TEST CASE GENERATION OPTIMIZATION USING COMBINATION PREDICTION MODEL AND PRIORITIZATION IDENTIFICATION FOR COMPLEX SYSTEM RETESTING

NURUL SHAZANI BINTI SAHIDAN

UNIVERSITI TEKNOLOGI MALAYSIA
TEST CASE GENERATION OPTIMIZATION USING COMBINATION PREDICTION MODEL AND PRIORITIZATION IDENTIFICATION FOR COMPLEX SYSTEM RETESTING

NURUL SHAZANI BINTI SAHIDAN

A thesis submitted in fulfilment of the requirements for the award of the degree of Masters in Science (Computer Science)

Faculty of Computing
Universiti Teknologi Malaysia

APRIL 2017
All praises to Allah the Almighty for
the strengths and His blessing in completing this thesis.

Specially dedicated to;
my beloved husband Mohd Farhan bin Ahmad
my adorable daughter Nur Jannah Humairah
my wonderful parents Sahidan Zakaria and Noorhayah Che Halim
my understanding parents in law Ahmad and Jamilah
my precious siblings Shahida, Shahrin, Shafizan, Shahanim, Shahiruddin, Shakirin
ACKNOWLEDGEMENT

“In the name of Allah, the most Gracious and the most Merciful”

This thesis would have not been completed without the help and splendid support from many individuals and teams. Firstly, my sincere gratitude goes to the backbone of my research, my supervisors, Dr. Hishammuddin Asmuni and Dr. Muhamad Razib Othman for their excellent supervision, knowledge, belief, patience and interest in the work and for pushing me farther than I thought I could go. To my beloved parents, thank you for always being there and never fail to give me words of encouragement. Your endless support is making me for who I am today. To my dearest colleagues and research-mates, I am thankful for the friendship, supportive comments and ideas in reviewing each other’s works and also fun times throughout last three and half year of this study. To my precious close friends and loved ones, thank you for helping me surviving all the stress and not letting me giving up. To everyone who has consistently giving support and advice directly or the other way round, including the team of Software Engineering Research Group (SERG) and GATES IT Solutions Sdn. Bhd., I refer my appreciation. My greatest thanks should also credit to all lecturers at the Faculty of Computing, Universiti Teknologi Malaysia (UTM) for their understanding and support. Last but not least, I appreciate the financial support from the GATES IT Solutions Sdn. Bhd. under GATES Scholar Foundation (GSF) and Malaysian Ministry of Higher Education under MyMaster funds.
ABSTRACT

Nowadays, the retesting process has become crucial in assessing the functionality and correctness of a system in order to ensure high reliability. Although many techniques and approaches have been introduced by researchers, some issues still need addressing to ensure test case adequacy. To determine test case adequacy, it is crucial to first determine the test set size in terms of number of test cases to prevent the system from failing to execute. It is also crucial to identify the requirement specification factor that would solve the problem of insufficiency and scenario redundancy. To overcome this drawback, this study proposed an approach for test case generation in the retesting process by combining two models, which would reveal more severe faults and improve software quality. The first model was enhanced through determining the test case set size by constructing a predictive model based on failure rate using seed fault validation. This model was then extended to requirement prioritisation. Next, it was used to schedule the test cases that focus on Prioritisation Factor Value of requirement specifications. The Test Point Analysis was used to evaluate test effort by measuring level of estimation complexity and by considering the relationship among test cases, fault response time, and fault resolution time. This approach was then evaluated using complex system that called as Plantation Management System as a project case study. Data of Payroll and Labour Management module that applied in 138 estates been collected for this study. As a result, the test case generation approach was able to measure test effort with High accuracy based on two combination model and it achieved a complexity level with 90% confidence bounds of Relative Error. This result proves that this approach can forecast test effort rank based on complexity level of requirement, which can be extracted from early on in the testing phase.
ABSTRAK

# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DECLARATION</td>
<td>ii</td>
<td></td>
</tr>
<tr>
<td>DEDICATION</td>
<td>iii</td>
<td></td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>iv</td>
<td></td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>v</td>
<td></td>
</tr>
<tr>
<td>ABSTRAK</td>
<td>vi</td>
<td></td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td>vii</td>
<td></td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>xi</td>
<td></td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>xiii</td>
<td></td>
</tr>
<tr>
<td>LIST OF SYMBOLS</td>
<td>xv</td>
<td></td>
</tr>
<tr>
<td>LIST OF ABBREVIATIONS</td>
<td>xvi</td>
<td></td>
</tr>
</tbody>
</table>

