LETTER

The Impact of Information Richness on Fault Localization

Yan LEI[†a)], ***Student Member***, **Min ZHANG**[†], **Bixin LI**[†b)], **Jingan REN**[†], ***and*** **Yinhua JIANG**[†], ***Nonmembers***

SUMMARY Many recent studies have focused on leveraging rich information types to increase useful information for improving fault localization effectiveness. However, they rarely investigate the impact of information richness on fault localization to give guidance on how to enrich information for improving localization effectiveness. This paper presents the first systematic study to fill this void. Our study chooses four representative information types and investigates the relationship between their richness and the localization effectiveness. The results show that information richness related to frequency execution count involves a high risk of degrading the localization effectiveness, and backward slice is effective in improving localization effectiveness.

key words: *fault localization, automated debugging, information richness, experimental study*

1. Introduction

Spectrum-based Fault Localization (SFL) [1] is a promising technique widely studied in the fault localization community. SFL usually utilizes Binary information of execution Count (BC), that is, information of each program statement being *executed* (denoted by the value '1') and *not executed* (represented by the value '0') by a particular test case. Based on BC and test results, SFL adopts a suspiciousness evaluation formula to evaluate the suspiciousness of each statement being faulty and gives a ranking list of all statements in terms of suspiciousness. Studies have empirically demonstrated that SFL is effective to improve fault localization effectiveness by reducing the percentage of code that needs to be examined to find the fault (*e.g.*, [1]–[3]).

One issue related to SFL is the effects of information richness on the effectiveness of fault localization. Information richness is associated with a specific information type, that is, information richness represents the amount of information expressed by a specific information type. Intuitively, a richer information type should contain more information and thus have a positive effect on fault localization effectiveness. In light of this intuition, some SFL techniques have adopted richer information types, rather than BC, to obtain some benefit to fault localization effectiveness. Lee *et al.* [4] propose a SFL technique using Frequency execution Count (FC) to refine the weights of each statement. FC is the number of times a statement is executed. Meanwhile, many SFL techniques have incorporated dependence information [5], [6] to define new rich information types to improve information richness for SFL (*e.g.*, def-use pairs [2], information flow [7] and context pattern [8]).

Despite the great progress in recent years, there still lacks a systematic study of the effects of information richness on the effectiveness of fault localization. It is necessary to investigate the effects of information richness on fault localization effectiveness to shed insight on how to enrich information for improving fault localization effectiveness, such as the reasonableness of the aforementioned intuition. Therefore, this paper aims to address the following three Research Questions (RQ):

RQ1: Does the general intuition always hold? In other words, does it always hold that a richer information type is used for fault localization, the more positive effects are made on fault localization effectiveness?

RQ2: If RQ1 does not hold, what type of information has a negative effect on fault localization effectiveness, and how much influence on fault localization?

RQ3: Furthermore, what type of information has the promising potential of having a positive effect on fault localization effectiveness, and how much benefit fault localization can obtain?

To achieve the above research goals, we present the results of our first experiment on the effects of information richness on fault localization effectiveness. The experiment first chooses three major information types used by SFL. The three types of information are BC [1], FC [4] and Backward Slice* (BS [9]). In addition, we follow the structure of SFL to construct a new SFL technique using a new information type that combines FC with BS (FC&BS).

Obviously, FC is richer than BC because FC contains frequency execution count rather than the information of each program statement being executed or not executed. BS uses dynamic dependence graphs [10] to show dynamic data/control dependence of the statements in a program whereas BC cannot show data/control dependence. In addition, dynamic dependence graphs can also show the information of a statement being executed and not executed by a test case. In this respect, BS contains more information than BC, that is, BS is richer than BC. Since FC&BS contains FC and BS, FC&BS should be the richest among the four types of information. FC cannot show the dependence information of statements and BS usually does not record the number of times a statement is executed for a test case. It demonstrates that FC and BS are two different directions of information richness for fault localization. Consequently, the richness relationship can be obtained among the four types

Manuscript received July 16, 2015.
Manuscript publicized October 14, 2015.

