MODEL-BASED SEMI-AUTOMATED TEST CASE GENERATION APPROACH USING UML DIAGRAMS

HUSSAM MOHAMED BASHIR MOHAMED ALI

UNIVERSITI TEKNOLOGI MALAYSIA

MODEL-BASED SEMI-AUTOMATED TEST CASE GENERATION APPROACH USING UML DIAGRAMS

HUSSAM MOHAMED BASHIR MOHAMED ALI

A thesis submitted in fulfilment of the
requirements for the award of the degree of
Master of Computer Science

School of Computing
Faculty of Engineering
Universiti Teknologi Malaysia

Jul2019

DEDICATION

To my beloved parents and siblings, whose spring of love and tenderness shaped my heart, instilled determination &confidence in me;
they always believed in me, and thus I started to believe

ACKNOWLEDGEMENT

In the Name of Allah SWT, The Most Gracious and The Most Merciful.
First and foremost, all praise to The Almighty, all praise to Allah swt for bestowing me with knowledge, good health, courage and strength to complete my master study. Alhamdullilah for His endless blessings throughout my entire research process. My sincere appreciation to my main supervisor, Associate Professor Dr. Dayang Norhayati Abang Jawawi for her supervision, guidance, countless hours in sharing understanding and patience throughout my research journey. I am thankful for her insightful comments, criticism, and advice during my learning process. I will always look up to them as my academic role model to achieve my ambitions.

I would also like to express my gratitude to my parents, Mohamed and Mona, my siblings who mean most to me, for their prayers of day and night, understanding, moral and financial support to complete my master study. Special thanks also to my close friends for their kindness, assistance, and support through thick and thin.

ABSTRACT

Software Testing, a process comprised of test case generation, execution and evaluation, is one of the imperative phases of the development life cycle, with its cost approximated to about 50% of the overall development cost. Researchers have automated it using models with the utmost focus put on Unified Modeling Language (UML) as the up-to-date de facto standard utilized in software modeling. Its diagrams include both behavioral and structural.

Recently, Model-Based Testing (MBT) application using Unified Modelling Language (UML) has achieved high ranking from many testers to use UML diagrams for test case generation. The benefit of this technique is to achieve early detection of faults, bugs, and errors in the design phase. Some UML diagrams have a limitation in generating test cases such that UML diagrams do not support looping and iteration activities. To avoid this issue, an integrated semi-automated test case generation technique has been proposed to generate test cases from UML sequence diagram that can support the looping process. The enhanced technique has been applied to the same case study as in the original technique. A matrix tool is then applied to the enhanced test cases to achieve better coverage.

ABSTRAK

Ujian Perisian adalah satu proses yang terdiri daripada penjanaan kes ujian, pelaksanaan dan penilaian, ia merupakan salah satu fasa penting dalam kitaran hayat pembangunan(life cycle) dengan kosnya kira-kira 50% daripada kos pembangunan keseluruhan. Penyelidik telah menggunakan model paling sesuai dengan memberi tumpuan sepenuhnya pada Unified Modeling Language (UML) sebagai piawaian yang sebenarnya sehingga digunakan dalam pemodelan perisian. Terdapat dua jenis kategori UML terdiri daripada gambarajah UML tingkahlaku(behavioral) dan gambarajah UML struktur(structural).

Selain itu, akhir-akhir ini, aplikasi Pengujian Berbasis Model (MBT) menggunakan Bahasa Pemodelan Bersepadu (UML) telah mencapai kedudukan yang tinggi daripada beberapa penguji(tester) untuk menggunakan rajah UML untuk penjanaan kes ujian.

Disamping itu juga, kebaikan dari teknik ini adalah untuk mencapai pengesanan awal kesalahan, pepijat, dan kesilapan dalam fasa reka bentuk. Sesetengah gambarajah UML mempunyai batasan dalam menjana kes ujian, supaya rajah UML boleh menyokong aktiviti gegelung(looping) dan lelaran(iteration). Untuk mengelakkan masalah ini, teknik yang dipertingkatkan telah dicadangkan untuk menjana kes ujian dari gambarajah urutan UML yang boleh menyokong proses gegelung(looping). Teknik tersebut telah digunakan untuk kajian kes yang sama seperti teknik asal. Kemudian alat matriks terpakai untuk kes ujian yang ditingkatkan untuk mengukur kualiti dan ketepatan kes ujian yang dihasilkan.