1 INTRODUCTION

1.1 Background 1
1.2 Challenges in Modeling Test Case Generation 3
1.3 Current Method in Modelling the Test Case Generation 5
1.4 Problem Statement 6
1.5 Objectives of the Study 7
1.6 Scope of the Study 8
1.7 Significance of the Study 8
1.8 Organization of the Thesis 9
2 LITERATURE REVIEW

2.1 Introduction 11
2.2 Understanding of Complex System 11
2.3 Mapping Test Case using Requirements Traceability 16
2.4 Structuring Test Cases Size Using Prediction Model 19
2.5 Identifying Test Case using Prioritization Model 24
2.6 Evaluation of Test Effort Estimation 30
   2.6.1 Test Point Analysis 33
2.8 Summary 36

3 RESEARCH METHODOLOGY

3.1 Introduction 37
3.2 Research Framework 38
3.3 Research Process 39
   3.3.1 Test Predictive Model 39
      3.3.1.1 Retesting Process 40
      3.3.1.2 Test Case Predictive Process 44
      3.3.1.3 Adapting the Test Predictive Model for Seed Fault Analysis 48
   3.3.2 Test Prioritization Model 52
      3.3.2.1 4FTC Prioritization Factor Implementation 53
      3.3.2.2 Prioritization Factor Value (PFV) 55
      3.3.2.3 4FTC Validation Metrics 56
      3.3.2.4 Acceptable Test Case Size 57
   3.3.3 Test Effort Estimation 58
      3.3.3.1 Test Point Analysis 59
   3.3.4 Case Study Selection and Data Source 61
3.5 Evaluation Metrics 63
3.6 Summary 64
TEST PREDICTIVE MODEL: UNDER/OVER

4  PREDICTION OF TEST CASE SIZE

ESTIMATION

4.1 Introduction 65
4.2 Implementation in Case Study 66
4.3 Discussion based on implementation 69
  4.3.1 Conservative versus Liberal predictions 70
  4.3.2 Functionality with Multiple Faults 71
4.4 Summary 72

SELECTION AND IDENTIFICATION OF

5  PRIORITIZATION FACTOR FOR TEST CASE

OPTIMIZATION

5.1 Introduction 73
5.2 Case Study Implementation and Discussion 74
  5.2.1 4FTC Comparative Analysis 76
5.3 Summary 78

EVALUATION OF TEST EFFORT ESTIMATION

6  BASED ON RANK OF COMPLEXITY

6.1 Introduction 79
6.2 Historical Data Process 80
6.3 Comparative Analysis with Different Approach 80
  6.3.1 Evaluation Test Effort for Random Model 81
  6.3.2 Evaluation Test Effort for Predictive Model 82
  6.3.3 Evaluation Test Effort for Combination of Predictive and 4FTC Model 82
6.4 Comparison Analysis and Discussion 86
  6.4.1 Rank of complexity level for test effort estimation 88
6.5 Summary 89
7 CONCLUSION

7.1 Concluding Remarks 90
7.2 Contributions of the study 92
7.3 Future Works 93

REFERENCES 95
APPENDIX 99
### LIST OF TABLES

<table>
<thead>
<tr>
<th>TABLE</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Features of ERP</td>
<td>13</td>
</tr>
<tr>
<td>2.2</td>
<td>Summary of complex system</td>
<td>15</td>
</tr>
<tr>
<td>2.3</td>
<td>Summary of RT Classification</td>
<td>18</td>
</tr>
<tr>
<td>2.4</td>
<td>Summary of predictive analytics model</td>
<td>23</td>
</tr>
<tr>
<td>2.5</td>
<td>Stakeholders that involve in prioritization</td>
<td>28</td>
</tr>
<tr>
<td>2.6</td>
<td>Summary of Prioritization algorithm’s characteristics</td>
<td>29</td>
</tr>
<tr>
<td>2.7</td>
<td>Evaluation of complexity level of test estimation</td>
<td>31</td>
</tr>
<tr>
<td>2.7</td>
<td>Summary of test effort estimation</td>
<td>35</td>
</tr>
<tr>
<td>3.1</td>
<td>Term input means for test case</td>
<td>42</td>
</tr>
<tr>
<td>3.2</td>
<td>Type of Reported Bugs</td>
<td>43</td>
</tr>
<tr>
<td>3.3</td>
<td>Status and resolution of bugs</td>
<td>43</td>
</tr>
<tr>
<td>3.5</td>
<td>Severity Value</td>
<td>56</td>
</tr>
<tr>
<td>3.6</td>
<td>Test case example</td>
<td>62</td>
</tr>
<tr>
<td>3.7</td>
<td>Evaluation Metrics</td>
<td>63</td>
</tr>
<tr>
<td>4.1</td>
<td>Defect error of Labour and Payroll</td>
<td>66</td>
</tr>
<tr>
<td>4.2</td>
<td>Fault failures rate of PMS system</td>
<td>67</td>
</tr>
<tr>
<td>4.3</td>
<td>Probability of fault based on fault rate</td>
<td>68</td>
</tr>
<tr>
<td>4.4</td>
<td>Fault Detection Effectiveness</td>
<td>68</td>
</tr>
<tr>
<td>4.5</td>
<td>Summary of seed fault analysis using <em>AppPerfect</em></td>
<td>69</td>
</tr>
<tr>
<td>4.6</td>
<td>Confidence levels at which null hypotheses (H_0)</td>
<td>71</td>
</tr>
</tbody>
</table>
can be rejected.

