

The copyright © of this thesis belongs to its rightful author and/or other copyright owner. Copies can be accessed and downloaded for non-commercial or learning purposes without any charge and permission. The thesis cannot be reproduced or quoted as a whole without the permission from its rightful owner. No alteration or changes in format is allowed without permission from its rightful owner.



**AN EVALUATION ON THE COMPREHENSIBILITY OF UML
ACTIVITY AND STATE CHART DIAGRAMS WITH REGARD
TO MANUAL TEST CASE GENERATION**

HAITHAM RAED IBRAHIM



UUM

Universiti Utara Malaysia

**SUPERVISOR
DR. NOR LAILY BINTI HASHIM**

**MASTER OF SCIENCE (INFORMATION TECHNOLOGY)
UNIVERSITI UTARA MALAYSIA
2017**

**AN EVALUATION ON THE COMPREHENSIBILITY OF UML
ACTIVITY AND STATE CHART DIAGRAMS WITH REGARD
TO MANUAL TEST CASE GENERATION**

A dissertation submitted to Dean of Awang Had Salleh

Graduate School

**in Partial Fulfillment of the required for
Master of Science (Information Technology)**

University Utara Malaysia



Universiti Utara Malaysia

By

Haitham Raed Ibrahim

Permission to Use

In presenting this dissertation in partial fulfillment of the requirements for a postgraduate degree from Universiti Utara Malaysia, I agree that the Universiti Library may make it freely available for inspection. I further agree that permission for the copying of this dissertation in any manner, in whole or in part, for the scholarly purpose may be granted by my supervisor(s) or, in their absence, by the Dean of Awang Had Salleh Graduate School of Arts and Sciences. It is understood that any copying or publication or use of this dissertation or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to Universiti Utara Malaysia for any scholarly use which may be made of any material from my dissertation.

Requests for permission to copy or to make other use of materials in this project dissertation, in whole or in part, should be addressed to:

Dean of Awang Had Salleh Graduate School of Arts and Sciences

UUM College of Arts and Sciences

Universiti Utara Malaysia

06010 UUM Sintok

Abstrak

Gambar rajah aktiviti dan *statechart* adalah gambar rajah UML yang paling kerap digunakan untuk menguji sistem berdasarkan spesifikasinya. Salah satu ciri penting gambar rajah UML adalah boleh difahami. Analisis kandungan kajian terdahulu menekankan kekurangan penilaian pakar mengenai kefahaman gambar rajah aktiviti dan *statechart* berkaitan dengan penjanaan kes ujian. Oleh itu, objektif utama kajian ini adalah bagi menilai kefahaman pakar penguji perisian ke atas gambar rajah aktiviti dan *statechart* UML dalam penjanaan kes ujian. Pertama, analisis kandungan telah dilakukan untuk mengenal pasti kriteria boleh difahami. Kriteria tersebut adalah berdasarkan kesukaran dan keyakinan subjektif. Seterusnya, satu set soalan penilaian direka berdasarkan analisis kandungan yang telah dilakukan. Kemudian, kes ujian dijana secara manual daripada gambar rajah aktiviti dan *statechart* satu kajian kes yang telah disesuaikan. Temu bual telah dijalankan dengan lima pakar untuk mengesahkan soalan penilaian yang dibentuk. Pakar tersebut menilai kefahaman ke atas gambar rajah aktiviti dan *statechart* dengan menggunakan soalan-soalan penilaian tersebut. Hasil kajian ini memberikan butiran khusus mengenai ciri yang berbeza daripada gambar rajah aktiviti dan *statechart*. Selain itu, ia mencadangkan bahawa gambar rajah aktiviti adalah lebih difahami daripada gambar rajah *statechart* dalam aspek penjanaan kes ujian. Hasil kajian ini diharapkan dapat memudahkan para penguji perisian untuk memilih satu daripada beberapa jenis gambar rajah pengujian yang sedia ada.

Abstract

The activity and state chart diagrams are the most frequently used UML diagrams for testing a system based on its specification. One of the key important qualities of the UML diagrams is their comprehensibility. The content analysis of previous studies highlighted the lack of experts' evaluation of the comprehensibility of activity and state chart diagrams with regard to test case generation. Thus, the main objective of this study is to evaluate the comprehensibility of the UML activity and state chart diagrams for test case generation. First, a content analysis was performed to identify the comprehensibility criteria. The criteria are perceived difficulty and subjective confidence. Next, a set of evaluation questions was designed based on the content analysis. Then, test cases were generated from activity and state chart diagrams manually of an adapted case study. An interview was conducted with five experts to validate the evaluation questions. The experts evaluated the comprehensibility of the activity and state chart diagrams by using the evaluation questions. The result of the study provided specific details of the different characteristics of activity and state chart diagrams. Further, it suggested that the activity diagram is more comprehensible than the state chart diagram in the aspect of test case generation. The finding of this study could assist software testers in choosing the appropriate UML diagrams for software testing.

Acknowledgement



*In the name of God, the Most Gracious, the Most
Merciful.*

All praises and thanks to the Almighty, Allah (SWT) for helping me to finish this study. Allah gives me the opportunity, strength and the ability to complete my Master degree after a long, continuous work. No volume of words is enough to express my gratitude towards my guide, Dr. Nor Laily Binti Hashim; without her knowledge and assistance plus her recommendations, this study would not have been successful. She has helped me to explore the topic in an organised manner and provided me with all the ideas on how to work towards a research-oriented endeavour.

It would not be possible for me to complete the study without the support and encouragement from my family and friends. First and foremost, my gratitude goes to my wife Maha for supporting and providing great inspiration for me to finish my master's study. May Allah bless her and my two lovely kids Ahmed and Dania. Secondly, my father and mother and their prayers for me, my aunt Nadia Putrus, my father-in-law and mother-in-law who motivated me and gave me their endless support. Finally, to my dearest brother's soul (Ahmed). To my great friends, especially Dr. Nassir Farhan; thanks for standing beside me and giving me support for all the periods of my study.

Special thanks to all who have helped or contributed to making this study a success.

Table of Contents

Permission to Use	iii
Abstrak.....	iv
Abstract	v
Acknowledgement	vi
Table of Contents	vii
Last of Tables.....	xi
List of Figures	xii
List of Appendices	xiii
List of Abbreviations	xiv
CHAPTER ONE INTRODUCTION	1
1.1 Overview	1
1.2 Introduction.....	1
1.3 Statement of Problem.....	3
1.4 Research Questions	6
1.5 Research Objectives.....	7
1.6 Scope of Study	7
1.7 Significance of Study	8
1.7.1 Body of knowledge	8
1.7.2 The practical support	9
1.8 Organisation of the Study	10
1.9 Summary	10
CHAPTER TWO LITERATURE REVIEW	11

2.1 Overview	11
2.2 Introduction to Software Testing	11
2.3 The Evaluation of Different Behavioural UML Diagrams With Regard to Test Case Generation	13
2.4 The Comprehensibility Evaluation Criterion	20
2.4.1 Criteria in Evaluating Comprehensibility	32
2.5 Test Cases Generation from UML Diagrams	33
2.5.1 Test Case Generation from UML Activity Diagram	35
2.5.2 Test Cases Generation from State Chart Diagram	40
2.6 Summary of Chapter Two	46
CHAPTER THREE RESEARCH METHODOLOGY	51
3.1 Overview	51
3.2 Research Design	51
3.2.1 Phase One	53
3.2.1.1 Investigation of Previous Studies	53
3.2.2 Phase Two	56
3.2.2.1 Instrumentation Design	56
3.2.2.2 Generating Manual Test Cases from Activity and State chart diagrams	64
3.2.2.3 Planning the One-to-One Interview	75
3.2.2.4 Conducting the Interview	77
3.2.2.5 Profile of Experts	78
3.2.2.6 Data Analysis	79
3.3 Summary	81

CHAPTER FOUR FINDINGS	82
4.1 Introduction.....	82
4.2 The Significant Findings from the Interview with the Experts.....	82
4.2.1 Evaluation Data from Open-Ended Questions.....	83
4.2.1.1 Perceived Difficulty of the UML Diagrams with regard to test case generation	83
4.2.1.2 Subjective Confidence of the UML Diagrams with Regard to Test Case Generation	90
4.2.2 Evaluation Data from Closed-Ended Questions	96
4.2.2.1 Perceived Difficulty of the UML Diagrams with Regard to Test Case Generation	97
4.2.2.2 Subjective Confidence of the UML Diagrams with regard to test case generation	100
4.3 Summary	104
CHAPTER FIVE DISCUSSION AND CONCLUSION	105
5.1 Introduction.....	105
5.2 Research Discussion	105
5.2.1 Achieving First Objective	105
5.2.2 Achieving Second Objective	108
5.3 Contribution of Study	111
5.3.1 Practical Contribution	112
5.3.2 Theoretical Contribution.....	113
5.4 Future Work.....	113
5.5 Limitation of the Study	114

5.6 Conclusion	114
REFERENCES.....	115
APPENDIX A	129
APPENDIX B	134



Last of Tables

Table 2.1: Summaries of Previous Studies Related to the Test Case Generation from Different Behavioural UML Diagrams	18
Table 2.2: Summaries of Previous Studies Related to the Comprehension of UML Diagrams	28
Table 2.3: Summaries of Previous Studies Related to the Test Case Generation Based on UML Activity Diagram.....	39
Table 2.4: Summaries of Previous Studies Related to the Test Case Generation Based on UML State Chart Diagram.....	45
Table 3.1: The Closed-Ended Questions to Evaluate the Comprehensibility of UML Diagrams with regard to test case generation	62
Table 3.2: The Open-Ended Questions to Evaluate the Comprehensibility of UML Diagrams with regard to test case generation	63
Table 3.3: NDT for Activity Graph	68
Table 3.4: Test Cases from Activity Graph	69
Table 3.5: NDT for State Chart Graph.....	73
Table 3.6: Test Cases from State Chart Graph	74
Table 4.1: Experts' Background	79
Table 4.2: The Experts' Responses to Evaluate the Perceived Difficulty of UML Activity and State Chart Diagrams with regard to test case generation.....	97
Table 4.3: The Experts' Responses to Evaluate the Subjective Confidence of UML Activity and State Chart Diagrams with regard to test case generation.....	101

List of Figures

Figure 2.2.1: Software Testing Life Cycle.....	13
Figure 2.3.1: Overview of UML Diagrams.....	34
Figure 2.5.1: Activity Diagram for Gumball Machine	36
Figure 2.5.2: State Chart Diagram for Gumball Machine.....	41
Figure 3.1: The Steps of the Research Methodology	53
Figure 3.2: Gumball Machine Described as UML Activity Diagram	65
Figure 3.3: Activity Graph Obtained from the Activity Diagram of Gumball Machine.....	67
Figure 3.4: Gumball Machine Described as UML State Chart Diagram.....	70
Figure 3.5: State Chart Graph Obtained from State Chart Diagram of Gumball Machine.....	72
Figure 3.6: NVivo Steps	81
Figure 4.1: NVivo Result of the Perceived Difficulty for Determining the Steps of Test Case Generation from Activity and State Chart UML Diagram.	86
Figure 4.2: NVivo Result of the Perceived Difficulty for Determining the Origin Diagrams for the Generated Test Cases from Activity and State Chart Diagrams.....	89
Figure 4.3: NVivo Result of the Experts' Certainty of the Generated Test Cases from Activity and State Chart Diagrams.	92
Figure 5.4 NVivo Result of Experts' Evaluation for the Comprehensibility of Activity and State Chart Diagrams in Generating the Test Cases	94

List of Appendices

Appendix A: Questionnaire.....	129
Appendix B: Summaries Of the Interview Sessions with Experts.....	134



List of Abbreviations

MBT	Model Base Testing
UML	Unified Modelling Language
DFS	Depth First Search
LR	Literature Review
GA	Genting Algorithm
OOAD	Object-Oriented Analysis and Design
AOAD	Aspect-Oriented Analysis and Design
IFD	Interaction Flow Diagram
ADT	Activity Dependency Table
AFG	Activity Flow Graph
TFG	Testing Flow Graph
ECFG	Extended Control Flow Graph
EFSM	Extended Finite State Machine
TeGeMiOOSc	Test Generation and Minimization for OO software with State Charts
NDT	Node Description Table

CHAPTER ONE

INTRODUCTION

1.1 Overview

This chapter provides an introduction to the study which begins with the background of the study, followed by the discussion of the problem. Subsequently, research questions are provided and used to construct the objectives. Finally, this chapter presents the scope as well as the significance of the research. This chapter is concluded with the summary of the main issue of this study.

1.2 Introduction

The software systems that exist throughout the world and their designs are rapidly developing and becoming more complex, a trend which very likely will continue in the near future (Meena, 2013). This development of complex software systems is a fault-prone process and these incur a great loss of time and money if neglected (Mailewa, Herath, & Herath, 2015). In this regard, Manaseer, Manaseer, Alshraideh, Abuhashish and Adwan (2015) and Jain, Jain and Dhankar (2014) remarked that software testing is the most widely used approach to ensure software quality that assists software faults detection.

On the same note, Bansal (2014) and Vashishtha, Singla and Singh (2014) stated that software testing typically consumes about 50% of the development effort, cost, and time to achieve a higher level of quality. Consequently, to reduce test challenges,

Model Based Testing (MBT) offers various significant approaches (Schweighofer & Hericko, 2014). Furthermore, Ingle and Mahamune (2015) and Singh (2014) asserted that the main challenge in software testing can be reduced by generating test cases from the Unified Modelling Language diagrams (UML), as one of the MBT approaches.

UML diagrams are the most commonly used diagrams in MBT to generate test cases (Schweighofer & Hericko, 2014). Generating test cases are the most significant process in software testing. It is a set of conditions under which a tester can determine whether a software system is working as its proposed requirements (Pahwa & Solanki, 2014; Gupta, 2014). A number of studies had generated test cases from different behavioural UML diagrams: Patel and Patil, (2013) and Jena, Swain, and Mohapatra, (2014) utilised activity diagram and Ali, Shaik, and Kumar, (2014) and Salman and Hashim, (2016) employed state chart diagram. Other studies, as in the generation of test cases from activity and sequence diagrams, used more than one diagrams (Tripathy & Mitra, 2013). These studies evaluated different UML diagrams with regard to test case generation based on evaluation criteria such as testing coverage (Ali et al., 2014), fault detection ability (Swain, Mohapatra, & Mall, 2010) and the comprehensibility (Scanniello, Gravino, Risi, Tortora, & Dodero, 2015) of UML diagrams.

Based on the fact that there is no single superior diagram in all cases of test case generation due to the different specifications of the diagrams, a tester must choose one model from the various types of models available and this chosen model must be based on the evaluation criteria (Nikfard, bin Ibrahim, Rohani & bin Selamat and Naz'ri, 2013).

The key quality of evaluating the UML diagrams is their comprehensibility, whereby if designers of modelling languages intend to have their creations to be used in real software projects, their modelling languages need to be evaluated based on comprehension (Aranda, Ernst, Horkoff, & Easterbrook, 2007; Liebel & Tichy, 2015). Furthermore, Budgen, Burn, Brereton, Kitchenham and Pretorius (2011) confirmed that comprehensibility is considered to be the most quality attribute of UML models that had been studied in the past. Comprehensibility should be considered in evaluating test case generation.

In this regard, researchers stated that evaluating the UML diagrams must be conducted based on their comprehensibility and this criterion is considered as one of the most important quality attributes of UML models that had been studied earlier (Aranda, Ernst, Horkoff, & Easterbrook, 2007; Budgen, Burn, Brereton, Kitchenham & Pretorius, 2011). In detail, the comprehensibility of UML diagrams is the understandability of users in generating the diagram of a software (Razali, Snook, Poppleton, Garratt, & Walters, 2007). However, there is still a shortage of comprehension evaluation with regard to test case generation despite the considerable importance of these two diagrams (Felderer & Herrmann, 2015). Many researchers inspected the comprehension of UML diagrams from the perspective of design and perspective of requirement specification (Condori-Fernandez, Daneva, Sikkil, & Herrmann, 2011) but not focusing on test case generation.

1.3 Statement of Problem

Software testing techniques are rapidly developing. Nevertheless, they are insufficient (Choudhary & Kumar, 2011). This inadequacy can be solved by launching fundamental

research and by using development methods and tools that can improve software testing methods (Verma, Yadav, & Tiwari, 2012; Lewis, 2016). Software testing encourages the reuse of the system modelling diagrams for testing purpose to expedite the process of software testing (Shirole & Kumar, 2013). Therefore, Shukla and Chandel (2012); Mailewa (2015) and Crowder, Carbone and Demijohn (2016) assured that one of the most used models of MBT is the UML that helps in reducing the challenges of software development and increases the effectiveness of software testing by providing generation of test cases in lesser time and effort. In this regard, Kansomkeat, Offutt, and Abdurazikand Baldini, (2008); Kansomkeat et al. (2008); Utting and Legeard, (2010); Kramer and Legeard (2016) affirmed that a tester must choose a model among the various types of models based on the basis of evaluation data to evaluate the effectiveness of testing models.

Research has shown that activity and state chart diagrams are the most frequently used UML models for testing an entire system based on its specification (Felderer & Herrmann, 2015). Moreover, Schweighofer and Hericko (2014) confirmed that there are findings of which UML diagram is the most suitable for a specific type of testing. The outcomes of their study revealed that there is a lack of research on the evaluation of different UML diagrams in finding the appropriate model of test case generation. Their study, therefore, aimed to ascertain which UML diagrams were the most commonly used for test case generation. Schweighofer and Hericko evaluated activity and state chart diagrams to test the software system using content analysis approach. Henceforth, this study strives to produce a more detailed content analysis of test case generation by focusing on these two diagrams.

Additionally, Felderer and Herrmann (2015) asserted that up to now these diagrams have not been compared in terms of comprehensibility with regard to test case generation. Based on the highlighted importance of these two diagrams and because of the lack of comprehension evaluation of these two diagrams, they consequently conducted an evaluation of the comprehensibility of these diagrams to find the appropriate model of test case generation from the testers' understanding. In this study, they evaluated the comprehension of these diagrams through participants who were undergraduate students. In their study, the reasons and specific characteristics that make activity diagram to be perceived as more comprehensible than the state chart diagram were not mentioned. Therefore, there is a need to conduct an additional evaluation to identify possible reasons for the different comprehension level for both of these diagrams.

From research point of view, replications research is one of the key mechanisms to confirm previous experimental findings (Mendonca et al., 2008; Delanote, Van Baelen Jose & Berbers, 2008; Robson & McCartan, 2016), and based on the issues aforementioned that is the need of evaluation studies of the comprehensibility of UML diagrams with regard to test case generation, the purpose of this study is to extend the evaluation study of the Felderer and Herrmann (2015)'s on comprehensibility of activity and state chart diagrams with regard to test case generation.

Moreover, this study aims to address the limitation that was mentioned by Felderer and Herrmann (2015), where they used inexperienced participants to evaluate the comprehensibility of the UML diagrams with regard to manual test case generation. The use of experts could have given a more accurate data (Creswell, 2012) and the

result can be used to complement the evaluation findings gathered by Felderer and Herrmann (2015). Accordingly, this study aims to use a group of experts in software testing who also have experiences in using UML diagrams for data collection. The experts will undergo an interview, answer evaluations questions, assess the comprehensibility of activity and state chart diagrams with regard to test case generation by examining two comprehensibility's variables (perceived difficulty and confidence).

Condori-Fernández, Daneva, and Herrmann (2011) asserted that *“there is a lack of underlying theory in the formulation of comprehensibility questions”*. This study, therefore, attempts to improve the theoretical part of the comprehensibility instrument of UML-based test case generation from the aforesaid content analysis that was conducted by Agarwal, De, and Sinha, (1999) and Aranda et al., (2007). Precisely, the study seeks to adapt evaluation questions of UML diagrams in terms of the comprehension of test case generation to support the collection of data. The questions strive to address the limitation of the study by Felderer and Herrmann (2015).

1.4 Research Questions

To resolve the issues that were discussed in the preceding section, this study puts forth the following questions:

- 1- What are the comprehension evaluation issues related to the UML activity and state chart diagrams with regard to manual test case generation?
- 2- How to evaluate the comprehensibility of the UML activity and state chart diagrams with regard to manual test case generation?

1.5 Research Objectives

The main objective of this study is to evaluate the comprehensibility of activity and state chart diagrams in terms of test case generation. The following specific objectives have been outlined in order to achieve the main aim:

- 1- To identify the comprehension evaluation issues related to the UML activity and state chart diagram with regard to manual test case generation using content analysis.
- 2- To evaluate the comprehensibility of the UML activity and state chart diagrams with regard to manual test case generation.

1.6 Scope of Study

The scope of this study can be classified under the field of the evaluation of the comprehension of UML diagrams with regard to test case generation. The following are further descriptions of the test case generation approaches and the evaluation methods of UML in test cases generation.

- The test case generation from UML diagrams

There are various approaches to test case generation. This study only focuses on the test case generation from UML activity and state chart diagrams which are considered as the most widely used diagrams of MBT.

- The evaluation of activity and state chart diagrams by comparing their comprehensibility of test case generation.

The two diagrams were evaluated using comprehensibility criterion through a set of comprehension questions. The responses to these questions were collected during One-to-One interview with the experts. Moreover, the interviews with software engineering experts who are experienced in software testing help to collect accurate evaluation data.

1.7 Significance of Study

This study focuses on the evaluation of two UML diagrams based on their comprehensibility with regard to test case generation with the assistance of experts of software testing and software engineering. The experts are intended to set the final recommendation and provide further description and understanding of MBT. This study also supports the body of knowledge as well as practice in several aspects. These are discussed further subsequently.

1.7.1 Body of knowledge

A content analysis of MBT is achieved to help in determining the following issues:

- 1) Current trends of using UML diagrams with regard to test case generation (Schweighofer & Hericko, 2014; Jena, Swain, & Mohapatra, 2014; Salman & Hashim, 2016).
- 2) Current trends in comprehension evaluation on UML diagram (Aranda et al., 2007; Budgen et al., 2011; Felderer & Herrmann, 2015).
- 3) The lack of evaluation studies for the comprehensibility of the UML activity and state chart diagrams with regard to test case generation.
- 4) Proposal of an instrument for comprehension evaluation for the experts on the two diagrams. From the existing study on comprehension evaluation, the

instrument formulated was adapted without much support from underlying theory (Condori-Fernandez et al., 2011).

Additionally, this study aims to improve the field of evaluating UML diagrams with regard to test case generation from both of the examined diagrams in terms of comprehensibility as there have been limited studies in this field. Further, this content analysis pursues more understanding of the different characteristics of activity and state chart diagrams in more specific details. The previous study by Felderer and Herrmann (2015) did not mention in detail the reasons and specific characteristics that make activity diagram to be perceived as more comprehensible than the state chart diagram with regard to manual test case generation. Moreover, the results obtained from this study based on the experts' evaluation can be used to complement the study conducted by Felderer and Herrmann (2015) in proving that UML diagrams provide the best comprehension with regard to test case generation.

1.7.2 The practical support

Testers must make practical efforts to choose among the various types of MBT diagrams (Kansomkeat et al., 2008; Utting and Legeard, 2010 and Kramer and Legeard, 2016). Therefore, evaluating the comprehensibility of UML activity and state chart diagrams with regard to test case generation seeks to unite the practical efforts of testers to determine their preference diagram for test case generation.

Testers' practical efforts are crucial in making the choice for the right model from the various types of MBT diagrams (Kansomkeat et al., 2008; Utting & Legeard, 2010; Kramer & Legeard, 2016). Therefore, evaluating the comprehensibility of UML

activity and state chart diagrams with regard to test case generation seeks to unite the practical efforts of testers to determine their preference diagram for test case generation.

1.8 Organisation of the Study

Chapter One, based on the background of the study, determines the research gap; hence formulating the research problem, the research objectives and the research questions. This chapter also describes the significance of the research and the scope of the current study.

Chapter Two reviews the literature related to test case generation, MBT and on UML activity diagram-based and state chart diagram-based generation of test cases, as well as the evaluation and comparison studies of UML-based test case generation.

Chapter Three emphasizes on the research methodology, which is developed in stages whereby problem identification, solution design, and finally, the collected data on the evaluation of the two diagrams are presented therein.

1.9 Summary

In this chapter, a brief introduction to the main issue of this study has been illustrated, that is the need for the evaluation of different UML diagrams in terms of comprehensibility in software testing. Additionally, this study aims to provide effective questions to evaluate the comprehensibility of these two diagrams.

CHAPTER TWO

LITERATURE REVIEW

2.1 Overview

This chapter describes software testing, the evaluation of different behavioural UML diagrams with regard to test case generation, the comprehensibility evaluation criterion, and test cases generation from UML diagrams. Section 2.2 gives an overview of software testing within the life cycle of a software system. The most important part of this chapter is the evolution of different behavioural UML diagrams with regard to test case generation as explained in Section 2.3. The comprehensibility evaluation criterion is explained in Section 2.4. UML-based test cases generation from activity and state chart diagrams are explained in Section 2.5 by discussing the previous studies and characteristics of the proposed diagrams as well as the differences between the proposed diagrams in terms of test case generation. This chapter is concluded with the summary in Section 2.6.

2.2 Introduction to Software Testing

With the development of technology, software becomes more advanced in code size and thus, turns to be more complex. In order to manage this development effectively, the modern process of this development is commonly proceeding in a particular development sequence activities by considering testing events (Wang, 2015). One of the challenges in the process of developing a large software system is the increasing number of errors (Yu, Xu, & Liu and Sheng, 2012). It is a fact that errors may occur at any stage of software development and these errors can incur great losses of time and

money if they are not identified and removed as soon as possible (Mailewa, 2015). Software testing is a process of finding software errors in a program in order to achieve a zero-defect software and obtain software quality (Garg, 2015). Software testing is a major measurement factor in the process of development in any software system (Dubey & Sharma, 2015; Gupta, & Yadav and Singh, 2016). Discovering all the software errors in the early stage of the development process is a primary activity that can be achieved through software testing; thereof reducing the challenges of software development and increases software quality (Sharma & Vishawjyoti, 2013).

