Estimating the Effectiveness of Spectrum-Based Fault Localization

Shuo Song
State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China
ss11@software.nju.edu.cn

ABSTRACT

Spectrum-Based Fault Localization (SBFL) techniques calculate risk values to predict buggy units in a program, but they may cause heavy manual work when the calculated risk values are not reasonable on some application scenarios. In this paper, presents a preliminary study to estimate the effectiveness of SBFL before manual code walk through, so that we can decide whether to adopt SBFL for a given application.

Categories and Subject Descriptors

D.2.5 [**Software Engineering**]: Testing and Debugging—*Fault Localization*

General Terms

Experimentation, Reliability

Keywords

Debugging, Fault localization, Risk formula, Estimator

1. INTRODUCTION

Spectrum-Based Fault Localization (SBFL) uses the execution profile and test case information to localize faults in a program [9]. Many empirical studies show the effectiveness of SBFL in debugging [8]. However, the effectiveness of SBFL techniques depends on many factors, such as the pass-fail ratio of tests, the coverage of tests and the structure of the program under test [2][4][13]. Therefore, the effectiveness of SBFL is not always guaranteed and it might be misleading if the result of fault localization is ambiguous and inaccurate [10]. This motivates us to estimate the effectiveness of SBFL in advance, so that we can decide whether to adopt SBFL for a given application.

In this paper, we present a simple formula to estimate the effectiveness of SBFL. The program spectrum (coverage information and test results) are transformed into color for each function in program. The color difference is used as an estimator of the effectiveness of SBFL. The effort of using estimator to estimate the effectiveness of SBFL is less onerous compared to the effort of walking through source code with inaccurate SBFL results. We design an empirical study with two C programs (*schedule*) and (*grep*). The experimental results indicate that our estimator can work well on some small programs and large programs. We also discuss how to sufficiently use colored call graph and design more comprehensive empirical studies in the future.

FSE'14, November 16–21, 2014, Hong Kong, China
Copyright 2014 ACM 978-1-4503-3056-5/14/11...$15.00

2. BACKGROUND AND RELATED WORK

Fault localization has been studied intensively and many methods have been proposed in the past decades [11]. Among all the methods, spectrum-based methods [9] have been widely studied due to their simplicity.

The empirical study by R. Abreu et al. indicated that the accuracy of SBFL would be affected by tests used [2]. Their further study showed that a few (around 10) failed tests can be sufficient to reach near-optimal debugging effectiveness and the effect of passed tests is unpredictable [1]. N. DiGiuseppe et al. studied the effect of multiple faults on fault localization techniques [4]. Y. Yu et al.[13] studied the effects of test suite reduction on fault localization. Their results showed that the effectiveness of SBFL varies depending on the test suite reduction strategy used. B. Jiang et al. presented an empirical study to examine the impact of test case prioritization on the effectiveness of fault localization [7]. Y. Miao observed the effect of coincidental correctness on fault localization and proposed a clustering-based strategy to weaken the affects [10]. X. Xie provided a theoretical analysis of the suspicious formulas on SBFL [12].

In [13], Y. Yu et al. assigned Tarantula suspiciousness and confidence metrics to each coverage entities, in which confidence value is to measure the degree of confidence in the given suspiciousness. The Tarantula confidence is greater if the entity is covered by more test cases. The Tarantula confidence is used in sorting coverage entities with same suspiciousness, instead of estimating SBFL effectiveness in the application as a whole.

All the above efforts study different factors to affect the effectiveness of SBFL. These studies show the effectiveness and the limitations of SBFL. We believe that SBFL can work well for some, but not all, application scenarios. Therefore, we propose a novel estimator on the effectiveness of SBFL.

3. APPROACH AND UNIQUENESS

Given a program $P = < e_1, e_2, \cdots, e_n >$ with some entities (functions used in this paper) executed by a test suite T. The coverage information and test results, so called program spectrum, are often collected for some software maintenance task, such as test suite reduction, test case prioritization, fault localization, etc. For each entity e_i, we use a_i^{ef}, a_i^{ep} to denote the number of test cases executing e_i with the result of *fail* or *pass*, respectively.