[†]The authors are with Logistical Engineering University, 401311, Chongqing, China.
a) E-mail: yanlei.cs@foxmail.com
b) E-mail: libixin000000@163.com (Corresponding author)

DOI: 10.1587/transinf.2015EDL8152

*In this paper, backward slice is the dynamic slice.

of information: BC<FC<FC&BS and BC<BS<FC&BS. As a reminder, BS is built on BC by eliminating irrelevant executed statements, that is, BS is equal to BC&BS in the methodology of SFL. Thus, we use BS for our study to denote them both.

Based on the four information types, four corresponding SFL techniques and five groups of the maximal SFL suspiciousness evaluation formulas are used for the experiment. The empirical results do not support an intuition that a rich information type should always have a positive effect on the effectiveness of fault localization. Our experiment also shows that richness related to FC can introduce a high risk of taking a negative effect on fault localization effectiveness, and BS is more helpful than the other three information types in improving fault localization effectiveness.

2. SFL Using FC&BS

Due to the space limit, this section just presents the new SFL technique using FC&BS. The details of SFL using BC, SFL using FC and SFL using BS can refer to [1], [4] and [9], respectively. First, we assume that a program P comprises a set of program statements $S = \{s_1, s_2, \ldots, s_M\}$ and runs against a set of test cases $T = \{t_1, t_2, \ldots, t_N\}$ (see Fig. 1). Let $bslice(t_i)$ be the backward slice [9] of the output of the test case t_i. Thus, $bslice(t_i)$ includes those statements whose execution affects the output of test case t_i according to data and control dependence.

In Fig. 1, the matrix $N \times (M+1)$ represents the input to SFL. An element x_{ij} is equal to the number of times statement s_j is executed (*i.e.* FC) for test case t_i if $s_j \in bslice(t_i)$ and 0 otherwise. It implies that x_{ij} records FC of those statements whose execution affects the output of test cases. The basic intuition is that a larger FC of a particular statement in a *bslice* of a test case indicates that the statement affecting the output of a test case is executed by the test case more frequently, and thus the statement should obtain a more weighted correlation with the influence on the output of the test case.

$$
\begin{array}{c}
\qquad\quad M \text{ statements} \qquad\quad \text{failed/passed} \\
N \text{ bslices} \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1M} \\ x_{21} & x_{22} & \cdots & x_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{NM} \end{bmatrix} \begin{bmatrix} r_1 & t_1 \\ r_2 & t_2 \\ \vdots & \vdots \\ r_N & t_N \end{bmatrix} \\
\qquad\quad \mathrm{s}_1 \quad \mathrm{s}_2 \quad \cdots \quad \mathrm{s}_M
\end{array}
$$

Fig. 1 Input to SFL.

Table 1 Maximal formulas of SFL.

Name		Formula
ER1'	Naish1	$\begin{cases} -1, \ if\ a_{01}(s_j) > 0 \\ a_{00}(s_j), \ if\ a_{01}(s_j) \le 0 \end{cases}$
	Naish2	$a_{11}(s_j) - \frac{a_{10}(s_j)}{a_{10}(s_j)+a_{00}(s_j)+1}$
	GP13	$a_{11}(s_j)(1 + \frac{1}{2a_{10}(s_j)+a_{11}(s_j)})$
ER5	Wong1	$a_{11}(s_j)$
	Russel&Rao	$\frac{a_{11}(s_j)}{a_{11}(s_j)+a_{01}(s_j)+a_{10}(s_j)+a_{00}(s_j)}$
	Binary	$\begin{cases} 0, \ if\ a_{01}(s_j) > 0 \\ 1, \ if\ a_{01}(s_j) \le 0 \end{cases}$
GP02		$2(a_{11}(s_j) + \sqrt{a_{00}(s_j)} + \sqrt{a_{10}(s_j)})$
GP03		$\sqrt{\lvert a_{11}(s_j)^2 - \sqrt{a_{10}(s_j)} \rvert}$
GP19		$a_{11}(s_j)\sqrt{\lvert a_{10}(s_j) - a_{11}(s_j) + a_{01}(s_j) - a_{00}(s_j) \rvert}$