4.7 Summary prediction of test case size 72
5.1 List of requirements after assign factor priority 74
5.2 TSFD using 4FTC for UAT phase 75
6.1 Service Level Agreement (SLA) process time 80
6.2 Data validation of test effort for random model 83
6.3 Data validation of test effort for predictive model 84
6.4 Data validation of test effort for combination of predictive and 4FTC model 85
6.5 Summary of comparison of model criteria value 86
6.6 Complexity Level 88
LIST OF FIGURES

<table>
<thead>
<tr>
<th>FIGURE</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Significance of the study</td>
<td>9</td>
</tr>
<tr>
<td>2.1</td>
<td>Complex system characteristic</td>
<td>12</td>
</tr>
<tr>
<td>2.2</td>
<td>ERP advantages and disadvantages</td>
<td>14</td>
</tr>
<tr>
<td>2.3</td>
<td>Relation of classification of RT</td>
<td>17</td>
</tr>
<tr>
<td>2.4</td>
<td>General approaches for predictive model</td>
<td>20</td>
</tr>
<tr>
<td>2.5</td>
<td>Relationship between training, optimism and true prediction error.</td>
<td>22</td>
</tr>
<tr>
<td>2.6</td>
<td>The structure of the next release</td>
<td>25</td>
</tr>
<tr>
<td>2.7</td>
<td>Illustration of Test Effort Cycle Estimation</td>
<td>32</td>
</tr>
<tr>
<td>3.1</td>
<td>Research Framework</td>
<td>38</td>
</tr>
<tr>
<td>3.2</td>
<td>Test Case Prediction Process</td>
<td>40</td>
</tr>
<tr>
<td>3.3</td>
<td>Test case selection process flow</td>
<td>41</td>
</tr>
<tr>
<td>3.4</td>
<td>Flow diagram for test execution</td>
<td>43</td>
</tr>
<tr>
<td>3.5</td>
<td>Bug tracking process</td>
<td>44</td>
</tr>
<tr>
<td>3.6</td>
<td>Legitimate input domain</td>
<td>49</td>
</tr>
<tr>
<td>3.7</td>
<td>Multi-stakeholders access control for Plantation</td>
<td>51</td>
</tr>
<tr>
<td>3.8</td>
<td>4FTC Prioritization Process</td>
<td>52</td>
</tr>
<tr>
<td>3.9</td>
<td>Test Cycle in PMS execution</td>
<td>62</td>
</tr>
<tr>
<td>5.1</td>
<td>Percentage of TSFD (4FTC versus Random)</td>
<td>77</td>
</tr>
<tr>
<td>6.2</td>
<td>Linear Regression Analysis of three data evaluation</td>
<td>87</td>
</tr>
</tbody>
</table>
LIST OF SYMBOLS