Function call graphs are generated by Egypt[1] and Graphviz [6][2] in Linux. Egypt makes use of GCC's capability to create an intermediate representation of the program being compiled into a Register Transfer Language (RTL) file. Egypt extracts information about function calls from the RTL file and converts it into the format used by Graphviz, which draws the graph.

The call graph is a static view of the program, in which a node represents a function and an edge represent a function call. We use the program spectrum to color the call graph, based on two formulas: $Red_i = 255 \times \frac{a_i^{ef} - min(a^{ef})}{max(a^{ef}) - min(a^{ef})}$ and $Green_i = 255 \times \frac{a_i^{ep} - min(a^{ep})}{max(a^{ep}) - min(a^{ep})}$, in which $max(a^{ef})$, $min(a^{ef})$, $max(a^{ep})$ and $min(a^{ep})$ are maximum a_i^{ef}, minimum a_i^{ef}, maximum a_i^{ep} and minimum a_i^{ep} for all i, respectively. Each node i is calculated with Red_i value and $Green_i$ value in RGB color model.

Given a faulty program P and test suite T, we use the color difference to construct the estimator for the effectiveness of SBFL. Intuitively, if the color difference of call graph is larger, that is the difference between a_i^{ef} and a_i^{ep} is larger for some functions, then it may be more suitable for using SBFL. We use $Red_i - Green_i$ to compute the color difference, and select the maximum one, i.e. $max(Red_i - Green_i)$ for all i, as the effectiveness estimator of SBFL. The larger estimating value indicates that SBFL can work better in this application.

4. EXPERIMENTAL RESULTS

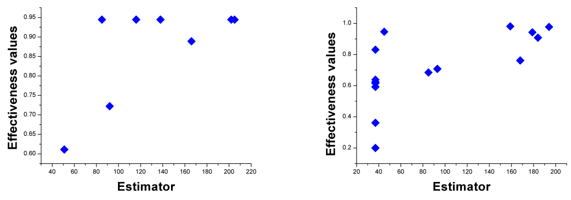
Two C programs *schedule* and *grep* from Software-artifact Infrastructure Repository (SIR) [5] are used to test the method. Along with source code, SIR also provides seeded version with faults and test suites. Program *schedule* has 412 lines of code, 18 functions and 9 separate faults. Program *grep*(v3) has 10068 lines of code, 146 functions and 16 separate faults. We execute the program under all test cases to collect function call information and construct the program spectrum for fault localization.

We use a popular risk formula *Jaccard* ($\frac{a_i^{ef}}{a_i^{ef} + a_i^{nf} + a_i^{ep}}$) [3] of SBFL to locate the seeded faults, in which a_i^{nf} denotes the number of tests not executing the function. All functions are sorted in descending order of risk values. The *expense* of SBFL is "rank of faulty function/number of executable functions" [8]. We use 1-*expense* to evaluate the actual effectiveness of SBFL, the higher 1-*expense*, the more effective SBFL is.

In order to investigate the correlations between estimating values and the actual effectiveness of SBFL, we draw the scatter diagrams for *schedule* and *grep*, as shown in Figure 1.

[1]http://www.gson.org/egypt/egypt.html
[2]http://www.graphviz.org/

Both two scatter diagrams indicate strong correlation coefficients between estimating values and actual effectiveness of SBFL. Furthermore, we calculate the Spearman's rank correlation coefficient values of two results and they are 0.6811 and 0.6632, respectively. These two scatter diagrams show that when the estimators are greater than 100, the effectiveness of SBFL are excellent (1-*expense*> 0.9 for *schedule* and 1-*expense*> 0.8 for *grep*) . It inspires us to find a boundary estimating value to decide whether to adopt SBFL in practice. This will be studied in our future work.



(a) *schedule*(R=0.6811) (b) *grep*(R=0.6632)

Figure 1: Experimental Results of *schedule* and *grep*

5. CONCLUSION AND DISCUSSION

In this paper, we present a simple formula to estimate the effectiveness of SBFL based on the color difference of call graph. The preliminary results show that our estimator can work well on both small program (*schedule*) and large program (*grep*).