In addition, software testing approaches have two main goals; first, demonstrating that the software meets its functional requirements, and second, to find the situations in which the behaviour of the software is mistaken, unwanted, or does not conform to its specifications (Sommerville, 2010). Software testing must be performed at several levels, which are: unit testing, integration testing, system testing and acceptance testing (Gulia & Chugh, 2015).

Software testing could be considered as an unavoidable part of any Software Development Life Cycle (SDLC) (Gulia & Chugh, 2015). Consolidating various studies on software testing leads to the categorization of software testing life cycle into four main phases: test planning, test design, test execution and test review phases (Afzal and Torkar, 2008). The categorization of software testing life cycle is summarised as in Figure 2.1.

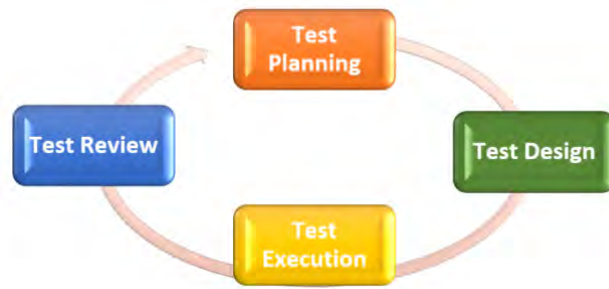


Figure 2.1: Software Testing Life Cycle

Source: (Chandu, 2015)

2.3 The Evaluation of Different Behavioural UML Diagrams With Regard to Test Case Generation

In this part, the closest research to this study is explained. The evaluation of two different UML diagrams based on evaluation criteria with regard to test case generation has been shown in previous studies. On the other hand, there are studies that aimed to enhance the efficiency of test case generation by combining more than one UML diagrams, and evaluating the result of the new integrated UML diagrams by comparing them with each single UML diagram. These details are explained in the following paragraphs.

In this section, the evaluation criteria of different UML diagrams are explained and discussed in the summary of the section to determine the evaluation criterion that is used during the current study. Some researchers, as illustrated in the forthcoming paragraphs, studied activity diagram, and others used state chart diagram. These two diagrams are considered as the scope of this study.

Kansomkeat et al. (2008) evaluated the state chart and sequence diagrams based on the number of generated test cases and the capabilities of fault detection belonging to test sets by inserting the faults by hand and then detecting the faults. The results showed

that the sets of state chart test achieved maximal capability of detecting faults than the sequence diagram sets within the unit testing level. On the other hand, the sets of sequence diagram test achieved a better capability of detecting faults than the state chart diagram sets within integration testing level. The results also showed that the state chart generated more test cases than the sequence diagrams.

In another study, Swain et al. (2010) proposed a novel technique of software testing through combined State-Activity Diagram (SAD) in order to examine the ability of fault detection. The results show that the generated approach had outperformed both activity path coverage and transition coverage used in this study. The manual selection of test data for a huge number of test cases, which took a long process and exhausted time, was mentioned as a limitation of this study.

In the same context, Tripathy and Mitra (2013) presented an approach to generate test cases via combining activity and sequence diagrams, based on the capability of an activity diagram having multiple paths and the feature of representing different interactions between the objects during the operation of the sequence diagram. Each of the diagrams used is transformed into a new graph, i.e., the Sequence Graph (SG) and the Activity Graph (AG). The two graphs integrated into a System Graph (SYG) that is used to generate the test cases. This study used a graph optimization technique, DFS, as an optimization algorithm for graph traversal. The results showed that the generated test cases via integrated method were optimized and were suitable for system testing as well as for faults detection. Even with the achieved results, the limitation of this study lies at the need to add one or more UML diagrams in the integrated approach.

Schweighofer and Hericko (2014) were concerned that *“A lot of papers present approaches for test case generation from different UML diagrams and researchers are trying to find the most optimal one”*. This study, therefore, presented the initial outcomes of a Systematic Literature Review (SLR) to explore the most commonly utilised UML diagrams to generate the test cases through different testing levels. The results displayed that the state diagram, activity diagram, sequence diagram and the integration of UML diagrams were considered as the most typically utilised diagrams with regard to test case generation.

Oluwagbemi and Asmuni (2015) claimed that several UML-based testing approaches are still suffering from inadequate criteria of test coverage and other limitations. Therefore, the authors proposed full coverage criteria in order to validate or determine the technique's performance of activity, sequence, use case, class, and state chart diagrams within four various case studies. The two metrics that were used to evaluate the techniques were, firstly, the correlations of the elements by measuring the quantity of covered and uncovered nodes as well as edges, and secondly, by measuring the coverage of the elements and nodes numbers across the mentioned UML diagrams. However, this study did not mention the evaluation for each of the used UML diagrams in terms of test case generation.

Felderer and Herrmann (2015) justified in the controlled experiment study that the selection of the right model from UML types should take into account testing aspects. They generated test cases from the two useful UML diagrams (activity and state chart) in terms of manual test case generation considering that *“activity diagrams and state machines have not been compared so far”*. Henceforth, their study evaluated the

comprehensibility of UML activity and state chart diagrams in the manual test case generation. Their experiment contained the idea of asking comprehensibility questions to a student group of 84 participants who were divided into three different groups, and the number of questions that were answered correctly were then measured. Subsequently, comprehensibility was evaluated based on four metrics: 1) the correctness of understanding that helps a participant to answer correctly the questions about the representation, 2) the measurement of the time required to understand the representation, 3) the subjective confidence of participants about the representation understanding and finally, 4) the subjective judgment of people regarding the ease to obtain information through the representation. The experiment resulted in the activity diagrams having easier Perceived difficulty as well as higher error proneness. However, the authors saw that their approach was not a standard that could compare the two diagrams when they anticipated that using more experienced testers could lead to fewer errors as the sample of students were only having a short training period with regard to test case generation. Moreover, the main aim of this study is observing the most comprehensibility errors to happen, for example missing testing steps.

Khurana, Chhillar, and Chhillar (2016) in their technique, generated and optimized the test cases by deriving the Use Case Diagram Graph (UCDG) from the use case diagram, Activity Diagram Graph (ADG) from the activity diagram and the Sequence Diagram Graph (SDG) from the sequence diagram. Then the three derived graphs were combined to generate the System Test Graph (SYTG) that were optimized by using GA in order to enhance the generation of test cases and faults detection. This study presented a new proposed GA to generate the test cases by integrating three behavioural diagrams and measured the results based on the maximum number of faults detection ability of the

newly generated approach. However, this study has recommended that future work should automate this integration approach with different UML models.



Table 2.2:

Summaries of Previous Studies Related to the Test Case Generation from Different Behavioural UML Diagrams

Author(s)	Year	Objective	Input Model	Method (s)	Evaluation criteria	Outcome	Limitation/ Future Work
Kansomkeat, et al.	2008	evaluate UML diagrams in different testing levels	-State chart diagram -Sequence diagram	Fault injection	- Number of test cases -Fault detection	- State charts generated more test cases and have better fault detection in unit level, and Sequence diagram has better detection in integration level	The study used only one project that limits the general conclusions.
Swain, et al.	2010	Compare fault detection of UML diagrams with their integration diagram	-State chart diagram -Activity diagram	Fault injection	Transition coverage and activity path coverage.	The generated approach had outperformed both diagrams	Selection of huge number of test cases was a boring process and time exhaustion
Tripathy & Mitra	2013	Compare test case generation of UML diagrams with their integration diagram	-Sequence diagram -Activity diagram	DFS algorithm	Covering all possibilities of the two combined diagrams	- Optimized, test cases suitable for system testing and detect interact, operational, scenario faults. Activity, state chart,	-There is a need to add one or more UML diagrams
Schweighofer & Hericko	2014	Compare UML diagrams systematically	-Activity -State chart -Sequence - integration diagrams	Systematic literature review	-Number of studies or LR regarding UML diagrams to generate test cases	sequence and integration diagrams are the most commonly used with regard to test case generation	Lack of empirical evaluation studies
Felderer, Herrmann	2015	Compare the manual test case generation of UML diagrams	-State chart diagram -Activity diagram	Experiment study	-The Comprehensibility	Activity diagrams have an easier Perceived difficulty as well as higher error proneness	-This approach is not a standard to compare the two diagrams - need to use more experienced testers.
Khurana, Chhillar, & Chhillar	2016	Compare the test case generation of integrating three UML diagrams	-Activity -Sequence -Use case diagrams	GA	Maximum number of faults detection ability	- Presented a new derived genetic Algorithm to generate the test cases	-The future work will be by automating this integration approach with different UML models.

As a summary of this subsection, the illustrated studies highlighted the importance of test case generation from different UML diagrams (Khurana, Chhillar, & Chhillar, 2016). The details have been demonstrated clearly in Table 2.1. On top of that, the assessments of different UML diagrams regarding the test case generation that aim to determine the choice for UML diagram should be achieved under specified evaluation criteria. In more detail, this study focuses on the activity diagram and the state chart diagram because of the lack of evaluation on the most frequently UML diagrams used for designing software systems (Schweighofer & Hericko, 2014; Felderer & Herrmann, 2015).

Examples of the evaluation criteria that are used to evaluate different UML diagrams are the percentage of coverage (Tripathy & Mitra, 2013), the maximum number of faults detection ability (Swain et al., 2010) and the comprehensibility of the diagrams with regard to manual test case generation (Felderer & Herrmann, 2015). Moreover, it is worth to mention that the lack of evaluating activity and state chart based on the comprehensibility criterion with regard to test case generation is highlighted in this subsection.

Regarding the importance of activity and state chart diagrams mentioned above, there were three assessment studies that evaluated both of these diagrams. In detail, the first study was conducted by Swain et al. (2010). In their study, they combined activity and state chart diagrams in one new proposed diagram and the proposed diagram was

compared with each single diagram based on the fault detection ability. However, they did not compare the diagrams directly based on this criterion.

The second study by Schweighofer and Hericko (2014) evaluated the activity and state chart diagrams based on the number of studies that were conducted using these diagrams without using practical evaluation criteria. Ultimately, there was one study highlighted by Felderer and Herrmann (2015) that evaluated the activity and state chart diagrams based on comprehension with regard to test case generation. Their study had a limitation whereby inexperienced participants, who were undergraduate students, were involved. They recommended experts as participants.

In this regard, the comprehensibility evaluation criterion and its related studies will be explained in the subsequent session.

2.4 The Comprehensibility Evaluation Criterion

UML has been designed with the goal of unifying the best features of various existing languages and notations. However, UML is not free of problems, and its efficacy to support program comprehension within the comprehensibility of UML diagrams has limitations. Therefore, several studies on evaluating the comprehensibility of UML diagrams have been conducted (Byckling, Gerdt, & Kuzniarzand, 2006).

Regarding the importance of this criterion and based on the issue that was aforementioned in Section 2.3, i.e., the lack of evaluation studies for UML activity and state chart diagrams based on the comprehensibility criterion, this section explains some of the previous studies that evaluated different UML diagrams based on

comprehensibility. These studies have provided foundations for the explanation and definition of this criterion and its measurement variables.

Comprehensibility is also known as understandability, by which could be defined as the degree to which information contained in representation can be easily understood by a stakeholder (Condori-Fernandez et al., 2011). Xie, Kraemer, and Stirewalt (2007) stated that the comprehensibility of UML diagrams is the search of determining the complication that most participants encounter against the learning and understanding of the represented diagrams, and that could be achieved through instructor interviews and observational studies of users' learning about the Diagram. Hadar and Hazzan (2004) expressed the bottom line of the comprehensibility as *"how well is the domain knowledge that is captured and represented in a model communicated to different stakeholders"*. Additionally, Budgen, Burn, Brereton, Kitchenham and Pretorius (2011) confirmed that comprehensibility is considered as the most important quality attribute of UML models that had been studied.

Anda and Sjøbergand (2001) evaluated comprehension of use case diagram through a set of guidelines in a controlled experiment. Each participant group, which consisted of 139 undergraduate students of software development, used one out of the three sets of guidelines when constructing a use case model from the requirement specification aspect. After completing the use case model, they answered a questionnaire to reflect their Correctness of Understanding. The results of the experiment indicated that guidelines based on templates support the construction of use case models and are easier to understand by the readers than guidelines without specific details on how to document each use case. The guidelines based on templates are also considered as the

most useful when constructing use cases. The results further indicate that it may be beneficial to combine the template guidelines with another set of guidelines that focus on the documentation of the flow of events of each use case.

Cox, Phalp and Shepperd (2001) proposed simplified guidelines of the use case in terms of comprehensibility of the use cases. They described a pilot experiment to explore whether the simplifications result in any loss of use case quality. The collected data, gained through a questionnaire answered by 24 postgraduate students of software engineering covered correctness of understanding. The results showed that the simpler guidelines had been performed at an acceptable comprehension level. Moreover, they mentioned that industrial case studies must be used to confirm whether the simplified approach warrants industrial adoption.

The empirical study of Otero and Dolado (2004) compared the semantic comprehension of three different notations for representing the dynamic behaviour in unified modelling language (UML): (a) sequence diagrams, (b) collaboration diagrams, and (c) state diagrams using eighteen students. The data was gathered through 31 final year undergraduate students of Computer Science. This study covered two variables of comprehensibility, namely, the correctness of understanding and time required for answering the questionnaire. The results showed that the software project design written in the UML notation was more comprehensible when the dynamic behaviour was modelled in a sequence diagram. Whilst if it was implemented using a collaboration diagram, the design turned out to be less comprehensible as the application domain, and consequently, the document became more complex. However,

more practical work with the models is needed in order to identify which diagrams provide the most appropriate semantics for each domain.

On the comprehension of UML diagrams to a software system, Hadar and Hazzan (2004) focused on the comprehension of use case, activity, class, sequence, collaboration, and state chart diagrams. Data was gathered from an experiment study on two groups of 55 senior computer science students by answering questions in which they were asked to rank different types of UML diagrams according to their importance. Results showed that the comprehension of the different UML diagrams varies among different people. It was also found that, when taken together, no one diagram type was globally less or more comprehension than the others. However, the differences in preference between the various teams cancelled out each other.

Kuzniarz and Staronand (2004) proposed an empirical study on using stereotypes to improve comprehension of UML models. This study elaborates this role of stereotypes from the perspective of UML, clarifies the role and describes a controlled experiment aimed at evaluation of the role in the context of model understanding. The experiment results were gained through measuring the number of correct answers in the tests checking the correctness of understanding and the required time to understand the representation by nine students of the Information Systems Programme. The results of the experiment support the claim that stereotypes with graphical icons for their representation play a significant role in the comprehension of models and show the size of the improvement.

In the same context, Razali, Snook, Poppleton, Garratt and Walters (2007) presented an investigation of evaluation into the usability of the formal notation, namely, UML-

B which allows the system properties and behaviours to be illustrated using the class and state chart diagrams. Usability in this context means the understandability, comprehensibility, learnability, operability and attractiveness of the method using an experiment that evaluates the comprehension of the produced model. The answered questionnaire showed that the method was able to achieve a higher comprehension of the participants who were ten Master's students of Software Engineering. However, the main objective was to help enhance the correctness of understanding of the method and discover any other factors that affect its use in terms of software design without referring to the software testing process.

Gravino, Scanniello and Tortora (2008), in a controlled experiment, reported the abstraction of comprehension of software requirements abstracted using a behavioural modelling approach. The subjects were 24 second year undergraduate students of Computer Science. The subjects judged, through a survey questionnaire on the correctness of understanding, the use of dynamic modelling as more useful to comprehend and interpret software requirements. Conversely, the analysis of the factors of interest revealed that there is no significant difference in the comprehension of system requirements achieved either by using or not using dynamic modeling.

Cruz-Lemus et al. (2011) presented a family of experiments to investigate whether the use of stereotypes improves the comprehension of UML sequence diagrams. The experiments consisted of one experiment and two replications that were carried out with 143 computer science undergraduates. Data were gathered through a questionnaire answered by the students who were divided into three groups. The statistical analysis and meta-analysis of the data obtained from each experiment separately indicate that

the use of the proposed stereotypes helps to improve the comprehension of the diagrams, especially when the subjects are not familiar with the domain. Introducing these stereotypes both in academia and industry could be an interesting practice for checking the validity of the results.

Shukla (2014) presented a comparative research of the effectiveness of UML AOAD (Aspect-Oriented Analysis and Design) versus UML OOAD (Object Oriented Analysis and Design), based on the comprehensibility of software systems. The class diagram has been used as the O-O and aspect-oriented modelling. Data were collected by measuring the correctness of understanding of the model and through the responses in the questionnaires which were distributed to 10 participants. The results showed that when the system is to be demonstrated to the end-user, OOAD artifacts would be more useful. On the other hand, for explaining the system to the development team, AOAD approach would be more useful and the degree of comprehensibility increases. This study also indicated that the experiment set can be increased so that more generalized conclusion can be offered.

Moreover, the comprehensibility of testing is used to evaluate a comparison of two or more UML diagrams. Felderer and Herrmann (2015) evaluated the activity and state chart diagrams through comprehensibility by measuring the correctness of understanding with regard to manual test case generation by 84 undergraduate participants. The required time to answer the questions was measured as the second variable of comprehensibility, while the third variable was the perceived difficulty. This study analyzed the manual test case generation from UML system models for the purpose of understanding what errors are made and which differences there are between

UML activity and state machine diagrams. Participants of the study were inexperienced students. The result of this controlled experiment indicated that the activity diagrams have the easier perceived difficulty than state chart diagrams. The participants expressed that the UML activity diagrams were more comprehensible but also prone to errors as compared to UML state machine diagram. In more detail, more errors occurred when generating test cases from UML activity diagrams than from UML state machine diagram during the case study. The authors justified that *"These results could mean, that the easier Perceived difficulty of activity diagrams led the participants to underestimate the carefulness demanded to derive test cases, while state machines demanded both, more care for comprehension and for test case derivation. Activity diagrams, probably due to lower formality and less rigid semantics, are on the one hand easier to understand than state machines, but on the other hand more ambiguous which may induce errors when manually deriving test cases"*. It is worth to mention that this study emphasized the use of experts in the evaluation of UML diagrams in order to get more accurate evaluation data.

Scanniello, Gravino, Risi, Tortora and Doderio, (2015) evaluated the comprehensibility of a family of a source-code using design pattern instances with UML class diagrams. This study evaluated comprehensibility based on the four comprehensibility variables aforementioned in the study by Felderer and Herrmann (2015). For the confidence comprehension variable, the study measured the participants' confidence regarding their own understanding of the source code in the comprehension task through indicating the "sure enough", "sure", and "very sure" as their own confidence level of source code understanding. The results indicate that documenting design pattern

instances achieved an improvement in the comprehensibility of source code for those participants with a sufficient level of experience.



Table 2.3

Summaries of Previous Studies Related to the Comprehension of UML Diagrams

Author	Year	Objective	Variables Measured and Method for Data Collection	Results	Diagram Type	Participant Type
Anda et al.	2001	Evaluate the comprehension of use case diagram through sets of guidelines	Correctness of Understanding through answering questionnaire	guidelines based on templates support the construction of use case models that are easier to understand for the readers, than guidelines without specific details on how to document each use case	Use Cases diagram for requirement specification	139 undergraduate students of software development and requirements engineering
Cox et al.	2001	Proposed some simplified Use Case guidelines in terms of comprehensibility of the use cases	Correctness of Understanding through answering questionnaire	the conducted simpler guidelines had been performed without significant differences of the CREWS guideline	Use Cases diagram for requirement specification	24 postgraduate students in software engineering course
Otero& olado	2004	Evaluate the comprehension of the collaboration, sequence and state chart diagrams in designing software	Correctness of Understanding and time required for answering questionnaire	-State chart is more comprehensible in R.T.S -Sequence is more comprehensible in M.I.S - Second study: by using the pair Sequence–State they gained higher comprehension.	collaboration, sequence and state chart diagrams in designing software for software modelling	31 final year undergraduates of BSc in Computer Science
Hadar&Hazzan	2004	Comprehension of: use case, activity, class, sequence, collaboration and state chart diagrams using two groups of senior computer science students	Answering questionnaire and retrieve information from UML diagram	Comprehension varies among different people	use case, activity, class, sequence, collaboration and state chart diagrams for software modelling	Group1:13 senior computer science students Group 2: 42 senior computer science students

Author	Year	Objective	Variables Measured and Method for Data Collection	Results	Diagram Type	Participant Type
Kuzniarz	2004	an empirical study on using stereotypes to improve understanding of UML models	Correctness of understanding and required time through answering questionnaire	the stereotypes with graphical icons for their representation play a significant role in the comprehension of models and show the size of the improvement	stereotypes UML models of the class diagram	Nine Information Systems students
Razali et al.	2007	The study aims to enhance the understanding of UML-B method in terms of software design,	Correctness of Understanding to answer questionnaire	The method is able to produce a comprehensible model.	UML-B of state chart diagram	Ten Masters' students of Software Engineering
Gravino et al.	2008	the comprehension of dynamic models of system requirement	Correctness of Understanding to answer questionnaire	-dynamic modelling is more useful to comprehend and interpret software requirements - there is no significant difference in the comprehension of system requirements achieved by using or not using dynamic modelling	class and object diagrams represent the identified problem domain. State chart and sequence diagrams used to represent the behaviour of the meaningful use cases presented in the functional models.	24 second year Bachelor's students of Computer Science
Cruz-Lemus et al.	2011	investigate whether the use of stereotypes improves the comprehension of UML sequence diagrams	Understanding through answering questionnaire	the use of the proposed stereotypes helps to improve the comprehension of the diagrams, especially when the subjects are not familiar with the domain	Stereotypes with sequence diagram	78, 29, 36 Computer Science undergraduates
Shukla	2014	Analyzing the Comprehensibility of Aspect-Oriented Modelling and Design of Software System. Using UML class diagram firstly based on OOAD, and secondly based on AOSD	Understanding through answering questionnaire	OOAD artifacts would be more useful and AOAD approach would be more useful and comprehensible	Design level for class diagrams of a software system	10 participants
Felderer & Herrmann	2015	evaluated the comprehension between activity and state chart diagram with regard to manual test case generation	Understanding, time for understanding, and perceived difficulty through practical questions	Activity diagrams have a higher comprehensibility and error-proneness than state chart diagrams with regard to test case generation	from activity and state chart diagrams with regard to test case generation	84 students divided into three groups at two institutions of Business Informatics and Computer Science

Author	Year	Objective	Variables Measured and Method for Data Collection	Results	Diagram Type	Participant Type
Scanniello et al.	2015	evaluated the comprehensibility of source-code using design pattern instances with UML class diagrams	Understanding, time for understanding, subjective confidence and perceived difficulty through practical questions	design pattern instances achieved an improvement in the comprehensibility of source code	source-code using design pattern instances with UML class diagrams	88 participants having different experiences (i.e., professionals, Bachelor, Master, and Ph.D. students)



The discussed studies evaluated the various UML diagrams in different perspectives. Most of the studies covered the aspect of designing level of a software (Otero & Dolado, 2004; Hadar & Hazzan, 2004; & Shukla, 2014). Some of the studies considered the aspect of dynamic modelling (Gravino et al., 2008). Other studies investigated the aspect of stereotypes with UML diagrams (Kuzniarz et al., 2004; Cruz-Lemus et al., 2011). The requirement specification aspects have been covered by Anda et al., (2001) and Cox et al., (2001). It is worth to mention that only one study covered the aspect of test case generation (Felderer & Herrmann, 2015). This particular study aims to evaluate the comprehension of activity and state chart diagrams with the aspect of test case generation as recommended by Felderer and Herrmann, (2015).

The results of the foregoing studies were different; some of these studies revealed that one of the compared diagrams has more comprehension than the others whilst some of them could not ascertain the comprehension because of the varying results. However, the four variables of comprehensibility did not cover all of those mentioned by Aranda et al. (2007) and Felderer and Herrmann (2015). The majority of studies illustrated above covered the correctness of understanding (Anda et al., 2001; Cox et al., 2001; Hadar & Hazzan, 2004; Gravino et al., 2008; Cruz-Lemus et al., 2011; Shukla, 2014). Other researchers explored the correctness of understanding and the time required for comprehensibility (Otero & Dolado, 2004; Kuzniarz et al., 2004). The perceived difficulty as the third comprehension variable was researched by Felderer and Herrmann (2015).

The majority of the abovementioned highlighted studies used inexperienced participants, undergraduate and postgraduate students. In addition, this plurality of studies collected their data by using questionnaires to research one or two comprehension variables. This lack of experts' evaluation motivates this study to use experts of UML diagrams and software testing. Therefore, data collection of this study was achieved through a questionnaire containing a number of closed-ended and open-ended questions. These evaluation questions were adapted from the foregoing studies.

In detail, the comprehensibility measuring variables will be explained particularly in the next section.

2.4.1 Criteria in Evaluating Comprehensibility

According to Table 2.3, the criteria that were used to evaluate comprehensibility do not exceed the main four metrics, which are: 1) correctness of understanding, 2) time required to understand the representation, 3) subjective confidence of participants regarding their own understanding of the representation, 4) the perceived difficulty of the representation by the participants.

The following comprehensibility measuring variables are proposed by (Aranda et al., 2007; Felderer & Herrmann, 2015; Scanniello et al., 2015):

- 1) The correctness of understanding: The degree to which participants can answer questions about the representation correctly. This variable is more suitably covered by participants' generating test cases to reflect their understanding (Anda et al., 2001).