TABLE OF CONTENTS

CHAPTER **TITLE** **PAGE**

Table of Contents

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CC	-	Cyclomatic Complexity.
MBT	-	Model Based Test Case Generation.
SW	-	Software Engineering.
SDLC	-	Software Development Life Cycle.
SD	-	Sequence Diagram.
TCG	-	Test Case Generation.
UML	-	Unified Modelling Language.

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Introduction

Nowadays individuals and societies rely on technologies that adopt advanced software systems in every aspect of daily life. Software industry has become a rapidly changing, large, complex, and rather integrated software system.
When new technologies take off, modern software become larger and more complex. Computer applications have spread into every sphere of life for manipulation of several sophisticated applications and most of these applications are very large and complex (Syafiqah et al., 2015).

Software testing has become the most important of the Software Development Life Cycle (SDLC) phases, because it determines whether the system or its components are satisfying the specific requirements or not. Software Development Life Cycle is a process followed by software industries to design, implement and test the software products. SDLC is the acronym for Software Development Life Cycle. SDLC process is followed strictly to ensure quality products are delivered to the customers and consumers within the planned budget and time estimates. SDLC has six stages, with each phase having its own deliverables which serve as input for the next phase as shown in Figure 1.1.

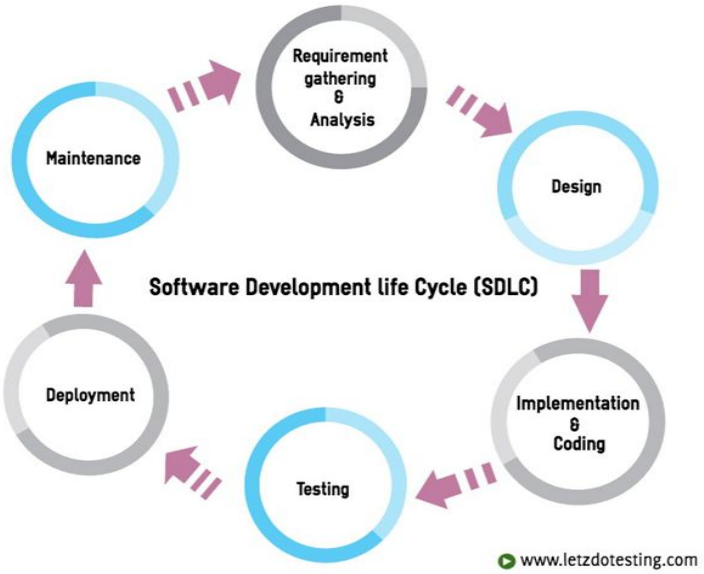

Figure 1. 1: Software Development Life Cycle (SDLC).

Software testing is believed to save up to 50% of time and reduce the cost by up to 50% Some researchers have defined software testing as a method, or a sequence of processes, of dynamically executing a program with given inputs, to ensure that the computer code does exactly that for which it is designed. Various applications are turning out to be ever more omnipresent, taking care of an extensive diversity of well accepted and safety-critical devices.

Testing is the frequently utilized technique used in authenticating software applications, and efficient testing methods may well be supportive for enhancing the reliability of such systems. Despite the ease of manual testing in terms of following the sequence of operations in the program step by step, it requires a lot of effort and time compared to automated testing. This is because automatic testing takes less time and less effort (high efficiency). Therefore, the automated testing in our time is most desired by the testers to apply in the software systems because it achieves the speed factor, which is an important factor to evaluate the efficiency of the program.

Unified Modelling Language provides various model to be applied in modelling Object Oriented Systems, such as Use Case Diagram, Class Diagram, Sequence Diagram, Activity Diagram, State Chart Diagram, Deployment Diagram. It is also used in the modelling of Object-Oriented Systems, and has been applied in the

design of tests in various phases such as requirements phase, unit, integration, and system.

Currently, many researchers are proposing innovative ways to reuse design models for the test generation process. Although there are many other techniques that use different models, this helps to increase the efficiency of the software testing process (time and effort reduction).

1.2 Problem Background

Testing assumes an indispensable job in ensuring the quality and dependability of a product (Jena, Swain, & Mohapatra, 2014). As the trouble and extent of frameworks extend can be ascribed to changing client's necessities, extra time and exertion are expected to perform sufficient testing. This also complicates the system with short period of time and tension due to inconsistencies, inaccuracies and ambiguities.

