TEST CASE PRIORITIZATION TECHNIQUE USING SEQUENCE DIAGRAM AND LABELED TRANSITION SYSTEMS IN REGRESSION TESTING

NUR FATIMAH AS'SAHRA

UNIVERSITI TEKNOLOGI MALAYSIA

TEST CASE PRIORITIZATION TECHNIQUE USING SEQUENCE DIAGRAM AND LABELED TRANSITION SYSTEMS IN REGRESSION TESTING

NUR FATIMAH AS'SAHRA

A dissertation submitted in partial fulfilment of the
requirements for the award of the degree of
Master of Science (Computer Science)

Faculty of Computing
Universiti Teknologi Malaysia

JANUARY 2015

I strongly dedicated this dissertation to my beloved parents for their supports, encouragement and love.

ACKNOWLEDGEMENT

Alhamdulillah hirobbil ‘alamin, all praises to Allah SWT for giving me the strength and health also His blessing in order to complete my dissertation entitled Test Case Prioritization Technique using Sequence Diagram and Labeled Transition Systems in Regression Testing.

I owe my deepest gratitude to my supervisor **Assoc. Prof. Dr. Dayang Norhayati Abang Jawawi** for her advice, guidance and support during my study at Universiti Teknologi Malaysia. She inspired me greatly to work on this dissertation. I have learned a lot from her and I am fortunate to have her as my mentor and supervisor.

Besides, I would like to thank to my parents, Husin Syah and Sumiati. They are the source of my power and they never stop giving me spiritually and physically support and advice. Not forgotten, for my brother, Al-amin Mohammad Syah Reza, thank you for the support.

Last but not least, I would like to thank my dear friend Hosein Abedinpourshotorban for the advice, ideas and helps that are priceless to me.

ABSTRACT

Model-Based Testing (MBT) utilizes the models of software to generate the test cases. In line with this, Unified Modeling Language (UML) is widely adopted as a modeling support for MBT and UML sequence diagram is one of the most important diagram in the creation of test cases under MBT umbrella (aided by intermediate model). However, MBT method in general tends to generate a large amount of test cases. It is impractical in testing to execute all of the test cases, moreover if the size is large. Also, it has a greater impact on model-based regression testing. Regression testing is a testing process that is applied after software is modified. As a software evolves, some modifications or new features are added to the software. Thus, it also tends to increase the number of test cases. Retesting a large-size of test cases during regression testing stage is even harder since the allocated time and cost are more limited. In order to overcome this issue, a similarity-based selection technique based on Labeled Transition Systems (LTS) intermediate model is introduced. It will select only the subset of test cases that are less similar and has a larger coverage. Nevertheless, this technique still has a drawback. It does not consider the modified parts of the software while selecting the test cases for regression testing. Thus, this technique is against the goal of regression testing, which the test cases supposed to target the modified part of the software. Therefore, a test case prioritization technique is proposed. In the proposed technique, the generated test cases derived from sequence diagram and LTS intermediate model are prioritized for regression testing. Also, the evaluation of both techniques is done based on a set of two case studies. As a result, the proposed technique is able to overcome the issues of original technique by maximizing early coverage of the modified code in regression testing as well as achieve the early fault detection.

