AUTOMATIC MODEL-BASED TEST CASE GENERATION FOR UML DIAGRAMS USING TREE TRAVERSAL ALGORITHM

OLUWAGBEMI OLUWATOLANI

UNIVERSITY TEKNOLOGI MALAYSIA

AUTOMATIC MODEL-BASED TEST CASE GENERATION FOR UML DIAGRAMS USING TREE TRAVERSAL ALGORITHM

OLUWAGBEMI OLUWATOLANI

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

Faculty of Computing
Universiti Teknologi Malaysia

APRIL 2016

Specially dedicated to *God Almighty*

The supplier of strength and grace.

I love you Lord!!!

ACKNOWLEDGEMENT

First and foremost, I thank God Almighty for giving me the required grace and strength to carry out this research work; without Him, it would have been impossible to achieve this task.

Secondly, I thank my supervisor; Dr Hishammuddin Asmuni whose constant guidance and suggestions have led to the completion of this thesis. Your suggestions were valuable and your guidance inestimable. I wish you long life and good health as you strive to be an expert of high reckoning in this field of computing.

I am highly indebted to my lovely husband who baptized me into academics and provided the necessary support and congenial atmosphere to achieve my academic dreams. Furthermore, I thank my dear daughters Success and Peace Achimugu who lost a lot of maternal cares during the time of undertaking this research. Your patience immensely contributed to the timely completion of this thesis and my love for you is immeasurable.

I must appreciate the kindness of Lead City University Ibadan, Nigeria where I work for releasing me to proceed on study leave with pay. I specifically, thank the Chairman, Governing Council, Vice Chancellor and Registrar for their support throughout the study years.

I am indeed grateful to my parents through whose biological union brought me to this world. They had suffered over the years to ensure that my future was never thwarted. My siblings are worthy of my thanks for the love and concern we all share with each other. I am also grateful to my parent and siblings in-law for their love, support and prayers.

ABSTRACT

The foundation of any model-based testing (MBT) with Unified Modelling Language (UML) diagrams is test case generation (TCG) which predicts the expected functionalities of a system under test (SUT). However, problems associated with existing test case generation methods are lack of integration with various UML diagrams and tools, inability to cover all the model elements of UML diagrams, failure to generate comprehensive test cases based on adequate coverage criteria and lack of support tools for automatic generation of test cases. To address these challenges, efficient mapping strategies for model elements that engenders effective artefacts extraction and test case generation processes were proposed. The methodology employed in this research comprised constructing relevant models and algorithms as well as implementing with the use of Java programming language. Specifically, an enhanced elements mapper, artefacts extractor (parser) and test case generator were developed and integrated to produce the support tool. The elements mapper yielded an accuracy result of 99.31%. The artefacts extractor recorded 99.64% accuracy while the test case generator recorded 100% accuracy. The improved methods proved to be more robust and efficiently generated quality test cases with eliminated redundancies based on all the descriptive attributes of UML diagrams. Limitations of existing the methods were addressed in the proposed method which is able to integrate more diagrams to generate quality test cases.