- 2) **Time:** Time required to understand the representation. This variable must be covered by measuring the time of answering understanding questions by participants. However, this variable can only be captured when conducting a case study and asking participants to design or generate the practical side of the representation as what have been done by Razali, Snook, and Poppleton (2007) and Felderer and Herrmann (2015) .
- 3) **Confidence:** The subjective confidence that participants display regarding their own understanding of the representation (Scanniello et al., 2015).
- 4) **Perceived Difficulty:** The subjective judgement that participants display regarding the ease to obtain information through the representation (Scanniello et al., 2015).

Furthermore, the adopted comprehensibility variables will be used to evaluate the UML activity and state chart diagrams with regard to the test case generation. The test case generation from activity and state chart diagrams will be explained in the following sections with summarizing the related previous studies.

2.5 Test Cases Generation from UML Diagrams

Software testing contains the implementation of the software on a group of test cases and verifies the results with the expected results (Ali and Shaikand, 2014). The test case is an explanation of how a test could be performed on required features that require the System under Test (SUT) to confirm that it runs as expected to meet the particular purpose of the system. The set of test cases is called the test suite in which the whole scenario of the test could be elucidated, having pre and post conditions and the failing and passing criteria (Lucantonio, 2015).

From the various techniques that are used in terms of test cases generation, some of these methods are commonly used and depended upon more than others by experts (Konka, 2012). One of the main approaches to software testing is the MBT which is applied at the design phase of system process that provides the early faults detection approach (Jena et al., 2014b). Generating a test case from design documents has the significant feature of allowing test cases to be available early in the software development life cycle and helps to minimize testing cost (Pandey & Mohapatra, 2012).

MBT case generation is more efficient and effective than the code-based test case generation (Wang, Jiang and Shi, 2015). With the increasing use of the UML in Object-Oriented systems, researchers have started the investigation on how the UML can be used in the testing phase in the process of software development (McQuillan & Power, 2005).

Model-based approaches identify respective test cases for the software with respect to the UML diagrams that can be categorized into two main types, behavioural diagrams and structural diagrams (Gupta, 2014) as shown in Figure 2.1.

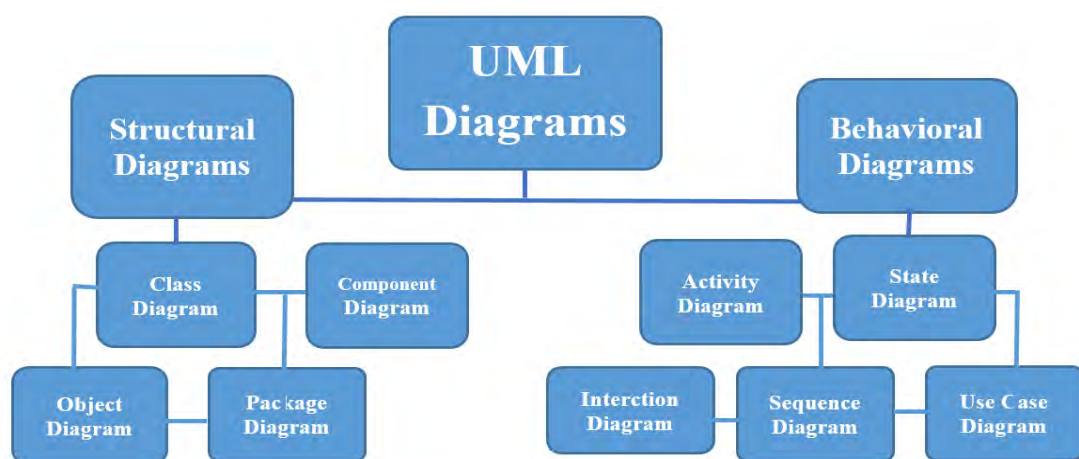


Figure 2.2: Overview of UML Diagrams

Source: (Gupta, 2014)

From Figure 2.2, two behavioural UML diagrams that are focused are activity and state chart diagrams. These two diagrams are considered as the most frequently used UML models for testing an entire system based on its specification (Felderer & Herrmann, 2015). Moreover, Schweighofer and Hericko (2014) confirmed that there are some findings of which UML diagram is the most suitable for each type of testing. However, Felderer and Herrmann (2015) highlighted the lack of evaluation studies for these two diagrams with regard to manual test case generation. This study, therefore, focuses only on these two diagrams as will be explained through the next coming Sections.

2.5.1 Test Case Generation from UML Activity Diagram

The UML activity diagram is used for describing behaviours of a software system by modelling the sequence of activities in the process, generating the test cases and also describing all possible flows of execution in a use case process (Jena et al., 2014; Schweighofer & Hericko, 2014). The usage of an activity diagram and its graphs is a good way to guarantee the production of the test cases (Hashim & Salman, 2011).

Activity diagrams are used to visualize the flow of controls in a system (Kaur & Bajaj, 2015). The flowchart of an activity diagram is a simple model that makes it very easy to be understood and can be used as the initial diagram to study systems (Patil & Ganeshwade, 2014). An activity diagram is normally used for generating test cases in system testing level (Mussa et al., 2009). In addition, it can also be used to generate test cases based on gray-box testing (Kundu et al., 2009).

Activity diagram consists of two main elements; activity and transition which could be used to measure the coverage of testing as shown in Figure 2.3 (Khandai & Acharyaand, 2011b). There are several studies that focus on the generation of test cases based on

activity diagram in various methods and intermediate graphs as elucidated in the following paragraphs and the main points of these studies are highlighted in Table 2.1.

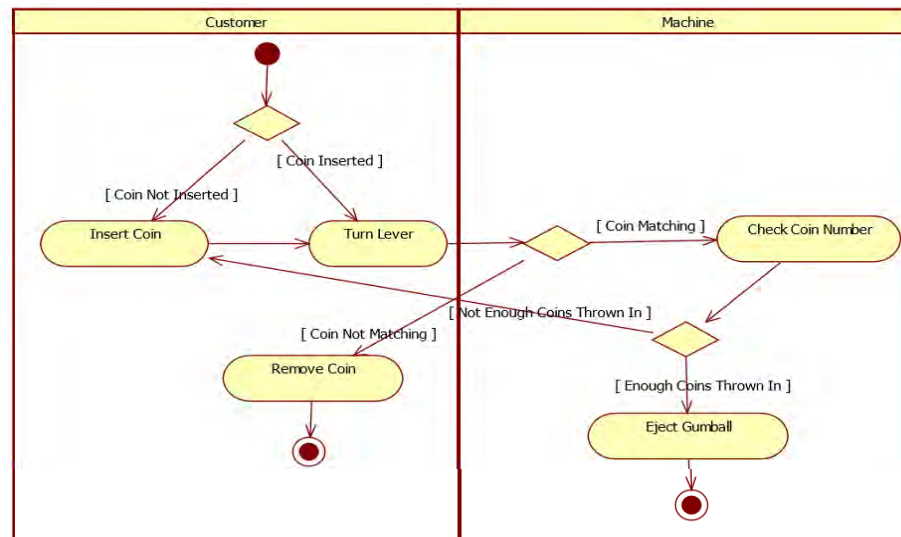


Figure 2.3: Activity Diagram for Gumball Machine

Source: (Felderer & Herrmann, 2015)

Referring to Debasish Kundu and Debasis Samanta (2009) study, they presented a way for creating test cases from an activity diagram via the activity path graph. They mainly depended on activity path coverage criteria as test evaluation criteria. Primarily, their approach aims to detect maximum errors and reduce testing efforts to enhance the quality of SUT.

On the same content, Heinecke et al. (2010) proposed a mechanism of test case generation based on the UML activity diagrams via the path coverage criteria by the Interaction Flow Diagram (IFD). Subsequently, they examined the test steps on the diagrams, accumulated the corresponding test plans and demonstrated the overall possibility of the proposed approach and evaluated their results.

In another study, Hashim and Salman (2011) proposed an enhanced technique for automatic generation of test cases directly from the UML activity diagram by the means of activity graphs. The produced test cases are created automatically, which can be compared to test cases that are generated manually so as to assess the usability and reliability of the technique. Their produced results showed that the test cases created automatically resemble those which are manually derived.

In the same context, Boghdady et al. (2011) proposed an automated mechanism to generate test cases from the UML activity diagram, and then the activity of Dependency Table (ADT) which is converted to a targeted graph, named activity dependency Graph (ADG). The branch coverage criteria are utilized to cover the path, and Depth First Search (DFS) navigation scheme is deployed on the graph to obtain all the possible test ways.

Shukla and Chandel, (2012) presented an idea of making test cases from the activity diagrams with the scope of use case and using path coverage criterion with the aim of detecting more errors (e.g. synchronization errors and loop errors).

Patel and Patil (2013) generated test cases from the UML activity diagrams from use case scope firstly, and automatic test case generation from the UML activity diagram by means of activity path secondly. The test cases are produced through activity path coverage criteria; such a path lies in the activity graph, enabling the discovery of errors in the test.

Jena et al. (2014b) in the Novel Approach, generated test cases from a UML activity diagram through a genetic algorithm that is used for early detection of errors. The authors produce an activity Flow Table (AFT) and then convert it to activity Flow

Graph (AFG) with the help of activity coverage criteria. They pass through the AFG and make test paths and finally, create the test cases from these paths by deploying their genetic algorithm.



Table 2.4:

Summaries of Previous Studies Related to the Test Case Generation Based on UML Activity Diagram

Author(s)	Year	Objective	Method (s)	Intermediate model	Outcomes
Kundu and Debasis Samanta	2009	To increase faults detection and reduce testing efforts	Activity diagram with use case scope	Activity graph	Detecting more faults and reducing testing effort
Heinecke, et al.	2010	To generate high-level test plans automatically	Business process	Interaction Flow Diagram (IFD).	Generating high-level test plans automatically from business processes
Nor Laily Hashim, Yasir D. Salman ²	2011	To compare usability and reliability of the automatically and manually generated test cases	An improved algorithm	Activity graph	The automatically generated test cases are the same as the one manually derived.
Boghdady et al.	2011	To save testing time and effort	Activity Dependency Table (ADT)	Activity Dependency Graph	Regression as well as integration testing
Shukla & Singh	2012	To create early detection of faults, reduce testing time	Activity diagram	Activity graph	Detecting more faults like synchronization faults, loop faults
Puneet Patel & Nitin Patel	2013	To automatically generate test cases	Comparing two activity diagram approaches	Executing loop from zero time until n+1	For each increment or decrement, operator of the loop is tested
Jena, et al.	2014	To create early detection of faults, reduce testing time, cost and efforts	Activity diagram with genetic algorithm	Activity Flow Graph (AFG)	Optimized generation of test cases from the paths using Genetic Algorithm

As a summary of this section, and from Table 2.4 above, it can be clearly noticed that past studies in the field of test cases generation from activity diagram shared some objectives like reducing test efforts, time and cost. These objectives were achieved through various methods, tools, and intermediate models. However, Heinecke et al. (2010) and Hashim and Salman (2011) used the automatic methods.

Various methods and tools have been used to achieve these targets and one of the most important ways is via an improved or genetic algorithms like the approaches of activity diagram of Hashim and Salman (2011) and Jena, et al. (2014b). Most of the techniques of these studies established an intermediate model such as generating the activity graph from activity diagram to ensure that each one independent path in the program is executed at least one time through the path coverage criterion. Some of the used algorithms focus on covering specific criteria and cover the maximum number of faults.

The importance of using UML diagrams was reflected in the previously discussed studies on the use of activity diagram.

2.5.2 Test Cases Generation from State Chart Diagram

UML state chart diagrams can be used to form a system's dynamic behavioural aspects and it consists of states, transitions, actions and events (Rumbaugh & Jacobsonand, 2004). Furthermore, each diagram ensures the flow of control from one state to another when each node represents the state and the arrow connecting the states representing the transition as shown in Figure 2.4 (Ali, Shaik, & Kumar; 2014). The state chart

diagram takes the lead position in a number of selected primary studies based on UML diagrams (Schweighofer & Hericko, 2014).

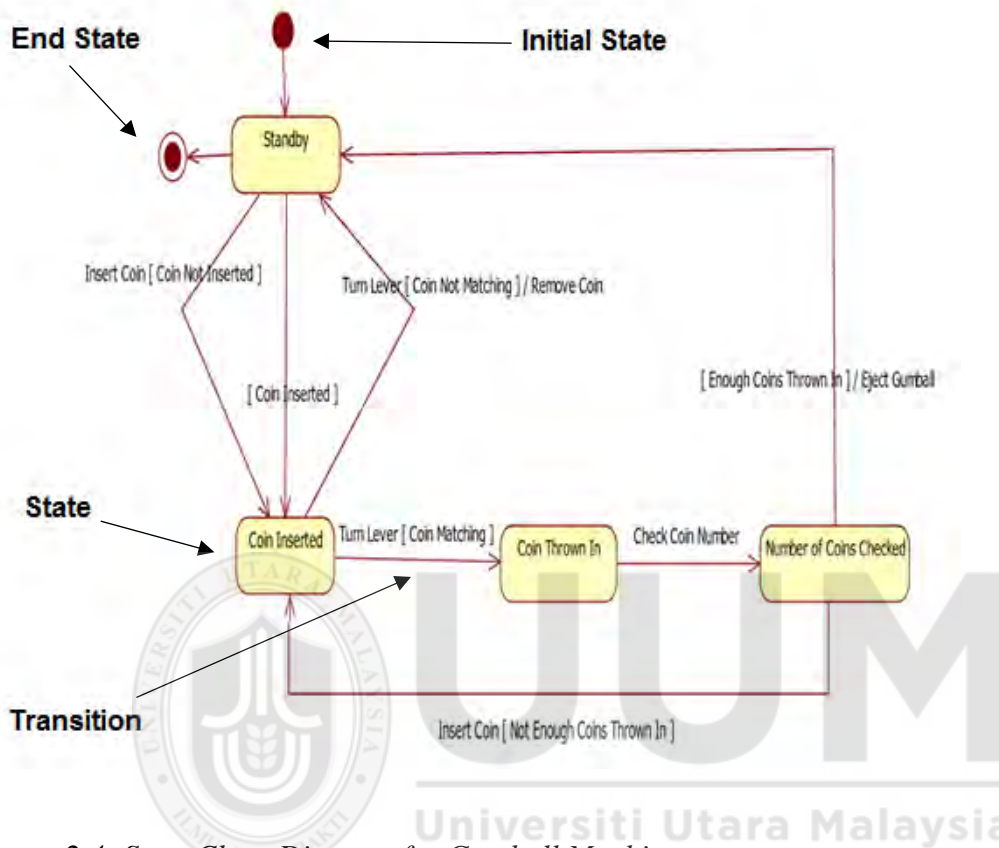


Figure 2.4: State Chart Diagram for Gumball Machine
Source: (Felderer & Herrmann, 2015)

State chart diagram is a suitable model for deriving test cases for unit testing level, and transitions are considered as the main building blocks of the state diagram (Khandai & Acharyaand, 2011a). However, system-level test cases can be generated initially from use case models and is then refined using state chart diagrams (Boghdady, Badr, & Hashemand, 2011b). This diagram results in large numbers of test cases, due to the consideration of every state that an object undergoes during its operation (Khandai et al., 2011b). Moreover, the state chart diagram has the state coverage, transition coverage, and path coverage criteria (Al Dallal & Sorenson, 2006).

State chart diagram is useful when the state of an object in its life cycle is important. It defines the sequence of states an object goes through in response to events. Events are external factors responsible for a state change (Kaur & Bajaj, 2015).

A state chart diagram is good at describing how the behaviour of an object changes across several use case executions and how the state of object changes in its lifetime (Mohanty & Acharyaand, 2011). There are several studies that focus on generating test cases based on state chart diagram in various methods and intermediate graphs as clarified in the following paragraphs and the main points of these studies were highlighted in Table 2.4.

Kansomkeat and Rivepiboon (2003) designed a method of transformation from the state chart diagram into an intermediary diagram to generate test cases automatically. The test cases measured the efficiency of the test case creation when the all-state coverage and all-transition coverage are used.

Doungsa-ard, Dahal, and Hossainand (2007) proposed a method for generating test cases from state chart diagram with the help of a genetic algorithm that aims to minimize the efforts of test case generation. In this method, each state comprises of a state name and a transition when the name of a state is used for stipulating a specific state. Their method of exploration for a transition and state coverage happens in an order.

Reza and Ogaardand (2008) proposed a prototype-based software testing via state chart diagram. They demonstrated how a model-based testing is utilized for the aim of software testing. However, they have not yet discovered the clarifications on the problem of concurrency modelling and the back-end modeling of web applications.

Kosindrdecha and Daengdeg (2010) proposed a novel scheme for generating test cases based on state chart diagrams, named “TGfMMD” scheme which is designed to validate the state chart diagrams prior to generating test cases from lengthy state chart diagrams. Nonetheless, this scheme is still not verified yet with a compound state chart diagram.

Shirole, Suthar and Kumar (2011) offered an approach of test case generation via state chart diagram using GA through the following steps: initially, converting the state chart diagram to Extended Finite State Machine (EFSM). Next, converting the EFSM into Extended Control Flow Graph (ECFG), and finally, with the help of GA, generating the test cases through data flow methods. They focus on the state coverage, transition cover, all definitions, and all du-paths coverage. However, all the routes coverage is not totally found in this study.

Swain and Beheraand (2012), proposed test case generation using state chart diagram. They changed the presented state chart diagram into state transition graph that is utilized to shape test orders and generate all the achievable routes. In conclusion, they reduce a group of test cases by computing node’s coverage for every order of test.

Ali, Shaik, and Kumar (2014) proposed a technique for test case creation, with the aims of minimizing the time and enhancing the consistency of software testing by transforming the state chart diagram to the finite state machine. In addition, the suggested scheme achieves the adequate test coverage without enhancing the number of test cases. Moreover, it can attain more significant coverage like transition coverage, transition pair coverage, and offers state coverage.

Salman and Hashim (2014) in their research illustrated the use of state chart diagram with path graph testing to generate test cases. The path graph is then converted to path

testing and is used for test case suites to minimize the created test paths. The test cases, which are appropriate for program testing, are then created. Path testing is a structural testing method traditionally followed in testing a system under test and the available test paths could give an idea to the software developer that one must assure that those paths are properly coded.



Table 2.5:

Summaries of Previous Studies Related to the Test Case Generation Based on UML State Chart Diagram

Author(s)	Year	Objective	Method (s)	Intermediate Model	Outcome
Kansomkeat, et al.	2003	To make the midway model TFG for test case generation	- Parsing TFG, mutation analysis -Rational Rose tool	Testing Flow Graph (TFG)	Based on their error detection capabilities, their test cases measure the efficiency of the test case creation
Doungsa, et al.	2007	To generate test data for the state chart diagram	GA	-	Useful tool for software to generate test data from state diagram
Reza, et al.	2008	To propose model based testing	Front end verifying links	Prototype based software testing	Test the Front-end functionality of a web application.
Kosindrdech, Daengdeg	2010	To reduce time and cost to generate testing	TGfMMD Method	Sketch Diagram	Generation of test case and test data based on state chart diagrams
Shirole, et al.	2011	To generate test cases that combine with information from state chart diagram	GA	ECFG	Automatic generation of feasible test paths and data
Swain, et al.	2012	To minimize time and cost for software testing	Test Generation and Minimization for O-O with State charts (TeGeMiOOSc)	State graph	Optimize test coverage by minimizing time and cost.
Azaharuddin Ali et al	2014	To minimize the time and enhance the consistency of testing	Mined information, pre and post condition to build test case	Finite state machine	Attainment of sufficient test coverage without increasing the number of test cases
Yasir Dawood, Nor Laily Hashim	2014	To enhance method and reduce the test paths for testing	Design specifications, and present a path testing for the test case.	-Path graph -path testing	The generated test cases are suitable for system testing and to detect interaction and scenario faults.

To summarize this section, and from Table 2.5 above, most past studies of generating tests from state chart diagram carry the same objectives which are reducing the efforts, cost and time for test generation as well as increasing the ability of faults detection. On top of that, various methods and tools have been applied in these studies, for instance, an improved or genetic algorithms as the approach of state chart diagram of Kosindrdecha and Daengdeg (2010); Shirole, et al. (2011) and Salman and Hashim, (2014). However, these past studies gained their results through the use of state chart diagram as a test case generation approach. As coverage criteria, there are several studies which used different criteria like transition coverage, state coverage, and path coverage. The path coverage was used by Shirole, et al. (2011); Swain and Beheraand (2012) and Salman and Hashim (2014). The intermediate graphs that have been used are Testing Flow Graph by Kansomkeat and Rivepiboon (2003) , state graph by Swain and Beheraand (2012) and path graph by Salman and Hashim (2014).

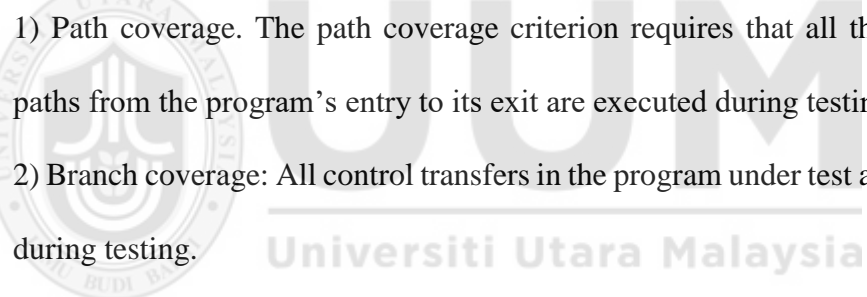
The importance of using UML diagrams and state chart diagram was reflected in the previously discussed studies as will be summarized in the subsequent paragraph.

2.6 Summary of Chapter Two

This chapter provides a review of past studies on software testing. Past studies on test case generation methods based on UML diagrams have been presented in detail. The focus of this chapter was on the two most widely used diagrams which are activity and state chart. On top of that, the empirical evaluation of the two mentioned diagrams had been discussed based on the proposed criteria.

The summary of section 2.3 and its subsections was on determining the gap in the previous studies, which is the lack of evaluation study between activity and state chart diagrams with regard to test case generation even though the importance of these two diagrams in the behaviours of a software system and with regard to test case generation were mentioned by past researchers. Moreover, these two diagrams consist of activity, states, and transition which could be used to achieve the path and edge coverage of test cases. However, the proposed solution of the expressed issue is elucidated in Chapter Three.

It is important to mention that the coverage of test cases that are highlighted by the aforementioned studies could be summarized as follows: (Zhu, Hall, & May, 1997)

- 
- 1) Path coverage. The path coverage criterion requires that all the execution paths from the program's entry to its exit are executed during testing.
 - 2) Branch coverage: All control transfers in the program under test are exercised during testing.
 - 3) State or node coverage: The shortest number of paths following which all the nodes will be covered is determined.

The illustrated testing coverage types provide assistance in preparing the comprehensibility evaluation questions. Testing coverage types reflect a practical guide for comprehensibility measuring variables (the perceived difficulty) as was shown in Chapter Three.

On the other hand, the focus of section 2.4 was on the importance of the comprehensibility criterion to evaluate the UML diagrams. The main part of this section referred to the importance of evaluating the comprehension of different UML

diagrams. It is interesting to note that Condori-Fernández, Daneva and Herrmann (2011) asserted that there is a lack of underlying theory in the formulation of comprehensibility questions.

In Summary, the analysis of the content of past studies that are related to the UML activity and state chart diagrams helped in determining the following issues:

1) The importance of the UML activity and state chart diagrams with regard to test case generation area (Schweighofer & Hericko, 2014; Jena, Swain & Mohapatra, 2014; Salman & Hashim, 2016). From Table 2.4 that summarized the past studies of test case generation from activity diagram, it can be clearly noticed that most of the past discussed studies highlighted the importance of the use of activity diagram as one of the most frequently used diagrams with regard of test case generation. Moreover, most of these studies established an intermediate model such as generating the activity graph from activity diagram that will be adopted during this study.

2) The importance of comprehensibility criterion to evaluate the UML diagrams (Aranda et al., 2007; Budgen et al., 2011). From Table 2.5 that summarized the past studies of test case generation from state chart diagram, it can be clearly noticed that most of the past discussed studies highlighted the importance of the use of state chart diagram as one of the most frequently used diagrams with regard of test case generation. Moreover, most of these studies established an intermediate model such as generating the activity graph from state chart diagram that will be adopted during this study.

- 3) The lack of evaluation studies for the comprehensibility of the UML activity and state chart diagrams with regard to test case generation (Felderer & Herrmann, 2015). From the discussed studies evaluated the various UML diagrams in different perspectives and as summarized in Table 2.3. There was only one particular study aims to evaluate the comprehension of activity and state chart diagrams with the aspect of test case generation. The majority of studies illustrated above covered the correctness of understanding as the most important criterion to evaluate the UML diagrams. Additionally, the lack of experts' evaluation that highlighted from the past studies motivates this study to use experts of UML diagrams and software testing.
- 4) The lack of underlying theory in the formulation of comprehensibility questions was highlighted by Condori-Fernandez et al. (2011) within the importance of this theory as highlighted by Agarwal, De, and Sinha, (1999) and Aranda et al., (2007).
- 5) The need to conduct evaluation research on the comprehensibility of UML activity and state chart diagrams with regard to test case generation by experts (Felderer & Herrmann, 2015). This issue was highlighted from past studies that summaries in Table 2.3.

Moreover, the evaluation questions will be adapted from the foregoing studies to collect the data for this study in order to treat the aforementioned issues of evaluating the comprehensibility of activity and state chart diagrams with regard to test case generation.

Ultimately, the aforesaid issues are shown in Chapter 3 with reference to the method to address the highlighted problem. The experts' evaluation of the comprehensibility of the UML activity and state chart diagrams with regard to test case generation are elaborated in Chapter 4.



