Empirical Effectiveness Evaluation of Spectra-based Fault Localization on Automated Program Repair

Yuhua Qi[1], Xiaoguang Mao[1]*, Yan Lei[1], Ziying Dai[1], Yudong Qi[2] and Chengsong Wang[1]
[1]National University of Defense Technology, [2]Naval Aeronautical and Astronautical University
[1]Changsha, China, [2]Yantai, China
{yuhua.qi, xgmao, yanlei}@nudt.edu.cn nudtdzy@gmail.com qiyudong@sina.com jameschen186@gmail.com

***Abstract*—Researchers have proposed many spectra-based fault localization (SBFL) techniques in the past decades. Existing studies evaluate the effectiveness of these techniques *from the viewpoint of developers*, and have drawn some important conclusions through either empirical study or theoretical analysis. In this paper, we present the first study on the effectiveness of SBFL techniques *from the viewpoint of fully automated debugging* including the program repair of automation, for which the activity of automated fault localization is necessary. We assess the accuracy of fault localization according to the repair effectiveness in the automated repair process guided by the localization technique. Our experiment on 14 popular SBFL techniques with 11 subject programs shipping with real-life field failures presents the evidence that some conclusions drawn in prior studies do not hold in our experiment. Based on experimental results, we suggest that Jaccard should be used with high priority before some more effective SBFL techniques specially proposed for automated program repair occur in the future.**

***Keywords*—Spectra-based fault localization; automated program repair; automated debugging**

I. Introduction

Debugging software after a failure has occurred is commonly recognized as a resource consuming process. Due to too much manual involvement during traditional debugging process, finding the fault, also called fault localization used to locate faulty lines of code, often consumes a great deal of time resources. To alleviate the debugging cost, numerous researchers have proposed various automated fault localization techniques to automate the process of finding fault. Of these techniques, a particularly effective approach is spectra-based fault localization (SBFL) [1], [2].

Recently, one popular area on fault localization is to evaluate and assess the effectiveness of various SBFL techniques through either empirical approach [3], [4] or theoretical analysis [2], [1]. Most existing literature evaluates the effectiveness *from the viewpoint of developers*, that is, measuring how many benefits that developers can obtain from SBFL technique used when debugging. To quantify the benefits, most existing work evaluates the effectiveness according to the percentage of the program code that needs to be examined before the faults are identified, which is referred to as the *EXAM* score [2], [1]. Obviously, a lower *EXAM* score indicates a better localization effectiveness. However, to our knowledge there exists no research work on evaluate and assess the effectiveness of various SBFL techniques from the viewpoint of fully automated debugging, which automates the process of program repair [5].

*Corresponding author.

The activity of automated fault localization is most often necessary especially when automating the repair of large-scale programs. Although the recent work [6], [7], [8] on automated program repair has noticed the important influence caused by the fault localization activity, there is no detail discussion on this kind of information.

In this paper we plan to empirically study the effects of popular SBFL techniques on automated program repair. We first present an evaluation measurement (i.e., *NCP* score in Section II-A) to evaluate the effectiveness of fault localization *from the viewpoint of fully automated debugging* (including the repair process of full automation). According to *NCP* score, we then evaluate the effects of 14 popular SBFL techniques through the experiment of automating the repair of 11 real-life programs shipping with field failures, the failures occur after deployment.

The main contributions of this paper are:

- The first study (to our knowledge) that evaluates the effectiveness of SBFL techniques from the viewpoint of fully automated debugging, rather than from the viewpoint of developers having done by much existing work.
- The evaluation measurement in term of *NCP* for evaluating and comparing the effectiveness of various fault localization techniques on automated program repair.
- Experimental Results show that these techniques that perform well in existing studies according to *EXAM* measurement do not have the lower *NCP* scores, justifying the necessity of studying fault localization from the viewpoint of fully automated debugging. In addition, we also find that Jaccard performs at least as good as the other 13 investigated techniques.

II. Experiment Evaluation

A. Repair System: GenProg-FL

To conduct the experiment, we have implemented a repair system called GenProg-FL, a tool that enables scalability to repair automatically C programs. GenProg-FL is the modification version of GenProg[1] [6]. GenProg has the ability of fixing bugs in deployed, legacy C programs without formal specifications. We modified the GenProg by adding the interface for accepting the localization information from various SBFL techniques.

We measure the repair effectiveness of GenProg-FL according to the Number of invalid Candidate Patches (*NCP*) generated before a valid patch is found in the repair process. The lower *NCP* score can be observed in the repair process, the better localization effectiveness of fault localization technique used by GenProg-FL.

B. Investigated SBFL techniques

In our experiment, we investigate 14 SBFL techniques, which includes most of popular techniques on SBFL. The 14 techniques come from the literature [1], [2], in which 30 SBFL techniques are systemically studied; other 16 techniques are eliminated because each of them is equivalent to some technique included by the 14 ones [2]. The more details on the definition of selected techniques in our experiment can be found in [1], [2].