In testing process, there are three areas, which are test case generation, execution, and appraisal. Generation of test case is the most troublesome development in testing as they choose the achievement of testing in making quality products that meet customer's desire.

Researchers have reliably created new techniques to address the difficulties of test case generation, intending to reduce testing effort and time. It is better to create test case at the design stage, which produces solid software (Jena et al., 2014). In view of this, MBT procedure has been introduced to help create value software. MBT includes an automatic generation of test case utilizing models from framework necessities (Cartaxo, Neto, & Machado, 2007).

Six MBT methods have been introduced by researchers to help create test case including Finite State Machine (FSM), Theorem Proving, Constraints Logic Programming and Symbolic Execution, Model Checking, Markov Chain, and Unified Modelling Language (UML). These days, model-based software development utilizing UML notations has attracted the attention of researchers,

whereby various researchers have started to separate valuable data from UML diagrams to create test case (Sawant & Shah, 2011).

Clearly, UML models assist developers with recognizing software structure and to find test data attributed to high-level abstraction models. Likewise, MBT methodology in creating test case enables testers to design testing at an early stage in SDLC and licenses parallel testing and coding (Kundu & Samanta, 2009). Moreover, when test case is delivered early, engineers are able to find abnormalities and ambiguities in requirement detail and design documents.

However, UML diagrams have a limitation mentioned by many authors (Hoseini & Jalili, 2014; Syafiqah et al., 2015) as it does not support a particular issue like looping, iteration activity in the software system. UML Sequence diagram as one of the commonly used diagrams to generate test cases has its limitations, as it needs a loop combination fragment to describe the looping process, and a combination fragment with the par operator is needed to show the parallel execution of the operation. Without using fragments, the sequence diagram will not support the looping activities such as demonstrated by (Sarma, 2007) in previous work.

Moreover, manual generation of test cases have limitation such as wasting time and effort, which is why nowadays automated test cases generation is highly ranked, as it provides an easy and efficient way for software testers to generate test cases from UML models (Schieferdecker, 2012). Testing plays an indispensable role in ensuring the quality and dependability of a product (Jena et al., 2014). As the trouble and extent of frameworks extend ascribed to changing client's necessities, extra time and exertion are expected to perform sufficient testing. This also complicates the system with short period of time and tension due to inconsistencies, inaccuracies and ambiguities.

As a result, various efforts have been made to automate the process, making it faster and more reliable, hence genesis of model-based testing (MBT). Numerous researchers have surveyed this field in depth with an intention of finding out exactly how the test data (input and output) is generated, executed and later evaluated against

any given system. Test data generation has proven the most challenging step that determines the correctness of the next phases.

1.3 Problem Statement

Against the above background, this research will emphasize the problem of how to generate test case from UML Sequence diagram to solve the looping limitation of considered UML Sequence diagrams in generating test cases using semi-automation test case generation method.

To tackle the aforementioned problem, this research addresses the following questions:

i. How can test cases be generated semi-automatically using UML Sequence diagram?
ii. How can test case generation be evaluated using coverage criteria?

1.4 Research aim and objectives

The aim of this research is to generate test case using UML Sequence diagram and to enhance the test cases generation of a given case study.
By addressing the problem and the questions based thereof, this research is targeted to achieve the following objectives:

i. Enhance test case generation techniques using proposed UML sequence diagram technique.
ii. Generate semi-automatic test cases from UML sequence diagram.
iii. Evaluate the coverage criteria of the test case generated.

1.5 Research Scope

This section elaborates the scopes of this research. Some of the research scopes are addressed in detail in later chapters.

i. Focuses on one of the black box testing techniques, which is MBT technique, where test cases are derived from UML diagrams that are used to model user's requirements.
ii. UML behavioural model sequence diagram have been implemented on looping and iteration cases to compare the effectiveness on generating test case.

1.6 Research contribution

This work will enhance the generation of test case from UML sequence diagram used in existing case study solving looping and iteration problems for providing better coverage. The enhanced test case generation technique will provide an easy and efficient way for software testers to generate test cases from UML sequence diagrams.

As a result, the knowledge gained from this research will benefit other researchers that are looking into exploring software testing area, particularly on test case generation using UML sequence diagrams. In addition, the enhanced methodology gives more test cases with same complexity in order to produce quality test cases and deliver the good software system to the user.

