Information and Software Technology 79 (2016) 79–105

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Evaluation and analysis of incorporating Fuzzy Expert System approach into test suite reduction

Chin-Yu Huang[a,b,*], Chung-Sheng Chen[a], Chia-En Lai[b]

[a] Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan
[b] Institute of Information Systems and Applications, National Tsing Hua University, Hsinchu, Taiwan

ARTICLE INFO

Article history:
Received 11 September 2015
Revised 4 June 2016
Accepted 21 July 2016
Available online 27 July 2016

Keywords:
Test suite reduction;
Tie-breaking;
Software testing;
Test suite minimization;
Fault detection effectiveness;
Fuzzy logic;
Test case prioritization

ABSTRACT

Context: Software has become increasingly important in our modern society. However, when new features are developed due to user requests, such requests could make the sizes of test-case pools bigger. Many techniques are proposed to solve this problem, such as test suite reduction. However, the ability to expose faults may be weakened when reducing the sizes of the test suites. In this paper, we propose some methods using fuzzy logic in order to improve existing test-suite reduction techniques.
Objective: The main purpose of this research is to use a Fuzzy Expert System approach in order to enhance the effectiveness of fault detection during software testing.
Method: Incorporating a Fuzzy Expert System into traditional test suite reduction techniques is presented and studied. More objective criteria are used in order to compare the performance of our proposed and selected test suite reduction methods. Some important measures (and metrics) will also be obtained and discussed. Application of the Fuzzy Expert System approach for test case prioritization is also discussed.
Results: The experiments in three improved test-suite reduction techniques show that the modified algorithms can reduce the size of test suites, which have improved the fault detection quality.
Conclusion: During software testing, test data are generally classified with Boolean logic. This method can classify data into groups easily. However, there may be ambiguity in classifications due to similar properties for certain data. Ambiguous data can be classified in each group. In this study, Boolean logic will be replaced by fuzzy logic. Incorporating the Fuzzy Expert System approach into three traditional test suite reduction techniques (i.e., HGS, GRE, and Greedy) is presented and evaluated. The experiments, based on nine real subject programs ranging from 173 LOCs to 35,545 LOCs, have demonstrated that our proposed Fuzzy-HGS, Fuzzy-GRE, and Fuzzy-Greedy algorithms can significantly reduce the sizes of test suites while also improving fault detection effectiveness. For instance, Fuzzy-HGS, Fuzzy-GRE, and Fuzzy-Greedy algorithms have almost the same reduction capability of test suite as traditional HGS, GRE, and Greedy algorithms. But in terms of the percentage of fault detection effectiveness loss (FDE loss), both Fuzzy-HGS and Fuzzy-GRE algorithms are averagely decreased by 21% and 5%, respectively. Additionally, Fuzzy-Greedy algorithm still provide the lower FDE loss for large subject programs compared to traditional Greedy algorithm. Based upon the integrated theoretical foundation, the approaches presented in this paper offer an efficient, useful software testing scheme in the testing and debugging phases.

1. Introduction

In today's technological world, software plays a key role in many safety-critical (or life-critical) systems that often requires an exceedingly rigorous certification process. Software testing is generally used to verify whether or not the developed software system meets those requirements. When a software project is entering the testing phase, the development team usually has a well-specified set of test cases that should be run within the minimal time. A test case is a document that has a set of test data, predetermined input, expected output, and behavior. According to the IEEE Standard 829-1998 [1], test cases usually identify the constraints on the test procedures resulting from a specific test case, which are then separated from test designs; separation is performed in order to allow for use in more than one design and for reuse in other situations. A test suite is a collection of test cases that are grouped and run together.

Traditionally, test data can be classified into groups by their properties. However, there is ambiguity in such a classification method due to similar properties for certain data. If ambiguous

* Corresponding author.
E-mail address: cyhuang@cs.nthu.edu.tw (C.-Y. Huang).

http://dx.doi.org/10.1016/j.infsof.2016.07.005
0950-5849/

data are classified into groups, they might be classified correctly. However in some cases, such data can be classified incorrectly into each group. A test suite typically contains both detailed instructions for selected test cases and important information on system configuration to be used during software testing. During the software development life cycle (SDLC), test suite reduction (or selection) demonstrates such a phenomenon. In many test cases, testers have to reduce them or select the most suitable test suites due to resource limitations. Thus, each test case must be classified into either the selected or the unselected groups. Yet there are some test cases in the unselected group that have better properties in testing, and vice versa.

Therefore, managing the size of test cases is an important issue in SDLC. More precisely, determining *which test case to pick, which test case to run next*, or *which test case to remove* is desirable. In the past, numerous approaches have been proposed to resolve this problem, such as *test suite reduction* (also called *test case minimization*). There are some heuristic Greedy algorithms applied to solve this problem by repeating the following steps: (1) pick a test case that satisfies the most requirements, or pick an arbitrary test case if there is more than one candidate, and (2) mark all of the requirements satisfied by the test case picked from step (1). The two steps are repeated until all requirements are satisfied [2,3].