ABSTRAK

Teras ujian berasaskan model (MBT) dengan gambar rajah Bahasa Pemodelan Bersatu (UML) merupakan penjanaan kes ujian (TCG) yang meramalkan fungsi jangkaan sistem di bawah ujian (SUT).Walau bagaimanapun, masalah yang berkaitan dengan kaedah penjanaan kes ujian yang sedia ada adalah kurangnya integrasi dengan pelbagai gambar rajah UML dan perkakasan, ketidakupayaan meliputi kesemua unsur model gambar rajah UML, kegagalan untuk menjana kes ujian yang komprehensif berdasarkan kriteria liputan yang memadai dan kekurangan perkakasan sokongan bagi penjanaan kes ujian automatik. Bagi menangani cabaran tersebut, strategi pemetaan yang cekap bagi unsur model yang diwujudkan oleh pengekstrakan artifak berkesan dan proses penjanaan kes ujian telah dicadangkan. Kaedah yang digunakan dalam kajian ini terdiri daripada pembinaan model yang sesuai dan algoritma serta melaksanakannya dengan menggunakan bahasa pengaturcaraan Java secara khusus, pemeta elemen yang dipertingkatkan, pengekstrak artifak (penghurai) dan penjana kes ujian telah dibangunkan serta bersepadu untuk menghasilkan perkakasan sokongan. Pemeta elemen menunjukkan ketepatan hasil kajian sebanyak 99.31%. Pengekstrak artifak mencatatkan ketepatan 99.64%, manakala penjana kes ujian mencatatkan ketepatan 100%. Kaedah yang dipertingkatkan ini terbukti lebih mantap dan secara cekap menjana kes ujian yang berkualiti dengan menghapuskan pertindihan berdasarkan semua sifat deskriptif gambar rajah UML. Had kaedah sedia ada dapat ditangani melalui kaedah yang dicadangkan yang mampu untuk menyepadukan lebih banyak gambar rajah untuk menjana kes ujian yang berkualiti.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AC	-	Activity Converter
ASSIST	-	Automatic State tranSItion teSTer
ATCGSGA	-	Activity Test Case Generation with Simple Genetic Algorithm
ATCUM	-	Automatic Test Case UML Model
AT	-	Artefacts Transformer
ATM	-	Automated Teller machine
BFS	-	Breadth-First-Search
CASE	-	Computer Aided Software Engineering
COM	-	Component Object Model
CORBA	-	Common Object Request Broker Architecture
DCOM	-	Distributed Component Object Model
DFT	-	Dependency Flow Tree
DFS	-	Depth-First-Search
EFSMs	-	Extended Finite State M
EVAG	-	Extended Variable Assignment Graph
FSM	-	Finite State Machine
GA	-	Genetic Algorithm
IBM	-	International Business Machines
IST	-	Invocation Sequence Tree
ITM	-	Intermediate Testable Model
IBT	-	Implementation-based Testing
IMR	-	Internal Model Representation
LHS	-	Left Hand Side
MBT	-	Model-Based Testing
MDA	-	Model Driven Architecture
MDD	-	Model Driven Design
MDE	-	Model Driven Environment

MOF	-	Meta Object Facility
M2T	-	Model-to-Text Transformation
MFG	-	Model Flow Graph
OMG	-	Object Management Group
OCL	-	Object Constraints Language
OOD	-	Object Oriented Design
PIM	-	Platform Independent Model
QoS	-	Quality of Service
QML	-	Qtronic Modeling Language
RHS	-	Right Hand Side
RUCM	-	Restricted Use Case Modeling
SBT	-	Specification-based Testing
SAD	-	State-Activity Diagram
SCOTEM	-	State Collaboration Test Model
SDLC	-	System Development Life Cycle
SeDiTeC	-	Testing Based on Sequence Diagram
SUT	-	System under Test
SVM	-	Support Vector Machine
TCG	-	Test Case Generation
TESTOR	-	Test Sequence Generator
TFG	-	Testing flow graph
TSGen	-	Test Scenario Generator
TDE	-	Transparent Data Encryption
TnT	-	Touch and Test
TECS	-	Test Environment for Complex Systems
UML	-	Unified Modeling Language
UBT	-	UML-Based Testing
UBTCG	-	UML Based Test Case Generator
UTG	-	UML Behavioral Test case Generator
UMLTGF	-	UML Test Generation Function
URI	-	Universal Resource Indicator
VAG	-	Variable Assignment Graph
XMI	-	XML Metadata Interchange
XML	-	Xtensible Mark-up Language

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Introduction

The fundamental activity of any black box testing is test case generation (Ingle and Mahamune 2015). This type of testing is designed based on the requirements specified and modelled for the software under test (SUT). Testing based on design models have the advantage that, test cases remain valid even when codes slightly changes (Kyaw and Min 2015). Design models are used as a basis for test case generation (Shanthi and Mohan Kumar 2012) and the technical name for this testing technique is known as Model Based Testing (MBT). However, the focus of this research is UML-based testing which is a sub set of MBT. It utilizes only UML diagrams for test case generation (Machado & Sampaio 2010).