ABSTRAK

Ujian Berasaskan Model (MBT) menggunakan model perisian untuk menghasilkan kes-kes ujian. Selaras dengan ini, Bahasa Pemodelan (UML) secara meluas dipakai untuk pemodelan MBT dan UML rajah jujukan adalah salah satu gambarajah yang paling penting untuk menghasilkan kes ujian dibawah payung MBT (dibantu oleh model perantaraan). Namun, kaedah MBT umumnya cenderung menghasilkan kes ujian dalam jumlah yang besar. Pengujian ini tidak dipraktikkan untuk dilakukan pada semua kes ujian, apalagi sekiranya saiz adalah besar. Ia juga mempunyai impak yang besar pada model asas pengujian regrasi. Pengujian regrasi adalah proses pengujian yang digunakan selepas perisian diubahsuai. Beberapa pengubahsuaian atau ciri-ciri baru ditambah ke dalam perisian. Oleh itu, ia cenderung untuk meningkatkan bilangan kes ujian. Pengujian semula kes ujian bersaiz besar adalah lebih susah disebabkan masa dan kos yang diperuntukkan adalah terhad. Untuk mengatasi masalah ini, satu teknik pemilihan berdasarkan persamaan dalam model perantaraan Sistem Peralihan Lebel (LTS) diperkenalkan. Ia hanya memilih sebahagian kecil dari kes-kes ujian yang lebih kurang sama dan mempunyai liputan yang lebih besar. Namun, teknik ini masih mempunyai kelemahan. Ia tidak mengambil kira pengubahsuian bahagian perisian semasa memilih kes ujian untuk pengujian regrasi. Oleh itu, teknik ini adalah bertentangan dengan matlamat ujian regrasi, yang mana kes-kes ujian sepatutnya mensasarkan bahagian perisian yang diubahsuai. Oleh itu, keutamaan teknik kes ujian adalah dicadangkan. Kes-kes ujian diperolehi dari jujukan rajah dan pengantaraan model LTS untuk ujian regrasi. Penilaian kedua-dua teknik dilakukan berdasarkan kepada kedua kajian kes. Kesimpulannya, teknik yang dicadangkan mampu mengatasi isu teknik asal dengan memaksimumkan liputan awal dalam mengubahsuai kod dalam pengujian regrasi serta mencapai pengesanan kerosakan awal.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AG	-	Activity Graph
ATM	-	Automatic Teller Machines
BFS	-	Breadth-First Search
BN	-	Bayesian Networks
CCFG	-	Concurrent Control Flow Graph
CCFP	-	Concurrent Control Flow Path
CCG	-	Concurrent Composite Graph
DFS	-	Depth-First Search
EFSM	-	Extended Finite State Machine
EOSDG	-	Extended Object-oriented System Dependence Graph
GA	-	Genetic Algorithm
LIS	-	Library Information System
LTS	-	Labeled Transition Systems
MBT	-	Model-Based Testing
MLTS	-	Modified Labeled Transition System
OCL	-	Object Constraint Language
OMG	-	Object Management Group
PTS	-	Prioritized of Test Suite
SCG	-	Structured Composite Graph
TC	-	Test Case
TS	-	Test Suite
UDG	-	Use case Dependency Graph
UML	-	Unified Modeling Language

CHAPTER 1

INTRODUCTION

1.1 Overview

The number of people utilized computer was relatively small back in the 1970s. It is inversely proportional to the situation today. Now, people can hardly complete their tasks without utilizing computer. In line with this, software has emerged and touched millions of people, assisting them to complete their tasks effectively and efficiently (Lyu, 2007).

However, software is a complex product that is hard to develop and test. Very often, software exhibits unexpected and undesired behaviors that may even lead to severe problems and damages (Fuggetta, 2000). Moreover, the accuracy of software applications' functionality, performance and usability gives a crucial task in software quality (Qian and Zheng, 2009). Therefore, software testing is vital means of software quality assurance that validate whether software behaves as intended and identify potential malfunctions (Bertolino, 2007). Also, Parvathi and Jenila (2011) added that testing is essential to software development, and even more than 50% of total cost and time have been spent on testing.

In addition, there are various methods to test the software. Generally, testing methods are expressed as white-box testing or black-box testing (Panchapakesan *et al.*, 2013). Nevertheless, in recent years, Model-Based Testing (MBT) or grey-box testing has progressively attracted the intention from various fields, such as industry and academia (Utting and Legeard, 2010). It is an evolution of white-box and black-box testing (Repasi, 2009).

Even though MBT is a relatively new method, but it has already gained the popularity. For instance, industries are adopting this method because of its advantages over the development life cycle of the product. Since MBT associated with models, it is useful in generating test cases at an early stage of software development. Besides, it helps to reveal the faults in requirements faster (Boghdady *et al.*, 2011). Furthermore, MBT method can be used during development testing stage, regression testing stage or both. A main difference between regression and development testing stage is that during regression testing stage, an established test suite may be available for reuse (Rothermel and Harrold, 1997).

