An Effective Test Case Selection for Software Testing Improvement

Adtha Lawanna
Department of Information Technology
Vincent Marry School of Science and Technology, Assumption University
Bangkok, Thailand
adtha@scitech.au.edu

Abstract— **One problem of testing software is selecting the suitable test cases from the test suit regarding the size of the programs. If the size of selected test cases is big, then it can affect the whole performance of software development life cycle. Accordingly, it increases testing time and produce many bugs. Therefore, this paper proposes the improvement of software testing for selecting the appropriate and small number of test cases by considering the amounts of the functions modified, lines of code changed, and numbers of bugs produced after modifying programs. The reason of proposing the software testing improvement model is to prepare effective algorithm, while numbers of bugs are lower than the traditional methods. According to the experimental results, the size of the selected test cases by using the proposed model is less than Retest All, Random, and a Safe Test about 98.70%, 87.86%, and 84.67% respectively. Moreover, the ability of STI is higher than the comparative studies about 1-20 times regarding the number of bugs found after modifying a program.**

Keywords—software testing; test case; code; bugs

I. INTRODUCTION

Software engineering is applied for several fields such as computer science, system dynamics, system science, and management system [1]. The software development life cycle is a methodology used in this field [2]. However, the major problem of software testing, which is studied in this paper refers to choosing the appropriate test cases, which contain bugs, functions, and any errors [3]. The serious problem is the size of the selected test cases is too big when modifying program each version [4-5]. This causes the testing time and errors increase. Therefore, this paper presents the model to solve this problem. The retest all method, random technique and a safe test are used for the comparisons. The studies show that method of retesting all possible cases is simple but it introduces time consuming during testing the software. While, the random technique is easier than the previous method, when testing some test cases selected from the whole program. Unfortunately, that it cannot guarantee the accuracy of auditing the software [6]. Another is a safe test technique, which gives the better performance in term of reducing many ineffective test cases, while few bugs are produced compared with the old approaches [7-8]. To the survey, some traditional test case selection techniques work effectively regarding the complexity of codes, environments, and user requirements [9]. This paper studies three factors that can affect the test case selection, which are functions, codes, and software versions. The traditional methods mentioned earlier can be applied for these environments. However, the size by chosen test cases and numbers of bugs after using these techniques need the improvement for better performance, especially in the process of software testing. Therefore, the proposed model named, Software Testing Improvement (STI) is developed for improving the ability of choosing the test cases, while the minimum test cases and bugs are reachable.

II. THE CONCEPT OF SELECTING THE TEST CASES

A. Dataset

The seven subject programs developed by the Siemen Suite are used in this paper as shown in Table I.

The details of each program can be downloaded from http://pleuma.cc.gatech.edu/aristotle/Tools/subjects.

TABLE I. THE SUBJECT PROGRAMS

Name	*F*	*L*	*V*
replace	21	516	32
print_token	18	402	7
print_token2	19	483	10
schedule2	16	297	10
schedule	18	299	9
totinfo	7	346	23
tcas	9	138	41

Definitions;
F is the numbers of function.
L is the lines of code.
V is the numbers of version.

B. Traditional Methods

Retest-All Method: RA

This technique tests all test cases in a test suite before writing the new code by considering functions that required by both users and developers. It suits for the smallest size with low complexities under certain changes. However, it is not appropriate technique, where numbers of function and code are large. This means testing needs long time and high cost of the maintenance phase. Besides, integrating parts of testing is a difficult task.

978-1-4673-7825-3/15/$31.00 ©2015 IEEE