Based on the new matrix, SFL using FC&BS follows the structure of SFL [4], and uses Eq. (1) and Eq. (2) to redefine the four sets and the four statistical variables.

$$
\begin{aligned}
&np(s_j) = \{i | (x_{ij} = 0) \wedge (r_i = 0)\}, \ ep(s_j) = \{i | (x_{ij} > 0) \wedge (r_i = 0)\} \\
&nf(s_j) = \{i | (x_{ij} = 0) \wedge (r_i = 1)\}, \ ef(s_j) = \{i | (x_{ij} > 0) \wedge (r_i = 1)\}
\end{aligned} \tag{1}
$$

$$
\begin{aligned}
&a_{00}(s_j) = \sum_{i \in np(s_j)} (1 - M_{ij}) + \sum_{i \in ep(s_j)} (1 - M_{ij}), \ a_{10}(s_j) = \sum_{i \in ep(s_j)} M_{ij} \\
&a_{01}(s_j) = \sum_{i \in nf(s_j)} (1 - M_{ij}) + \sum_{i \in ef(s_j)} (1 - M_{ij}), \ a_{11}(s_j) = \sum_{i \in ef(s_j)} M_{ij} \\
&\qquad\qquad Where, \ M_{ij} = \begin{cases} \frac{1}{e^{-\alpha * x_{ij}} + 1}, \ if\ x_{ij} > 0 \\ 0, \ otherwise \end{cases}
\end{aligned} \tag{2}
$$

Hence, in the context of SFL using FC&BS, $a_{00}(s_j)$ and $a_{01}(s_j)$ represent the cumulative weights related to those test cases whose output is *not affected* by the execution of statement s_j with a *passed* and *failed* result respectively. While $a_{10}(s_j)$ and $a_{11}(s_j)$ denote cumulative weights associated with those test cases whose output is *affected* by the execution of statement s_j with a *passed* and *failed* result respectively.

Recent research [11] has theoretically proven that nine suspiciousness evaluation formulas are the most efficient formulas (referred to as the maximal formulas), among which, three formulas are equivalent and constitute a group ER1', and other three equivalent formulas compose a group ER5. Table 1 describes the nine maximal formulas: Naish1 [3], Naish2 [3], GP13 [11], Wong1 [3], Russel&Rao [3], Binary [3], GP02 [11], GP03 [11] and GP19 [11], and shows how the suspiciousness of s_j is computed.

3. Experimental Study

3.1 Information Types and Subject Programs

To understand the impact of information richness on fault localization effectiveness, this study chooses four representative information types (BC, FC, BS and FC&BS), and the four corresponding SFL techniques. The experiment selects two widely used benchmarks (`Siemens` and `space`) in the field of fault localization. Table 2 shows the information of subject programs, and lists the programs, the number of faulty versions, the lines of code, the size of test pool, and the functional descriptions of the corresponding program. Because *print_tokens* and *print_tokens2* have similar structure and functionality, and each has only a few faulty versions, the experiment shows their combined results to give meaningful statistics. Following the same condition, we

Table 2 The summary of subject programs.

Program	Versions	Loc	Test	Description
print_tokens (2 ver.)	15	570/726	4,115/4,130	Lexical analyzer
replace	30	564	5,542	Pattern recognition
schedule (2 ver.)	18	374/412	2,650/2,710	Priority scheduler
tcas	37	173	1,608	Altitude separation
tot_info	19	565	1,052	Info. measure
space	35	6,199	13,585	ADL interpreter

