Chapter 6

Conclusions and Future Work
CONCLUSIONS AND FUTURE WORK

6.1 Introduction

This chapter covers the conclusions of this research and provides directions for the possible future work. The conclusions are drawn on proposed algorithms aimed at fulfilling the objectives such as whole test suite generation and mutation testing, the automatic discovery of dependency structures and test case prioritization for increasing the fault detection ratio.

6.2 Conclusions

The main aim of this thesis is to build a comprehensive testing tool that has underlying algorithms for the whole test suite generation and mutation testing, the automatic discovery of dependency structures and test case prioritization to improve the fault detection ratio. Literature review insights paved the way for achieving the objectives of this research. The objectives of the research are described here.

1. To propose an algorithm that can generate the whole test suite, which is representative of all test goals besides reducing the size of the test suite.
2. To develop an algorithm for the automatic discovery of open and closed dependency structures from the SUT.
3. To exploit the dependency structures to prioritize test cases so as to increase the fault detection ratio.

The following conclusions are drawn with the help of experience gained from this research.

1. A genetic algorithm is built that takes care of the generation of representative test suite at a time in order to have smaller size test suite besides ensuring full code coverage. The generated test suite can also be used for mutation testing. Then a tool is built that is designed to be modular and planned to be extended in future research. It is meant for demonstrating the proof of the concept with respect to test suite generation with minimal size and full coverage. Mutant killing test cases are directly generated from the given source code. From the experiments in software testing, it is evident that the optimization of whole
test suite generation is far better than the traditional method of targeting one coverage goal.

2. Haidry and Miller proposed a family of test case prioritization techniques that are used for dependency information from a test suite to prioritize that test suite. The nature of the techniques preserves the dependencies in the test ordering. Dependencies in test cases can have their impact on the discovery of faults in software. The authors implemented and demonstrated the methodology with their empirical results. However, they do not automate the extraction of dependency structures among the test suits that can help in effective prioritization of functional test suites. We proposed a methodology that automates the process of extraction of dependency structures from the test cases that will result in the increase the rate of fault detection. Thus the number of bugs uncovered from the SUT is improved. This leads to the improvement of quality of software.

3. A comprehensive tool is built that facilitates automatic testing of object oriented applications. The tool includes features such as automatic test suite generation, automatic discovery of dependency structures and prioritization of test cases. The tool is used to integrate the experiments conducted for all objectives of research. The tool is used to perform empirical study pertaining to whole test suite generation, automatic discovery of dependency structures and test case prioritization. The empirical results reveal that the automatic discovery of dependency structures can help in complete automation of test case prioritization.

6.3 Recommendations for the Future Work

The proposed software testing tool which focused on the Object Oriented Applications has been tested and also considered that the proposed tool needs to be improved future needs. Due to the emergence of mobile applications and cloud based applications, it is important to make the tool to be cloud ready and mobile ready. The following are therefore recommendations provided for future work.

1. As most of the applications in the future gets deployed in cloud, ability to consider such applications as case study for the whole test suite generation,
automatic discovery of dependency structures and test case prioritization. The cloud-aware tool will serve this purpose. The tool needs to be updated with cloud-ready algorithms. The rationale behind this is that the cloud computing is associated with a new programming model known as MapReduce which exploits parallel power of modern Graphics Processing Units (GPUs) in order to perform operations in parallel.

2. Another important recommendation is to improve the tool to be mobile-ready. This will enable the tool to generate test cases and prioritize them for mobile applications. The peoples used the mobiles in all walks of life, it is essential to have support for testing mobile applications so as to make the tool more useful.

6.4 Summary

The previous chapter mainly focused on the results of research whereas this chapter has provided the conclusions related to the proposed research aimed at building a comprehensive testing tool that has underlying algorithms for the whole test suite generation and mutation testing, the automatic discovery of dependency structures and test case prioritization to improve the fault detection ratio. The general picture of this chapter is the conclusion pertaining to the hypothesis “test case prioritization can help in improving the fault detection ratio” and other related research activities such as whole test suite generation and mutation testing, and the automatic discovery of dependency structures.