H - High
L - Low
M - Medium
VH - Very High
VL - Very Low
**LIST OF ABBREVIATIONS**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>4FTC</td>
<td>Four Factor Test Case</td>
</tr>
<tr>
<td>AE</td>
<td>Accuracy Estimation</td>
</tr>
<tr>
<td>CP</td>
<td>Customer-assigned priority</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>CR</td>
<td>Change Request</td>
</tr>
<tr>
<td>DR</td>
<td>Defect Request</td>
</tr>
<tr>
<td>DRL</td>
<td>Defect Request Logical</td>
</tr>
<tr>
<td>DRS</td>
<td>Defect Request Syntax</td>
</tr>
<tr>
<td>ERP</td>
<td>Enterprise Resource Planning</td>
</tr>
<tr>
<td>FPA</td>
<td>Function Point Analysis</td>
</tr>
<tr>
<td>IC</td>
<td>Developer-Perceived Implementation Complexity</td>
</tr>
<tr>
<td>IT</td>
<td>Incomplete Task</td>
</tr>
<tr>
<td>LSE</td>
<td>Least Square Estimation</td>
</tr>
<tr>
<td>MLE</td>
<td>Maximum Likelihood Estimation</td>
</tr>
<tr>
<td>MRE</td>
<td>Mean Relative Error</td>
</tr>
<tr>
<td>MSF</td>
<td>Mean Square Faults</td>
</tr>
<tr>
<td>NC</td>
<td>No Category</td>
</tr>
<tr>
<td>PC</td>
<td>Process Configuration</td>
</tr>
<tr>
<td>PFV</td>
<td>Prioritization Factor Value</td>
</tr>
<tr>
<td>PMS</td>
<td>Plantation Management System</td>
</tr>
<tr>
<td>QC</td>
<td>Quality Control</td>
</tr>
<tr>
<td>RE</td>
<td>Relation Error</td>
</tr>
<tr>
<td>RT</td>
<td>Requirement Traceability</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>RV</td>
<td>Requirement Volatility</td>
</tr>
<tr>
<td>SLA</td>
<td>Service Level Agreement</td>
</tr>
<tr>
<td>TPA</td>
<td>Test Point Analysis</td>
</tr>
<tr>
<td>TSFD</td>
<td>Total Severity Fault Detection</td>
</tr>
<tr>
<td>UAT</td>
<td>User Acceptance Test</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION

1.1 Background

Software testing is a form of investigation that is conducted to provide stakeholders with information about the quality of the product or service under test. Boehm et al. (2003) states that software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a programme or application with the intent of finding software bugs (errors or other defects), and to verify that the software product is fit to use. In this part, the software tester and business analyst play important roles to make sure all features of the application given by the end user works correctly. Usually, the software tester will execute manual testing to detect any defects in the system. To ensure completeness of testing, the tester often follows a written test plan that leads them through a set of important test cases. With this informal approach, the tester does not follow any rigorous testing procedure, but rather explores the user interface of the application using as many of its features as possible, and using information gained in prior tests to intuitively derive additional tests. The success of exploratory manual testing relies heavily on the domain expertise of the tester, because a lack of knowledge will lead to incompleteness in testing.
Large-scale engineering projects that rely on manual software testing follow a more rigorous methodology in order to maximise the number of defects that can be found. A systematic approach focuses on predetermined test cases and generally involves steps such as prediction of test case size and prioritisation of test case running number. A major issue when dealing with incomplete testing is the shortage of taxonomies to achieve a satisfactory level of information about defects in the requirements phase. Thus, it can be concluded that no matter how good the subsequent phases are, the quality of the requirements phase will still be the main determinant that affects the overall quality of the subsequent phases, including the testing phase of the software development process. The Software Requirement Specification (SRS) document defines the capabilities of the provided software (Alshazly et al., 2014). Therefore, if an analyst or a developer does not share the same understanding regarding the requirements, the outcome of the development process will not satisfy customer needs (Gutierrez et al., 2004).

Besides that, the software tester usually generates the test suites based on a coverage criterion that is given without considering the issue of ensuring that parts of the model are exhaustively exercised. Implementation of impracticable testing will allow errors to occur when generating many or way too many test cases in a set of test suites. This situation leads to a decrease in the likelihood of selected test suites uncovering errors in the implemented system. For efficiency of the testing process, the goal is to choose test cases from the test suite in order to establish the correctness of the modification (Gutierrez et al., 2004). Such test suite reuse, in the form of retesting, takes up as much as one-half of the cost of software maintenance (Boehm et al., 2003). For this reason, researchers have considered various techniques to reduce the cost of the retesting process. Retesting is used to verify alternations and to ensure that the changes have not corrupted other functionalities of the software (Peng et al., 2014). Logically, the increment in test cases size will lead to increased project size and time. Due to this reason, project development will face difficulty in predicting and managing time (Xie et al, 2003). It will also become the main problem to address when the testing deals with a vast and complex system. As is generally known, the test cycle is an important factor in testing; late changes or additions to function at the final moments of the testing phase can incur high costs in
the execution process. Therefore, intelligent planning and decision-making must be thoroughly done throughout the generation of the test case in order to achieve optimisation.