Table 3 Statistical results on effectiveness relationship

Formula	Comparison	2-tailed	1-tailed(right)	1-tailed(left)	Conclusion
ER1'	SFL(FC&BS) v.s. SFL(FC)	1.91E-05	9.54E-06	0.99999	WORSE
	SFL(FC) v.s. SFL(BC)	1.07E-07	5.34E-08	1	WORSE
	SFL(BC) v.s. SFL(BS)	2.89E-07	1.44E-07	1	WORSE
ER5	SFL(FC) v.s. SFL(FC&BS)	0.01	0.005	0.99501	WORSE
	SFL(FC&BS) v.s. SFL(BC)	1.16E-05	5.81E-06	0.99999	WORSE
	SFL(BC) v.s. SFL(BS)	0	0	1	WORSE
GP02	SFL(FC) v.s. SFL(BC)	5.22E-06	2.61E-06	1	WORSE
	SFL(BC) v.s. SFL(FC&BS)	0.03425	0.01712	0.99888	WORSE
	SFL(FC&BS) v.s. SFL(BS)	0.00223	0.00112	0.98291	WORSE
GP03	SFL(FC) v.s. SFL(BC)	3.01E-05	1.51E-05	0.99998	WORSE
	SFL(BC) v.s. SFL(FC&BS)	0.04934	0.02467	0.97538	WORSE
	SFL(FC&BS) v.s. SFL(BS)	1.41E-04	7.06E-05	0.99993	WORSE
GP19	SFL(FC) v.s. SFL(BC)	2.39E-06	1.19E-06	1	WORSE
	SFL(BC) v.s. SFL(FC&BS)	2.29E-04	1.15E-04	0.99989	WORSE
	SFL(FC&BS) v.s. SFL(BS)	1.44E-05	7.19E-06	0.99999	WORSE

also combine the results of *schedule* and *schedule2*. In total, 154 faulty versions of the programs, obtaining from the Software-artifact Infrastructure Repository[†], were used for the experiment.

3.2 SFL Formulas and Evaluation Metric

Since ER1', ER5, GP02, GP03 and GP13 are the maximal suspiciousness evaluation formulas for SFL [11], our study uses these maximal formulas in the study, that is, we use Nash 1 out of the three equivalent formulas in ER1', Binary out of the three equivalent maximal formulas in ER5 and the other three formulas GP02, GP03 and GP19 (see Table 1). The experiment evaluates the effectiveness of SFL using the four information types with these five maximal formulas. In light of the equivalence in each group, the following section uses ER1' and ER5 to represent Naish1 and Binary respectively.

We evaluate fault localization effectiveness by using the widely used metric, that is, the percentage of executable code that needs to be examined before finding the actual faulty statement (referred to as the *EXAM* score [3]). A lower *EXAM* score indicates higher effectiveness.

3.3 Results and Discussion

We compare the *EXAM* between each two SFL techniques by adopting the paired Wilcoxon-Signed-Rank test [12]. Thus, we can obtain the effectiveness relationship among the four SFL techniques, and Table 3 shows the statistical results on this relationship. Take SFL(FC&BS) v.s. SFL(FC) in ER1' as an example. The p values of 2-tailed, 1-tailed(right) and 1-tailed(left) are 1.91E-05, 9.54E-06 and 0.99999 respectively. It means that the *EXAM* of SFL(FC&BS) is significantly greater than that of SFL(FC). Therefore, we obtain a WORSE conclusion, that is, SFL(FC&BS) performs worse than SFL(FC) in ER1'. Based on the results in Table 3, we can obtain the effectiveness relationship as follows:

- **ER1:** SFL(FC&BS)<SFL(FC)<SFL(BC)<SFL(BS).
- **ER5:** SFL(FC)<SFL(FC&BS)<SFL(BC)<SFL(BS).
- **GP02, GP03 and GP19:** SFL(FC)<SFL(BC)<SFL(FC&BS)<SFL(BS).