CHAPTER THREE

RESEARCH METHODOLOGY

3.1 Overview

This chapter focuses on the research methodology that is applied in this study. The research methodology is a structured set of guidelines or activities to help in generating a valid and reliable research results (Mingers, 2001). The preceding chapter provided a review on the related studies. The analysis of the content of past studies offers the understanding on the issues that are related to the area of the study. This chapter discusses the method that have been utilised and the processes that are involved in attempting to answer the research questions and achieve the research objectives as illustrated in Chapter 1. Chapter Three presents the design of the research methodology in Section 3.2, followed by the phases of the study in sub-sections 3.2.1 and 3.2.2 and finally, Section 3.3 which summarizes the whole chapter.

3.2 Research Design

Research design provides plans and procedures for a research to address the research problem. The strategies involved span from broad assumptions to detailed methods of data collection and analysis (Creswell, 2012). There are three main approaches to data collection and analysis of the data: qualitative, quantitative, and mix method. This study attempts to investigate the comprehension of two behavioural UML diagrams during their process of test case generation using the one-to-one interview in order to collect data. The interview is a part of the qualitative method approach. Many common

qualitative research instruments can be used to collect qualitative data, including participant observation, one-to-one interview, email-interview, and focus group interviews. One-to-One interviews is useful for interviewing the experts who are courageous talkers and who provide rich data straightforwardly (Creswell, 2012).

It is worth to mention that in order to collect qualitative data from interviews, Creswell (2012) explained that “ *you may ask some questions that are closed-ended and some that are open ended. The advantage of this type of questioning is that your predetermined closed-ended responses can net useful information to support theories and concepts in the literature. The open-ended responses, however, permit you to explore reasons for the closed-ended responses and identify any comments people might have that are beyond the responses to the closed-ended questions*”. Therefore, groups of close-ended and open-ended questions are used in collecting the experts’ response as listed in Table 3.1 and 3.2. Moreover, the closed-ended questions are used to evaluate the results of open-ended questions from the interviews sessions.

Furthermore, the methodology of this study is adapted from Hadar and Hazzan (2004) which are divided into problem identification, solution design, data gathering and data analyzing as described in Figure 3.1.

- 1) The problem identification: based on past studies.
- 2) The solution design: the experts’ evaluation are adopted to solve the problem through evaluation questions.
- 3) Data gathering and analyzing: collecting the evaluation data through interviewing experts. To analyze these qualitative data, NVivo is used.

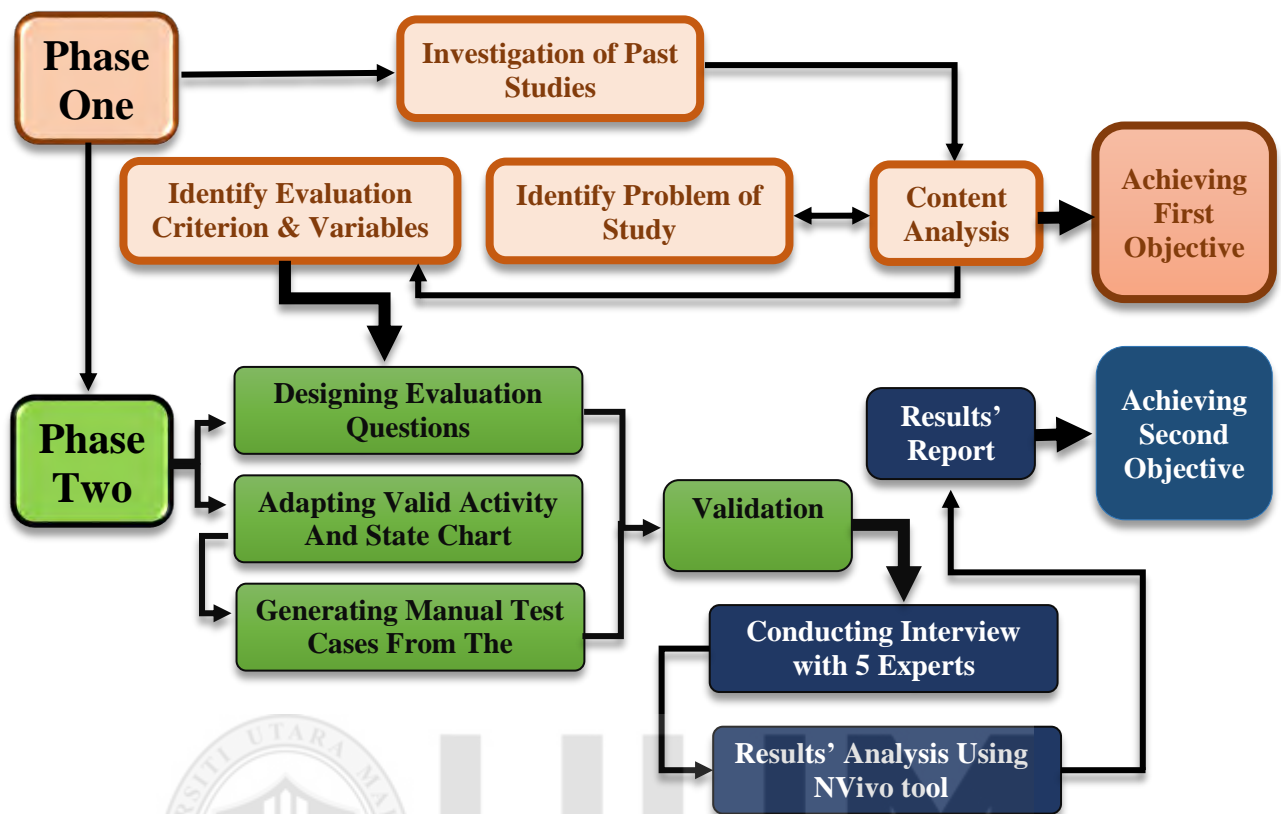


Figure 3.1: The Steps of the Research Methodology

3.2.1 Phase One

In this phase, the first objective is realized through the following steps:

3.2.1.1 Investigation of Previous Studies

Past research were reviewed in order to identify the issues and gaps related to the domain of the study. Consequently, the main ideas were gained through the literature in both printed and online references. Among them are journals, proceeding papers, and books. Based on the knowledge gained, the problem and scope of the study were defined in Sections 1.3 and 1.6, of Chapter 1.

One of the activities that was conducted was identifying the lack of evaluation studies for the comprehensibility of activity and state chart diagrams in test cases generation. The reviews on related work in Chapter Two have strengthened the need to propose a solution to this issue.

i) Content Analysis

Content analysis involves discussion and summarisation of the outcomes of the previous studies related to this research. According to Habib (2009), content analysis is *“a research technique for making replicative and valid inferences from data to their context”*.

In this regard, the outcomes of analyzing the content and results of the previous studies are summarized and shown in Section 2.3 and its subsections 2.3.1 and 2.3.2, complete with their summary tables in Table 2.1 and Table 2.2. In addition, Section 2.3.1 is summarized as in Table 2.3 and Section 2.3.3 is summarized as in Table 2.4. Finally, all the explained summaries emerged as the main issues of the study in Section 2.4.

Among the activities that were conducted is identifying the significance of using UML diagrams in software testing in order to reduce the challenges of software testing' for example, reducing the costs and efforts (Ingle & Mahamune, 2015). Notwithstanding the above, there are still needs for evaluation studies of UML diagrams, especially for activity and state chart diagrams with regard of test case generation as asserted by Felderer and Herrmann (2015).

Additionally, the comprehensibility evaluation criterion was described as the key quality of UML diagrams by Aranda, Ernst, Horkoff, and Easterbrook (2007) and

Liebel and Tichy (2015). In this regard, Condori-Fernandez et al. (2011) noted that there is a lack of underlying theory in the formulation of comprehensibility questions. Subsequently, the current study strived to deal with these issues by adapting a number of evaluation questions from past evaluation studies. The adapted questions took into account the specific definitions of the used variables of comprehensibility that are listed by Aranda et al. (2007) to be validated for use during the interview.

Ultimately, the collected data from the interview sessions will improve the content analysis of this study through the discussion and the specific characteristics of the evaluated diagrams.

ii) Problem Identification

The problem criteria are important in order to determine the research gaps (Macintosh & Colemanand, 2009). There are two significant issues related to this area that have been highlighted. Firstly, the need to collect experts' evaluation results of the comprehensibility of activity and state chart diagrams as asserted by Felderer and Herrmann (2015). This is considered as the main focus of the current study. The second important issue that is related to the initial study is the lack of underlying theory in the formulation of comprehensibility questions (Condori-Fernandez et al., 2011).

iii) Identifying Evaluation Criterion and Variables

Previous studies have highlighted that the key quality of evaluating the UML diagrams is their comprehensibility (Aranda, Ernst, Horkoff, & Easterbrook, 2007; Liebel & Tichy, 2015). In addition, Budgen, Burn, Brereton, Kitchenham and Pretorius (2011) confirmed that comprehensibility is considered as the most quality attribute of UML

models. Therefore, the assessment of activity and state chart diagrams are conducted based on this comprehension criterion.

3.2.2 Phase Two

This phase provides the solution design of the evaluation study. The second phase is regarded as the outset of the evaluation research. The evaluation study was conducted by adapting a group of evaluation questions that were designed based on the comprehensibility criterion. The case study was adapted from Felderer and Herrmann (2015) and the test cases were generated from activity and state chart diagrams from the adapted case study. Experts' responses were gathered during the interview. The illustrated steps are considered as the initiation of the second objective of this research. During this phase, the assessment instrumentation was designed in order to collect experts' data to solve the aforementioned problem. Finally, one-to-one interview with the experts were conducted, as explained in Section 3.2.2.2 and its subsections.

3.2.2.1 Instrumentation Design

Interview questions can be defined as a set of questions that are answered by the respondents, whose responses are recorded (Sekaran & Bougie, 2010). During the interview, the researcher asks a small number of common questions that elicit replies from the participants. This study involved the administration of one-to-one interviews with four experts (Creswell, 2012). The interview questions were prepared based on the comprehensibility evaluation criterion that has been identified from the content analysis in Section 2.4.1, which are 1) correctness of understanding, 2) time required to understand the representation, 3) subjective confidence of participants regarding their

own understanding of the representation, 4) the perceived difficulty of the representation by the participants.

Two of the four substantial variables were used for the interview questions in this study. The variables are: 1) the subjective confidence and 2) the perceived difficulty. The following points are the justifications for limiting to only two variables:

- 1) Aranda et al. (2007) confirmed that *“there are many comprehensibility variables to consider, and it may not be feasible to evaluate them all in a single study. Thus, the choice of which of these should be addressed is up to the researcher”*. Therefore, the initial study will not choose all of the elaborated variables.
- 2) Felderer and Herrmann (2015) in their evaluation study highlighted that the participants had expressed that the activity diagram gives more comprehension than state chart diagram with regard to manual test case generation. On the contrary, the rates of errors as conveyed by the participants of the case of study are higher in activity diagram. The authors believed that the contrast in the results led the participants to pay less attention to the generation of the test cases as compared to state chart diagram that needs more attention because of its difficulty. However, this contradiction in the results leads us to avoid falling into the same predicament of using a case of study and measuring the correctness of generating the test cases by participants in addition to the required time for that. Ultimately, this study used the subjective confidence and the perceive difficulty as the comprehension measuring variables.

In this regard, a group of questions was used to achieve the proposed evaluation. The question form includes both closed-ended and open-ended questions. These questions examined the practical understanding of the of test case generation between activity and state chart diagrams. The closed-ended questions as in Table 3.1 are contained in Section A whereby they evaluate the comprehension of the activity diagram; whilst Section B evaluates the comprehension of state chart diagram. Each Section A and B has two parts; the first part are three questions to evaluate the perceived difficulty of the diagrams, and the second is for the subjective confidence of comprehension of the diagram. The closed-ended questions are prepared based on an ordinal scale that provided the respondents with five possible levels of answers from (-2) for Strongly Disagree or very difficult to (2) for Strongly Agree or very easy. The sources of the adapted closed-ended questions are listed in Table 3.1 and Table 3.2 is for open-ended questions.

I. The Questions of Perceived Difficulty

The first comprehensibility variable of this study is perceived difficulty which is judged by the participants based on the information related to the representation (Scanniello et al., 2015). Moreover, to measure the perceived difficulty, participants were asked to express their responses to the questions by explaining whether the representation was easy or difficult (Figl & Laue, 2011). Additionally, this variable is checked with a simple question: the researcher asked the participants to express their satisfaction with the performance of the representation. This reflect their individual perceptions through questions that examine their understandability of the information that they gained on the practical execution of the representation (Ribiero & Yarnal, 2010).

The closed-ended and open-ended questions in Tables 3.1 and 3.2 respectively examine the perceived difficulty of the participants when they generate test cases from the activity and state chart diagrams. The closed-ended questions 1, 2, and 3 contain the difficulty of learning the test case generation, the difficulty of comprehending the test case generation and the difficulty of achieving different test coverage from the examined UML diagrams.

The first closed-ended question asks about the degree of difficulty to learn the test case generation from activity diagram or from state chart diagram (Siau & Cao, 2001; Ribiero & Yarnal, 2010). The second checks on the degree of difficulty that is the required tasks of test case generation from activity diagram or state chart diagram; this is adapted from Razali et al. (2007) and Ribiero & Yarnal, (2010). The third question aims to examine the level of difficulty to achieve different testing coverage when the experts generate test cases from activity diagram or state chart diagram, which is adapted from Felderer and Herrmann, (2015) and Ribiero and Yarnal (2010). These three questions basically reflect the difficulty of understanding the process of test case generation from the adapted UML diagrams, they simulate whether the representation is easy or difficult (Figl & Laue, 2011). Essentially, these three closed-ended questions support the two open-ended questions.

The first open-ended question examines the perceived difficulty of the participants through the obtained information of test case generation steps from the evaluated UML diagrams that was adapted from Razali et al., (2007), Felderer and Herrmann, (2015) and Ribiero and Yarnal, (2010). In detail, the steps of test case generation are the input data and the expected results that reflect the perceived difficulty. The second open-

ended question is to determine the origin of the generated test cases from activity or state chart diagrams, as adapted from Razali et al., (2007) and Ribiero and Yarnal, (2010). This question examines the perceived difficulty through the obtained information of the practical process of test case generation from activity and state chart diagrams to reflect the perceived difficulty (Aranda et al., 2007). Therefore, the three (3) closed-ended questions in Section A1 for Table 3.1 match the first two (2) open-ended questions for Table 3.2 in measuring the perceived difficulty of the examined diagrams with regard to test case generation through asking about the information and process of test case generation and test coverage.

II. The Questions of Subjective Confidence

The second comprehensibility variable of this study is the subjective confidence that refers to the subjective confidence that people display regarding their own understanding of the representation (Scanniello et al., 2015). The closed-ended and open-ended questions in Tables 3.1 and 3.2 respectively examine the subjective confidence of the participants when they generate the test cases from the activity and state chart diagrams. In this regard, the first closed-ended question aims to examine the degree of confidence of the experts regarding test case generation from activity diagram or state chart diagram, if the experts get the task of explaining test case generation from UML diagrams to others. This question was adapted from Shukla (2014), and Aranda et al. (2007).

Koriat (2011) explained that the preferred choice (favoured choice) of the participant reflects the subjective confidence on this choice (across a sample of representations of the item) based on the participant's understanding. In detail, the second closed-ended

question examines if the activity diagram or state chart diagram is the preferred choice of an expert to conduct test case generation of a software system. This was adapted from Shukla, (2014), and Koriati, (2011).

Additionally, this variable refers to the situations in which subjects have a high awareness about the representation (Kouider, De Gardelle, Sackur, & Dupoux, 2010). In this regard, the third closed-ended question investigates the agreement level of the expert if that activity diagram or state chart diagram increases the degree of awareness of test case generation, as adapted from Shukla (2014) and Kouider et al. (2010).

These three questions reflect the subjective confidence of the participants' understanding of test case generation from the adapted UML diagrams. On the other hand, the first open-ended question examines the degree of certainty (subjective confidence) and the understanding of the participants regarding test case generation from the proposed diagrams and their own understanding of the representation (Aranda et al., 2007; Shukla, 2014). The fourth open-ended question examines the subjective confidence of a tester to evaluate the test case generation from activity and state chart diagrams based on the tester's own understanding (Aranda et al., 2007) and referring to the proffered choice (Koriati, 2011), as adapted by Hadar and Hazzan, (2004) and Aranda et al., (2007). Therefore, the three (3) closed-ended questions in Section A2 for Table 3.1 match the second two (2) open-ended questions for Table 3.2 in measuring the subjective confidence of the examined diagrams with regard to test case generation through asking the degree of confidence, awareness, and preferences.

Table 3.1

The Closed-Ended Questions to Evaluate the Comprehensibility of UML Diagrams
with regard to test case generation

A1) Evaluate the Perceived Difficulty Variable of UML Activity Diagram/ State chart Diagram	Please tick (✓)				
	Very Difficult	Difficult	Neither Agree Nor Disagree	Easy	Very Easy
1) How difficult is it to learn the test case generation from activity diagram / state chart diagram? Source: (Siau & Cao, 2001) ; (Ribiero & Yarnal, 2010)					
2) How difficult are the required tasks of test case generation from activity diagram / state chart diagram? Source: (Razali et al., 2007); (Ribiero & Yarnal, 2010)					
3) How difficult is it to achieve different testing coverage when you are generating test cases from activity diagram / state chart diagram? Source: (Felderer & Herrmann, 2015); (Ribiero & Yarnal, 2010)					
A2) Evaluate the Subjective Confidence Variable of UML Activity Diagram / State chart Diagram	Please tick (✓)				
	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
1) If you are in a task of explaining the test case generation from UML diagrams to others, do you agree that you will be more confident explaining test case generation from activity diagram / state chart diagram? Source:(Shukla, 2014); (Aranda et al., 2007)					
2) Do you agree that activity diagram / state chart diagram is your preferred choice of test case generation of a software system? Source: (Shukla, 2014); (Koriat, 2011)					
3) Do you agree that activity diagram / state chart diagram increases your degree of awareness of test case generation? Source: (Shukla, 2014); (Kouider et al., 2010)					

Table 3.2

The Open-Ended Questions to Evaluate the Comprehensibility of UML Diagrams with regard to test case generation

Open-Ended Questions to Evaluate the Comprehensibility of Activity and State Chart Diagrams	Variables
<p>1) Which UML diagram (activity diagram or state chart diagram) do you think is more difficult for defining test case generation steps like input data and expected results?</p> <p>Please explain your response in detail.</p> <p>Source: (Razali et al., 2007); (Felderer & Herrmann, 2015); (Ribiero & Yarnal, 2010)</p>	Perceived difficulty
<p>2) If you have test cases that are generated from both activity and state chart diagrams for the same system, how difficult is it to determine the origin of the generated test cases? Which diagram do the test cases belong to: activity or state chart diagram?</p> <p>Please explain your response in detail.</p> <p>Source: (Razali et al., 2007); (Ribiero & Yarnal, 2010)</p>	Perceived difficulty
<p>3) Which UML diagram (activity diagram or state chart diagram) increases your certainty of the generated test cases?</p> <p>Please explain your response in detail.</p> <p>Source: (Shukla, 2014) ; (Aranda et al., 2007)</p>	The subjective confidence
<p>4) Based on your preference and your own understanding, please evaluate the comprehensibility of activity and state chart diagrams in generating test cases?</p> <p>Please explain your response in detail, in terms of comprehensibility aspect.</p> <p>Source: (Hadar & Hazzan, 2004) ; (Aranda et al., 2007)</p>	The subjective confidence

Finally, to validate the instrument, a pre-test technique could be used, in which the questions must be shown to a number of evaluators to check factors related to the construct of writing strategies, and those related to the research instrument and reliability check method (Alderson & Banerjee, 1996). Therefore, after making amendments based on the evaluators' comments, the adapted questions were verified

by three evaluators (one evaluator from the software engineering and software testing domain, one evaluator from the information system domain and one from the English language domain). The interview sessions were conducted only after the instruments had been updated and verified by the three evaluators.

3.2.2.2 Generating Manual Test Cases from Activity and State chart diagrams

This phase contains the following steps: the generation of test cases through both activity diagram and state chart diagrams separately. The test cases were used as a case study during the interviews and discussion session to evaluate the comprehensibility of the activity and state chart diagrams. However, the test cases were generated from adapted diagram of activity and state chart diagrams of Gumball machine that was prepared by Felderer and Herrmann, (2015).

The Gumball machine design is considered as a simple case because of the fewer number of activities for activity diagram and fewer number of states for state chart diagram. Adapting simple case study avoids reflecting the complexity of any difficult case study.

Furthermore, the approach of generating test cases from activity diagram of Nayak and Samanta (2011) was used whereas for state chart diagram, the approach by Salman and Hashim (2014) was applied. As a justification, to evaluate the comprehensibility of the two UML diagrams with the aspect of test case generation, this study intended to use the same process of generating the test cases to ensure the same evaluation framework. The second justification is because of the inherent feature of intermediate graphs-based test case generation with the activity and state chart diagrams which is considered as the most widely used approach for that purpose (Shirole & Kumar, 2013). In this regard,

the two adopted approaches of test cases for this study utilise the same main steps of generating test cases from activity and state chart diagrams which are:

- 1) Creating the activity or state chart diagrams.
- 2) Deriving the activity graph from the activity diagram and deriving the state chart graph from the state chart diagram.
- 3) Generating the test cases from the activity graph and state chart graph by following all paths coverage.

i) Generating Test Case from Activity Diagram

The activity as shown in the Figure 3.2 expresses the Gumball machine design starting from the first point which is inserting the coin until the final point which is ejecting the gumball or rejecting the coin.

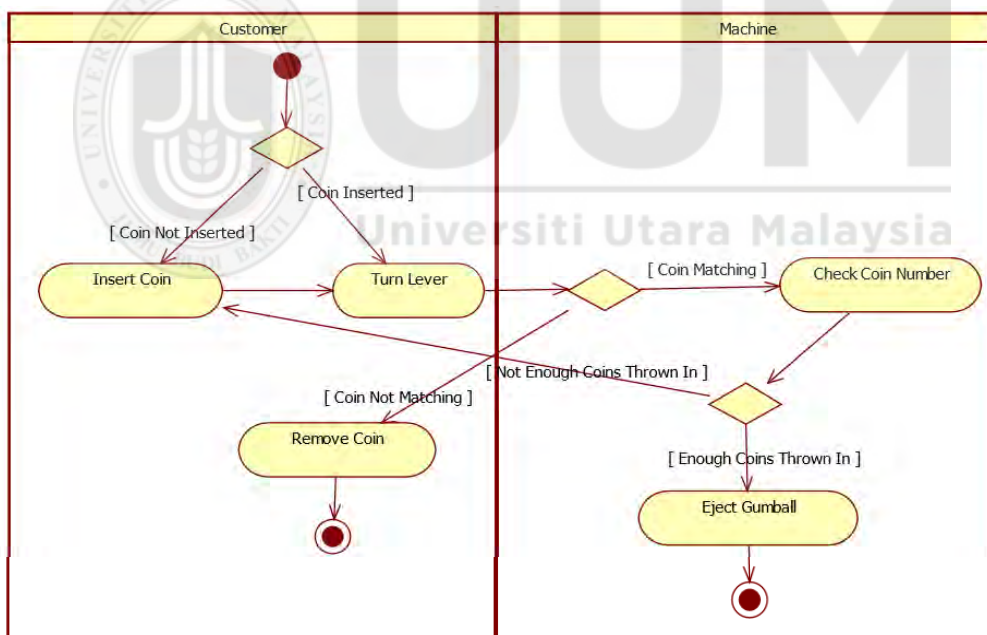


Figure 3.2: Gumball Machine Described as UML Activity Diagram

To generate the test cases from activity diagram, the process of generation are adapted from Nayak and Samanta (2011). They generated test cases from activity diagram based on three main steps:

- 1) Creating the activity diagram for the representation case.
- 2) Deriving the activity graph from the activity diagram using GA (Graph of activity diagram).
- 3) Generating the test cases from the activity graph by following all paths coverage.

The activity diagram of the adopted case study has been adapted from Felderer and Herrmann (2015) to ensure the verification of the diagram. Secondly, the transformation of the activity diagram into activity graph as shown in Figure 3.3 expresses the conversion of each element of activity diagram into a node of the graph.

The elements of the activity diagram in this case study are:

- 1) Initial node: Node with no incoming edge.
- 2) Flow final node: Node with no out coming edge.
- 3) Decision node: Node with one incoming edge and outgoing edges.
- 4) Guard condition node: Node that is associated with condition string. Its parent node is the decision node.