We should study the colored call graph more deeply in the future. The call relations are a key part of a program. We believe that combining the structure information and color information can construct better estimators. Furthermore, the estimator formula is simple in this paper and it can be improved with more insights. Both fault localization and fault understanding are important to program debugging. The colored call graph can be used not only to locate, but also to help understand a fault in a program. The colored call graph will be a promising direction for automatic debugging.

As we discussed before, the effectiveness of SBFL may be affected by many factors. Hence, we should design more sufficient estimator and design more comprehensive empirical studies to verify our approach in the future. We should select programs under test with different sizes and different structures. We also use test suites, with different passed tests and failed tests for the same program to investigate the impact of test suites. We use *Jaccard* as the risk formula of SBFL in this paper. A number of risk formulas [12] should be studied to design a specified estimator for a given risk formula. We also believe the fault types will affect the effectiveness of SBFL techniques. All the factors will be considered in our future work.

6. ACKNOWLEDGMENT

The author would like to thank Dr. Zhenyu Chen, Kai Zhou, and Yang Feng for their valuable comments. This work is partially supported by the National Natural Science Foundation of China (61170067, 61373013).

7. REFERENCES

[1] R. Abreu, P. Zoeteweij, R. Golsteijn, and A. J. Van Gemund. A practical evaluation of spectrum-based fault localization. *Journal of Systems and Software*, 82(11):1780–1792, 2009.

[2] R. Abreu, P. Zoeteweij, and A. J. Van Gemund. On the accuracy of spectrum-based fault localization. In *Testing: Academic and Industrial Conference Practice and Research Techniques-MUTATION, 2007. TAICPART-MUTATION 2007*, pages 89–98. IEEE, 2007.

[3] M. Y. Chen, E. Kiciman, E. Fratkin, A. Fox, and E. Brewer. Pinpoint: Problem determination in large, dynamic internet services. In *Dependable Systems and Networks, 2002. DSN 2002. Proceedings. International Conference on*, pages 595–604. IEEE, 2002.

[4] N. DiGiuseppe and J. A. Jones. On the influence of multiple faults on coverage-based fault localization. In *Proceedings of the 2011 international symposium on software testing and analysis*, pages 210–220. ACM, 2011.

[5] H. Do, S. G. Elbaum, and G. Rothermel. Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact. *Empirical Software Engineering: An International Journal*, 10(4):405–435, 2005.

[6] J. Ellson, E. Gansner, L. Koutsofios, S. C. North, and G. Woodhull. GraphvizâĂŤopen source graph drawing tools. In *Graph Drawing*, pages 483–484. Springer, 2002.

[7] B. Jiang, Z. Zhang, T. Tse, and T. Y. Chen. How well do test case prioritization techniques support statistical fault localization. In *Computer Software and Applications Conference, 2009. COMPSAC'09. 33rd Annual IEEE International*, volume 1, pages 99–106. IEEE, 2009.

[8] J. A. Jones and M. J. Harrold. Empirical evaluation of the tarantula automatic fault-localization technique. In *Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*, pages 273–282. ACM, 2005.

[9] J. A. Jones, M. J. Harrold, and J. Stasko. Visualization of test information to assist fault localization. In *Proceedings of the 24th international conference on Software engineering*, pages 467–477. ACM, 2002.

[10] Y. Miao, Z. Chen, S. Li, Z. Zhao, and Y. Zhou. A clustering-based strategy to identify coincidental correctness in fault localization. *International Journal of Software Engineering and Knowledge Engineering*, 23(05):721–741, 2013.

[11] W. E. Wong and V. Debroy. A survey of software fault localization. *Department of Computer Science, University of Texas at Dallas, Tech. Rep. UTDCS-45-09*, 2009.

[12] 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)*, 22(4):31, 2013.

[13] Y. Yu, J. A. Jones, and M. J. Harrold. An empirical study of the effects of test-suite reduction on fault localization. In *Proceedings of the 30th international conference on Software engineering*, pages 201–210. ACM, 2008.