However, MBT method in general will derive huge number of test cases. It is impractical in testing to execute all of the test cases, moreover if the size of test suite is large (Cartaxo *et al.*, 2011). In line with this, it has a greater impact on model-based regression testing. Regression testing is defined as the testing process which will be applied after software is modified. As a software evolves, some modifications or new features are added to the software. Thus, it tends to increase the number of test cases in test suite. Retesting a large-size of test suite during regression testing stage is even harder since the allocated time and cost are more limited (Yoo and Harman, 2012). Therefore, many researches have been conducted in order to address the issue of test suite size during development or regression testing stage.

1.2 Problem Background

According to Myers *et al.* (2011), software testing becomes one of the most challenging issues in software engineering domain. It is due to the vast array of operating systems, programming languages, as well as hardware platforms that have evolved. Thereby, conducting an appropriate testing that meet the quality of the software product is essential. However, selecting the appropriate testing method is not an easy task. It needs to be identified correctly because each technique will lead into different quality aspects of software (Luo, 2001).

As mentioned earlier, MBT is one of the testing methods that have already gained the popularity among the researchers and industries. It utilizes the models of software in order to derive the test cases (Anand *et al.*, 2013). The Unified Modeling Language (UML) or Non-UML models can be used as long as it depicts the behavior of software. However, UML has become widely adopted as a modeling support for MBT (Gross, 2005). For UML-based testing, the testing approaches are classified based on types of diagram. These diagrams then need to be converted into intermediate model in order to generate test cases. The reason is that even though UML diagrams are useful in depicting graphical representations of the system specifications, but it is hard to be used directly to generate test cases (Shirole and Kumar, 2013).

According to Kim *et al.* (2007), the most used UML diagram in MBT are sequence diagram, activity diagram, state diagram, class diagram, and use case diagram. Among these diagrams, the performance of the sequence diagram outperforms the others. For instance, it is very useful for visualizing the way several objects collaborate to get a job done. In addition, the sequence diagram (aided by intermediate model) is one of the most important UML models in the creation of test cases under MBT umbrella (Samuel *et al.,* 2007). Furthermore, Shirole and Kumar (2013) stated that the behavioral model like sequence a diagram has already become an extensive research and used by many researchers so that the generation of test cases can be conducted more effectively.

One of the existing works that utilize sequence diagram to generate test cases is proposed by Cartaxo *et al.* (2007). This work focused on generating functional test cases for feature testing of mobile-based applications. In line with this, procedural of this work is defined by converting sequence diagram into intermediate model namely Labeled Transition Systems (LTS). All paths from LTS later will be used to generate test cases.

Hence, it is essential to identify each path properly. As a result, this work can reduce the cost and time of testing as well as achieve the full coverage criteria.

Similarly, Khandai *et al.* (2011) proposed work that utilize sequence diagram in order to generate test cases, especially to be used in concurrent systems. Thus, sequence diagram is converted into Concurrent Composite Graph (CCG) intermediate model. Then, both Breadth-First Search (BFS) and Depth First Search (DFS) traversal techniques are used to traverse the paths in CCG using message sequence path criteria. In short, this work is proven to control the test case explosion problem in concurrent systems effectively.

On the other hand, Swain *et al.* (2010) proposed work that utilizes both sequence and use case diagram to generate test cases for integration as well as system testing. Thus, it begins with constructing the Use case Dependency Graph (UDG) from use case diagram. Subsequently, sequence diagram is converted to Concurrent Control Flow Graph (CCFG). Lastly, test cases are generated based on the information gathered from previous steps. As a result, this work can achieve the full predicate coverage criteria.

Even though various works proposed by researchers aimed to aid the test case generation activities, but some limitations and challenges still exist in the existing works. The major issue in MBT is the number of generated test cases in test suite is huge. It is impractical to execute all of the test cases since the time and cost of testing are restricted. In line with this, it has a greater impact on model-based regression testing as well since the allocated time and cost to retest the test cases are even more limited.

Regression testing is a testing process which is applied after software is modified. Usually, it is considered as a special type of testing activity (Yoo and Runeson, 2014). A main difference between regression and development testing is that during regression testing stage, an established test suite may be available for reuse (Rothermel and Harrold, 1997). Furthermore, it is conducted during software maintenance phase with the purpose to validate the modifications introduced in software (Korel and Koutsogiannakis, 2009). However, this gigantic task involves an infeasible number of test cases. As a software evolves, some modifications or new features are added to the software. Thus, it tends to increase the number of test cases in test suite. Similar to the testing conducted during development testing stage, it is almost impossible to re-execute the entire test cases in a test suite during regression testing, moreover if the size of test suite is large. Retesting the large size of test suite during regression testing is very expensive and time-consuming.