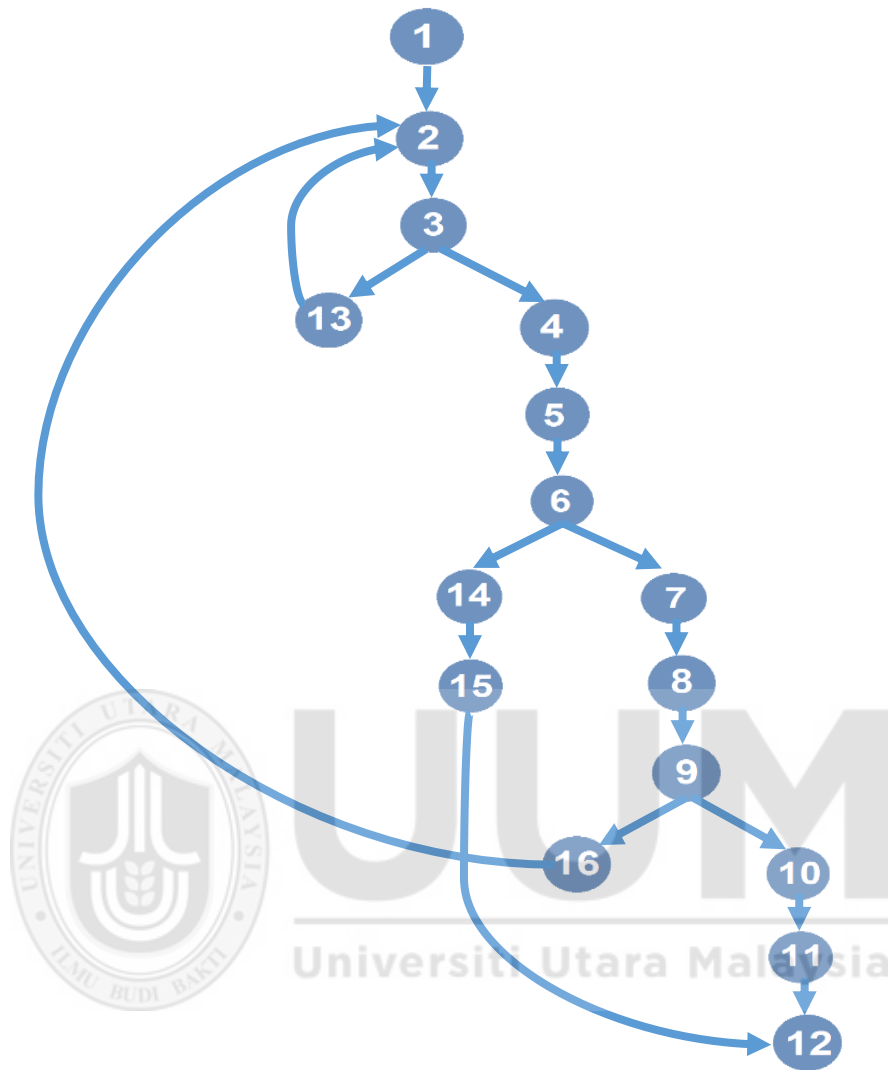


Figure 3.3: Activity Graph Obtained from the Activity Diagram of Gumball Machine

The nodes of activity graph that is shown in Figure 3.3 are stored with their details in Table 3.3. This table is called Node Description Table (NDT).

Table 3.3:
NDT for Activity Graph

Node Index	Activity diagram components
1	Initial state
2	Insert coin
3	Condition
4	Coin inserted
5	Turn lever
6	Condition
7	Coin matching
8	Check coins' number
9	Condition
10	Enough coins thrown in
11	Eject gumball
12	Final state
13	Coin not inserted
14	Coin not matching
15	Remove coin
16	Not enough coins thrown in

The activity path (P) is a path in an activity graph that is considered as the conduct relations between the activities (Linzhang et al., 2004). The paths (Ps) are used to write down the test cases based on the sequence of nodes in the activity graph as the following:

P1: 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 10 → 11 → 12

P2: 1 → 2 → 3 → 13 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 10
11 → 12

P3: 1 → 2 → 3 → 4 → 5 → 6 → 14 → 15 → 12

P4: 1 → 2 → 3 → 13 → 2 → 3 → 4 → 5 → 6 → 14 → 15 → 12

P5: 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 16 → 2 → 3 → 4
5 → 6 → 7 → 8 → 9 → 10 → 11 → 12

P6: 1 → 2 → 3 → 13 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 16
2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 10 → 11 → 12

P7: 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 16 → 2 → 3 → 4
5 → 6 → 14 → 15 → 12

P8: 1 → 2 → 3 → 13 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 16
2 → 3 → 4 → 5 → 6 → 14 → 15 → 12

The test cases from activity diagram are listed in Table 3.4 based on the sequence of activities, the sequence of branches as input data and the expected result for each case.

Table 3.4:
Test Cases from Activity Graph

T.C No	Sequence of Activities	Sequence of Branches	Expected Result
1	Insert Coin, Turn Lever, Check Coin, Eject Gumball	Coin Inserted, Coin Matching, Enough Coin Thrown in	Eject Gumball
2	Insert Coin, Turn Lever, Check Coin, Eject Gumball	Coin Not inserted, Coin Inserted, Coin Matching, Enough Coin Thrown in	Eject Gumball
3	Insert Coin, Turn Lever, Remove Coin	Coin Inserted, Coin Not Matching	Remove Coin
4	Insert Coin, Turn Lever, Remove Coin	Coin Not inserted, Coin Inserted, Coin Not Matching	Remove Coin
5	Insert Coin, Turn Lever, Check Coin, Eject Gumball	Coin Inserted, Coin Matching, Enough Coin Thrown in	Eject Gumball
6	Insert Coin, Turn Lever, Check Coin, Eject Gumball	Coin Not Inserted, Coin Inserted, Coin Matching, Enough Coin Thrown in	Eject Gumball
7	Insert Coin, Turn Lever, Check Coin, Remove Coin	Coin Inserted, Coin Matching, Not Enough Coin Thrown in, Coin Not Matching,	Remove Coin
8	Insert Coin, Turn Lever, Check Coin, Remove Coin	Coin Not Inserted, Coin Inserted, Coin Matching, Not Enough Coin Thrown in, Coin Not Matching,	Remove Coin

Table 3.4 explains the test cases with all possible cases in order to cover the whole test cases for the adopted case study. Moreover, this table includes eight (8) test cases, four (4) of the test cases are to reflect the ejection of Gumball. On the other hand, the rest (4) test cases end with removing the coin. Finally, the generated test cases from the activity diagram was evaluated by a specialist tester from UML diagrams. The expert scrutinized the test cases and approved them.

ii) Generating Test Case from State Chart Diagram

The state chart diagram as shown in the Figure 3.4 expresses the Gumball machine design starting from the first point which is inserting the coin until the final point which is ejecting the gumball or rejecting the coin.

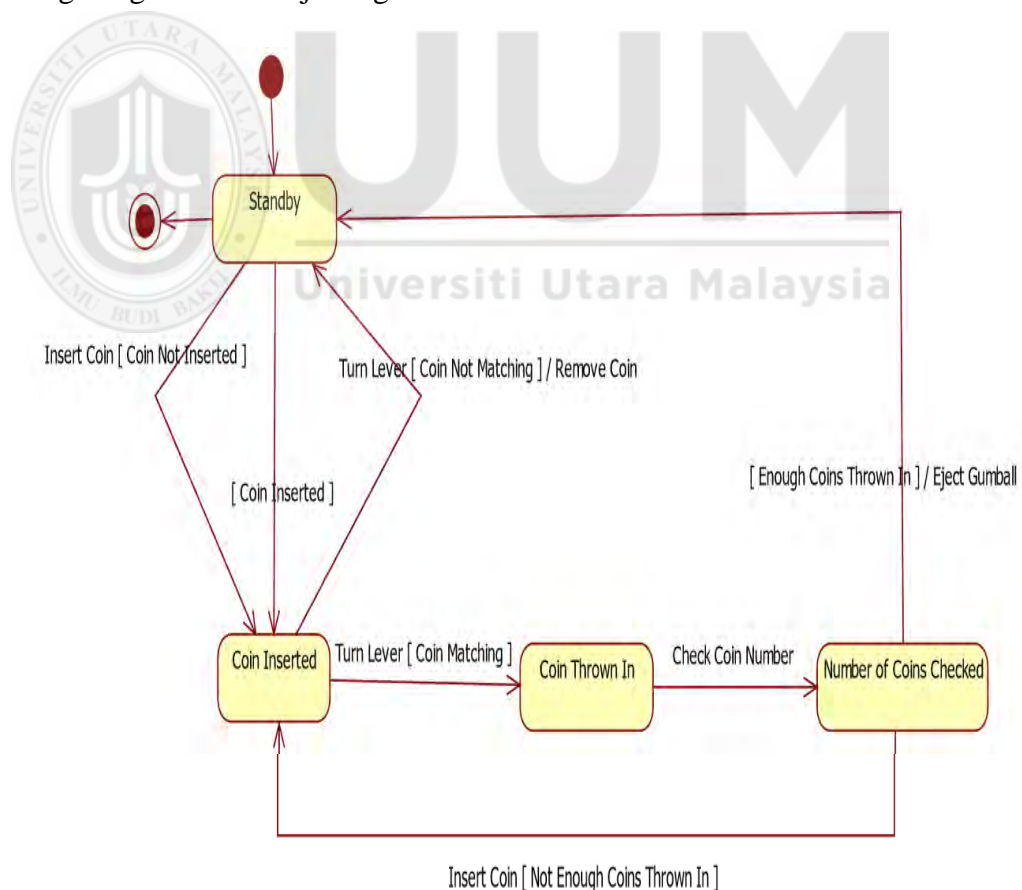


Figure 3.4: Gumball Machine Described as UML State Chart Diagram

To generate the test cases from state chart diagram, the process of generation are adapted from Salman and Hashim (2014). They generated test cases from activity diagram base on three main steps which are:

- 1) Creating the state chart diagram for the representation case.
- 2) Deriving the activity graph from the activity diagram.
- 3) Generating the test cases from the state chart graph by following all paths coverage.

The state chart diagram of the adopted case study was adapted from Felderer and Herrmann (2015) to ensure the verification of the diagram. Secondly, the transformation of the state chart diagram into state chart graph as shown in Figure 3.5 expresses the conversion of each element of state chart diagram into a node of the graph.

The elements of state chart diagram in this case study are:

- 1) Initial node: Node with no incoming edge.
- 2) Flow final node: Node with no out coming edge.
- 3) Decision node: Node with one incoming edge and outgoing edges.
- 4) Guard condition node: Node that is associated with condition string. Its parent node is the decision node.

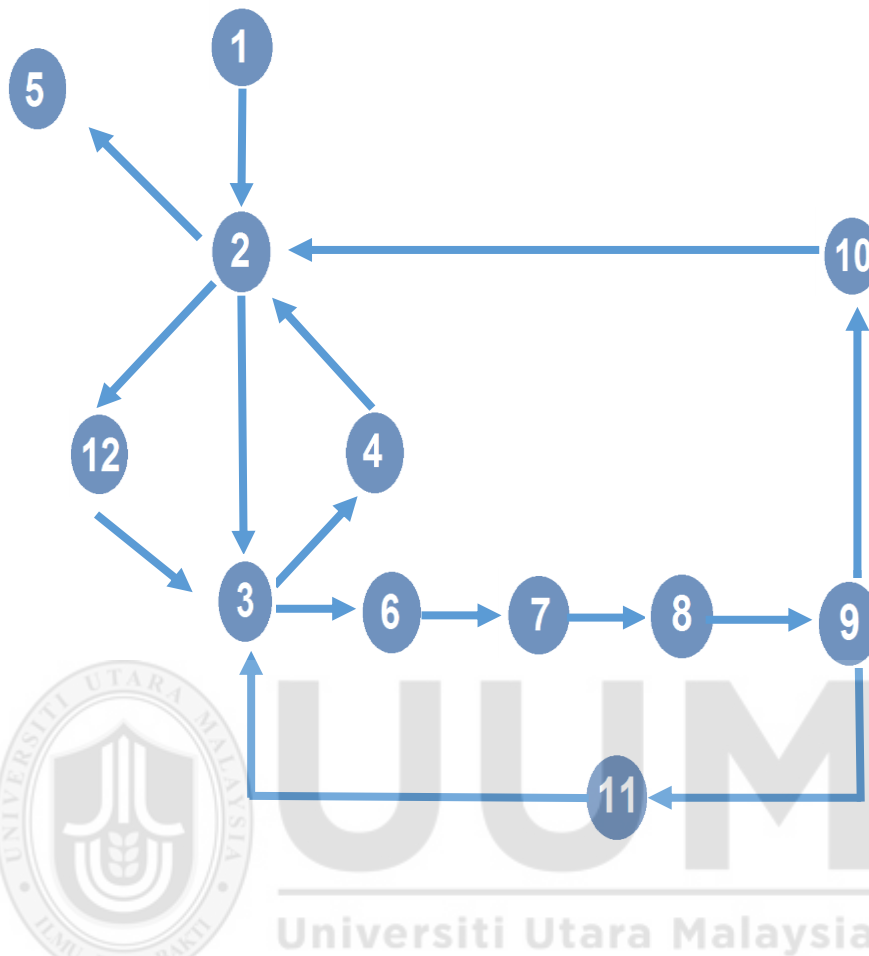


Figure 3.5: State Chart Graph Obtained from State Chart Diagram of Gumball Machine

The nodes of state chart graph that is shown in Figure 3.5 are stored with their details in Table 3.5. This table is called Node Description Table (NDT).

Table 3.5
NDT or State Chart Graph

Node Index	state chart diagram components
1	Initial state
2	Standby state (insert coin)
3	Coin inserted transition
4	Turn lever/ coin not matching [remove coin]
5	Final state
6	Turn lever [coin matching]
7	Coin thrown in state
8	Check coins' number
9	Number of coins Checked state
10	Enough coins thrown in
11	Not Enough coins thrown in [insert coin]
12	Coin not inserted [insert coin]

The state path (P) is a path in a state chart graph that is considered as the transitions to match relations between the states and transitions (Linzhang et al., 2004). The paths (Ps) are used to write down the test case based on the sequence of nodes in the state chart graph as the following:

P1: 1 → 2 → 3 → 4 → 5

P2: 1 → 2 → 3 → 6 → 7 → 8 → 9 → 10 → 5

P3: 1 → 2 → 3 → 6 → 7 → 8 → 9 → 11 → 3 → 6 → 7 → 8 → 9 →
10 → 5

P4: 1 → 2 → 3 → 6 → 7 → 8 → 9 → 11 → 3 → 4 → 5

P5: 1 → 2 → 12 → 3 → 4 → 5

P6: 1 → 2 → 12 → 6 → 7 → 8 → 9 → 10 → 5

P7: 1 → 2 → 12 → 6 → 7 → 8 → 9 → 11 → 3 → 6 → 7 → 8 → 9 → 10 → 5

P8: 1 → 2 → 3 → 12 → 7 → 8 → 9 → 11 → 3 → 4 → 5

The test cases from state chart diagram are listed in Table 3.6 based on the sequence of activities, the sequence of branches as input data and the expected result for each case.

Table 3.6

Test Cases from State Chart Graph

T.C No	Sequence of States	Sequence of Transitions	Expected Result
1	Initial State, Stand by, Coin Inserted, Final State	Coin inserted, Turn lever/ coin not matching [remove coin]	Remove Coin
2	Initial State, Stand by, Coin Inserted, Coin Thrown in, Number of Coins Checked, Final State	Coin inserted, Turn lever [coin matching], Check coin number, Enough coins thrown in [Eject Gumball]	Eject Gumball
3	Initial State, Stand by, Coin Inserted, Coin Thrown in, Number of Coins Checked, Final State	Coin inserted, Turn lever [coin matching], Check coin number, Not Enough coins thrown in, Enough coins thrown in [Eject Gumball]	Eject Gumball
4	Initial State, Stand by, Coin Inserted, Coin Thrown in, Number of Coins Checked, Final State	Coin inserted, Turn lever [coin matching], Check coin number, Not enough coins thrown in, Turn lever/ coin not matching [remove coin]	Remove Coin
5	Initial State, Stand by, Coin Inserted, Final State	Coin not inserted, Coin inserted, Turn lever/ coin not matching [remove coin]	Remove Coin
6	Initial State, Stand by, Coin Inserted, Coin Thrown in, Number of Coins Checked, Final State	Coin not inserted, Coin inserted, Turn lever [coin matching], Check coin Number, Enough coins thrown in [Eject Gumball]	Eject Gumball
7	Initial State, Stand by, Coin Inserted, Coin Thrown in, Number of Coins Checked, Final State	Coin not inserted, Coin inserted, Turn lever [coin matching], Check coin number, Not enough coins thrown in, Enough coins thrown in [Eject Gumball]	Eject Gumball
8	Initial State, Stand by, Coin Inserted, Coin Thrown in, Number of Coins Checked, Final State	Coin not inserted, Coin inserted, Turn lever [coin matching], Check coin number, Not Enough coins thrown in, Turn lever/ coin not matching [remove coin]	Remove Coin

Table 3.6 explains the test cases with all possible cases in order to cover the whole test cases for the adopted case study. Moreover, this table includes eight (8) test cases, four (4) of the test cases are supposed to reflect the ejection of Gumball. On the other hand, the rest (4) test cases end with rejecting the coin. Finally, the generated test cases from activity diagram was evaluated by a specialist tester from UML diagrams. The expert scrutinized the test cases and approved them.

3.2.2.3 Planning the One-to-One Interview

As mentioned earlier in Chapter 1, the second objective of this study is to collect the experts' responses in order to evaluate the UML activity and state chart diagrams based on the comprehensibility regarding the test case generation. An expert review was conducted to get valuable data. A thorough planning is needed to effectively implement the interview. The interview planning involved four (4) activities: defining the objectives, identifying the participants, scheduling the meeting, reminding the participants and finally, preparing the materials for the interview. These activities are further elaborated in the subsequent subsections.

Moreover, Chism, Douglas, and Hilson Jr. (2008) stated that "*interview encounters between a researcher and a respondent in which an individual is asked a series of questions relevant to the subject of the research*". The interview is one of the most powerful and widely used tools of the qualitative researcher (Willig & Stainton-Rogers, 2007). Further, during the interview, the interviewer also has better control over the types of information received, because the interviewer can ask specific questions to elicit specific information (Creswell, 2012).

The data were collected through a group of questions, as shown in Tables 3.1 and 3.2. The resources of the questions are also listed in these tables. Furthermore, the interview contained questions in relation to the respondents' experience in software testing with UML diagrams.

i) Defining the Objectives of the Interview

Basically, the objective of the interview is to evaluate the comprehensibility of activity and state chart diagrams with regard to testing case generation. In more detail, this study aims to advance the evaluation study of Felderer and Herrmann (2015) and addresses the limitation that is mentioned in their study; i.e., the use of inexperienced participants to evaluate the comprehensibility of the UML diagrams. Moreover, Felderer and Herrmann (2015) asserted that the use of experts could give more accurate evaluation data for these two diagrams with regard to manual test case generation.

Accordingly, through the interview, the experts evaluated the comprehensibility of activity and state chart diagrams. The comprehensibility with regard to test case generation was covered through answering the adapted evaluation questions about the perceived difficulty and the subjective confidence as measurement variables.

ii) Identifying the Participants

The participants of the interview were chosen by using purposive sampling (Creswell, 2012). They were chosen based on their experience in software testing and UML diagrams. Eight invitations were sent out but only five accepted the invitation and completed the interview. This number, according to Shneiderman, (1992) and Creswell (2012) is substantial. Board (2013) asserted that the expert in software testing is “... *a person with the special skills and knowledge representing mastery of a particular*

testing subject. Being an expert means possessing and displaying special skills and knowledge derived from training and experience”.

Invitations to become experts for the study were sent through e-mails. The related documents were then sent to the experts who agreed to participate in this study. Feedbacks were provided by the experts during the interview.

iii) Meeting Scheduling and Reminding

The interviews were scheduled by the experts. The meeting place was identified by the expert for each interview. They were reminded about the meeting and their attendance was confirmed one day before conducting the interviews. This was to ensure that they would not miss the session as well as to make them feel important in attending the session (Creswell, 2012).

iv) Preparation of the Interview Guide and Materials

Prior to conducting the interview, the materials that were used during the interview session were prepared; namely the presentation slides and documents for the participants. The interview was started with a general topic; the introduction to the study followed by showing the test case generation from the UML activity and state chart diagrams. Next in the interview agenda was to obtain the evaluation data represented in the evaluation of the UML diagrams based on the proposed criterion.

3.2.2.4 Conducting the Interview

The interviews were conducted on the scheduled day and time. Five participants turned up to attend the interviews. The experts were provided with the materials that were needed for the interview. The participants were also reminded that the data gathered

from them will be confidential and will only be used strictly for the study purposes. They were briefed about the objectives of the evaluation. They were encouraged to express their experience and points of view freely and spontaneously. After the discussion of each listed question, the experts filled the form containing the questions. The answers were submitted by the experts to the researcher at the end of each interview.

3.2.2.5 Profile of Experts

This section describes the demographic profiles of the respondents who participated in the study. Prior to reporting the main findings of the survey, the demographic profiles of the respondents must be identified. The demographic profiles include participants' position, experience, field of expertise, years of experience in software testing and UML diagrams.

Each expert represents different field of expertise: The first expert is a software analyst, the second and third are experts in software development and the fourth is from the field of web engineering. The fifth expert is a lecturer in software testing, who has been involved in the field for more than 10 years. Their involvement in software testing varies from 2 to 8 years, and their experience in UML diagrams varies from less than 1 year to 15 years.

Table 3.7
Experts' Background

No	Gender	Company/ Institution	Field of expertise	Experience Years in Software Testing	Experience Years in UML Diagrams
1	Female	Uniutama Solution Sdn Bhd	Software analyst/ programmer	6	6
2	Male	MST	Software development	8	6
3	Male	SOC.UUM	Commercial Software development	3	Less than 1 year
4	Female	SOC.UUM	Web engineer	2	15
5	Female	UUM	1) S.W tester 2) Lecturer of S. testing	10	More than 10 Years

3.2.2.6 Data Analysis

In this subsection, the second objective of this study was achieved; by documenting the collected data of the interviews based on the adapted questions. This collected data helped to get the final evaluation of the comprehension level of activity and state chart diagrams.

Upon completion of data collection, the filled question forms and the recorded data were transformed into textual data. Eventually, all data were entered into NVivo10 for storing, analyzing, sorting, and representing or visualizing the data.

The QSR NVivo software program combines efficient management of non-numerical, unstructured data with powerful processes of indexing, searching, and theorizing. Designed for researchers making sense of complex data, NVivo offers a complete toolkit for rapid coding, thorough exploration, and rigorous management and analysis. Especially valuable is the ability of the program to create text data matrixes for comparisons. It also provides visual mapping categories identified in the analysis (Creswell, 2012; 2013).

i) QSR-NVivo

The advancement in computer technology has led to the development of a range of software packages that assist in analyzing qualitative data. NVivo is the latest version of the software by QSR International, which helps analyse, manage and shape qualitative data (Creswell, 2013, Lewins & Silver, 2007). It provides security by storing the database and files together in a single file.

NVivo software enables a researcher to use multiple languages. It has a merge function for team research, and enables researchers to easily manipulate and search data (Lewins and Silver, 2007). It can also facilitate the qualitative research process by making all phases of investigations open to public inspection (Constas, 1992). It can be used to support the analysis processes involved in a literature review (Di Gregorio, 2000; Creswell, 2013). Besides, Bazeley and Jackson (2013) found that NVivo is programmed with a high degree of flexibility; allowing visuals, Figures, or tables to be

analyzed (Creswell, 2009). Figure 3.6 highlights the steps for analysis via NVivo software.

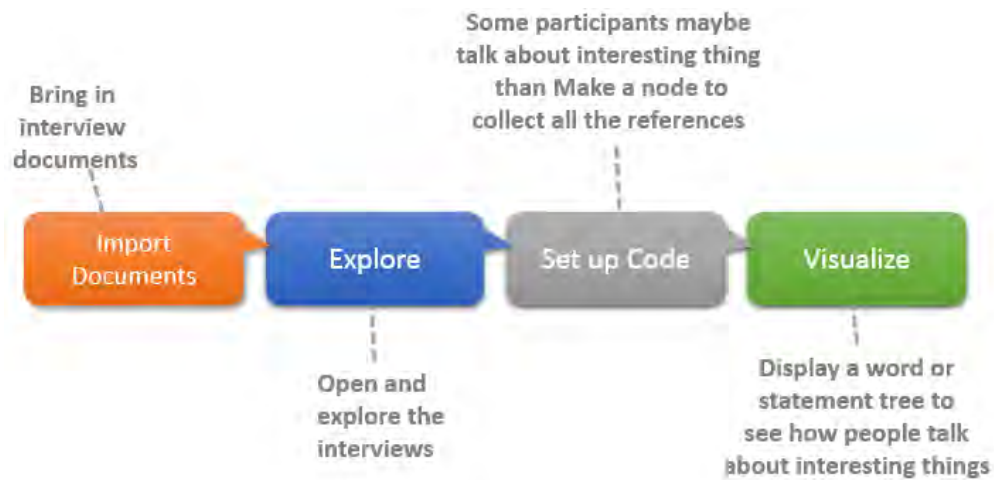


Figure 3.6: NVivo Steps

Based on those aspects of NVivo mentioned in the preceding paragraph, this study utilises NVivo 10 for analyzing the data collected. This is also influenced by the fact that NVivo can facilitate the qualitative research process by making all investigation phases open to public inspection (Sinkovics & Alfoldi, 2012).

3.3 Summary

This chapter provided the methodology and research design of this study. In detail, this chapter presented the steps that were used to evaluate the comprehensibility of test case generation from activity diagram and state chart diagram by using qualitative methodology. Analyzing the contents of past studies highlighted the main issues of this study. The test cases are generated from the abovementioned diagrams. One-to-one interview was the research instrument used to collect the evaluation data from the experts. Furthermore, the NVivo data analysis tool was presented in this chapter and it was used in chapter 4 to analyze the collected data.

CHAPTER FOUR

FINDINGS

4.1 Introduction

This chapter presents the results on the evaluation of the comprehension of UML activity and state chart diagrams with regard to test case generation which were gathered through the one-to-one interview. Subsequently, data analysis was carried out to identify the strengths and weaknesses of the studied comprehension variables. NVivo was used to analyse the data. The report on the sections that follow is based on the data provided by the content analysis of past studies, and further data collected from the experts during the interviews.

This chapter starts with the experts' evaluation data that was collected through the interviews in Section 4.3. The visible data and charts are presented in Section 4.4. This chapter ends with a summary in Section 4.4.