1.7 Research organization

In order to conduct a thorough study that emphasizes generating test case from UML Sequence diagram to solve the looping limitation of considered UML Sequence diagrams in generating test case and enhance a given case study as a primary objective, the structure of this thesis is comprised of 5 chapters as follows:

Chapter 1 enumerates the study overview and states the research problem, questions, objectives, significance, scope as well as the selected approach and design of the research.

REFERENCES

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), 7–11. https://doi.org/10.5120/ijais2016451599

Ard, J., Davidsen, K., & Hurst, T. (2014). Simulation-based embedded agile development. *IEEE Software*, *31*(2), 97–101. https://doi.org/10.1109/MS.2014.42

Arora, P. K., & Bhatia, R. (2018). Mobile agent-based regression test case generation using model and formal specifications. *IET Software*, *12*(1), 30–40. https://doi.org/10.1049/iet-sen.2016.0203

Boberg, J. (2008). Early fault detection with model-based testing. *Erlang'08: Proceedings of the 2008 SIGPLAN Erlang Workshop*, 9–20. https://doi.org/10.1145/1411273.1411276

Boghdady, Pakinam N., Badr, Nagwa L., Hashem, Mohamed, and Tolba, M. F. T. (2011). A Proposed Test Case Generation Technique Based on Activity Diagrams. *International Journal of Engineering & Technology*, *11*(3), 37–57.

Cartaxo, E. G., Neto, F. G. O., & Machado, P. D. L. (2007). Test case generation by means of UML sequence diagrams and labeled transition systems. *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, 1292–1297. https://doi.org/10.1109/ICSMC.2007.4414060

Chanda, C., Vivek, S., & Parminder, S. S. (2012). Test Case Generation based on Activity Diagram for Mobile Application. *International Journal of Computer Applications*, *57*(23), 975–8887. https://doi.org/10.5120/9436-3563

Dalai, S. (2011). *Test Case Generation For Concurrent Object-Oriented Systems Using Combinational Uml Models*. *3*(5), 97–102.

Dhineshkumar, M., & Galeebathullah. (2014). An approach to generate test cases from sequence diagram. *Proceedings - 2014 International Conference on Intelligent Computing Applications, ICICA 2014*, 345–349. https://doi.org/10.1109/ICICA.2014.77

Elallaoui, M., Nafil, K., & Touahni, R. (2017). Automatic generation of TestNG tests cases from UML sequence diagrams in Scrum process. *Colloquium in Information Science and Technology, CIST*, 65–70. https://doi.org/10.1109/CIST.2016.7804972

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. https://doi.org/10.1016/j.infsof.2014.12.005

Gutiérrez, J. J., Escalona, M. J., Mejías, M., & Torres, J. (2006). An approach to generate test cases from use cases. *Proceedings of the 6th International Conference on Web Engineering - ICWE '06*, 113. https://doi.org/10.1145/1145581.1145606

Hoseini, B., & Jalili, S. (2014). Automatic Test Path Generation from Sequence Diagram Using Genetic Algorithm. *2014 7th International Symposium on Telecommunications, IST 2014*, 106–111. https://doi.org/10.1109/ISTEL.2014.7000678

Jena, A. K., Swain, S. K., & Mohapatra, D. P. (2014). A novel approach for test case generation from UML activity diagram. *Issues and Challenges in Intelligent Computing Techniques (ICICT), 2014 International Conference On*, (April 2016), 621–629.

Khurana, N., Singh Chhillar, R., & 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*(3), 242–250. https://doi.org/10.17706/jsw.11.3.242-250

Kim, H., Kang, S., Baik, J., & Ko, I. (2007). Test cases generation from UML activity diagrams. *Proceedings - SNPD 2007: Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, *3*, 556–561. https://doi.org/10.1109/SNPD.2007.525

Kundu, D., & Samanta, D. (2009). A Novel Approach to Generate Test Cases from UML Activity Diagrams. *The Journal of Object Technology*, *8*(3), 65. https://doi.org/10.5381/jot.2009.8.3.a1

Li, L., Li, X., He, T., & Xiong, J. (2013). Extenics-based test case generation for UML activity diagram. *Procedia Computer Science*, *17*, 1186–1193. https://doi.org/10.1016/j.procs.2013.05.151

Li, V. (2014). *T he Research on Test Case Generation Technology ofUML Sequence Diagram*. (Iccse), 1067–1069.

