Test Case Reduction Using
Data Mining Technique

Ahmad A. Saifan, Department of Computer Information Systems, Faculty of IT, Yarmouk University, Irbid, Jordan
Emad Alsukhni, Department of Computer Information Systems, Faculty of IT, Yarmouk University, Irbid, Jordan
Hanadi Alawneh, Department of Computer Information Systems, Faculty of IT, Yarmouk University, Irbid, Jordan
Ayat AL Sbaih, Department of Computer Information Systems, Faculty of IT, Yarmouk University, Irbid, Jordan

ABSTRACT

Software testing is a process of ratifying the functionality of software. It is a crucial area which consumes a great deal of time and cost. The time spent on testing is mainly concerned with testing large numbers of unreliable test cases. The authors’ goal is to reduce the numbers and offer more reliable test cases, which can be achieved using certain selection techniques to choose a subset of existing test cases. The main goal of test case selection is to identify a subset of the test cases that are capable of satisfying the requirements as well as exposing most of the existing faults. The state of practice among test case selection heuristics is cyclomatic complexity and code coverage. The authors used clustering algorithm which is a data mining approach to reduce the number of test cases. Their approach was able to obtain 93 unique effective test cases out a total of 504.

KEYWORDS

Clustering, Coverage, Cyclomatic Complexity, Redundant Test Cases, Test Case Reduction

1. INTRODUCTION

Software quality is an important issue that all developers of software systems want to achieve. It currently attracts a lot of attention since software is everywhere and affects our lives on a daily basis. Software testing is the main factor in enhancing and increasing the quality of software, for which it is necessary to generate different test cases according to certain coverage criteria such as graph, logic, input space, and syntax coverage (Amman & Offut, 2008). The size and complexity of software systems is growing dramatically, in addition to which, the existence of automated tools leads to the generation of a huge number of test cases, the execution of which causes huge losses in cost and time (Lilly & Uma, 2010). According to Rothermel et al. (Rothermel et al., 2001), a product of about 20,000 lines of code requires seven weeks to run all its test cases. Ultimately, the challenge is to find a way to reduce the number of test cases or to order the test cases to validate the system being tested.

The main goal of software testing is to ensure that the software is almost free from errors. The test process can be said to be effective when the test cases are able to locate any errors. Several tools have been seen in the literature which automatically generate thousands of test cases for a simple program in a few seconds, but executing those test cases takes a great deal of time. Moreover, the tools could also generate redundant test cases (Muthyala et al., 2011). The problem is compounded when we have complex systems, where the execution of the test cases may take several days to complete. Moreover, it should be noted that most of the time is spent in executing redundant or unnecessary test cases.
To identify the redundant test cases a technique such as data mining (Lilly & Uma, 2010) is required to understand the properties of test cases, with a view to determining the similarities between them and removing the redundant ones.

This paper aims to deal with this issue, of reducing the number of test cases in order to minimise the time and cost of executing them. Several techniques can be used to reduce test cases such as information retrieval, pairwise testing (Yoo at al., 2009) and data mining. We used the data mining approach, mainly because of the ability of data mining to extract patterns of test cases that are invisible.

We present our approach, concentrating on the two most effective attributes of test cases, coverage and complexity (Kameswari et al., 2011). An empirical study presented in (Jeffery & Gupta, 2007) suggested that during test case reduction, using several coverage criteria rather than single coverage is more effective in selecting test cases that are able to expose different faults.

We start by collecting the test cases for a given system and then we build the dataset by selecting coverage and complexity. Next, we use data mining technique, K-clustering, to group several test cases into a particular cluster. Finally, redundant test cases that have the same distance to the cluster centre point are removed. To evaluate our approach, we calculate the coverage ratio of the original test cases and compare it with the coverage ratio of the reduced test cases.

2. PROBLEM DESCRIPTION AND COMPARISON

2.1. Description of the Problem

Given a program \( P = \{C_1, C_2, \ldots, C_n\} \) consists of a set of classes \( C \), a class that consists of a set of methods \( C = \{M_1, M_2, \ldots, M_m\} \). Every method consists of a set of lines such as \( M = \{l_1, l_2, \ldots, l_c\} \), in which jump instructions (i.e. IF and FOR) defines branches within program traces such as \( b = \{1, 2, \ldots, l_b\} \subseteq M \) and \( l \subseteq \{IF, FOR, WHILE, ?\} \).

A set of test cases is defined to test program traces including instructions, methods and branches. The set of test cases is defined as \( T = \{t_1, t_2, \ldots, t_n\} \), where \( t_i = \{a_1, a_2, \ldots, a_t\} \) is a test case defined to cover one of the following:

- Line Coverage: \( Cov(t_i) = |L| \) covered by \( t_i \)
- Branch Coverage: \( Cov(t_i) = |b| \) covered by \( t_i \)
- Method Coverage: \( Cov(t_i) = |M| \) called by \( t_i \)
- Instruction coverage: number of codes that have been executed or missed.

In order to reduce the number of generated test cases according to their coverage, K-MEANS clustering is applied using Euclidian distance that measure the distance between any given test case and the centroid. Specifically, the set of test cases is divided into \( k - \) centroids where the distance between every test case and its related centroid is computed.

\[
\text{dist} = \sqrt{\sum_{i=1}^{n} (d_{b_i} - c_j)^2}, \quad \text{where } c_j \text{ is the cluster center that } d_{b_i} \text{ is related to} \tag{1}
\]
Related Content

Formal Approach to Ensuring Interoperability of Mobile Agents
[www.igi-global.com/chapter/formal-approach-ensuring-interoperability-mobile/66482?camid=4v1a](www.igi-global.com/chapter/formal-approach-ensuring-interoperability-mobile/66482?camid=4v1a)

Integrating Integrity Reporting Into Industrial Control Systems: A Reality Check
[www.igi-global.com/chapter/integrating-integrity-reporting-into-industrial-control-systems/186914?camid=4v1a](www.igi-global.com/chapter/integrating-integrity-reporting-into-industrial-control-systems/186914?camid=4v1a)
Power Transmission Analysis in Wireless Sensor Networks Using Data Aggregation Techniques
www.igi-global.com/article/power-transmission-analysis-in-wireless-sensor-networks-using-data-aggregation-techniques/220456?camid=4v1a

Adaptive Modeling and Dynamic Targeting for Real Time Analytics in Mobile Advertising
Donald Kridel, Dan Dolk and David Castillo (2017). International Journal of Systems and Service-Oriented Engineering (pp. 24-39).
www.igi-global.com/article/adaptive-modeling-and-dynamic-targeting-for-real-time-analytics-in-mobile-advertising/190411?camid=4v1a