C. Subject Programs

In this experiment, we selected the subject programs used in the most recent work [6] on GenProg as the experimental benchmarks[2], all of which are written in C language and are different real-world systems with real-life bugs from different domains. These programs come with different sizes of positive test cases ranging from 12 to 8,471, which facilitates the application of SBFL.

Complete experimental evaluation on all the programs can take too much time (see [6, Table II]); the authors in [6] used Amazon's EC2 cloud computing infrastructure including 10 trials in parallel for their experiment. What is worse, in our experiment, for each subject program we need 14 times more computing resource compared to the experiment in [6], because the effectiveness of all the 14 investigated SBFL techniques need to be statistically observed. Given the expensive testing computation, we selected 11 faulty programs including `libtiff`, `php`, `python`, and `wireshark` in our experiment. Our experiment ran on three Ubuntu 10.04 machines in parallel, and we spent about one and a half month in collecting these experimental data.

D. Experimental Results

For each SBFL technique, we separately ran GenProg-FL to repair the 11 subject programs with the guidance of SBFL. All the experimental parameters for GenProg-FL in our experiment are similar to those settings in [6].

Experimental results show that 1) Ochiai did not always perform better over either Jaccard or Tarantula, which is inconsistent with prior empirical studies [4], 2) neither Naish or Russel_rao has the better effectiveness over other 12 techniques, although the two techniques has been theoretically proven to be optimal under the evaluation measurement of *EXAM* [2], [1], 3) Jaccard performs at least as good as the other 13 investigated techniques, with the effectiveness improvement up to "large" effect size.

After investigating the outcome, we find that Jaccard has more sharp score, which allows Jaccard to keep bigger suspiciousness of faulty statements and smaller suspiciousness of most normal statements. Although some techniques rank faulty statements higher, they also produce not too low suspiciousness of many normal statements (i.e., more flat score), which drastically increases the risk of modifying normal statements in the repair process, resulting in new faults introduced.

III. Conclusions and Future Work

Although our evaluation has shown some important observations on the effects of 14 SBFL techniques on automated program repair, more experimentation must be conducted to confirm the effects in general. In the future we plan to verify the conclusions on more real-life programs. In addition, we also plan to develop a new SBFL technique aiming to improve the repair effectiveness in term of lower *NCP* score from the viewpoint of fully automated debugging. However, according to the experimental results, we suggest that Jaccard should be used with high priority before some more effective SBFL techniques specially proposed for automated program repair occur in the future.

Acknowledgments

This research was supported in part by National Natural Science Foundation of China (Grant Nos. 61133001, 91118007) , National High Technology Research and Development Program of China (Grant Nos. 2011AA010106, 2012AA011201), and Program for New Century Excellent Talents in University.

References

[1] L. Naish, H. J. Lee, and K. Ramamohanarao, "A model for spectra-based software diagnosis," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 20, no. 3, pp. 11:1–11:32, 2011.

[2] X. Xie, T. Y. Chen, F.-c. Kuo, and B. Xu, "A theoretical analysis of the risk evaluation formulas for spectrum-based fault localization," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 2013 (to appear).

[3] S. Ali, J. H. Andrews, T. Dhandapani, and W. Wang, "Evaluating the accuracy of fault localization techniques," in *International Conference on Automated Software Engineering (ASE)*, 2009, pp. 76–87.

[4] R. Abreu, P. Zoeteweij, R. Golsteijn, and A. J. van Gemund, "A practical evaluation of spectrum-based fault localization," *Journal of Systems and Software (JSS)*, vol. 82, no. 11, pp. 1780 – 1792, 2009.

[5] M. Harman, "Automated patching techniques: the fix is in: technical perspective," *Communications of the ACM*, vol. 53, no. 5, pp. 108–108, 2010.

[6] C. Le Goues, M. Dewey-Vogt, S. Forrest, and W. Weimer, "A systematic study of automated program repair: fixing 55 out of 105 bugs for $8 each," in *International Conference on Software Engineering (ICSE)*, 2012, pp. 3–13.

[7] Y. Wei, Y. Pei, C. A. Furia, L. S. Silva, S. Buchholz, B. Meyer, and A. Zeller, "Automated fixing of programs with contracts," in *International Symposium on Software Testing and Analysis (ISSTA)*, 2010, pp. 61–72.

[8] A. Arcuri, "Evolutionary repair of faulty software," *Applied Soft Computing*, vol. 11, no. 4, pp. 3494 – 3514, 2011.

[1]Available: http://dijkstra.cs.virginia.edu/genprog/

[2]https://church.cs.virginia.edu/genprog/archive/genprog-105-bugs-tarballs/