Based on the above results, we can observe that SFL using FC performs worse than SFL using BC, and the effectiveness of SFL using BS decreases after incorporating FC. This shows that the information type FC has a negative effective on fault localization effectiveness despite the fact that FC is richer than BC and FC&BS is richer than BS. Let us take the faulty program in Fig. 2 as an example to understand the problem. This program with a fault at statment s_8 runs against five test cases. The cells below each statement indicate the execution times of the statement in a test case, and the rightmost cells represent whether the execution of a test case is failed or not. As shown in Fig. 2, s_5, s_6, s_7 and s_8 are executed by failed test cases t_1 and t_3, and they are not executed by passed test cases t_2, t_4 and t_5. Since s_5, s_6, s_7 and s_8 have the same binary information of execution count in the five test cases, SFL using BC [1] will assign the same suspiciousness value to these statements. However, the number of execution times of s_5, s_6, s_7 in failed test cases t_1 and t_3 is much higher than that of s_8. SFL using FC [4] will assign more weights to s_5, s_6 and s_7, and thus the suspiciousness values of these statements are higher than the suspiciousness value of s_8. Hence, the rank of the faulty statement s_8 will become lower in SFL using FC compared with the rank of s_8 in SFL using BC, that is, SFL using FC decreases the effectiveness of SFL using BC. Recall that the basic intuition of using FC is that the larger FC of a particular statement in a test case should have a larger correlation or affection with the output of the test case. Nevertheless, the FC of a statement is not generally consistent with the correlation or affection of a statement on the output of the test case. Obviously, SFL using FC or FC&BS can perform well in those cases that faulty statements need to be frequently executed to accumulate effects for triggering a failure. The experimental study shows that such cases do not always hold in practice. Furthermore, the frequently executed non-faulty statements, such as loop statements, can always obtain more benefits than the infrequently executed faulty statements in light of FC, and thus these statements have a high probability of surpassing faulty statements in suspiciousness being faulty. That is the reason why SFL using FC or FC&BS performs worse than that SFL using BC or BS despite the previous preliminary experiment [4] showing that SFL using FC can obtain more benefit than SFL using BC.

On the other hand, we observe that SFL using BS [9] performs the best among the four SFL techniques and shows a significant improvement in terms of fault localization effectiveness. Recall that the intuition of using BS is that the execution of a particular statement in a test case should have

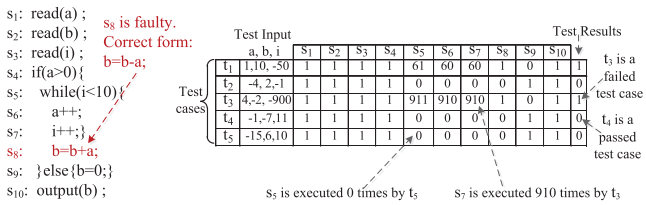

Fig. 2 An illustrative example.

[†]http://sir.unl.edu/portal/index.php

a strong correlation with the output of the test case when the execution of the statement affects the output of the test case. In contrast, in the framework of SFL using code coverage (e.g. BC and FC), the execution of a statement should have a strong correlation with the output of the test case when the statement is just or frequently executed. Nevertheless, the execution of a statement in a test case does not necessarily mean that the execution of the statement affects the output of the test case. In general, faulty statements cannot trigger a program failure unless their execution affects the output. Thus, SFL using BS adopts stronger semantics to evaluate the correlation between statements and failures compared to SFL using code coverage. In addition, it has been demonstrated that BS is effective to capture the influence of the execution of a statement on the output and can locate a wide spectrum of faults [13], that is, BS is generally and strongly correlated with the output of a test case. That is the reason why SFL using BS shows the highest fault localization effectiveness among the four SFL techniques.

3.4 Answers to RQs

Answer to RQ1 Given the same evaluation structure of SFL, the results show that the effectiveness relationship of the four SFL techniques does not conform to the richness relationship of the four corresponding information types used. In other words, it does not always hold that, as a richer information type is used for fault localization, the more positive effects are made on fault localization effectiveness. In fact, some rich information types can have a significant negative effect on fault localization effectiveness. For example, SFL with the richest information type FC&BS does not perform the best among the four SFL techniques and even performs the worst in ER1 compared to the other three SFL techniques. It suggests that researchers should carefully choose or propose a rich information type to enrich information for fault localization and avoid negative effects on fault localization effectiveness.

Answer to RQ2 After incorporating FC into SFL, SFL using FC performs worse than SFL using BC, and SFL using FC&BS also performs worse than SFL using BS. The reason behind this phenomenon is that it does assume that faulty statements need to be frequently executed to accumulate effects for triggering a failure. However, the current methodology of using FC presents a bias towards those frequently executed statements. If the FC of faulty statements is lower than that of many non-faulty statements, the negative effects on fault localization effectiveness would accumulate and finally lead to a significant effectiveness decrease. Consequently, information richness related to FC has a negative effect on fault localization effectiveness. Researchers should be aware of the negative effects of FC on fault localization effectiveness, that is, researchers should be cautious of using FC, at least in the framework of SFL, to enrich information for improving fault localization effectiveness.

