Towards Reusing Hints from Past Fixes

An Exploratory Study on Thousands of Real Samples

Hao Zhong
Department of Computer Science and Engineering
Shanghai Jiao Tong University, China
zhonghao@sjtu.edu.cn

Na Meng
Department of Computer Science
Virginia Tech, USA
nm8247@cs.vt.edu

ABSTRACT

Researchers have recently proposed various automatic program repair (APR) approaches that reuse past fixes to fix new bugs. However, some fundamental questions, such as how new fixes overlap with old fixes, have not been investigated. Intuitively, the overlap between old and new fixes decides how APR approaches can construct new fixes with old ones. Based on this intuition, we systematically designed six overlap metrics, and performed an empirical study on 5,735 bug fixes to investigate the usefulness of past fixes when composing new fixes. For each bug fix, we created *delta dependency graphs* (*i.e.*, program dependency graphs for code changes), and identified how bug fixes overlapped with each other in terms of *the content, code structure, and identifier names of fixes*. Our results show that if an APR approach composes new fixes by fully or partially reusing the content of past fixes, only 2.1% and 3.2% new fixes can be created from single or multiple past fixes in the same project, compared with 0.9% and 1.2% fixes created from past fixes across projects. However, if an APR approach composes new fixes by fully or partially reusing the code structure of past fixes, up to 41.3% and 29.7% new fixes can be created.

ACM Reference format:
Hao Zhong and Na Meng. 2018. Towards Reusing Hints from Past Fixes. In *Proceedings of ICSE '18: 40th International Conference on Software Engineering , Gothenburg, Sweden, May 27-June 3, 2018 (ICSE '18),* 1 pages.
DOI: 10.1145/3180155.3182550

1 INTRODUCTION

Due to the usage of version control systems, many bug fixes have accumulated over the years. From past bug fixes, researchers [1–3] proposed some automatic program repair (APR) approaches that reused past fixes to fix new bugs. Although these approaches show promising results, they are limited in the following two aspects: (1) existing empirical studies are based on manual or simple automatic analysis; and (2) some hypotheses of APR approaches are not fully explored. In this paper, by comparing each fix's delta dependency graphs (DDGs) produced by GRAPA [5], we performed a comprehensive empirical study on 5,735 bug fixes in four popular projects. Our comparison checks for six types of overlap between bug fixes *i.e., a bug fix may fully or partially overlap with one or multiple fixes in terms of code content, syntactic structures, and identifier names.* Our major findings are as follows:

- **Extent of constructing fixes from past fixes.** Bug fixes have more overlap in syntactic structures than code content and identifier names. The syntactic structures of 41% fixes can be synthesized from the syntactic structures of multiple past fixes. A new fix is often relevant to only several useful past fixes. However, if we require exact matches, only 3% fixes can be constructed from past fixes, as most bugs need creative repairs that never appear in past fixes.
- **Preparing a repository of past fixes.** We consider a bug fix to be useful, if it is repetitive. If we build the repository from fixes of the same project, most fixes are useful, but it can be challenging to extract useful repair actions, since a fix typically has both useful repair actions and useless repair actions. The usefulness of a repository may not increase with the number of collected bug fixes. This is because when a repository contains many past fixes, it can become challenging to identify useful ones among the fixes. It is worthy identifying useful fixes, since an identified fix can be useful to repair many fixes.
- **Learning from other projects.** From other projects, it is feasible to learn many code structure changes. However, the combination of multiple past fixes does not achieve as good results as the combination of fixes from the same projects. Furthermore, as different projects typically have different client code names, it becomes more challenging to learn how to replace those API elements correctly.

Our journal paper [4] presents more details of the empirical study.

ICSE '18, Gothenburg, Sweden
 978-1-4503-5638-1/18/05…$15.00
DOI: 10.1145/3180155.3182550

ACKNOWLEDGEMENT

Hao Zhong is sponsored by the 973 grant 2015CB352203, the NSFC grant 61572313, and the STCSM grant 15DZ1100305. Na Meng is sponsored by the grants NSF CCF-1565827 and ONR-N00014-17-1-2498.

REFERENCES

[1] Dongsun Kim, Jaechang Nam, Jaewoo Song, and Sunghun Kim. 2013. Automatic patch generation learned from human-written patches. In *Proc. 35th ICSE*. 802–811.
[2] Fan Long and Martin Rinard. 2016. Automatic patch generation by learning correct code. In *Proc. 43rd POPL*. 298–312.
[3] Matias Martinez and Martin Monperrus. 2013. Mining software repair models for reasoning on the search space of automated program fixing. *Empirical Software Engineering* 20, 1 (2013), 176–205.
[4] Hao Zhong and Na Meng. 2018. Towards Reusing Hints from Past Fixes -An Exploratory Study on Thousands of Real Samples. *Empirical Software Engineering* (2018).
[5] Hao Zhong and Xiaoyin Wang. 2017. Boosting complete-code tool for partial program. In *Proc. ASE*. 671–681.