages? To answer the above questions, in this paper, we conduct the first empirical study on mirror bugs. In this study, we manually analyze 402 bugs that are collected from four projects, and each project releases a Java implementation and C# implementation. Our study presents answers to two interesting research questions. According to our results, there is a timely need for a tool that assists in detecting mirror bugs. Indeed, we find that some programmers already manually identify and fix mirror bugs, even without any tool support.

ACM Reference Format:
Honghao Chen, Ye Tang, and Hao Zhong. 2024. An Empirical Study on Cross-language Clone Bugs. In *2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion '24), April 14–20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3639478.3643075

1 INTRODUCTION

To attract more users, projects can be implemented in multiple programming languages. For example, many popular mobile applications release both iOS versions and Android versions. Even if a project is implemented in a single language, outsiders can re-implement the project in other languages. For example, `LocationTech` implements Java Topology Suite (JTS) [6], a library that provides geometric functions. Meanwhile, outsiders implement, Net Topology Suite [7], a C# correspondence of `JTS`. When a project has implementations in multiple languages, they are typically ported from the implementation in a single language. As a result, these implementations in different languages can contain many similar code fragments. Indeed, researchers [8, 12] have detected many cross-language clones. When they analyze bugs in the same language, researchers [11, 13] found that many bugs appear in similar code fragments. Furthermore, researchers [10, 15] have proposed approaches to extract bug signatures and detect similar bugs in clones. Similarly, bugs can appear in similar code fragments, even if they are implemented in different languages, and a tool is desirable to detect such bugs. However, to the best of our knowledge, no study has been conducted to analyze mirror bugs. Compared with bugs in clones, languages supplement interesting elements to the studies on mirror bugs. For example, memory leaks are less found in Java code than in C code, since Java provides the garbage collection mechanism to manage memory resources. To deepen the knowledge of bugs, we explore the following questions:



(a) One-sided and two-sided **(b) Reported and identified**

Figure 1: The distribution

- **RQ1. How many mirror bugs are there?** The answers are useful for understanding the significance of mirror bugs.
- **RQ2. How many mirror bugs are fixed?** The answers are useful for understanding the awareness of mirror bugs.

2 METHODOLOGY

Dataset. In this study, we select 4 real-world projects. From the four projects, we have sampled 402 bugs from these projects. From the projects, we select bug reports in order of their priorities or from the latest ones. For each implementation, we terminate the selection, if all candidates are inspected or 90 bug reports are selected. As the analysis of mirror bugs involves many complicated issues, we cannot implement a tool to automate the analysis. Due to the heavy effort of manual analysis, we cannot afford the analysis of more bugs. Indeed, from the four projects, we have collected 402 bugs, which are already more than other empirical studies [9, 14].

Protocol. Our manual inspection includes reading the introductions and manuals of the projects; inspecting bug reports to understand the symptoms; and inspecting patches to understand the causes. Specifically, for each bug report in one of the paired projects p_l, we search the bug reports of the other project $p_{l'}$ for a

ICSE-Companion '24, April 14–20, 2024, Lisbon, Portugal

ACM ISBN 979-8-4007-0502-1/24/04…$15.00
https://doi.org/10.1145/3639478.3643075

report whose symptoms are similar. If we can find such a report, we check out their fixing commits to determine whether it is a two-sided bug. If we cannot find a bug report with similar symptoms, we locate its buggy files in p_l according to its fixing commit. After that, we search the latest source files of $p_{l'}$ for equivalent source files. If we can reproduce a similar symptom on the equivalent source files of $p_{l'}$, we classify this bug report as a two-sided bug. If not, we identify it as a one-sided bug. Also, while classifying, we pay special attention to whether two-sided bugs are fixed.

3 EARLY RESULT

RQ1. Overall Distribution. Figure 1a shows the distribution of different projects. In `Lucene` and `Hibernate`, two-sided bugs are around 10%. However, we find around 60% sampled bugs from `JTS` are mirror bugs. We notice that some programmers from `NTS` participate in the development of `JTS` and actively learn the fixed bugs of `JTS`. When `Log4j` upgrades from `1.x` to `2.x`, most source files are rewritten. As we select recent bugs, the bugs from `Log4net` are similar to `Log4j 1.x`, but we find no two-sided bugs between `Log4j2` and `Log4net`. In total, we find that 14.7% of sampled bugs appear in both the Java and C# implementations. As the implementations in two languages have many differences during their independent evolution, the percentage is high, and can motivate many research topics on cross-language bugs.

RQ2. Fixed or Unfixed. We analyze how many two-sided bugs were fixed in this research question. Figure 1b shows the distribution of different projects. `JTS` and `NTS` have no reported mirror bug, but they actively identified mirror bugs. `Lucene` and `Hibernate` tell a different story. For example, Figure 2 shows a reported mirror bug. Although their reporters are different, the buggy code lines and even their repairs are quite similar. The similarity highlights the importance of detecting mirror bugs. In total, 45 (76.2%) of our found two-sided bugs are already fixed. 9 have bug reports in both Java and C# implementations, but 36 bugs have reports on only one side. Among unfixed bugs, we find some bugs have minor symptoms, and programmers of the other implementation may not fix them. For example, the symptom of Lucene-10118 [5] is that the log messages in `ConcurrentMergeScheduler` are too simple. The programmers of `Lucene` may not pay attention to this minor bug. More details are presented on our website:
https://github.com/chenhh021/cross-language-poster

4 WORK PLAN

To extend this work to a full paper, our work plan is as follows:

1. Analyze the causes for one-sided bugs. The results are useful for understanding the differences that are caused by languages. For example, the difference in implementations of different languages. We also plan to build the taxonomy for the causes. The results are useful for designing a detection tool for mirror bugs. For example, after knowing all the causes, a tool can improve its accuracy in detecting mirror bugs.

