Assisting Students in Finding Bugs and their Locations in Programming Solutions

Long H. Pham, Ho Chi Minh City University of Technology, Ho Chi Minh, Vietnam
Giang V. Trinh, Ho Chi Minh City University of Technology, Ho Chi Minh, Vietnam
Mai H. Dinh, Ho Chi Minh City University of Technology, Ho Chi Minh, Vietnam
Nam P. Mai, Ho Chi Minh City University of Technology, Ho Chi Minh, Vietnam
Tho T. Quan, Ho Chi Minh City University of Technology, Ho Chi Minh, Vietnam
Hung Q. Ngo, State University of New York at Buffalo, Buffalo, NY, USA

ABSTRACT

Teaching experience shows that programming is time consuming and can be acquired with substantial practice. Besides, students need to know whether their solutions are correct or incorrect and the root causes of their errors. Thus, teaching programming in a large class requires considerably many teaching assistants, which is costly. More importantly, a communication means that can support students virtually any time is also desirable. In order to handle these problems, a static method was applied to build an online intelligent tutoring system that can assist students checking their solutions. In addition, when detecting the errors, this system can suggest students to investigate the suspected code. This feature is really significant for students to self-practice and improve their learning.

Keywords: Fault Localization, Group Testing, Intelligent Tutoring System, Program Verification, Programming Exercises

INTRODUCTION

Programming is essential for any computer science study. Research and experience show that practicing with problems is the best method to master the programming skills. In traditional education, class tutors have to read students’ programs to verify their correctness. However, the number of programs is often too much and code reading is error-sensitive. This explains why the method is less effective in programming courses.

Automated assessment system is a good solution for this problem. This system checks whether the students’ programs qualify the test cases in an automatically generated test suite.
(Ala-Mutka, 2005; Douce et al., 2005; Jurado et al., 2012; Ihantola et al., 2010; Kaushal & Singh, 2012). However, this approach has some disadvantages. For example, the test suite must be large enough to cover all possible errors in the program and executing possibly buggy program is potentially dangerous for the system.

To overcome such obstacles, **static methods** are proposed to statistically verify the programs correctness without running the programs. Two statistical methods referred as **theorem proving** and **model checking** are combined to build a web-based tutoring system (Quan et al., 2009). These methods are also known as **formal methods**, whereby mathematics-based techniques are used to check the program properties. While theorem proving can verify the program correctness, model checking can generate counter examples to help trace down the faults (bugs) via the corresponding execution paths if the program is false.

However, this system can only help learners to be aware of programs correctness. It cannot help locate the root causes of the problems effectively since the generated counter examples are too complicated for students to follow. Moreover, determining the root cause locations alongside the execution paths provided by the counter examples is non-trivial, especially for novice learners in programming.

Thus, it is intuitively more convenient if identification of the program parts is identified. This will most likely contain the root causes and can be showed to the students for further investigation. This was the motivation for the researchers to develop a framework in which the above system is combined with **group testing** and **slicing spectrum-based fault localization** to achieve the goal. Group testing is a simple yet powerful method to locate faults in the program, slicing spectrum-based fault localization on the contrary, is more complex. However, it can help to find bugs in case group testing gives the wrong answer. So group testing and slicing spectrum-based fault localization act as complements in the system. The above framework was experimented with non-trivial exercises in Programming Fundamentals course and positive results were recorded. This paper presents the proposed framework. Subsequently, group testing and slicing spectrum-based fault localization are also presented. This is followed by the presentation of two case studies. The final section concludes the paper.

**THE PROPOSED FRAMEWORK**

Figure 1 depicts the proposed framework on three components: Teacher, Learner and Coordinator. The Teacher provides the programming problems presented in the Problem Description Module. When the Learners visit the system, they can try to solve these problems. The Learner’s submitted program is verified by the Correctness Proving module. The Model Checking component will identify and show the execution paths which lead to error, if the program is false.

When the Learner wants to find buggy portions in the program, Group Testing and Fault Localization modules will detect them. The process of detecting buggy portions uses test cases generated by Test Cases Generator module. In this module, test cases generated by **constraint-based test-cases generation algorithm** (Le et al., 2013) were combined with random test cases (although test cases are generated randomly, they are chosen to cover different execution paths in the program) to have a suitable test that can identify as many failed execution paths in the program as possible. In the system, instead of actually running the program, a simulator is used to find the program output according to the test cases. With this way, the system is safe and students can be influenced not to use bad programming structures.

The Coordinator can use the system information such as common errors or behaviors of active learners to assess the course performance. This information is stored and analyzed in the Analysis Module. The Information Exchange module helps other modules exchange data. These data are XML-based and they are stored in the database for convenient. The above framework has already been implemented as
Technology Adoption in Engineering Design for Distance Education
Amanullah M.T.O., Jaideep Chandran and Alex Stojcevski (2014). International Journal of Quality Assurance in Engineering and Technology Education (pp. 54-64).
www.igi-global.com/article/technology-adooption-in-engineering-design-for-distance-education/111949?camid=4v1a

The Smart Women – Smart State Strategy: A Policy on Women’s Participation in Science, Engineering and Technology in Queensland, Australia
www.igi-global.com/chapter/smart-women-smart-state-strategy/43200?camid=4v1a