UML based testing (UBT) consist of 3 flows of procedural events which include: (i) the UML diagram used in modelling user's requirements (ii) the parser required in extracting artefacts from model files of UML diagrams and (iii) a test case generation algorithm. The essence of creating models in UBT is to aid precise and comprehensive description of user's requirements (Sawant and Shah 2011; Wehrmeister and Berkenbrock 2013). Parser aids the extraction of artefacts from the model files of UML diagrams which could be in XMI or .MDL format and stored in a tree or graph (Sawant and Shah 2012, Li et al. 2013) while test case generation process consist of algorithms that traverses the storage mediums to generate test cases (Priya and Sheba 2013).

The rest of the chapter is organized as follows: Section 1.2 discusses the motivation for undertaking this research work. Section 1.3 articulates the research problem statements while Section 1.4 deals with the research objectives. The scope of the study is presented in Section 1.5 while Section 1.6 provides the thesis structure.

1.2 Motivation

The complexities associated with testing have led to the need for automatic generation of test cases. This is because, user's requirements are becoming larger and organizations are demanding for robust systems that can serve the needs of their customers irrespective of their geographical locations. Therefore, testing a fully implemented system with large requirements manually, can prove to be a difficult task (Jena et al. 2014). With the constant increase in system sizes, the concept of automatic generation of test cases is attracting serious research attention (Ingle and Mahamune, 2015). Correctly generated test cases may not only detect errors in a software system, but also minimizes the high cost and time associated with software testing process (Kyaw and Min 2015). Furthermore, conducting testing from UML diagrams have a major advantage; that is, testing can be initiated as soon as the requirements/design documents becomes available; thus, saving time, cost, and detecting errors early during the development span (Kyaw and Min 2015; Schweighofer and Heričko 2014; Kulkarni and Joglekar 2014).

With these motivations, improved methods for test case generation based on UML diagrams, considering adequate coverage criteria is proposed. Therefore, each output of a coding exercise can be compared to the generated test cases in order to determine whether the system under development is behaving as expected or not.

1.3 Statements of the Problem

The study was conducted in the area of test case generation with particular emphasis on UML diagrams driven by the problems arising from existing methods such as lack of integration with various UML diagrams and tools, inability to cover all model elements of UML diagrams, failure to generate comprehensive test cases based on adequate coverage criteria and lack of support tools for automatic generation of test cases. In this study, the problems of existing test case generation methods were addressed to ultimately provide mechanism of mapping generated test cases to the modelled requirements so as to verify the correctness of the SUT. Subsequently, the main research question of the study is:

How can test cases be systematically generated from UML diagrams?

Figure 1.1 shows the four basic problems that was addressed by this research. The first problem has to do with lack of support for integrated generation of test cases from various UML diagrams. This is very crucial because, requirements could be modelled in structural or behavioural diagrams or both. Therefore, there is need to develop a method that can support diagrams in both categories to execute testing at various levels. The second problem has to do with the fact that UML diagrams could be complex in nature, hence the need to develop robust parsers capable of executing complete extraction of artefacts from the model files of UML diagrams. Also lack of adequate coverage criteria has led to the generation of incomplete test cases. In the proposed method, an improved algorithm was developed to enhance faster and reliable generation of quality test cases, devoid of erroneous elements. Proposing efficient support tool for automatic generation of test cases has to do with accurate implementations of mapping and extraction rules; hence this research.