2. Find and fix new bugs. We plan to try to manually find new bugs based on mirror bugs, and if we succeed, we will attempt to fix them according to the known patch. After that, we will report them to developers to collect their feedback. This manual process will provide references for the development of detection tools and repair tools. The feedback from developers would help in understanding the significance of researching such tools.

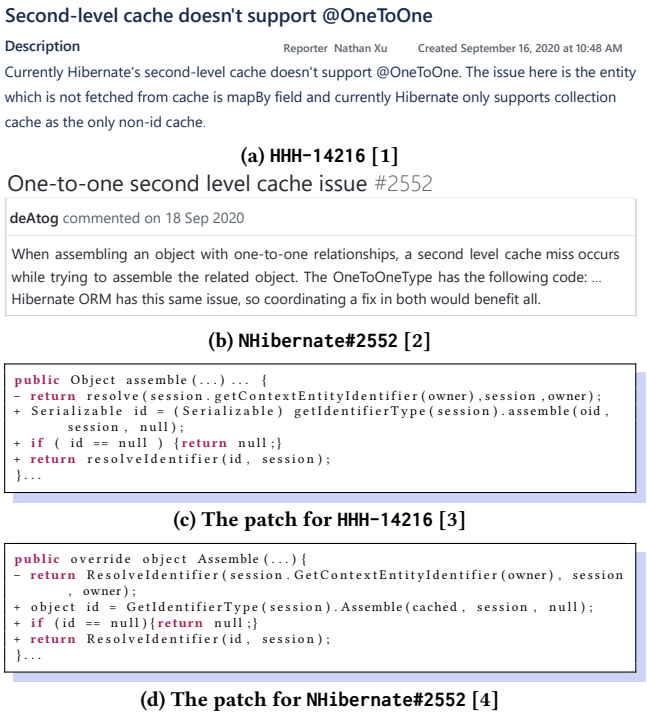


Figure 2: A mirror bug.

ACKNOWLEDGEMENT

This work is sponsored by National Nature Science Foundation of China No. 62232003 and 62272295.

REFERENCES

[1] 2020. https://hibernate.atlassian.net/browse/HHH-14216.
[2] 2020. https://github.com/nhibernate/nhibernate-core/issues/2552.
[3] 2020. https://github.com/hibernate/hibernate-orm/pull/3590.
[4] 2020. https://github.com/nhibernate/nhibernate-core/pull/2576.
[5] 2021. https://issues.apache.org/jira/browse/LUCENE-10118.
[6] 2022. https://locationtech.github.io/jts/.
[7] 2022. https://github.com/NetTopologySuite/NetTopologySuite.
[8] Xiao Cheng, Zhiming Peng, Lingxiao Jiang, Hao Zhong, Haibo Yu, and Jianjun Zhao. 2016. Mining revision histories to detect cross-language clones without intermediates. In *Proc. ASE*. 696–701.
[9] Li Jia, Hao Zhong, Xiaoyin Wang, Linpeng Huang, and Xuansheng Lu. 2021. The symptoms, causes, and repairs of bugs inside a deep learning library. *Journal of Systems and Software* 177 (2021), 110935.
[10] Yanjie Jiang, Hui Liu, Nan Niu, Lu Zhang, and Yamin Hu. 2021. Extracting concise bug-fixing patches from human-written patches in version control systems. In *Proc. ICSE*. 686–698.
[11] Sunghun Kim, Kai Pan, and E. James Whitehead Jr. 2006. Memories of bug fixes. In *Proc. ESEC/FSE*, Michal Young and Premkumar T. Devanbu (Eds.). ACM, 35–45.
[12] Nicholas A. Kraft, Brandon W. Bonds, and Randy K. Smith. 2008. Cross-language Clone Detection. In *Proc. SEKE*. 54–59.
[13] Tung Thanh Nguyen, Hoan Anh Nguyen, Nam H Pham, Jafar Al-Kofahi, and Tien N Nguyen. 2010. Recurring bug fixes in object-oriented programs. In *Proc. 32nd ICSE*. 315–324.
[14] Xiao Xuan, Xiaoqiong Zhao, Ye Wang, and Shanping Li. 2015. An Empirical Study of Bugs in Industrial Financial Systems. *IEICE Trans. Inf. Syst.* 98-D, 12 (2015), 2322–2327.
[15] Hao Zhong, Xiaoyin Wang, and Hong Mei. 2020. Inferring bug signatures to detect real bugs. *IEEE Transactions on Software Engineering* 48, 2 (2020), 571–584.