Therefore, an additional effort in addressing the test suite size problem is required in order to improve software testing activity during development and regression testing stage. A number of different techniques have been studied to address the test suite size problem, which are test case selection, test case prioritization, and test case reduction technique. Test case selection seeks to identify the test cases that are relevant to some set of recent changes, while test case prioritization seeks to order test cases in such a way that early fault detection is maximized. Finally, test case reduction seeks to eliminate redundant test cases in order to reduce the number of test cases to run (Yoo and Harman, 2012). These three techniques aim to keep the size of test cases in an intended size.

The researchers have proposed several works in this area. For instance, one of the works on test case selection technique is proposed by Cartaxo *et al.* (2011). The strategy presented in the work of Cartaxo *et al.* (2011) is actually originated from Cartaxo *et al.* (2007) with preliminary experiments emphasizing on the use of sequence diagram and LTS intermediate model in order to generate test cases. However, the problem exists while applying this technique for bigger features with bigger LTS where the set of test cases will be greater and application functionalities test coverage is hard to be achieved.

Therefore, Cartaxo *et al.* (2011) introduced a similarity-based selections technique in MBT with an aim to overcome the test suite size problem that arises in its preliminary work. It begins with utilizing the LTS intermediate model to obtain the test cases. Then, the similarity-based selections technique will select the subset of test cases that are less similar (the most different ones) in test suite based on a given coverage criteria. Thus, it reduces the number of test cases to be executed.

On the other hand, as for the test case prioritization technique, Korel *et al.* (2007) proposed one work called model-based test prioritization technique. This work integrated both the original and modified system models together with information gathered during execution of the modified model. Subsequently, all of this information is used to prioritize test cases for retesting the modified software system. As a result, the goal of early fault detection in the modified software is achieved.

Likewise, Panigrahi and Mall (2010) have presented work on model-based regression test prioritization technique (M-RTP) in order to increase the rate of fault detection. However, the main focus of this work is on the object-oriented programs. The technique involves constructing a graph model of the source code to represent control and data dependences, as well as static object relations such as inheritance, aggregation, and association. When a change occurs in a program, both the original and modified programs are compared by a code differencer to find the modified statements. The identified changes are marked on the model. In line with this, a model namely Extended Object-oriented System Dependence Graph (EOSDG) is introduced and followed by the construction of forward and backward slice of the model. Lastly, a test case, that covers a maximum number of affected model elements, will be assigned as a higher priority.

In addition, Mirarab and Tahvildari (2007) have also presented a work to prioritize test cases in order to increase the rate of fault detection during regression testing stage. However, this work is based on the probability theory that concerns on the probabilistic specification of the problem. Thus, the test cases will be prioritized by using a special type of probabilistic graphical models, namely Bayesian Networks (BN). It is used to integrate source code changes, software fault-proneness, and test coverage data into a unified model. Subsequently, the prioritization technique will order the test cases based on their success probability.

Another interesting technique has also been proposed by Ma *et al.* (2005). This work integrates Genetic Algorithm (GA) and test case reduction technique. Moreover, this work is focused on code-based of regression testing. With an aim to reduce the cost of regression testing, this technique will create a population according to test history. Then, it computes the fitness value using cost and coverage information. Finally, breed the successive generations using GA. This process will keep on repeating to find minimized test cases. Thus, it helps to reduce the size of test cases, reduce the cost of regression testing, and attain good cost-effectiveness. However, to the best of knowledge gathered in this research, there is only code-based regression testing for test case reduction techniques. There is no work has been found which use the test case reduction technique in MBT.

As a conclusion, even though various works, that address the test suite size problem in MBT (either used during development or regression testing stage), have been proposed by the researchers, some improvements are still needed in order to overcome the limitations or challenges faced in existing works. It implies that it is still an open issue and demand for more researches conducted in this area is needed.