Ma, C., & Provost, J. (2017). A model-based testing framework with reduced set of test cases for programmable controllers. *13th IEEE Conference on Automation Science and Engineering (CASE)*, 944–949.

Maheshwari, V., & Prasanna, M. (2015). Generation of test case using automation in software systems - A review. *Indian Journal of Science and Technology*, *8*(35). https://doi.org/10.17485/ijst/2015/v8i35/72881

Meiliana, Septian, I., Alianto, R. S., Daniel, & Gaol, F. L. (2017). Automated Test Case Generation from UML Activity Diagram and Sequence Diagram using Depth First Search Algorithm. *Procedia Computer Science*, *116*, 629–637. https://doi.org/10.1016/j.procs.2017.10.029

Nian-Fa, M. (2015). The Test System Design of Real-Time Embedded Software System. *2015 Seventh International Conference on Measuring Technology and*

Mechatronics Automation, 1321–1324. https://doi.org/10.1109/ICMTMA.2015.323

Peltola, J., Sierla, S., Aarnio, P., & Koskinen, K. (2013). Industrial evaluation of functional Model-Based Testing for process control applications using CAEX. *2013 IEEE 18th Conference on Emerging Technologies Factory Automation (ETFA), 62424*, 1–8. https://doi.org/10.1109/ETFA.2013.6647997

Qin, Y., & Xu, R. (2008). GSPN-based modeling and analysis for robotized assembly system. *2008 IEEE International Conference on Robotics and Biomimetics, ROBIO 2008*, 1070–1075. https://doi.org/10.1109/ROBIO.2009.4913149

Sarma, M. (2007). *Automatic Test Case Generation from UML Sequence Diagrams*. 60–65. https://doi.org/10.1109/ADCOM.2007.68

Sarma, M., & Mall, R. (2007a). Automatic test case generation from UML models. *Proceedings - 10th International Conference on Information Technology, ICIT 2007*, *49*, 196–201. https://doi.org/10.1109/ICOIT.2007.4418295

Sarma, M., & Mall, R. (2007b). Automatic Test Case Generation from UML Models. *10th International Conference on Information Technology (ICIT 2007)*, 196–201. https://doi.org/10.1109/ICIT.2007.26

Sawant, V., & Shah, K. (2011). Construction of test cases from UML models. *Communications in Computer and Information Science*, *145 CCIS*, 61–68. https://doi.org/10.1007/978-3-642-20209-4_9

Schieferdecker, I. (2012). Model-based fuzz testing. *Proceedings - IEEE 5th International Conference on Software Testing, Verification and Validation, ICST 2012*, 814. https://doi.org/10.1109/ICST.2012.180

Shirole, M., & Kumar, R. (2013). UML behavioral model based test case generation. *ACM SIGSOFT Software Engineering Notes*, *38*(4), 1. https://doi.org/10.1145/2492248.2492274

Syafiqah, N., Binti, Z., Rahman, A., Dayang, N. A., Azurati, N., & Salleh, A. (2015).

Jurnal Teknologi SELECTING UML MODELS FOR GENERATION OF TEST CASES : AN EXPERIMENTS OF TECHNIQUE TO GENERATE TEST CASES. *1*(1), 1–6.

Wang, C., & Liu, M. T. (1993). *Generating Test Cases for*. 774–781.

Wang, S. Y., Sun, J. Z., & Zhang, J. (2016). The method of generating web link security testing scenario based on UML diagram. *Proceedings - 15th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, 10th IEEE International Conference on Big Data Science and Engineering and 14th IEEE International Symposium on Parallel and Distributed Proce*, 1831–1838. https://doi.org/10.1109/TrustCom.2016.0281

Wang, X., Jiang, X., & Shi, H. (2015). Prioritization of test scenarios using hybrid genetic algorithm based on UML activity diagram. *Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS*, *2015-Novem*, 854–857. https://doi.org/10.1109/ICSESS.2015.7339189

Xing, Y., Gong, Y. Z., Wang, Y. W., & Zhang, X. Z. (2015). The application of iterative interval arithmetic in path-wise test data generation. *Engineering Applications of Artificial Intelligence*, *45*, 441–452. https://doi.org/10.1016/j.engappai.2015.07.021

Zhang, C., Duan, Z., Yu, B., Tian, C., & Ding, M. (2016). A test case generation approach based on sequence diagram and automata models. *Chinese Journal of Electronics*, *25*(2), 234–240. https://doi.org/10.1049/cje.2016.03.007