Software Fault Localization: Issues and Limitations

Abubakar Zakari\textsuperscript{A,B}, Sai Peck Lee\textsuperscript{A}
\textsuperscript{A} Department of Software Engineering, Faculty of Computer Science and Information Technology, University of Malaya, 50603, Kuala Lumpur, Malaysia. 
\textsuperscript{B} Kano University of Science and Technology, Wudil, Kano, Nigeria. 
E-mail: abubakar.zakari@yahoo.com; saipeck@um.edu.my

ABSTRACT
Fault localization is the most challenging and tedious activity in program debugging. This activity is vital in maintaining software quality. Existing fault localization techniques assume that a program contains only one fault during the localization process. Realistically, a program failure can be caused by multiple active faults. However, for a program with multiple faults, interference do occur between faults that mask failures, this causes the existing fault localization techniques to lose their localization effectiveness with a great margin. Hence, techniques that try to solve this problem are suffering from lack of scalability and high computational complexity. In this paper, we discussed and analyzed the existing fault localization techniques and their application to multiple fault programs, and highlight some of their issues and limitations. Based on our findings, existing fault localization techniques based on multiple faults are shallow and more improvement is needed. Nevertheless, techniques that are relatively effective in localizing multiple faults lack practical generality.

Keywords: Automated Fault Localization, Debugging, Fault Interference, Spectrum-based Fault Localization

INTRODUCTION
Software systems are complex and this complexity contributes to the increase in the number of faults in the system that led to the increase in software maintenance cost (Cleve and Zeller 2005). Program debugging is a very important activity in reducing software maintenance cost (Kuhn et al. 2004; Zakari et al. 2016). Program debugging has three main activities which are, failure detection, fault localization and fault repair (Lian et al. 1997). Fault localization is a costly activity in program debugging, a developer is tasked with identifying the exact locations of faults in a software program. Primarily, manual techniques are utilized by software developers to identify faults locations (Hennessy 1982; Rosenblum 1995), but due to the high complexity of software programs, the utilization of manual techniques is very difficult.

Various studies have been done to automate the fault localization process (Jones and Harrold 2005; Jones et al. 2002; Neelofar et al. 2017; Pearson et al. 2017; Podgurski et al. 2003; Wong and Qi 2009). Nonetheless, the existing automated fault localization techniques effectiveness decreases once used on programs with multiple faults, due to
fault interference phenomenon (DiGiuseppe and Jones 2011b). Test cases that failed in the presence of single fault passed if two faults are active, and test cases that passed in the presence of single faults fails when a single fault is active. The former has the highest rate of occurrence. Empirical studies have been done to investigate this phenomenon (Debroy and Wong 2009; DiGiuseppe and Jones 2011a; Xue and Namin 2013).

Furthermore, various existing techniques use the one-fault-at-a-time approach to localize multiple faults. Based on this approach, a fault will be localized and fixed independently, then the program will be re-tested to find and fixed the rest of the faults. This process will be repeated until the program is fault-free. But, this approach contributes to the introduction of more faults to the program (Jones et al. 2007). Model-based techniques have good accuracy in identifying the locations of software faults, but they have issues of scalability and high computational complexity that limit their application in practice.

In this paper, we discussed existing fault localization techniques in detail and highlight some of their issues and limitations. The paper is organized as follow. Section 2 contains the background of our study. Section 3 discusses issues and limitations. Lastly, Section 4 provides the conclusion.

BACKGROUND

Manual debugging techniques are utilized in the early years for fault localization. Using this technique, developers use assertion or breakpoints to identify faults locations (Hennessy 1982; Rosenblum 1995). However, the techniques are not sustainable because of the increase in software complexity (Jones et al. 2002). To decrease maintenance cost, researchers have proposed various automated fault localization techniques to aid in effective localization of software faults. The earlier techniques proposed are slice based techniques (Agrawal et al. 1995), this techniques aids developers in reducing the search space in identifying software faults. There are two main slice based techniques which are static slicing and dynamic slicing based techniques (Agrawal and Horgan 1990).