1.3 Problem Statement

As a part of MBT method, the role of regression testing cannot simply be ignored. Even though regression testing is considered as an expensive testing, but it holds a very important role to validate that the modifications introduced in software are correct and do not adversely affects the unchanged portion of the software (Elbaum *et al.*, 2002). Generally, MBT method is known to simply generate a large amount of test cases even from small models (Grieskamp, 2006). It is impractical in testing to execute or re-execute all of the test cases in the test suite, moreover if the size of test suite is large.

Therefore, various works that address the test suite size problem in MBT (either used during development or regression testing stage) have been proposed. However, the work presented by Cartaxo *et al.* (2011) still has a drawback. The similarity-based selection technique in this work does not consider the modified parts of software in selecting test cases for regression testing. Hence, this technique is unable to be used for regression testing stage. This approach only focused on selecting the test cases based on path coverage percentage criteria with similarity-based selection function. It means that it will only select the test cases that has more coverage and less similar (the most different ones) among each other. Therefore, this technique is against the goal of regression testing, which the test cases should target the modified part of the software. In addition, this selection technique is the only technique to address the test suite size problem of generated test cases based on LTS intermediate model.

Therefore, the research question posed is *"How to prioritize the test cases for regression testing to achieve the goal of early fault detection based on MBT?"*

1.4 Research Aim and Objectives

The aim of this research is to propose a model-based test case prioritization technique for regression testing in order to achieve early fault detection so that it is able to address the shortcoming of test case selection technique for regression testing in the work of Cartaxo *et al.* (2011).

Furthermore, this research consists of a set of objectives that lead to the research process as follows:

- To identify the issue of test case selection technique for regression testing.
- To propose a test case prioritization technique for regression testing in order to achieve early fault detection by using sequence diagram and labeled transition systems.
- To compare the result of the proposed test case prioritization technique with the original test case selection technique to support regression testing.

1.5 Scope of Study

In this research, the boundary of the research is defined. The following are significant.

- This research focused on generated the test cases in MBT using UML sequence diagrams and LTS intermediate model, leaving out the rest of the other models.
- This research focused on test case selection and test case prioritization techniques in order to address the test suite size problem for regression testing.
- This research focused on comparing the original test case selection technique with the proposed test case prioritization technique.

1.6 Significance of Study

As early fault detection can provide faster feedback on the system under regression test and let software engineers begin locating and correcting faults earlier, then it helps the software industries to reduce the required resources, such as cost, time, and effort while conducting the testing activity (Elbaum *et al.*, 2000).

Therefore, this research supports the software industries by achieving early fault detection. Moreover, it is beneficial to the researchers, those who are interested in carrying out the study in this area.

1.7 Dissertation Organization

This research is made up of six chapters. In Chapter 1, it discusses on the research introduction, problem background, problem statement, aim, objectives, scope and significance of the study. Similarly, Chapter 2 presents the overview of software testing, further discussion on MBT, UML sequence diagram, LTS intermediate model, techniques for addressing test suite size problem, as well as regression testing. In addition, the literature review of the current works has been discussed as well. In Chapter 3, the research methodology is explained in sequence of phases. Moreover, Chapter 4 presents the proposed test case prioritization technique for regression testing. Furthermore, a comparison between the original and the proposed technique have been illustrated in Chapter 5. However, Chapter 6 presents the study contributions and future work.

REFERENCES

Anand, S., Burke, E. K., Chen, T. Y., Clark, J., Cohen, M. B., Grieskamp, W., et al. 2013. An orchestrated survey of methodologies for automated software test case generation. *Journal of Systems and Software, 86*(8), 1978-2001.

Baker, P., Dai, Z. R., Grabowski, J., Haugen, Ø., Schieferdecker, I., and Williams, C. 2008. *Model-driven testing*: Springer.

Bertolino, A. 2007. *Software testing research: Achievements, challenges, dreams.* 2007 Future of Software Engineering, 85-103.

Binder, R. 2000. *Testing object-oriented systems: models, patterns, and tools*: Addison-Wesley Professional.

Boberg, J. 2008. *Early fault detection with model-based testing.* Proceedings of the 7th ACM SIGPLAN workshop on ERLANG, 9-20.

Boghdady, P. N., Badr, N. L., Hashim, M. A., and Tolba, M. F. 2011. *An enhanced test case generation technique based on activity diagrams.* Computer Engineering & Systems (ICCES), 2011 International Conference on, 289-294.