Answer to RQ3 For all investigated formulas, SFL using BS performs the best among the four SFL techniques, that is, BS is helpful in improving fault localization effectiveness. Since BS can capture the influence of the execution of a statement on the output, SFL using BS can define statistical variables with stronger semantics for suspiciousness evaluation formulas as compared to SFL using code coverage (e.g. BC and FC). BS owes this advantage to the use of dependence information because dependence information can construct a causal chain of how data and control propagates in a program. In other words, dependence information can introduce more semantics into the analysis process of fault localization than code coverage. Thus, dependence information have more positive effects than code coverage on fault localization effectiveness. With this useful guidance, researchers are encouraged to use dependence information to improve the effectiveness of fault localization.

4. Conclusion

In this paper, we present the first experimental study that evaluates the effects of information richness on fault localization effectiveness. We provide the evidence that contradicts an intuition that a rich information type should always be helpful in improving fault localization. We also show that different types of information richness can have different (positive or negative) effects on fault localization effectiveness, providing advices for researchers on how to enrich information for improving fault localization effectiveness.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (Nos. 61502015 and 61502296), the STCSM project (No.15ZR1417000), and the Funding Scheme for Training Young Teachers in Shanghai Colleges (No. ZZegd14001).

References

[1] R. Abreu, P. Zoeteweij, and A. Van Gemund, "On the accuracy of spectrum-based fault localization," Proc. Testing: Academic and Industrial Conference Practice and Research Techniques-MUTATION, pp.89–98, IEEE, 2007.
[2] R. Santelices, J. Jones, Y. Yu, and M. Harrold, "Lightweight fault-localization using multiple coverage types," Proc. 31st International Conference on Software Engineering (ICSE 2009), pp.56–66, IEEE, 2009.
[3] L. Naish, H. Lee, and K. Ramamohanarao, "A model for spectra-based software diagnosis," ACM Trans. Software Engineering and Methodology (TOSEM), vol.20, no.3, p.11, 2011.
[4] H.J. Lee, L. Naish, and K. Ramamohanarao, "Effective software bug localization using spectral frequency weighting function," Proc. 34th Annual Computer Software and Applications Conference (COMPSAC 2010), pp.218–227, IEEE, 2010.
[5] M. Weiser, "Program slicing," IEEE Trans. Softw. Eng., vol.10, no.4, pp.352–357, 1984.
[6] B. Korel and J. Laski, "Dynamic program slicing," Information Processing Letters, vol.29, no.3, pp.155–163, 1988.
[7] W. Masri, "Fault localization based on information flow coverage," Software Testing, Verification and Reliability, vol.20, no.2, pp.121–147, 2010.
[8] X. Wang, S. Cheung, W. Chan, and Z. Zhang, "Taming coincidental

correctness: Coverage refinement with context patterns to improve fault localization," Proc. 31st International Conference on Software Engineering (ICSE 2009), pp.45–55, IEEE Computer Society, 2009.
[9] X. Mao, Y. Lei, Z. Dai, Y. Qi, and C. Wang, "Slice-based statistical fault localization," J. Systems and Software, 2014.
[10] H. Agrawal and J.R. Horgan, "Dynamic program slicing," Proc. ACM SIGPLAN 1990 Conference on Programming Language Design and Implementation (PLDI 1990), pp.246–256, ACM, 1990.
[11] X. Xie, F.C. Kuo, T.Y. Chen, S. Yoo, and M. Harman, "Provably optimal and human-competitive results in sbse for spectrum based fault localisation," in Proc. 5th Symposium on Search-Based Software Engineering (SSBSE 2013), pp.224–238, Springer, 2013.
[12] G.W. Corder and D.I. Foreman, Nonparametric statistics for non-statisticians: a step-by-step approach, John Wiley & Sons, 2009.
[13] X. Zhang, N. Gupta, and R. Gupta, "A study of effectiveness of dynamic slicing in locating real faults," Empirical Software Engineering, vol.12, no.2, pp.143–160, 2007.