However, in static based technique the slice can sometimes be the whole program, therefore, dynamic slicing technique was proposed to further reduce the fault search space (Korel and Laski 1988). Dynamic slicing technique uses dynamic execution behavior to reduce the fault search space (Agrawal and Horgan 1990). Spectrum-based fault localization technique (SBFL) was proposed in (Jones et al. 2002). SBFL techniques are regarded as one of the most effective fault localization techniques (Wong et al. 2016). The techniques use program spectra and test execution result (pass/fail) to identify program statements that may contain faults. SBFL techniques are also named as lightweight fault localization techniques because of their simplistic nature. Wong et al (Wong et al. 2014) proposed a DStar method for effective fault localization. It is an enhanced form of kulczynski coefficient metric (Abreu et al. 2007).
On the other hand, Model-based diagnosis techniques (MBD) have good localization accuracy with strong mathematical background (Abreu and van Gemund 2009; Abreu et al. 2009). MBD techniques produce program models based on program execution behavior to help in the identification of failure root causes. Machine learning techniques (Briand et al. 2007; Wong et al. 2012; Wong and Qi 2009; Zheng et al. 2016) are robust and adaptive and are one of the most effective fault localization techniques. In some cases, they showed better performance than SBFL techniques (Zheng et al. 2016).

**Localizing Multiple Faults**

This section highlights the existing techniques/approaches proposed for localizing multiple faults.

Parallel debugging was proposed by Jones et al in (Jones et al. 2007). Failed test cases are clustered into fault focused clusters, and each fault focused cluster is combined with all passed test cases for fault localization. Hence, each combination will be given to developers to locate the faults in parallel. More parallel debugging approaches are proposed such as (Högerle et al. 2014; Srivastav et al. 2010) to improve debugging. A weighting approach is proposed in (Lee et al. 2016), the approach includes information mined from failed test cases that were caused by multiple faults and weight them to enhance localization effectiveness. In this study (Lamraoui and Nakajima 2016), a formula-based approach is proposed. The approach uses SAT-based formula verification techniques and Reiter’s model-based diagnosis theory to understand the root causes of faults in a program. The main limitation of this approach is its high computational complexity which limits the approach utilization to only small programs with few hundred lines of code.

To improve localization efficiency in the context of multiple faults a parameter-based combination approach is proposed in (Wei and Han 2013). Failed test cases are clustered into fault-focused clusters with different combinations, a crosstab-based fault localization technique is used for fault identification. In this studies (DiGiuseppe and Jones 2011a; DiGiuseppe and Jones 2011b; Xue and Namin 2013), one-fault-at-a-time approach is utilized for fault localization and fault interference phenomenon is rigorously investigated in relation to impact on localization effectiveness. The authors concluded that interference between faults do occur with destructive interference being the most prevalent. In this study (Abreu et al. 2009), Zoltar-M was proposed for localization of multiple faults.

**ISSUES AND LIMITATIONS**

In this section, we highlight some vital issues and limitations of existing fault localization techniques. SBFL is one of the most effective fault localization techniques. Using this technique, researchers generally utilized the one-fault-at-a-time approach to localize multiple faults. This contributes to the increase in both time-to-delivery and software maintenance cost. Moreover, the effectiveness of SBFL techniques decreases when one-fault-at-a-time approach is utilized in the context of multiple faults because of fault
interference phenomenon. Also, the use of one-fault-at-a-time approach in a program with thousands or million line of codes is time-consuming and costly. Therefore, simultaneous approaches are needed to further localize faults simultaneously. MBD techniques mostly localize multiple faults simultaneously, but due to their high computational complexity their utilization on large software programs is limited.

CONCLUSION

Software fault localization is important in maintaining software quality and reducing cost in software development. This paper discusses some state-of-the-art fault localization techniques and insights are given on the effects of fault interference phenomenon on localization effectiveness. Moreover, we highlight existing multiple faults localization techniques and approaches on their contributions in improving localization effectiveness in the context of multiple faults. Our findings show that, scalability and high computational complexity the major limitations of MBD techniques. Additionally, studies on SBFL mostly used one-fault-at-a-time approach on multiple fault programs. Moreover, this approach contributes to the loss of effectiveness of SBFL techniques. Conclusively, for the existing fault localization techniques to be practically applicable in the software industry, these vital issues need to be addressed.

REFERENCES


Abubakar Zakari, Sai Peck Lee