Bouquet, F., Grandpierre, C., Legeard, B., Peureux, F., Vacelet, N., and Utting, M. 2007. *A subset of precise UML for model-based testing.* Proceedings of the 3rd international workshop on Advances in model-based testing, 95-104.

Cartaxo, E. G., Machado, P. D., and Neto, F. G. O. 2011. On the use of a similarity function for test case selection in the context of model-based testing. *Software Testing, Verification and Reliability, 21*(2), 75-100.

Cartaxo, E. G., Neto, F. G. O., and Machado, P. D. 2007. *Test case generation by means of UML sequence diagrams and labeled transition systems.* SMC, 1292-1297.

Chen, T. Y., and Lau, M. F. 1998. A new heuristic for test suite reduction. *Information and software technology, 40*(5), 347-354.

Chen, Y., Probert, R. L., and Ural, H. 2009. Regression test suite reduction based on SDL models of system requirements. *Journal of Software Maintenance and Evolution: Research and Practice, 21*(6), 379-405.

de Vries, R. G., and Tretmans, J. 2000. On-the-fly conformance testing using SPIN. *International Journal on Software Tools for Technology Transfer, 2*(4), 382-393.

Dhir, S. 2012. Impact of UML Techniques in Test Case Generation. *International Journal of Engineering Science and Advanced Technology, 2*(2), 214-217.

Dias Neto, A. C., Subramanyan, R., Vieira, M., and Travassos, G. H. 2007. *A survey on model-based testing approaches: a systematic review.* 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, 31-36.

Elbaum, S., Malishevsky, A. G., and Rothermel, G. 2000. *Prioritizing test cases for regression testing* (Vol. 25): ACM.

Elbaum, S., Malishevsky, A. G., and Rothermel, G. 2002. Test case prioritization: A family of empirical studies. *Software Engineering, IEEE Transactions on, 28*(2), 159-182.

Fenton, N. E., and Neil, M. 2000. *Software metrics: roadmap.* Proceedings of the Conference on the Future of Software Engineering, 357-370.

Fraikin, F., and Leonhardt, T. 2002. *SeDiTeC-testing based on sequence diagrams.* Automated Software Engineering, 2002. Proceedings. ASE 2002. 17th IEEE International Conference on, 261-266.

Fuggetta, A. 2000. *Software process: a roadmap.* Proceedings of the Conference on the Future of Software Engineering, 25-34.

Grieskamp, W. 2006. Multi-paradigmatic model-based testing. In *Formal Approaches to Software Testing and Runtime Verification* (pp. 1-19): Springer.

Gross, H.-G. 2005. *Component-based software testing with UML* (Vol. 44): Springer.

Hemmati, H., Briand, L., Arcuri, A., and Ali, S. 2010. *An enhanced test case selection approach for model-based testing: an industrial case study.* Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering, 267-276.

Herzner, W., Schlick, R., Ait Austrian, H. B., and Wiessalla, J. 2011. Towards Fault-based Generation of Test Cases for Dependable Embedded Software.

Jard, C., and Jéron, T. 2005. TGV: theory, principles and algorithms. *International Journal on Software Tools for Technology Transfer, 7*(4), 297-315.

Juristo, N., Moreno, A. M., and Strigel, W. 2006. Software testing practices in industry. *IEEE software, 23*(4), 19-21.

Khandai, M., Acharya, A. A., and Mohapatra, D. P. 2011. *A novel approach of test case generation for concurrent systems using UML Sequence Diagram.* Electronics Computer Technology (ICECT), 2011 3rd International Conference on, 157-161.

Kim, H., Kang, S., Baik, J., and Ko, I. 2007. *Test cases generation from UML activity diagrams.* Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007. SNPD 2007. Eighth ACIS International Conference on, 556-561.

Kim, J.-M., and Porter, A. 2002. *A history-based test prioritization technique for regression testing in resource constrained environments.* Software Engineering, 2002. ICSE 2002. Proceedings of the 24rd International Conference on, 119-129.

Korel, B., and Koutsogiannakis, G. 2009. *Experimental comparison of code-based and model-based test prioritization.* Software Testing, Verification and Validation Workshops, 2009. ICSTW'09. International Conference on, 77-84.