In addition, the test cases must also be prioritised to the best new positions based on software requirement specifications, so as to reduce the need for additional test efforts. These techniques let testers order their test cases so that the test cases with the highest priority are executed earlier in the retesting process (Peng et al., 2014). Besides that, test maturity has also become a significant factor that effects project size. Test maturity is the process that is done to ensure a system achieves stability. In the progress of reaching stability, changes in the function will keep occurring in the project development life cycle and this can increase the test execution phase and time pressure (Elbaum et al., 2001).

1.2 Challenges in Modelling Test Case Generation

Even with the current technology and sophisticated tools for generating a test case for testing areas nowadays, there are still issues and challenges that the researcher needs to address in order to reduce test efforts and save time and cost. Techniques like root cause analysis and orthogonal defect classification are some of the commonly used practices.

Firstly, there is a significant challenge in identifying the adequate test case size. The number of test case size must be taken into consideration when dealing with system testing. The application of a few test cases or overdrawn test cases can result to inaccuracy rate of fault detection. As a consequence, the assumption for the test case size is important for improving the ability of detecting the fault, thus reducing the cost and time for the retesting process. There are some defects that
affect the option selection of test case size, which is due to lack of knowledge on the part of the tester in defining multiple scenarios when dealing with the requirement phase or inadequate testing executed by the internal user.

The second challenge involves planning for the test cases to be executed to achieve the performance goal. The importance of increasing the performance goal can be done by producing high rates of fault detection in the system. With this, when the test case is executed based on complexity, the fault can be detected at the earliest time in the testing phase (Krishnamoorthi & Sahaaya 2009). The best way to assist the testing process is by prioritising its functionality based on the requirement criteria. Prior planning of prioritisation is one of the test strategies that contribute to improving the rate of fault detection with the aim of increasing the performance and bettering the quality in the testing phase (Boehm et al., 2003). The idea of this technique is to release the test case with higher priority so it is executed first, which is then followed by the lowest priority test case. The level of priority is based on the complexity of the requirement. Since test case prioritisation techniques do not discard test cases, the drawbacks of test case minimisation techniques can be avoided.

Last but not least, approach of generating the test cases can also become a major challenge, particularly in an effort to increase software reliability. Software reliability is the failure probability of the software operation. The lower the percentage of failure of the system, the higher the reliability of the product (Lo & Huang 2006). The important aspects that need to be considered include identification of measures, formulation of theories, capturing of historical data and assessment of how effective those effort estimation models in order to achieve realistic effort estimates for the successful management of software development.
1.3 Current Method in Modelling Test Case Generation

In general, the current methods for test case generation can be categorised into three: requirement traceability, probabilistic estimation, and multiple performance metrics.

a) Requirement traceability is the process of mapping between requirement and test cases that will be generated. Practically, test cases are used to demonstrate the flow of the requirement provided by the client. It will be difficult for testers to determine whether or not the requirement is adequate for testing if the test cases have no connection to the individual requirement (Vaysburg, 2001). Traceability, as introduced by Gotel et al. (1994), has two different criteria, which are (pre-RS) traceability and (post-RS) traceability. Both represent the encompassing solution and provide the basic framework to illustrate the nature of the issue. The authors successfully proved that poor requirement traceability as a widely reported problem. Since traceability is an important characteristic (Krishnamoorthi and Sahaaya Arul Mary, 2009) and end-to-end traceability is derived from software requirements, test cases and their associated defects in detecting the most severe faults must be discovered at the earliest possible moment in the testing life cycle.

b) Probabilistic estimation is the result of using a retesting model and is used to 'predict/estimate missing or out-of-sample y-values, where y is defined as the dependent variable. In practice, it is very difficult to estimate failure rates with such accuracy. Therefore, Debroy and Wong (2011) introduced the idea of predicting the defect based on historical data. Observation of the failure rates was originally based on the entire set of available test cases against a faulty version of a programme. Historical data can be applied in the requirement specifications by identifying the failure in cumulative number when the test cases are run.

c) The validation metrics measures an organisation's activities and performance. In project management, validation metrics is used to assess the health of the project and consists of the measuring of seven criteria: safety, time, cost, resources,
scope, quality, and actions (Neville, 2008). In testing, quality becomes a great concern before a product is released to the customer. One criticism of performance metrics is that the value of information is computed using mathematical methods based on historical data that have been collected from an industrial study.