4.2 The Significant Findings from the Interview with the Experts

This section describes the findings from the conducted interview. During the interview sessions, the researcher briefly explained the background and the objectives of the study to the participants. They were then introduced to the comprehension of UML diagrams of software testing based on two examined variables (the perceived difficulty and the subjective confidence). The main instrument used for this review are the closed-ended and the open-ended questions (see appendix A) that were adapted from past research in the field of evaluating the comprehensibility as described in Section 3.2.2.1. Next, the experts were required to answer all the questions in the evaluation instrument. After

completing the interview sessions with the expert, the data obtained from the interviews were explored (see appendix B) and then analyzed via NVivo and represented in charts.

The five interviewed experts agreed on the importance of activity and state chart diagrams with regard to test case generation, even though the level of comprehensibility among them differs as shown in Figures 4.1, 4.2, 4.3 and 4.4. The responses to the evaluation questions enquired on the comprehensibility of activity and state chart diagrams are presented in the forthcoming sections.

4.2.1 Evaluation Data from Open-Ended Questions

This section illustrates the evaluation data that were collected via the open-ended questions from the interviews with the experts. Moreover, the representations of the findings are shown in Figures 4.1, 4.2, 4.3 and 4.4. Qualitative studies often display their findings visually (Miles & Huberman, 1994) by using figures or pictures that augment the discussion.

4.2.1.1 Perceived Difficulty of the UML Diagrams with regard to test case generation

The first open-ended question queries on the assessment of the perceived difficulty of the diagrams with regard to test case generation. This question examines participants' understandability of the information that they acquire from the practical execution of the representation (Ribiero & Yarnal, 2010).

With regard to the first question of the interview, Expert 1 stated that the state chart diagram is more difficult in determining input data and expected results. In addition,

Expert 1 also explained that the activity diagram is easier in determining test case generation because of the easy process of activity diagram to generate the test cases. Expert 1 also added that the second reason is because of the easiness of determining the valid and invalid input and output data for each test case. Furthermore, Expert 1 added that the consideration of the transitions between the system and the user (business aspect) aspects increase the degree of easiness of listing the activities and the branches of activity diagram, as well as the expected result for each test case. For this, Eriksson and Penker (2000) confirmed that activity diagram is the most important UML diagram for business aspect.

Expert 2 declared that the state chart diagram is more difficult, and this might lead to the difficulty of determining the steps of test case generation. In addition, Expert 2 mentioned that the activity diagram is easier in determining test case generation steps because of the clearness of the process of activity diagram in generating the test cases. Expert 2 also indicated the ability of the activity diagram to represent more information on the activities and to provide clear decision points together with the possibilities to be used as a reference to test case generation steps. In this regard, Swain, Mohapatra, and Mall (2010) asserted that *“activity diagram presents the concept at a higher abstraction level of the system”*.

Contradicting the argument mentioned by Expert 1 and Expert 2, Expert 3 asserted that state chart diagram is very simple when compared to the redundancy of information captured by the activity diagram. Expert 3 clarified that activity diagram is more difficult in determining the steps of test case generation since the activity diagram is dealing with a wider process that includes business and system aspects. The second reason for the

difficulty of activity diagram is because of the variety of components of activity, for example, initial node, decision node, guard condition node and flow final node. This is in agreement with Yang, Yu, Sun and Qian (2010) claim that a tester needs to deal with each node to generate test cases.

Expert 4 shared the same perspective as Expert 3 whereby she stated that the state chart diagram is easier in determining the main building blocks of the test case generation (e.g. inputs and the expected results). Expert 4 asserted that the minimization of steps of state chart diagram makes it easier in listing the steps of generating test cases. Furthermore, Expert 4 pointed out that activity diagram has many condition statements that cause loops paths which creates difficulty in writing the test case steps. A large number of components for activity diagram was also mentioned by Mingsong, Xiaokang, and Xuandong (2006).

The last expert (Expert 5) stated that the activity diagram is easier in determining test case generation. In addition, through the interview session, Expert 5 also explained that the state chart diagram is more difficult in determining input data and expected results. Expert 5 justified the easiness of activity diagram to the suitability of this diagram with black box testing that makes the determining of test case generation easier. Having a similar view is Linzhang et al. (2004) who stated that activity diagram is suitable for black box testing. Figure 4.1, illustrated the outcome of the first interview question.

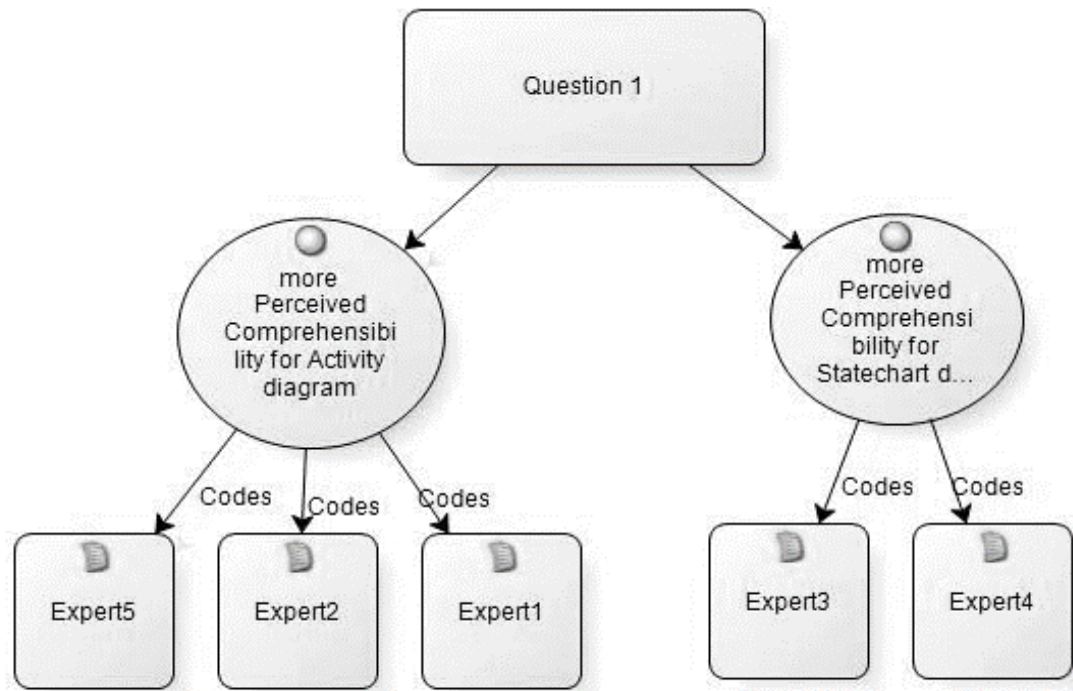


Figure 4.1: NVivo Result of the Perceived Difficulty for Determining the Steps of Test Case Generation from Activity and State Chart UML Diagram.

The second question tries to examine the ability of these two UML diagrams to determine the origin of the generated test cases to their UML diagrams from the sequence of activities and states, as well as the transitions and branches that were used to generate the test cases. The easiness of reading a test case and matching it to its diagram reflects the easiness of the diagram in the same manner it refers to the obtained information for the representation that was mentioned by Felderer and Herrmann (2015). On the other hand, the difficulty reflects the perceived difficulty of the diagram.

Expert 1 stated that the activity diagram is easier in generating the test cases and the generated test case is easier in determining its origin diagram. Expert 1 also mentioned that this is due to the easiness of the process of generating the test case from the activity diagram. Furthermore, this expert indicated that it is quite difficult for the state chart

diagram to determine its test case origin because of the difficulty in the process of generating the test cases from this diagram. In this regard, Kundu and Samanta, (2009) stated that the main reasons for using activity diagram for model-based testing are to make it easier for software testers to better understand the system and find test information.

In the same vein, Expert 2 stated that it would be easier to determine the origin of generated test cases from activity diagram to their diagram because of the easiness of determining the steps of generating the test case from activity diagram. Also, Expert 2 highlighted that the state chart diagram is quite difficult in this regard. In this aspect, Shukla and Chandel (2012) explained that the benefits of using activity diagram in model-based testing are to help software testers to better understand and to easily find test information.

Expert 3 also supported the claimed stated by the other experts (Expert 1 and Expert 2) when he stated that it would be easier to determine the generated test cases from activity diagram to their origin diagram, since this test case reflects more information from the view of business aspect, and that makes the test case easy to read and easy to match to its diagram. In this regard, Kingston and Macintosh (2000) claimed that activity diagram represents how business processes are performed and where communication occurs between processes, and this multi aspect of UML diagrams are required to improve the comprehensibility. Additionally, Expert 3 asserted that the test case that is generated from state chart diagram reflects the developer's perspective, and this makes the test case to become more complex to determine its origin diagram. In this regard, Swain et al. (2010) explained that the state chart diagram is difficult to be implemented in code

and the state chart-based faults are difficult to be detected from the software code. Consequently, it is difficult for state chart to test based on code.

Expert 4 indicated that the simplicity of tracking the events and states of state chart diagram makes test cases easier to determine their origin from the diagram used. On the other hand, this expert declared that testers must pay more attention to the generated test cases from activity diagram since they take into account the perspective of the user (business) within the perspective of the system. In addition, Expert 4 explained that the higher the number of activities and condition of activity diagram, the harder it is to generate test cases. Therefore, the test cases from activity diagram are difficult to match with their origin diagram. In this respect, Eriksson and Penker (2000) explained that activity diagram is the most important UML diagram for doing business aspect (user aspect) in addition to the system aspect. Moreover, a large number of components for activity diagram are also mentioned by Booch (2005).

Expert 5 stated that it would be harder to determine the origin diagram for the test cases that are generated from activity diagram. Expert 5 explained that activity diagram shows many components are involved in generating test cases. Furthermore, the expert showed that the minimization process of generating test cases from state chart diagram makes it quite easy to determine its test case. The minimization of test case generation refers to using less number of components of state chart diagram, as indicated by Booch (2005). Figure 4.2 summarises the outcome through NVivo.

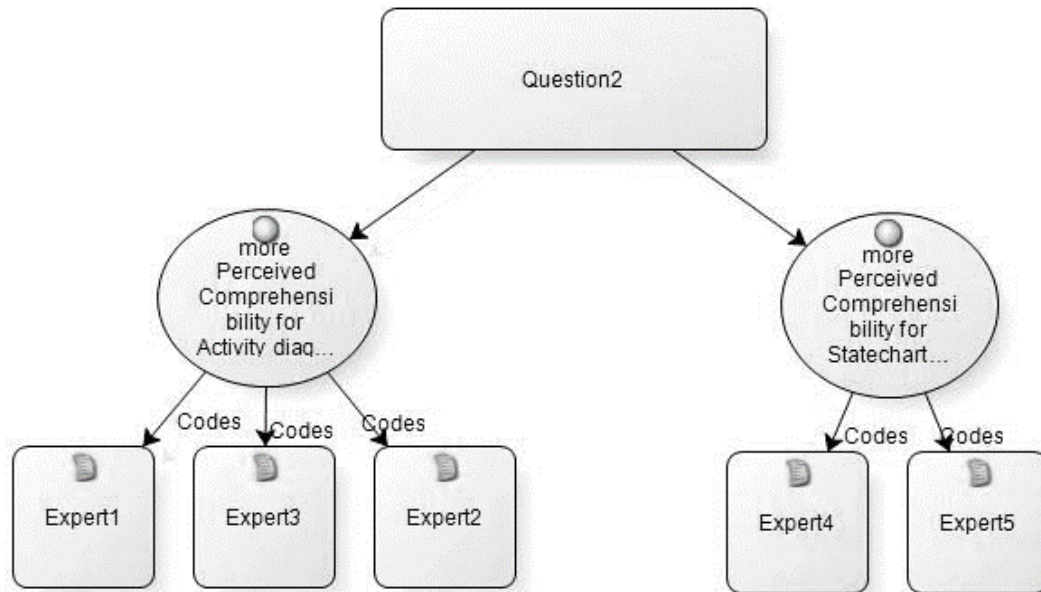


Figure 4.2: NVivo Result of the Perceived Difficulty for Determining the Origin Diagrams for the Generated Test Cases from Activity and State Chart Diagrams.

The conclusion of this section will summarise the experts' responses for the perceived difficulty of the diagrams with regard to test case generation. Based on the responses to the first question, three (3) experts assured that by presenting both system and business aspects, activity diagram is easier in determining the test case generation steps because of the clear vision of activity diagram. On the other hand, two (2) of the experts found that state chart diagram is easier to determine test case generation steps because of the minimization of steps against the wide process of activity diagram.

Based on the responses to the second question, three of the experts stated that activity diagram is easier to trace back to its origin diagram as compared to state chart diagram, while two (2) experts claimed the contrary. The three experts substantiated that it is easier than state chart diagram due to the clear process of generating test cases from activity diagram, which involves generating the activity diagram and transferring it to

activity graph and writing the specification of the test case, as mentioned by Kundu & Samanta, (2009). Another reason is that activity diagram reflects more view of business aspect, thus matching the test case to their origin diagram easier. Relatively, the second question leads to the conclusion that the activity diagram is more comprehensible than state chart diagram.

From the results elaborated above, in measuring comprehensibility between activity and state chart diagram in generating test cases, state chart diagram can be perceived as more difficult when compared to activity diagram. This is because state chart holds a harder process in determining test case generation steps (e.g. data input and expected results are easier when applied on activity diagram). It is also more difficult to determine the origin of the generated test cases.

It is worth mentioning that the results of evaluating the perceived difficulty of the activity and state chart diagrams with regard to manual test case generation matched with what was mentioned by the Felderer and Herrmann (2015), where they claimed that the activity diagram is perceive to be easier than state chart diagrams in the aspect of test case generation.

4.2.1.2 Subjective Confidence of the UML Diagrams with Regard to Test Case Generation

The third evaluation question of this study tests the subjective confidence as a comprehensibility measuring variable. This question enquires about the certainty of the tester towards the generated test cases. The question inspects if there is a difference in the certainty level of the generated test cases from activity diagram and the generated test cases from state chart diagram.

From the interview session, Expert 1 confirmed that the generated test cases from activity diagram have higher certainty when compared to the test cases that were generated from state chart diagram. This expert also indicated that confidence comes from the ability of activity diagram to indicate the control flow of the system (Linzhang et al., 2004), and the right flow of the system leads to the right test case generation.

Moreover, Expert 2 stated that the certainty of the generated test cases depends on the clarity of the diagram. This expert mentioned that activity diagram gives a clear flow of the system. Expert 2 further explained that the generated test cases from activity diagram have higher certainty level than state chart diagram. In this concern, Swain et al., (2010) specified that activity diagram represents concepts at a higher abstraction level of the system.

Expert 3 asserted that the state chart diagram increases the certainty of the generated test cases since it describes the dynamic behaviour of the system when software is executed. Moreover, this expert added that state chart diagram describes how the system changes if an appropriate trigger is applied to the system, thus the correct state chart can be easily converted to code. However, Swain, Mohapatra and Mall (2010) refuted Expert 3 claimed by explaining that state chart diagram is difficult to implement in code.

Expert 4 asserted that the test cases that are generated from state chart diagram achieve higher certainty than the test cases generated from activity diagram. This expert also justified that higher test coverage of state chart diagram is obtained when generating the test cases. However, Swain, Mohapatra, and Mall (2010) argued that “*generating test cases from state chart diagrams only deal with testing a single object and can be used to easily achieve transition and state coverage for any single class*”. Therefore,

the claim by Expert 4 contradicted with the claim by Swain et al (2010). This expert also mentioned that the state chart achieves the goal of testing in a shorter time because of the easiness of generating the test cases from state chart diagram.

Expert 5 confirmed that the generated test cases from activity diagram have higher certainty when compared to test cases generated from state chart diagram. This expert also specified that certainty comes from the ability of activity diagram to achieve high testing coverage. On the other hand, the expert marked high coverage ability of state chart diagram only in terms of unit testing. The higher level of testing coverage for activity diagram was stated by Kim, Kang, Baik, and Ko (2007) and the high coverage ability of state chart diagram was mentioned by Samuel, Mall, & Bothra (2008).

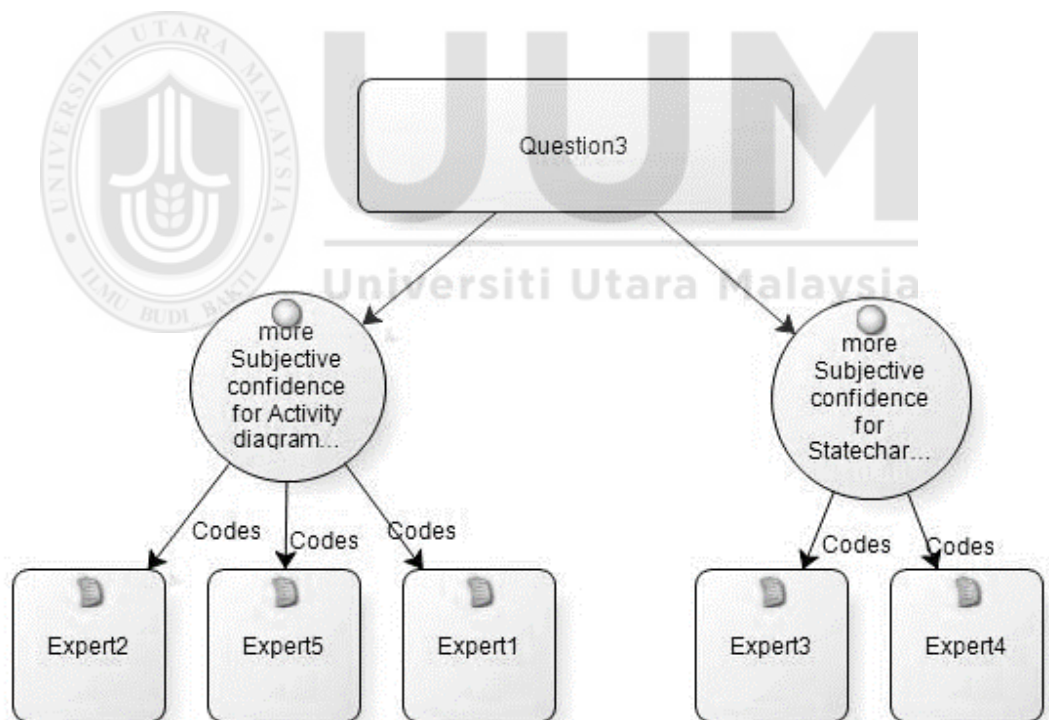


Figure 4.3: NVivo Result of the Experts' Certainty of the Generated Test Cases from Activity and State Chart Diagrams.

The fourth interview question aims to evaluate the subjective preference of the experts to evaluate the comprehensibility (understandability) of the two examined UML diagrams in generating test cases.

The activity diagram easiness of understanding the test case generation is stated by Expert 1. Expert 1 justified that her confidence in choosing activity diagram in generating test cases is due to the fact that this diagram is able to provide higher understandability from business (user aspect) perspective. In addition, Expert 1 asserted that the state chart diagram has less understandability with regard to test case generation even though it is very important to represent programmer's perspective (behaviour of the system) as stated by Harel and Politi (1998).

Expert 2 preferred activity diagram to generate the test cases because of the higher comprehensibility in generating the test cases. Expert 2 specifically mentioned that the higher comprehensibility of activity diagram is due to the simplicity of activity diagram as mentioned by Swain, Mohapatra and Mall (2010).

Expert 3 had also favoured activity diagram in generating test cases. This expert justified his preference by stating that activity diagram possesses the ability of minimizing the number of test cases, similar to the assertion made by Li, Li, He, and Xiong (2013) after using the enhancement algorithms.

Expert 4 selected activity diagram in generating the test cases from activity diagram, disregarding state chart diagram. This expert justified that this preference after taking into consideration the importance of user aspect (business) in generating test cases from activity diagram.

Lastly, Expert 5 preferred activity diagram in generating the test cases from activity diagram more than state chart diagram based on the experts' own opinion. The expert indicated this preference and the higher subjective confidence due to the characteristics of activity diagram which imitates the process flow of the system, similar to what has been mentioned by Eriksson and Penker (2000). Figure 4.4 highlights the NVivo result for question 4.

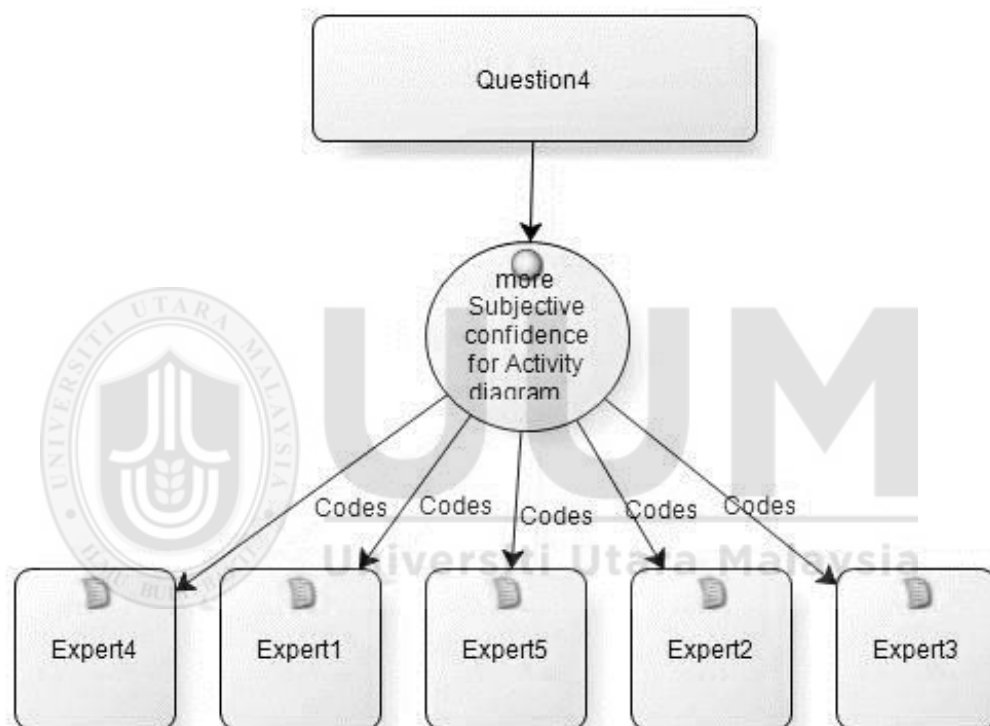


Figure 4.4 NVivo Result of Experts' Evaluation for the Comprehensibility of Activity and State Chart Diagrams in Generating the Test Cases

The conclusion of this sub-section will summarize the experts' responses to the subjective confidence of the diagrams with regard to test case generation. Based on the responses on the first question, three (3) experts assured that activity diagram increases the certainty of the test case generation because of the ability of activity diagram to indicate the control flow of the system that leads to the right test case generation.

Additionally, one expert mentioned that the high test coverage of activity diagram in generating the test cases also increases the certainty level of the generated test case.

On the other hand, two (2) of the experts found that state chart diagram increases the certainty of test case generation because of the higher test coverage of state chart diagram. Moreover, one expert stated that the ability of state chart diagram for describing the dynamic behaviour of the system. He explained how the system changes if an appropriate trigger is applied to the system; thus, the correct state chart diagram can be easily converted to code and increases the certainty on the generated test cases.

Based on the responses on the second question, it is overwhelming to note that all of the five (5) interviewed experts believed that the activity diagram is more comprehensible in overall view of test case generation. The three experts substantiated that the ability of activity diagram of taking the aspect of the business (user aspect) increases the higher confidence of activity diagram with regard to test case generation. Moreover, the higher subjective confidence is also due to the characteristic of the activity diagram that imitates the process flow of the system. Additionally, the minimization ability of activity diagram for a number of test cases is also one of the reasons stated by one the experts. Finally, one expert confirmed that higher subjective preference diagram is provided by the activity diagram because of its higher simplicity.

The inference of the experts, higher subjective confidence (certainty) of the test case generation from activity diagram provides merit to the comprehensibility of the diagram. The state chart diagram, on the other hand, do not seem to be comprehensible.

Felderer and Herrmann (2015) investigated the perceived difficulty comprehension variable but their study did not encompass the subjective confidence comprehension

variable. In this study, it was found that the activity diagram achieved more subjective confidence (certainty) with regard to test case generation than state chart diagram, hence contributing to the body of knowledge regarding comprehensibility of activity and state chart diagrams with regard to test case generation.

4.2.2 Evaluation Data from Closed-Ended Questions

The closed-ended questions were utilised to convey the experts' assessment of the comprehensibility of UML activity and state chart diagrams. The closed-ended questions covered the two comprehensibility variables which are the perceived difficulty and subjective confidence. The closed-ended questions that are shown in Table 4.2 and Table 4.2 are divided into four (4) sections. Sections A1 and B1 include the questions on the perceived difficulty and sections A2 and B2 include questions on the subjective confidence. The results of the experts' responses are presented in Table 4.2 and Table 4.3. Table 4.2 presents the evaluation on perceived difficulty of activity and state chart diagrams while Table 4.3 presents the evaluation on the subjective confidence of activity and state chart diagrams.