Korel, B., Koutsogiannakis, G., and Tahat, L. H. 2007. *Model-based test prioritization heuristic methods and their evaluation.* Proceedings of the 3rd international workshop on Advances in model-based testing, 34-43.

Korel, B., Koutsogiannakis, G., and Tahat, L. H. 2008. *Application of system models in regression test suite prioritization.* Software Maintenance, 2008. ICSM 2008. IEEE International Conference on, 247-256.

Korel, B., Tahat, L. H., and Harman, M. 2005, 26-29 Sept. 2005. *Test prioritization using system models.* Software Maintenance, 2005. ICSM'05. Proceedings of the 21st IEEE International Conference on, 559-568.

Leung, H. K., and White, L. 1989. *Insights into regression testing [software testing].* Software Maintenance, 1989., Proceedings., Conference on, 60-69.

Li, B.-l., Li, Z.-s., Qing, L., and Chen, Y.-H. 2007, 15-19 Dec. 2007. *Test Case Automate Generation from UML Sequence Diagram and OCL Expression.* Computational Intelligence and Security, 2007 International Conference on, 1048-1052.

Luo, L. 2001. Software testing techniques. *Institute for software research international Carnegie mellon university Pittsburgh, PA, 15232*(1-19), 19.

Lyu, M. R. 2007. *Software Reliability Engineering: A Roadmap*. 2007 Future of Software Engineering.

Ma, X.-y., Sheng, B.-k., and Ye, C.-q. 2005. Test-Suite reduction using genetic algorithm. In *Advanced Parallel Processing Technologies* (pp. 253-262): Springer.

Martin, S., Bleck, R., Dislis, C., and Farren, D. 1999, 1999. *The evolution of a system test process [for Motorola GSM products].* Test Conference, 1999. Proceedings. International, 680-688.

Mirarab, S., and Tahvildari, L. 2007. A Prioritization Approach for Software Test Cases Based on Bayesian Networks. In M. Dwyer and A. Lopes (Eds.), *Fundamental Approaches to Software Engineering* (Vol. 4422, pp. 276-290): Springer Berlin Heidelberg.

Myers, G. J., Sandler, C., and Badgett, T. 2011. *The art of software testing*: John Wiley & Sons.

Naslavsky, L., Ziv, H., and Richardson, D. J. 2009, 20-26 Sept. 2009. *A model-based regression test selection technique.* Software Maintenance, 2009. ICSM 2009. IEEE International Conference on, 515-518.

Nayak, A., and Samanta, D. 2010. Automatic test data synthesis using uml sequence diagrams. *journal of Object Technology, 9*(2), 75-104.

Nielsen, B. 2014. *Towards a Method for Combined Model-based Testing and Analysis.* Proceedings of the 2nd International Conference on Model-Driven Engineering and Software Development.

Nirpal, P. B., and Kale, K. 2011. A Brief Overview Of Software Testing Metrics. *International Journal on Computer Science & Engineering, 3*(1).

Panchapakesan, A., Abielmona, R., and Petriu, E. 2013. *Dynamic white-box software testing using a recursive hybrid evolutionary strategy/genetic algorithm.* Evolutionary Computation (CEC), 2013 IEEE Congress on, 2525-2532.

Panigrahi, C. R., and Mall, R. 2010. Model-based regression test case prioritization. *ACM SIGSOFT Software Engineering Notes, 35*(6), 1-7.

Parvathi, M., and Jenila, A. 2011. *A survey of software testing in refactoring based software models.* Nanoscience, Engineering and Technology (ICONSET), 2011 International Conference on, 571-573.

Petre, M. 2013. *UML in practice.* Proceedings of the 2013 International Conference on Software Engineering, 722-731.

Qian, H.-m., and Zheng, C. 2009. *A Embedded Software Testing Process Model.* Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on, 1-5.

Repasi, T. 2009. *Software testing-state of the art and current research challanges.* Applied Computational Intelligence and Informatics, 2009. SACI'09. 5th International Symposium on, 47-50.