1.4 Problem Statement

The problem in representing modelling of the test case generation, specifically in the retesting process, is described as follows:

“Given a large and complex system that has issues regarding budgetary cost and time in testing effort, the challenge is to predict test case size in terms of the number of test cases that might suffer from the redundancy process. Also, to prioritise test planning that can detect the most severe faults at the earliest moment in the testing life cycle. Lastly, the method must be able to measure the test effort based on complexity level with a high prediction capability and yielding fairly accurate results.”

The first challenge is related to insufficient user story for the testing execution when generating the test cases. Thus, this study aims to predict the test case size of the test cases. Test case size can greatly impact the budgeting of cost and time in testing if testers underestimate the prediction of test case size. Hence, the relationship between test case size and fault detection are both taken into consideration in this study. Poor fault detection will occur if too few test cases are generated and the use of too many test cases might incur expensive cost and result in time constraints.

The second challenge involves no interdependencies between user requirements and their prioritisation based on their relative ranking or grouping on a
specified criterion or criteria that may suffice. The current requirements for engineering techniques for prioritisation of software requirements implicitly assume that each user requirement will have an independent and symmetric impact on user satisfaction. For example, it is assumed that implementing a high-priority user requirement will positively impact his/her satisfaction and not implementing a high-priority user requirement will negatively impact his/her satisfaction. For this reason, this study aims to generate the test case based on requirement priority collected from the stakeholders themselves. The factor with high value is recognised as having a complex requirement that must be prioritised in order to increase test efficiency.

The third challenge is the measurement of effort without using any approach in order to estimate effort level. It is important to estimate level of effort to increase the reliability and quality of the product. This study therefore determines the level of test effort based on time lag from the detection and correction of faults that occur in the system. In actual fact, the time taken for bug correction will become shorter if the test effort focuses more on the allocation between the detection and correction of faults. However, in the development life cycle, it is nearly impossible to make the system bug-free but attempts by the developer to reduce the percentage of fault detection will ensure a high reliability for the software.

1.5 Objectives of the Study

The goal of this research is to develop an approach for test case generation for the retesting process using a predictive and prioritisation model to reveal more severe faults and to improve customer-perceived software quality. Therefore, there are three objectives of this study that need to be achieved:
i. To estimate the optimum test case size required to detect the faults in the system

ii. To prioritise test cases based on the requirement specification to increase test effectiveness

iii. To evaluate the test effort and assess its effectiveness by implementing an industrial case study

### 1.6 Scope of the Study

In order to achieve the objectives stated, the limitations below bind the scope of this study:

- a) This study focuses on improving the test case optimisation that can reduce the time and cost for the retesting process.
- b) Test prediction is applied to assess the optimisation of the test case based on 669 test case sizes.
- c) Test prioritisation is integrated with requirement specifications to determine the adequate test planning before retesting is executed.
- d) A Plantation Management System (PMS) that focuses on the labour and payroll module is used in this study to demonstrate the proposed model.
- e) Historical data from the User Acceptance Test of the PMS system are taken into consideration for the evaluation phase of this study.
- f) The test effort is evaluated via 37 test suites with 651 fault detections in the system.

### 1.7 Significance of the Study

The significance of this study can be divided into three different categories, which are: (i) System performance; (ii) Software tester; and (iii) In-house software organisation. The benefit of these respective categories is simplified in Figure 1.1:
Figure 1.1 Significance of the study

1.8 Organisation of the Thesis

This thesis is organised into seven chapters. A brief description of each chapter is given as follows:

i. Chapter 1 defines the challenges, problems, current methods, objectives, scope, and significance of the study.

ii. Chapter 2 reviews the main issues of interest, which include requirement traceability, test prediction model, test case prioritisation, and test effort estimation techniques.

iii. Chapter 3 presents the design of the computational method that supports the objectives of the study. This includes the research framework, data collection, and instrumentation and analysis.

iv. Chapter 4 describes the scheme for developing a prediction model to estimate the reliability of the retesting process and to determine the test case size using failure rate so as to improve fault detection in the system.

v. Chapter 5 discusses the implementation of the test case prioritisation model by considering four factors of requirement complexity.

vi. Chapter 6 evaluates the test effort based on fault detection and correction model by showing the relationship between the flow of test cases, fault
response time, and fault resolution time, which are demonstrated using three types of comparative analyses.

vii. Chapter 7 draws the overall conclusions from the obtained results and presents the contributions of the study as well as recommends potential directions for future study.
REFERENCES

Averous, J. (2012). Large Complex Projects. LNCS 7343, 350–354,


Verification and Reliability, 320-337