4.2.2.1 Perceived Difficulty of the UML Diagrams with Regard to Test Case Generation

Table 4.2

The Experts' Responses to Evaluate the Perceived Difficulty of UML Activity and State Chart Diagrams with regard to test case generation

		Frequency				
A1) Evaluate the Perceived Difficulty		Percent %				
Variable of UML Activity Diagram		Very Difficult	Difficult	Neither	Easy	Very Easy
1) How difficult is it to learn the test case generation from activity diagram?		-	-	2	3	-
		-	-	40%	60%	-
2) How difficult are the required procedures of generating the test case from activity diagram?		-	-	1	4	-
		-	-	20%	80%	-
3-) How difficult is it to achieve higher testing coverage when you are generating test cases from activity diagram?		-	1	1	3	-
		-	20%	20%	60%	-
B1) Evaluate the Perceived Difficulty		Frequency				
Variable of UML State Chart Diagram		Percent %				
		Very Difficult	Difficult	Neither	Easy	Very Easy
1) How difficult is it to learn the test case generation from state chart diagram?		-	3	-	2	-
		-	60%	-	40%	-
2) How difficult are the required procedures of generating the test case from state chart diagram?		1	2	-	2	-
		20%	40%		40%	-
3) How difficult is it to achieve higher testing coverage when you are generating test cases from state chart diagram?		-	1	2	2	-
		-	20%	40%	40%	-

Sections A1 and B1 contain the experts' responses to the perceived difficulty of activity and state chart diagrams with regard to test case generation through answering three (3) questions for each diagram as shown in Table 4.2.

Based on the first question, the experts were asked about the difficulty of learning the test case generation from activity diagram that reflects the perceived difficulty of the diagram. The experts' responses show that 2 experts (40%) responded Neither to the perceived difficulty for activity diagram with regard to test case generation when 3 experts (60%) responded Easy. On the other hand, the perceived difficulty of state chart diagram with regard to test case generation received difficult response from 3 experts (60%) and Easy response from 2 experts (40%). Thus, it can be said that the majority of the experts believe that it will be easier to learn the test case generation from activity diagram.

Based on the second question, one expert (20%) responded Neither to the perceived difficulty of the required procedure of generating the test case of activity diagram with regard to test case generation while 4 experts responded Easy (80%). On the other hand, the perceived difficulty of the required procedure of state chart diagram with regard to test case generation obtained Very Difficult response from 1 expert (20%) and Difficult response from 2 experts (40%) while 2 other experts (40%) responded Easy. This means that the majority of experts find the procedure of test case generation from activity diagram is easier than state chart diagram.

Based on the third question of the perceived difficulty, the result shows that the perceived difficulty of activity diagram in achieving higher testing coverage when performing test case generation received Easy response from three experts (60%) while

one expert responded Neither (20%) and one other expert responded Difficult (20%). On the other hand, the perceived difficulty of state chart diagram in achieving higher testing coverage when performing test case generation obtained Neither response from two experts (40%) and Difficult response from one expert (20%) while two other experts (40%) responded Neither. This means that the achievement of higher testing coverage is easier than activity diagram.

From the data that aforementioned, regarding to the first closed-ended question, 60% from the interviewed experts explained that activity diagram perceived easier while only 40% from the experts stated that for state chart diagram. According to the second closed-ended question, 80% from the interviewed experts explained that activity diagram perceived easier while only 40% from the experts stated that for state chart diagram. Ultimately, based on the third closed-ended question, 60% from the interviewed experts explained that activity diagram perceived easier while only 40% from the experts stated that for state chart diagram.

The conclusion for the measurement of the comprehensibility between activity and state chart diagram in generating test cases include the results of experts' responses to the closed-ended and the open-ended questions. Regarding the first comprehension variable (the perceived difficulty), activity diagram can be perceived as more comprehensible when compared to state chart diagram based on the open-ended questions while the closed-ended questions confirmed these results and the details are illustrated as follows:

- Activity diagram is more difficult to determine test case generation steps.

- The origin diagram of Activity diagram's generated test case is easier to be determined
- It will be easier to learn the test case generation from activity diagram.
- The required procedure of generating test case from activity diagram is easier.
- It will be easier to achieve higher testing coverage when one is generating test cases from state chart diagram.

It is worthwhile to note that the results of evaluating the perceived difficulty of the activity and state chart diagrams matched with what was mentioned by the Felderer and Herrmann (2015), where they claimed that the activity diagram is perceived to be easier than state chart diagrams for the understanding of the aspect of test case generation.

4.2.2.2 Subjective Confidence of the UML Diagrams with regard to test case generation

Section A2 and B2 contain the experts' responses to the subjective confidence of activity and state chart diagrams with regard to test case generation through three (3) questions for each diagram as shown in Table 4.3.

Table 4.3

The Experts' Responses to Evaluate the Subjective Confidence of UML Activity and State Chart Diagrams with regard to test case generation

A2) Evaluate the Subjective Confidence Variable of UML Activity Diagram	Frequency				
	Percent %				
	Strongly Disagree	Disagree	Neither	Agree	Strongly Agree
1) If you are in the task of explaining the generated test case from one of UML diagrams to others, do you agree that you will be more confident explaining test case generation from activity diagram?	1	-	1	3	-
	20%	-	20%	60%	-
2) Do you agree that activity diagram is your preferred choice to generate the test case for a project requirement or design?	-	-	-	5	-
	-	-	-	100%	-
3) Do you agree that generating test case from activity diagram increases your degree of awareness of UML test case generation?	-	-	3	2	-
	-	-	60%	40%	-
B2) Evaluate the Subjective Confidence Variable of UML State Chart Diagram	Frequency				
	Percent %				
	Strongly Disagree	Disagree	Neither	Agree	Strongly Agree
1) If you are in the task of explaining the generated test case from one of UML diagrams to others, do you agree that you will be more confident explaining test case generation from state chart diagram?	-	1	1	3	-
	-	20%	20%	60%	-
2) Do you agree that state chart diagram is your preferred choice to generate the test case for a project requirement or design?	-	1	2	2	-
	-	20%	40%	40%	-
3) Do you agree that generating the test case from state chart diagram increases your degree of awareness of UML test case generation?	-	1	3	1	-
	-	20%	60%	20%	-

Based on the first question of the subjective confidence of activity diagram with regard to test case generation, the result shows that three experts (60%) responded Agree that activity diagram has high subjective confidence with regard to test case generation, while one expert (20%) responded Neither and one other expert (20%) responded Strongly Disagree. On the other hand, the question of the subjective confidence of state chart diagram with regard to test case generation shows that three experts (60%) responded Agree that State chart diagram has high subjective confidence with regard to test case generation, while one expert (20%) responded Disagree, and one other expert (20%) responded Neither. This means that the state chart diagram outperforms the activity diagram with the experts' subjective confidence.

The second question of the subjective confidence of activity diagram with regard to test case generation shows that the five interviewed experts (100%) responded Agree that activity diagram has high subjective confidence. On the other hand, the question of the subjective confidence of state chart diagram with regard to test case generation shows that two experts (40%) responded Agree that activity diagram has high subjective confidence with regard to test case generation, two other experts (40%) responded Neither and one expert (20%) responded Disagree. This means that the teachers are completely satisfied with the activity diagram for test case generation as their preferred choice.

The third question of the subjective confidence of activity diagram with regard to test case generation shows that two experts (40%) responded Agree that activity diagram has high subjective confidence with regard to test case generation, while three other experts (60%) responded Neither. On the other hand, the question of the subjective

confidence of state chart diagram with regard to test case generation shows that one expert (20%) responded Agree that activity diagram has high subjective confidence with regard to test case generation, while one other expert (20%) responded Disagree and three experts (60%) responded Neither. This means that the activity diagram increases the degree of experts' awareness of UML test case generation.

The conclusion of measuring the comprehensibility between activity and state chart diagram in generating test cases include the results of experts' responses to the closed-ended and the open-ended questions. Regarding the second comprehension variable (the subjective confidence), activity diagram received more experts' confidence (certainty) when compared to state chart diagram based on the open-ended questions while the closed-ended questions confirmed these results and the details are illustrated as follows:

- Most of the interviewed experts agreed that activity diagram increases their certainty of the generated test cases.
- Most of the interviewed experts agreed that activity diagram is more comprehensible based on their own understanding and preference.
- Most of the interviewed experts preferred to generate test cases from activity diagram.
- Most of the interviewed experts agreed that activity diagram increases their own degree of awareness of UML test case generation.

This variable was not included during the only related study by Felderer and Herrmann (2015). This study therefore has contributed to this comprehension variable to evaluate the activity and state chart diagrams with regard to manual test case generation.

The results supported the claim whereby there are significant differences in the comprehension between activity and state chart diagrams with regard to test case generation. These results are consistent with the result obtained by Felderer and Herrmann (2015) where activity diagram is more comprehensible than state chart diagram with regard to manual test case generation.

4.3 Summary

This chapter described the evaluation interviews with the experts to evaluate the comprehensibility of activity and state chart diagrams in the case of generating test cases. The evaluation questions of the interview were constructed based on the evaluation criterion of their variables. The experts' evaluation data were analyzed by using NVivo and descriptive analysis. Findings from the study revealed there are significant differences in the comprehensibility of both of the proposed diagram with regard to test case generation. Besides, the excellence of activity diagram with regards to this study are disclosed as a consequence from evaluating the experts' perceived difficulty and the experts' subjective confidence of this UML diagram. Accordingly, these findings from the aforementioned interviews led to the consideration that the activity diagram is more comprehensible than state chart diagram with regard to manual test case.

CHAPTER FIVE

DISCUSSION AND CONCLUSION

5.1 Introduction

This chapter discusses results of the study depending on the outcome of the interview which was conducted with academic and industrial experts. The analysis based on the data gathered from the interviews and figures generated using NVivo10 were presented in the preceding chapter. This chapter is divided into four sections; each respectively discussing the achievement of the objectives, the contribution and the vision for future work. The fourth section highlights the limitation of this study. The final section concludes Chapter 5 and the current study.

5.2 Research Discussion

This study aimed to highlight the current issues related to the comprehensibility evaluation of the UML activity and state chart diagram with regard to test case generation using content analysis. In addition, the study also aimed to conduct experts' evaluation of the comprehensibility of activity and state chart diagrams in terms of test case generation. The following subsections discuss the achievement of the objectives.

5.2.1 Achieving First Objective

The first objective of this study aimed to highlight the comprehensibility evaluation issues related to the UML activity and state chart diagram with regard to test case generation based on the importance of these diagrams for test case generation by analyzing the content of related past studies. The purpose of this objective is to improve

the field of evaluating UML diagrams with regard to test case generation from both of the proposed diagrams in terms of comprehensibility. Furthermore, the content analysis of past research indicated that there is a lack of evaluation studies for the comprehensibility of the UML activity and state chart diagrams with regard to test case generation. As highlighted in the problem statement section in Chapter 1, many existing studies on comprehensibility evaluation on UML diagrams are more focused on the aspect of system design.

Furthermore, the content analysis aimed to provide more understanding of the different characteristics of activity and state chart diagrams in more specific detail. The study by Felderer and Herrmann (2015) did not mention in detail the reasons and specific characteristics that make activity diagram to be perceived as more comprehensible than the state chart diagram. Therefore, the initial study provided elaborated details about the evaluated diagrams and reasons that explained the different comprehension for the diagrams from the collected evaluation data by specialist experts as explained in Section 4.3.1.

The results of this objective are represented by prominent current trends with regard to test case generation using UML diagrams that were emphasized by Schweighofer and Hericko, (2014); Jena, Swain, and Mohapatra, (2014); Salman and Hashim, (2016). The content analysis of past studies explained the importance of activity and state chart diagrams respectively with regard to test case generation for reducing testing challenges (e.g. efforts, time and cost) through different methods.

Moreover, the findings also highlighted the importance of comprehension evaluation on UML diagrams as summarized in Table 2.4. Comprehensibility is the key quality

criterion to evaluate the UML diagrams as asserted by Aranda, Ernst, Horkoff, and Easterbrook, (2007); Budgen et al., (2011); Reinhartz-Berger and Sturm, (2014) and Liebel and Tichy, (2015). In addition, Budgen, Burn, Brereton, Kitchenham and Pretorius (2011) confirmed that comprehensibility is considered the most important quality attribute of UML models for had been studied.

Therefore, in order to achieve the evaluation for this study, the analysis of the content of past studies helped in proposing an instrument for comprehension evaluation by the experts on the two diagrams. The instrument, as illustrated in Table 3.1 was adapted from the relevant past studies. Existing studies on the comprehensibility of these two diagrams were designed to be used on inexpert participants (Felderer & Herrmann (2015). Thus, the substance of this study is achieved by evaluating the examined diagrams with regard to test case generation via the two determined measuring variables of comprehensibility (perceived difficulty and subjective confidence) that were illustrated by Aranda et al., (2007) and Felderer and Herrmann (2015). Aranda et al (2007) in their studies had proposed these two comprehensibility measurement variables when they created a flexible framework to evaluate the comprehensibility of model languages. Their studies, however, has never undergone any evaluation process. Felderer and Herrmann (2015) only evaluated perceived difficulty and not on subjective confidence.

Further, this content analysis provided specific details of the different characteristics of activity and state chart diagrams, in addition to the reasons of the different comprehension levels for the examined diagrams from the collected evaluation data by the specialist experts as discussed in the subsequent subsection. However, the study by

Felderer and Herrmann (2015) did not mention in detail the reasons and specific characteristics that make activity diagram to be perceived more comprehensible than the state chart diagram.

5.2.2 Achieving Second Objective

This section aimed to achieve the evaluation of comprehensibility for the UML activity and state chart diagrams with regard to test case generation. The evaluation data were conducted by experts in software testing and UML diagrams in order to conduct accurate evaluation and at the same to complement the effort done by Felderer and Herrmann (2015), who had conducted a similar study with inexperienced participants. The purpose of this objective is to unite the practical efforts of testers to choose among various types of MBT diagrams.

The experts' evaluation data were gathered using both closed-ended and open-ended questions that aimed to address the highlighted issue of the lack of evaluation study regarding to the comprehensibility of activity and state chart diagrams with aspect of test case generation. In this regard, Creswell (2012) explained that the use of this type of instrument in the interviews helps to collect useful information to support theories and concepts in the literature, in addition to explaining responses, exploring more detailed data supported by reasons by the open-ended questions. In evaluating the comprehensibility of the UML diagrams, Razali, Snook and Poppleton (2007), Gravino et al., (2008), Cruz-Lemus et al., (2011) and Shukla (2014) recommended using these questions in the research instrument. Therefore, this study adapted questions from past research to measure the comprehensibility for each diagram, albeit the lack of

underlying theory in the formulation of comprehensibility questions as highlighted by Condori-Fernandez et al. (2011).

The test cases from activity and state chart diagrams were generated to help conduct the evaluation of the examined diagrams with regard to test case generation as explained in Section 3.2.2.2. Subsequently, the interview sessions were arranged with the five (5) experts; the data were collected through filled forms and discussion notes were recorded and analyzed by NVivo.

The results show that the experts interviewed highlighted the high comprehensibility level for both activity and state chart diagrams with regard to test case generation, even with the recorded superiority for the comprehensibility of activity diagram. However, in detail, the experts marked more comprehensibility for activity diagram than comprehensibility for state chart diagram with regard to test case generation.

Furthermore, the experts assured that activity diagram is perceived to be easier with regard to test case generation. The experts justified that activity diagram is presentable in both system and business aspect, and it has the ability to show the transitions between the system and the user aspects. The experts also explained that these characteristics clarify the determination of test case generation steps and the valid and invalid input and output data for each test case. For this regard, Eriksson and Penker (2000) confirmed that activity diagram is the most important UML diagram for business aspect (user aspect), in addition to the system aspect. Moreover, its ability to present more information on the activities and the clear decision points and the effects on the easiness of test case generation steps was also mentioned by Mingsong, Xiaokang and Xuandong (2006). The experts indicated the easiness of the process of generating the test case

from activity diagram, which include generating the activity diagram and transferring it to activity graph and write the specification of the test case, similar to what was mentioned by Kundu and Samanta (2009). One expert mentioned the easiness and suitability of activity diagram to black box testing (Thanki & Shinde, 2014), making determination of test case generation easier.

The experts found that state chart diagram perceived difficulty is easier to test case generation steps because of the minimization of test case generation steps that contain lesser number of components for state chart diagram against the wide number of components for activity diagram. This argument is supported by Booch (2005). However, the results of the Perceived difficulty of this study manifest the same results as of Felderer and Herrmann (2015) i.e. the higher perceived comprehensibility of activity diagram in comparison with state chart diagram.

In the discussion on the subjective confidence measuring variable, the two examined diagrams with regard to the generated test cases have an uneven level of certainty based on their special features as mentioned by the experts and explained in Section 4.3.1. Through the interview session, the majority of experts preferred activity diagram with regard to test case generation based on subjective confidence and their own understanding. Also, most of the experts stated that activity diagram achieves a higher level of certainty based on the high clarity of the diagram in generating test cases and the ability of activity diagram to show the flow of the system, resembling the findings of Eriksson and Penker (2000). On top of that, the experts highlighted significant characteristics for activity diagram (e.g. the high simplicity and the ability to indicate the process flow of a system) and the aspect of the user (business) to generate test cases

and the high level of testing coverage (Kim et al., 2007). On the other hand, the experts gave a high level of certainty of generating test cases from state chart diagram based on its high test coverage ability, in addition to the ability to describe the dynamic behaviour of the system (Samuel et al., 2008).

In conclusion, the two questions for measuring the experts' perceived difficulty of the examined diagrams with regard to generation of test cases recorded higher comprehensibility for activity diagram based on the lower rate of experts' perceived difficulty as explained in Section 4.3.1.1. Besides that, the two questions for measuring the experts' subjective confidence recorded higher comprehensibility for activity diagram based on the higher rate of experts' subjective confidence on generating a test case from activity diagram. On the other hand, the state chart diagram looked more difficult in generating test cases and it recorded less comprehensibility. Also, the state chart diagram marked fewer experts' subjective confidence in generating test cases and that recorded less comprehensibility for state chart diagram.

It deserves to be mentioned that the closed-ended questions support the outcomes of the open-ended question. In detail, the overview of the experts' responses via the closed-ended questions highlighted that the activity diagram is more comprehensible with regard to test case generation through both of the Subjective confidence and the perceived difficulty comprehension's variables.

5.3 Contribution of Study

This research investigated the current issues associated with the need to evaluate the comprehensibility of the UML activity and state chart diagram with regard to test case

generation through the analysis of the content of past research related to the UML-based test case generation. The evaluation data was gathered with the assistance of experts in software testing who have experiences with the UML diagrams. This study supports the body of knowledge as well as the practice in several aspects, which are explained further subsequently.

5.3.1 Practical Contribution

Evaluating the comprehensibility of UML activity and state chart diagrams with regard to test case generation aimed to unite the practical efforts of testers to determine their preference diagram for test case generation from various types of MBT, as stated by Kansomkeat et al., (2008) and Kramer and Legeard, (2016). Therefore, this study has produced a comprehensibility instrument that contains evaluation questions even though there is scarcity of underlying theory for comprehensibility. The test cases from both activity and state chart diagrams were provided during this study to assist in evaluating the comprehensibility of the UML diagrams. Moreover, the essential contribution to this research is the evaluation of the UML diagrams by experts of software testing who have experiences in using UML diagrams, which has overcome the limitation highlighted by Felderer & Herrmann (2015).

Finally, the current study measured the subjective confidence comprehension's variable that has not been covered via the one and only evaluation study of Felderer and Herrmann (2015).

5.3.2 Theoretical Contribution

The content analysis approach of UML-based test case generation, which was involved in achieving the theoretical contribution of this study helps to determine the main issues related to the UML-based test case generation. The current trends in comprehension evaluation on UML diagrams highlighted the importance of comprehensibility as the key quality criterion for UML diagrams (Liebel & Tichy, 2015). Moreover, the necessity of evaluating the comprehensibility of the UML activity and state chart diagram with regard to test case generation by experts was highlighted through the analysis of the content of past studies related to the UML-based test case generation. Therefore, this study aimed to improve the field of evaluating the UML diagrams with regard to manual test case generation from both of the proposed diagrams in terms of comprehensibility as there have been limited studies in this field (Felderer & Herrmann, 2015). Further, this content analysis has provided more understanding of the different characteristics of activity and state chart diagrams in more specific details, enabling the adaptation of the evaluation questions that are related to the comprehensibility of UML diagrams even without substantial underlying theories. The experts' discussion has also provided insights into the different comprehension levels for the evaluated diagrams.

5.4 Future Work

The current study attempted to investigate the evaluation of behavioural UML diagrams based on the comprehension with regard to test case generation. This study envisions future work as follows:

- 1) Evaluate the comprehensibility of more than two behavioural UML diagrams.

- 2) Compare the comprehensibility of designing aspect and the testing aspect for both of the examined UML diagrams.

5.5 Limitation of the Study

Although this study provided an effective evaluation to represent comprehensibility of UML activity and state chart diagrams with regard to test case generation, there are still some limitations. The limits of this study are:

- 1) The limited number of diagrams that were examined during this study.
- 2) There is a need to compare the comprehensibility for the UML activity and state chart diagrams with regard to test case generation aspect with their comprehensibility in designing aspect.

5.6 Conclusion

This study, based on two comprehension variables (i.e. the perceived difficulty and the subjective confidence) albeit the lack of evaluation studies for the comprehensibility of the UML activity and state chart diagrams with regard to test case generation has revealed that both activity and state chart diagrams marked high comprehensibility with regard to test case generation despite the highlighted superiority for activity diagram. The activity diagram, besides being more comprehensible, possesses more subjective confidence in test cases generation.

REFERENCES

- Afzal, W., & Torkar, R. (2008). Incorporating metrics in an organizational test strategy. In *Software Testing Verification and Validation Workshop, 2008. ICSTW'08. IEEE International Conference on* (pp. 304–315).
- Agarwal, R., De, P., & Sinha, A. P. (1999). Comprehending object and process models: An empirical study. *IEEE Transactions on Software Engineering*, 25(4), 541–556.
- Al Dallal, J., & Sorenson, P. (2006). Generating class based test cases for interface classes of object-oriented black box frameworks. *Transactions on Engineering, Computing and Technology*, 16, 90–95.
- Alderson, J. C., & Banerjee, J. (1996). How might impact study instruments be validated. *Unpublished Manuscript Commissioned by UCLES*.
- Ali, M. A., Shaik, K., & Kumar, S. (2014). Test Case Generation using UML State Diagram and OCL Expression. *International Journal of Computer Applications*, 95(12)1-6.
- Anda, B., Sjoberg, D., & Jorgensen, M. (2001). Quality and understandability of use case models. In *European Conference on Object-Oriented Programming* (pp. 402–428).
- Aranda, J., Ernst, N., Horkoff, J., & Easterbrook, S. (2007). A framework for empirical evaluation of model comprehensibility. In *Proceedings of the International Workshop on Modeling in Software Engineering* (p. 7-12).
- Bansal, A. (2014). A Comparative Study of Software Testing Techniques. *International Journal of Computer Science and Mobile Computing*, 3, 579–584.
- Board, Q. (2013). Certified Tester Expert Level Modules Overview. *International Software Testing Qualifications Board, Version 1*.

- Boghdady, P. N., Badr, N. L., Hashem, M., & Tolba, M. F. (2011a). A proposed test case generation technique based on activity diagrams. *International Journal of Engineering & Technology IJET-IJENS*, 11(3)92-97.
- Boghdady, P. N., Badr, N. L., Hashem, M., & Tolba, M. F. (2011b). Test case generation and test data extraction techniques. *Inter. J. Electr. Comput. Sci*, 11(3), 87–94.
- Booch, G. (2005). *The unified modeling language user guide*. Pearson Education: India.
- Budgen, D., Burn, A. J., Brereton, O. P., Kitchenham, B. A., & Pretorius, R. (2011). Empirical evidence about the UML: a systematic literature review. *Software: Practice and Experience*, 41(4), 363–392.
- Byckling, P., Gerdt, P., Kuzniarz, L., & Sajaniemi, J. (2006). Increasing comprehensibility of object models: Making the roles of attributes explicit in UML diagrams. *Nordic Journal of Computing*, 13(3), 149-161.
- Chandu, P. (2015). An Analytical Way to Improve Test Execution and Review of Software Metrics for The Software Quality. *Journal of Theoretical and Applied Information Technology*, 73(1)1-6.
- Chism, N. V. N., Douglas, E., & Hilson Jr, W. J. (2008). Qualitative research basics: A guide for engineering educators. *Rigorous Research in Engineering Education NSF DUE-0341127*.
- Choudhary, D., & Kumar, V. (2011). Software testing. *Journal of Computational Simulation and Modeling*, 1(1), 1-5.
- Condori-Fernandez, N., Daneva, M., Sikkel, K., & Herrmann, A. (2011). Practical relevance of experiments in comprehensibility of requirements specifications. In *Workshop on Empirical Requirements Engineering (EmpiRE 2011)* (pp. 21–28).

- Cox, K., Phalp, K., & Shepperd, M. (2001). Comparing use case writing guidelines. In *7th International Workshop on Requirements Engineering: Foundation for Software Quality, Interlaken, Switzerland* (Vol. 45, p. 101-112).
- Creswell. (2012). *Educational Research Planning, Conducting, and Evaluating Quantitative and Qualitative Research*. Sage.
- Crowder, J. A., Carbone, J. N., & Demijohn, R. (2016). Systems Engineering Tools and Practices. In *Multidisciplinary Systems Engineering* (pp. 89–103). Springer.
- Cruz-Lemus, J. A., Genero, M., Caivano, D., Abrahão, S., Insfrán, E., & Cars'i, J. A. (2011). Assessing the influence of stereotypes on the comprehension of UML sequence diagrams: A family of experiments. *Information and Software Technology*, 53(12), 1391–1403.
- Delanote, D., Van Baelen, S., Joosen, W., & Berbers, Y. (2008). An automatic test data generation from UML state diagram using genetic algorithm. *2008 13th IEEE International Conference on Engineering of Complex Computer Systems*.
- Doungsa-ard, C., Dahal, K. P., Hossain, M. A., & Suwannasart, T. (2007). An automatic test data generation from UML state diagram using genetic algorithm. *The University of Bradford Institutional Repository*, (<http://bradscholars.brad.ac.uk>).
- Dubey, S. K., & Sharma, D. (2015). Software Quality Appraisal Using Multi-Criteria Decision Approach. *I.J. Information Engineering and Electronic Business*, 1530-1362/04.
- Eriksson, H.-E., & Penker, M. (2000). Business modeling with UML. *Business Patterns at Work, John Wiley & Sons: New York, USA*.
- Felderer, M., Büchler, M., Johns, M., Brucker, A. D., Breu, R., & Pretschner, A. (2015). Security Testing: A Survey. *Advances in Computers*.