Rodrigues Barbosa, J., Eduardo Delamaro, M., Carlos Maldonado, J., and Marcelo Rizzo Vincenzi, A. 2011. *Software Testing in Critical Embedded Systems: a Systematic Review of Adherence to the DO-178B Standard.* VALID 2011, The Third International Conference on Advances in System Testing and Validation Lifecycle, 126-130.

Rothermel, G., and Harrold, M. J. 1997. A safe, efficient regression test selection technique. *ACM Transactions on Software Engineering and Methodology (TOSEM), 6*(2), 173-210.

Rothermel, G., Untch, R. H., Chu, C., and Harrold, M. J. 1999. *Test case prioritization: An empirical study.* Software Maintenance, 1999.(ICSM'99) Proceedings. IEEE International Conference on, 179-188.

Sabharwal, S., Singh, S. K., Sabharwal, D., and Gabrani, A. 2010. *An event-based approach to generate test scenarios.* Computer and Communication Technology (ICCCT), 2010 International Conference on, 551-556.

Samuel, P., Mall, R., and Kanth, P. 2007. Automatic test case generation from UML communication diagrams. *Information and software technology, 49*(2), 158-171.

Sawant, V., and Shah, K. 2011. Construction of Test Cases from UML Models. In K. Shah, V. R. Lakshmi Gorty and A. Phirke (Eds.), *Technology Systems and Management* (Vol. 145, pp. 61-68): Springer Berlin Heidelberg.

Schieferdecker, I. 2012. Model-Based Testing. *IEEE software, 29*(1).

Shirole, M., and Kumar, R. 2013. UML behavioral model based test case generation: a survey. *SIGSOFT Softw. Eng. Notes, 38*(4), 1-13.

Singhal, A., Bansal, A., and Kumar, A. 2013. A critical review of various testing techniques in aspect-oriented software systems. *ACM SIGSOFT Software Engineering Notes, 38*(4), 1-9.

Swain, S. K., Mohapatra, D. P., and Mall, R. 2010a. Test case generation based on use case and sequence diagram. *International Journal of Software Engineering, IJSE, 3*(2), 21-52.

Swain, S. K., Pani, S. K., and Mohapatra, D. P. 2010b. Model Based Object-Oriented Software Testing. *Journal of Theoretical & Applied Information Technology, 14*.

Tassey, G. 2002. The economic impacts of inadequate infrastructure for software testing. *National Institute of Standards and Technology, RTI Project, 7007*(011).

Tretmans, J. 2008. Model based testing with labelled transition systems. In *Formal methods and testing* (pp. 1-38): Springer.

Tretmans, J. 2011. Model-based testing and some steps towards test-based modelling. In *Formal Methods for Eternal Networked Software Systems* (pp. 297-326): Springer.

Tsui, F. F. 2013. *Essentials of software engineering*: Jones & Bartlett Publishers.

Tucker, A. B. 2004. *Computer science handbook*: CRC press.

Utting, M. 2005. Position paper: Model-based testing. *Verified Software: Theories, Tools, Experiments. ETH Zürich, IFIP WG, 2*.

Utting, M., and Legeard, B. 2010. *Practical model-based testing: a tools approach*: Morgan Kaufmann.

Utting, M., Pretschner, A., and Legeard, B. 2012. A taxonomy of model-based testing approaches. *Software Testing, Verification and Reliability, 22*(5), 297-312.

Walcott, K. R., Soffa, M. L., Kapfhammer, G. M., and Roos, R. S. 2006. *Timeaware test suite prioritization.* Proceedings of the 2006 international symposium on Software testing and analysis, 1-12.

Wong, W. E., Horgan, J. R., London, S., and Agrawal, H. 1997. *A study of effective regression testing in practice.* Software Reliability Engineering, 1997. Proceedings., The Eighth International Symposium on, 264-274.

Yoo, S., and Harman, M. 2012. Regression testing minimization, selection and prioritization: a survey. *Software Testing, Verification and Reliability, 22*(2), 67-120.

Yoo, S., and Runeson, P. 2014. Guest editorial: special section on regression testing. *Software Quality Journal, 22*(4), 699-699.

Zander-Nowicka, J., Pérez, A. M., Schieferdecker, I., and Du Dai, Z. 2007. *Test design patterns for embedded systems.* 10th International Conference on Quality Engineering in Software Technology, 183-200.