For example, there is a heuristic algorithm proposed by Harrold et al. [4] which finds a representative test suite from a test case pool and that satisfies all requirements. However, this method makes for a great deal of effort in the recursive test case selection. For these existing heuristic algorithms, the size of test suites can be reduced. But in practice, the project managers or developers need further information about how well the test suites have been reduced when compared with the original test suites. Since one of the purposes of software testing is to detect and predict the remaining faults, the fault detection capability has to be measured and evaluated. Although much research has focused on test suite reduction, selecting appropriate test cases- such as those with better fault detection capability- is more useful than adding more test cases in reduced test suites. Further, due to the growth of test case pools, execution time in testing is also an important issue to discuss. In the past, Lin and Huang [8] reported that a tie occurs if more than one test case has the same importance, and thus have proposed a tie-breaking technique.

Based on our past research, in this paper, we further propose and adopt the Fuzzy Expert System (FES) approach [9] in order to enhance the fault-detection capability, which uses the concept of partial members in each group to classify the test cases. We will incorporate the FES approach into some traditional (and well known) test suite reduction algorithms, such as the Harrold–Gupta–Soffa (HGS) algorithm proposed by Harrold et al. [4], the GRE algorithm presented by Chen and Lau [10,11], and the Greedy algorithm [2,3]. Experiments are performed and discussed based on nine real subject programs ranged from 173 LOCs to 35,545 LOCs [12,13]: *tcas, schedule2, schedule, totinfo, printtokens2, printtokens, replace, space*, and *make*. In order to check the performance of our proposed methods, make a fairly comprehensive comparison with other existing test suite reduction methods, and avoid bias, we use five criteria in this paper.

Our experiments show that our proposed Fuzzy-HGS, Fuzzy-GRE, and Fuzzy-Greedy algorithms not only have the better test suite reduction capability but also have better overall performance on the fault detection capability and the execution time. For example, the original suite size for *tcas* program of 173 LOCs is 38.62. After performing the suite size reduction, the reduced suite size is 5.00 and 5.07 for traditional HGS and our proposed Fuzzy-HGS algorithms, respectively. With almost the same reduced suite size, the Fuzzy-HGS algorithm just takes 1.331 s for performing the suite size reduction, which is far more less than 5.718 s for traditional HGS algorithm. Additionally, the Fuzzy-HGS algorithm still has better fault detection capability compare to traditional HGS algorithm. And further, the original suite size for *space* program of 9564 LOCs is 1817.44. Similarly, after performing the suite size reduction, the reduced suite size is 119.98 and 119.44 for traditional HGS and our proposed Fuzzy-HGS algorithms, respectively. We can see that the Fuzzy-HGS algorithm only takes 64.962 s for performing the suite size reduction, but traditional HGS algorithm takes up to 3037.13 s. It is obvious that our proposed Fuzzy-HGS algorithm consistently performs better than traditional HGS algorithm.

The rest of this paper is organized as follows. In Section 2, we give a detailed survey of the test suite reduction techniques. The reviews of some traditional test suite reduction techniques are described in Section 3. Section 4 discusses and shows how the traditional test suite reduction algorithms can be integrated with the FES approach. Experiments based on real subject programs are performed in order to assess the performance of our proposed approaches in Section 5. Additionally, the discussions on threats to validity are also given in Section 5. The application of the FES approach to prioritize test cases is presented in Section 6. Finally, some concluding findings are described in Section 7.

2. Related works

Software testing is typically used to verify whether or not the developed software meets its original requirements. In the phase of software testing, testers usually have to ensure the correctness of software by using different testing techniques and test cases. Generally, developers need to find the major root causes of these detected faults and then eliminate them in order to reduce the reoccurrence of faults and the risks of software-failure. Altogether, the quality of developed software can be significantly increased and the risk of project-failure can be greatly reduced if we are able to do more tests and identify the most error-prone modules that are difficult to maintain as early as possible. However, this could cause the size of test cases to become larger during software testing, while large test case pools usually extend the project release time. It is also time-consuming to rerun all the test cases when a subject program is modified.

There is much research on test suite reduction techniques. For example, Rothermel [5] observed that the test suite reductions can compromise fault detection effectiveness. Wong [6] reported that fixed size test sets may lead to a larger size reduction due to block minimization than those of fixed coverage test sets, because of the way they are generated. Wong also found that there is little or no reduction in fault detection effectiveness for both types of test sets. Additionally, Jeffrey and Gupta [7] proposed an approach called Reduction with Selective Redundancy (RSR) in order to improve fault detection capability. RSR selectively retains test cases during test suite reduction. However, the extra process will increase test time and overhead of reusing test suites. Heimdahl and George [14] performed an experiment using a large case example of a Flight Guidance System, generated reduced test-suites for a variety of structural coverage criteria while preserving coverage, and recorded their fault finding effectiveness. Their evaluation results showed that the size of the specification-based test-suites was dramatically reduced and that the fault detection of the reduced test-suites was adversely affected.

Additionally, Sampath and Bryce [15] proposed ordering the test cases in a reduced test suite in order to increase its rate of fault detection, and Mansour and El-Fakih [16] adapted a hybrid genetic algorithm to the test suite reduction problem. On the other hand, Harder et al. [17] proposed a method to minimize the test suites by considering the operational abstraction. Harder et al. argued that an operational abstraction is a formal mathematical description of the actual behavior of a program; further, the test case(s) that