- Felderer, M., & Herrmann, A. (2015). Manual test case derivation from UML activity diagrams and state machines: A controlled experiment. *Information and Software Technology*, 61, 1–15, 61, 1–15.
- Figl, K., & Laue, R. (2011). Cognitive complexity in business process modeling. In *International Conference on Advanced Information Systems Engineering* (pp. 452–466).
- Garg, P. (2015). Role of Testing Strategies in Improving Quality of Software. *International Journal of Research*, 2(12), 800–805.
- Gravino, C., Scanniello, G., & Tortora, G. (2008). An empirical investigation on dynamic modeling in requirements engineering. In *International Conference on Model Driven Engineering Languages and Systems* (pp. 615–629).
- Gulia, P., & Chugh, J. (2015). Comparative Analysis of Traditional and Object-Oriented Software Testing. *ACM SIGSOFT Software Engineering Notes*, 40(2), 1–4.
- Gupta, J. (2014). *An Investigation of Test Cases Generation from Activity Diagram*. Thesis. Thapar University Patiala.
- Gupta, N., Yadav, V., & Singh, M. (2016). A Review on Automated Regression Testing. *International Journal of Engineering Technology Science and Research*, 3, 46–50.
- Habib, Z. (2009). *The critical success factors in implementation of software process improvement efforts: CSFs, motivators & obstacles*. University of Gothenburg.
- Hadar, I., & Hazzan, O. (2004). On the contribution of UML diagrams to software system comprehension. *Journal of Object Technology*, 3(1), 143–156.

- Harel, D., & Politi, M. (1998). *Modeling reactive systems with statecharts: the STATEMATE approach*. McGraw-Hill, Inc.
- Hashim, N. L., & Salman, Y. D. (2011). An Improved Algorithm With regard to test case generation From Uml Activity Diagram Using Activity Path. In *Proceedings of the 3rd International Conference on Computing and Informatics, ICOCI*.
- Heinecke, A., Bruckmann, T., Griebe, T., & Gruhn, V. (2010). Generating test plans for acceptance tests from uml activity diagrams. In *Engineering of Computer Based Systems (ECBS), 2010 17th IEEE International Conference and Workshops on* (pp. 57–66).
- Ingle, S. E., & Mahamune, M. R. (2015). An UML Based software Automatic Test Case Generation: Survey. *International Research Journal of Engineering and Technology*, 2, 971–973.
- Jain, A., Jain, M., & Dhankar, S. (2014). A Comparison of RANOREX and QTP Automated Testing Tools and their impact on Software Testing. *Nternational Journal of Engineering, Management & Sciences (IJEMS) ISSN-2348*, 1(1), 8–12.
- Jena, A. K., Swain, S. K., & Mohapatra, D. P. (2014a). ? In *Issues and Challenges in Intelligent Computing Techniques (ICICT), 2014 International Conference on* (pp. 621–629).
- Jena, Swain, & Mohapatra. (2014b). A novel approach for test case generation from UML *International Conference on activity diagram*. In *(ICICT)*, (pp. 621–629).
- Kansomkeat, S., Offutt, J., Abdurazik, A., & Baldini, A. (2008). A comparative evaluation of tests generated from different UML diagrams. *International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2008. SNPD'08. Ninth ACIS* (pp. 867–872).

- Kansomkeat, S., & Rivepiboon, W. (2003). Automated-generating test case using UML statechart diagrams. In *Proceedings of the 2003 annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology* (pp. 296–300).
- Kaur, G., & Bajaj, A. (2015). A Notation of Unified Modeling Language in various Areas. *Asian Journal of Research in Social Sciences and Humanities*, 5(3), 195–205.
- Khandai, M., Acharya, A. A., & Mohapatra, D. P. (2011a). A novel approach of test case generation for concurrent systems using UML Sequence Diagram. In *Electronics Computer Technology (ICECT), 2011 3rd International Conference on* (Vol. 1, pp. 157–161).
- Khandai, M., Acharya, A. A., & Mohapatra, D. P. (2011b). Test case generation for concurrent system using UML combinational diagram. *International Journal of Computer Science and Information Technologies, IJCSIT*, 2-11-18.
- Khurana, N., Chhillar, R. S., & Chhillar, U. (2016). A Novel Technique for Generation and Optimization of Test Cases Using Use Case, Sequence, Activity Diagram and Genetic Algorithm. *Journal of Software*, 11, 242–250.
- Kim, H., Kang, S., Baik, J., & Ko, I. (2007). Test cases generation from UML activity diagrams. In *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007. SNPD 2007. Eighth ACIS International Conference on* (Vol. 3, pp. 556–561).
- Kingston, J., & Macintosh, A. (2000). Knowledge management through multi-perspective modelling: representing and distributing organizational memory. *Knowledge-Based Systems*, 13(2), 121–131.

- Konka, B. B. (2012). *A case study on Software Testing Methods and Tools*. University of Gothenburg.
- Koriat, A. (2011). Subjective confidence in perceptual judgments: a test of the self-consistency model. *Journal of Experimental Psychology: General*, 140(1), 117.
- Kosindrdecha, N., & Daengdeg, J. (2010). A test generation method based on state diagram. *Journal of Theoretical and Applied Information Technology*, 28–44.
- Kouider, S., De Gardelle, V., Sackur, J., & Dupoux, E. (2010). How rich is consciousness? The partial awareness hypothesis. *Trends in Cognitive Sciences*, 14(7), 301–307.
- Kramer, A., & Legeard, B. (2016). *Model-Based Testing Essentials-Guide to the ISTQB Certified Model-Based Tester-Foundation Level*. John Wiley & Sons.
- Kundu, D., & Samanta, D. (2009). A Novel Approach to Generate Test Cases from UML Activity Diagrams. *Journal of Object Technology*, 8(3), 65–83.
- Kundu, D., Sarma, M., Samanta, D., & Mall, R. (2009). System testing for object-oriented systems with test case prioritization. *Software Testing, Verification and Reliability*, 19(4), 297–333.
- Kuzniarz, L., Staron, M., & Wohlin, C. (2004). An empirical study on using stereotypes to improve understanding of UML models. In *Program Comprehension, 2004. Proceedings. 12th IEEE International Workshop on* (pp. 14–23).
- Lewins, A., & Silver, C. (2007). *Using Software for Qualitative Data Analysis: A Step-by-Step Guide*. Thousand Oaks, Calif.: Sage.
- Lewis, W. E. (2016). *Software testing and continuous quality improvement*. CRC press.
- Li, L., Li, X., He, T., & Xiong, J. (2013). Extenics-based test case generation for UML activity diagram. *Procedia Computer Science*, 17, 1186–1193.

- Liebel, M. T. G., & Tichy, M. (2015). Comparing Comprehensibility of Modelling Languages for Specifying Behavioural Requirements. In *First International Workshop on Human Factors in Modeling (HuFaMo 2015)*. CEUR-WS (pp. 17–24).
- Linzhang, W., Jiesong, Y., Xiaofeng, Y., Jun, H., Xuandong, L., & Guoliang, Z. (2004). Generating test cases from UML activity diagram based on gray-box method. *Software Engineering Conference, 2004. 11th Asia-Pacific*, 284–291.
- Lucantonio, V. (2015). Enanching the consistency between requirements and test cases through the definition of a Controlled Natural Language. *Digital Vetenskapliga Arkivet*.
- Macintosh, A., Coleman, S., & Schneeberger, A. (2009). eParticipation: The research gaps. In *Electronic participation* (pp. 1–11). Springer.
- Mailewa, A. B. (2015). Reducing Software Testing Time with Combinatorial Testing and Test Automation. *St. Cloud State University the Repository at St. Cloud State*.
- Mailewa, A., Herath, J., & Herath, S. (2015). A Survey of Effective and Efficient Software Testing. *Micsymposium.org*. Retrieved from <http://www.micsymposium.org>
- Manaseer, S., Manaseer, W., Alshraideh, M., Abuhashish, N., & Adwan, O. (2015). Automatic Test Data Generation for Java Card Applications Using Genetic Algorithm. *Journal of Software Engineering and Applications*, 8(12), 603-609.
- McQuillan, J. A., & Power, J. F. (2005). A survey of UML-based coverage criteria for software testing. *Department of Computer Science. NUI Maynooth, Co. Kildare, Ireland*.

- Meena, D. K. (2013). *Test Case Generation From UML Interaction Overview Diagram and Sequence Diagram*. National Institute of Technology Rourkela.
- Mendonca, M. G., Maldonado, J. C., De Oliveira, M. C. F., Carver, J., Fabbri, S. C. P. F., Shull, F., ... Basili, V. R. (2008). A framework for software engineering experimental replications. In *Engineering of Complex Computer Systems, 2008. ICECCS 2008. 13th IEEE International Conference on* (pp. 203–212).
- Miles, M. B., & Huberman, A. M. (1994). *Qualitative data analysis: An expanded sourcebook*. Sage.
- Mingers, J. (2001). Combining IS research methods: towards a pluralist methodology. *Information Systems Research*, 12(3), 240–259.
- Mingsong, C., Xiaokang, Q., & Xuandong, L. (2006). Automatic test case generation for UML activity diagrams. In *Proceedings of the 2006 international workshop on Automation of software test* (pp. 2–8).
- Mohanty, S., Acharya, A. A., & Mohapatra, D. P. (2011). A model based prioritization technique for component based software retesting using uml state chart diagram. In *Electronics Computer Technology (ICECT), 2011 3rd International Conference on* (Vol. 2, pp. 364–368).
- Mussa, M., Ouchani, S., Al Sammane, W., & Hamou-Lhadj, A. (2009). A survey of model-driven testing techniques. In *Quality Software, 2009. QSIC'09. 9th International Conference on* (pp. 167–172).
- Nayak, A., & Samanta, D. (2011). Synthesis of test scenarios using UML activity diagrams. *Software & Systems Modeling*, 10(1), 63–89.

- Nikfard, P., bin Ibrahim, S., Rohani, B. D., bin Selamat, H., & Naz'ri, M. (2013). An Evaluation for Model Testability approaches. *International Journal Of Computers & Technology*, 9(1), 938–947.
- Oluwagbemi, O., & Asmuni, H. (2015). An Approach for Automatic Generation of Test Cases from UML Diagrams. *International Journal of Software Engineering and Its Applications*, 9(8), 87–106.
- Otero, M. C., & Dolado, J. J. (2004). Evaluation of the comprehension of the dynamic modeling in UML. *Information and Software Technology*, 46(1), 35–53.
- Pahwa, N., & Solanki, K. (2014). UML based test case generation methods: A review. *International Journal of Computer Applications*, 95(20)311-316.
- Pandey, N., & Mohapatra, D. P. (2012). Test Case Generation from UML Interaction Diagrams. In *International Conference on Computing* (p. 17-24).
- Patel, P. E., & Patil, N. N. (2013). Testcases Formation using UML Activity Diagram. In *Communication Systems and Network Technologies (CSNT), 2013 International Conference on* (pp. 884–889).
- Patil, K., & Ganeshwade, M. M. (2014). Generating Automated Test Cases using Model Based Testing. *International Journal of Enhanced Research in Science Technology & Engineering*, 3, 18–24.
- Razali, R., Snook, C. F., & Poppleton, M. R. (2007). Comprehensibility of uml-based formal model: A series of controlled experiments. In *Proceedings of the 1st ACM international workshop on empirical assessment of software engineering languages and technologies: Held in conjunction with the 22nd IEEE/ACM international conference on automated software engineering (ASE) 2007* (pp. 25–30).

- Razali, R., Snook, C., Poppleton, M., Garratt, P., & Walters, R. (2007). Usability assessment of a UML-based formal modelling method. In *19th Annual Psychology of Programming Workshop (PPIG'07)* (pp. 56–71).
- Reza, H., Ogaard, K., & Malge, A. (2008). A model based testing technique to test web applications using statecharts. In *Fifth International Conference on Information Technology: New Generations* (pp. 183–188).
- Ribiero, N. F., & Yarnal, C. M. (2010). The Perceived Difficulty Assessment Questionnaire (PDAQ): Methodology and Applications for Leisure Educators and Practitioners. *Schole*, 25.
- Robson, C., & McCartan, K. (2016). *Real world research*. John Wiley & Sons.
- Rumbaugh, J., Jacobson, I., & Booch, G. (2004). *Unified Modeling Language Reference Manual*. Pearson Higher Education.
- Salman, Y. D., & Hashim, N. L. (2014). An Improved Method of Obtaining Basic Path Testing for Test Case Based On UML State Chart. *Science International*, 26(4)1-8.
- Salman, Y. D., & Hashim, N. L. (2016). Automatic Test Case Generation from UML State Chart Diagram: A Survey. In *Advanced Computer and Communication Engineering Technology* (pp. 123–134). Springer.
- Samuel, P., Mall, R., & Bothra, A. K. (2008). Automatic test case generation using unified modeling language (UML) state diagrams. *IET Software*, 2(2), 79–93.
- Scanniello, G., Gravino, C., Risi, M., Tortora, G., & Doderio, G. (2015). Documenting design-pattern instances: a family of experiments on source-code comprehensibility. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 24(3), 14.

- Schweighofer, T., & Hericko, M. (2014). Approaches for Test Case Generation from UML Diagrams. In *SQAMIA* (pp. 91–98).
- Sekaran, U., & Bougie, R. (2010). Research Design. *JW Ltd, Research Methods for Business-*, 110.
- Sharma, S., & Vishawjyoti, M. (2013). Study And Analysis Of Automation Testing Techniques. *Journal of Global Research in Computer Science*, 3(12), 36–43.
- Shirole, M., & Kumar, R. (2013). UML behavioral model based test case generation: a survey. *ACM SIGSOFT Software Engineering Notes*, 38(4), 1–13.
- Shirole, M., Suthar, A., & Kumar, R. (2011). Generation of improved test cases from UML state diagram using genetic algorithm. In *Proceedings of the 4th India Software Engineering Conference* (pp. 125–134).
- Shneiderman, B. (1992). Designing the user interface: strategies for effective human-computer interaction, 2nd edn. Addison. *Reading, MA*.
- Shukla, D. (2014). Analyzing the Comprehensibility of Aspect-Oriented Modelling and Design of Software System. *International Journal of Computer Applications*, 95(21).
- Shukla, & Chandel. (2012). A Systematic Approach for Generate Test Cases Using UML Activity Diagrams. *International Journal of Research in Management & Technology (IJRMT)*, ISSN: 2249-9563, 2, 469–475.
- Siau, K., & Cao, Q. (2001). Unified modeling language: A complexity analysis. *Journal of Database Management (JDM)*, 12(1), 26–34.
- Sinkovics, R. R., & Alfoldi, E. A. (2012). Progressive focusing and trustworthiness in qualitative research. *Management International Review*, 52(6), 817–845.

- Sommerville, I. (2010). Software testing. *Ian Sommerville 2004, Software Engineering, 7th Edition, Prechelt@inf.fu-Berlin.de*, 7.
- Swain, R. K., Behera, P. K., & Mohapatra, D. P. (2012). Minimal TestCase Generation for Object-Oriented Software with State Charts. *arXiv Preprint arXiv:1208.2265*.
- Swain, S. K., Mohapatra, D. P., & Mall, R. (2010). Test Case Generation Based on State and Activity Models. *Journal of Object Technology*, 9(5), 1–27.
- Thanki, M. H. J., & Shinde, S. M. (2014). Test case generation using UML activity diagram-A survey, 3, 292–300.
- Tripathy, A., & Mitra, A. (2013). Test Case Generation Using Activity Diagram and Sequence Diagram. In *Proceedings of International Conference on Advances in Computing* (pp. 121–129).
- Utting, M., & Legeard, B. (2010). *Practical model-based testing: a tools approach*. Morgan Kaufmann.
- Vashishtha, V., Singla, T., & Singh, S. (2014). Software Testing. *International Journal of Research*, 1(8), 1258–1264.
- Verma, S., Yadav, K. P., & Tiwari, U. K. (2012). Software Testing Techniques for Finding Errors. *International Journal of Research Review in Engineering Science and Technology*, 2, 433–439.
- Wang, L. (2015). GUI test automation for Qt application. *Digital Vetenskapliga Arkivet*.
- Wang, X., Jiang, X., & Shi, H. (2015). Prioritization of test scenarios using hybrid genetic algorithm based on UML activity diagram. In *Software Engineering and Service Science (ICSESS), 2015 6th IEEE International Conference on* (pp. 854–857).

- Willig, C., & Stainton-Rogers, W. (2007). *The SAGE handbook of qualitative research in psychology*. Sage.
- Xie, S., Kraemer, E., & Stirewalt, R. E. K. (2007). Empirical evaluation of a UML sequence diagram with adornments to support understanding of thread interactions. In *15th IEEE International Conference on Program Comprehension (ICPC'07)* (pp. 123–134).
- Yang, N., Yu, H., Sun, H., & Qian, Z. (2010). Mapping uml activity diagrams to analyzable petri net models. In *2010 10th International Conference on Quality Software* (pp. 369–372).
- Yu, L., Xu, X., Liu, C., & Sheng, B. (2012). Using grounded theory to understand testing engineers' soft skills of third-party software testing centers. In *Software Engineering and Service Science (ICSESS), 2012 IEEE 3rd International Conference on* (pp. 403–406).
- Zhu, H., Hall, P. A. V, & May, J. H. R. (1997). Software unit test coverage and adequacy. *Acm Computing Surveys (Csur)*, 29(4), 366–427.

APPENDIX A

QUESTIONNAIRE



AN ASSESSMENT OF THE COMPREHENSIBILITY OF BEHAVIORAL UML DIAGRAMS IN TEST CASE GENERATION

Dear Respondents,

This evaluation is to collect your responses in view of investigating behavioral UML diagrams based on the comprehension in test case generation. Your responses will help us in determining the success of the evaluation. Hence, your responses are to be as objective as possible.

In this evaluation, two groups of questions are used. The first contains closed-ended questions that are further divided into two sections A and B. The two sections aim to evaluate two variables of comprehension which are the perceived comprehensibility and the subjective confidence of activity and state chart diagrams. Each question in both sections carries five possible responses. The second question group carries open-ended questions that aim to rank the comprehensibility of activity and state chart diagrams for test case generation through the same detailed variables.

Thank you for your willingness to participate in this evaluation.

Name:

Gender: **Age:**

E-mail: **Mobile phone:**

Qualification:

Position:

Company/ Institution:

Briefly describe your experience in conducting software testing:

Years of Experience in conducting software testing:

Briefly describe your experience in using UML diagrams:

Years of Experience in using UML diagrams:

Date: / 11 / 2016

Signature: _____

II. The Close-Ended Questions

A1- Evaluate the Perceived Difficulty Variable of UML Activity Diagram	Please tick (✓)				
	Very Difficult	Difficult	Neither	Easy	Very Easy
1- How difficult is it to learn the test case generation from activity diagram?					
2- How difficult are the required procedure of generating test case from activity diagram?					
3- How difficult is it to achieve higher testing coverage when you are generating test cases from activity diagram?					
A2- Evaluate the Subjective Confidence Variable of UML Activity Diagram	Please tick (✓)				
	Strongly Disagree	Disagree	Medium	Agree	Strongly Agree
1- If you are in a task of explaining the generated test case from one of UML diagrams to others, do you agree that you will be more confident when you are explaining test case generation from activity diagram?					
2- Do you agree that activity diagram is your preferred choice to generate the test case for a project requirement or design?					
3- How you agree that generating test case from activity diagram increases your degree of awareness of UML test case generation?					

B1- Evaluate the Perceived Difficulty Variable of UML Statechart Diagram	Please tick (✓)				
	Very Difficult	Difficult	Neither	Easy	Very Easy
1- How difficult is it to learn the test case generation from state chart diagram?					
2- How difficult are the required procedure of generating test case from state chart diagram?					
3- How difficult is it to achieve higher testing coverage when you are generating test cases from state chart diagram?					
B2- Evaluate the Subjective Confidence Variable of UML Statechart Diagram	Please tick (✓)				
	Strongly Disagree	Disagree	Medium	Agree	Strongly Agree
1- If you are in a task of explaining the generated test case from one of UML diagrams to others, do you agree that you will be more confident when you are explaining test case generation from state chart diagram?					
2- Do you agree that state chart diagram is your preferred choice to generate the test case for a project requirement or design?					
3- How you agree that generating test case from state chart diagram increases your degree of awareness of UML test case generation?					

III. The discussion Questions:

1- Which UML diagram (activity diagram or state chart diagram) would you believe that more difficult to determine test case generation steps like input data and expected results?

Please explain your response in detail.

2- If you have two test cases that are generated, the first from activity diagrams and the second from state chart diagrams for the same system, how difficult is it to determine the origin of the first generated test cases from activity and the second generated test cases from state chart diagram?

Please explain your response in detail.

3- Which UML diagram (activity diagram or state chart diagram) increases your certainty of the generated test cases?

Please explain your response in detail.

4- Based on your own understanding, please evaluate the comprehensibility of activity and state chart diagrams in generating the test cases?

Please explain your response in detail, in terms of comprehensibility aspect.

APPENDIX B

SUMMARIES OF THE INTERVIEW SESSIONS WITH EXPERTS

Participant	Question1	Question2	Question3	Question4
Expert1	<p>Activity diagram has more Perceived comprehensibility (less perceived difficulty) in test case generation because of:</p> <ol style="list-style-type: none"> 1) the easiness of determining the valid and invalid input and output data for each test case. 2) the ability to show the transitions between the system and the user aspects. 	<p>Activity diagram has more Perceived comprehensibility (less perceived difficulty) in determining the origin of the generated test case because of:</p> <ul style="list-style-type: none"> - the easiness of the process of generating the test case from activity diagram. 	<p>Activity diagram has more Subjective confidence (certainty) in test case generation because of:</p> <ul style="list-style-type: none"> - the ability of activity diagram to indicate the control flow of the system and the right flow of the system that leads to the right test case generation. 	<p>Activity diagram has more Subjective confidence based on experts' preference and own understanding in test case generation because of:</p> <ul style="list-style-type: none"> - the ability of activity diagram taking the aspect of business (user aspect) that increases the higher confidence of activity diagram in test case generation.
Expert2	<p>Activity diagram has more Perceived comprehensibility (less perceived difficulty) in determining test case generation steps because of:</p> <ol style="list-style-type: none"> 1) the clarity of the steps of generating the test cases from activity diagram. 	<p>Activity diagram has more Perceived comprehensibility (less perceived difficulty) in determining the origin of the generated test case because of:</p> <ul style="list-style-type: none"> - the easiness of the process of generating the test case from activity diagram. 	<p>Activity diagram has more Subjective confidence (certainty) in test case generation because of:</p> <ul style="list-style-type: none"> - the ability of activity diagram in showing a clear flow of the system that reflects high certainty on the generated test case. 	<p>Activity diagram has more Subjective confidence based on experts' preference and own understanding in test case generation because of:</p> <ul style="list-style-type: none"> - the higher comprehensibility of activity diagram due to the simplicity of activity diagram.

	2) the ability to present more information on the activities and the clear decision points and the effects on the easiness of test case generation steps.			
Expert3	State chart diagram has more Perceived comprehensibility (less perceived difficulty) in determining test case generation steps because of: - the less number of components of state chart diagram.	Activity diagram has more Perceived comprehensibility (less perceived difficulty) in determining the origin of the generated test case because of: - activity diagram reflects more view of business aspect that makes the test case easy to read and easy to match to their diagram.	State chart diagram has more Subjective confidence (certainty) in test case generation because of: - the ability of state chart diagram in describing the dynamic behaviour of the system that explains how the system changes if an appropriate trigger is applied to the system, thus, the correct state chart can be easily converted to code.	Activity diagram has more Subjective confidence based on experts' preference and own understanding in test case generation because of: - the minimization ability of activity diagram in test case generation
Expert4	State chart diagram has more Perceived comprehensibility (less perceived difficulty) in determining test case generation steps because of: - the minimization of steps of state chart diagram makes it easier in listing the steps of generating the test cases.	State chart diagram has more Perceived comprehensibility (less perceived difficulty) in determining the origin of the generated test case because of: - the simplicity of tracking the events and states of state chart diagram which make their test cases easier to determine their origin diagram.	State chart diagram has more Subjective confidence (certainty) in test case generation because of: - the high-test coverage of state chart diagram in generating the test cases.	Activity diagram has more Subjective confidence based on experts' preference and own understanding in test case generation because of: - the higher subjective confidence believing on the importance of taking the aspect of user (business) to generate test cases from activity diagram.

Expert5	<p>Activity diagram has more Perceived comprehensibility (less perceived difficulty) in determining test case generation steps because of:</p> <ul style="list-style-type: none"> - the suitability of this diagram with black box testing that makes the determining of test case generation easier. 	<p>State chart diagram has more Perceived comprehensibility (less perceived difficulty) in determining the origin of the generated test case because of:</p> <ul style="list-style-type: none"> - the minimization of this diagram. 	<p>Activity diagram has more Subjective confidence (certainty) in test case generation because of:</p> <ul style="list-style-type: none"> - the high-test coverage of activity diagram in generating the test cases. 	<p>Activity diagram has more Subjective confidence based on experts' preference and own understanding in test case generation because of:</p> <ul style="list-style-type: none"> - The higher subjective confidence believing on the characteristic of the diagram that imitates the process flow of the system.
----------------	--	--	---	---