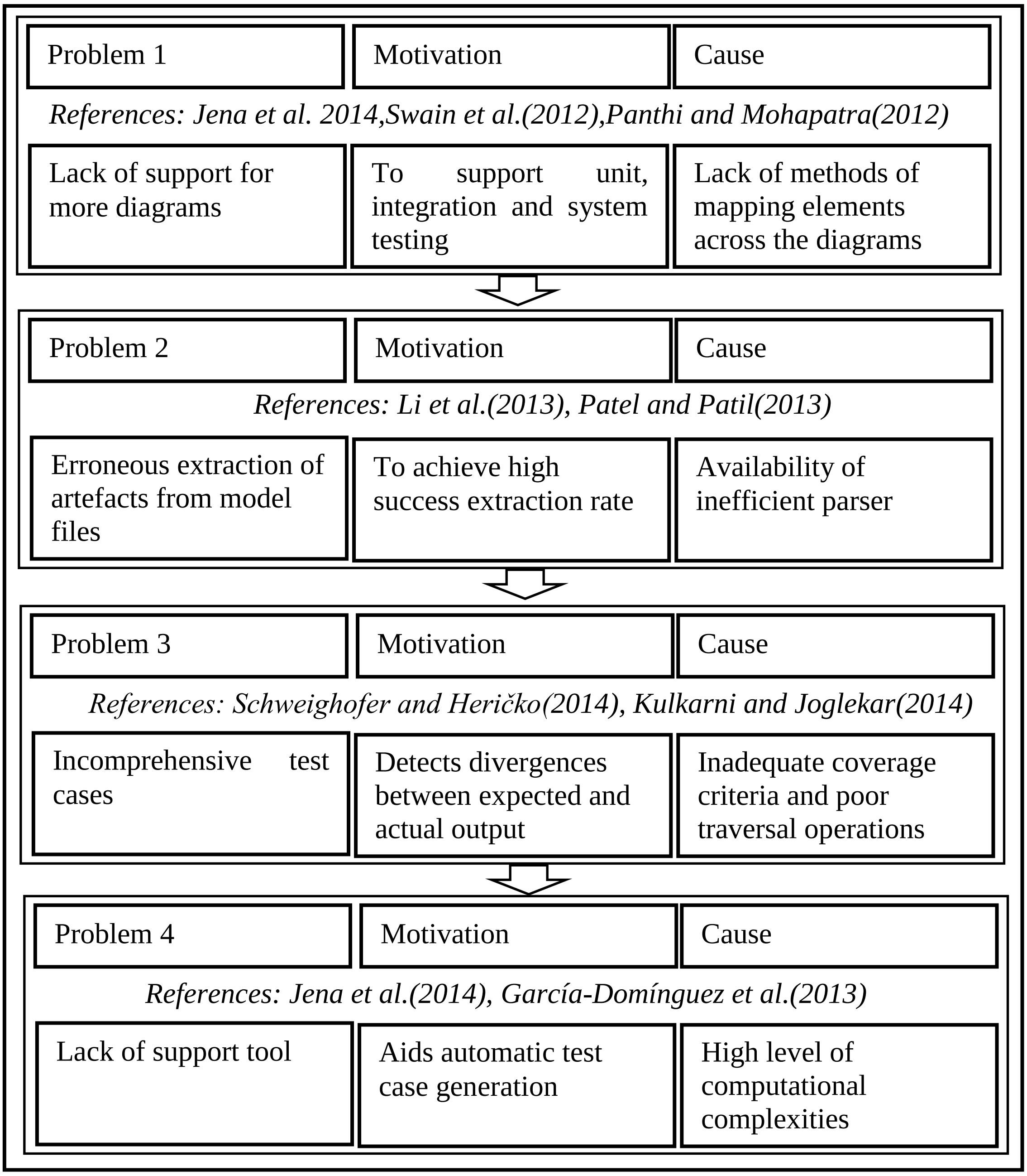

Figure1.1 Diagrammatic illustration of research problems

Based on the main research question, the following sub research questions (RQs) were formulated to aid the development of improved solutions to the problems identified.

RQ1 How can the coverage for more UML diagrams be achieved?

With UML, developers design systems with varieties of diagrams (both structural and behavioural) to present different views of the system model. Therefore, UML diagrams can individually or collectively be used to model requirements and test cases must be generated from them. For example, statechart diagrams could be used for unit testing while activity and sequence diagrams could be used for integration or system testing. However, existing methods are not integrated with the various modeling diagrams; therefore, generated test cases do not tally with the artefacts from the software development document. Lack of such methods makes practical adoption of testing tools difficult and manual integration between tools results in high costs.

RQ2 How can complete extraction of artefacts be achieved?

Artefacts from descriptive links of objects, states, activities, use cases and classes are expected to be visited once in an adequate extraction process. This ensures that all the artefacts associated with any two objects or entities are extracted. Thus, for each artefact; it is necessary to account for the corresponding test case. But existing methods are deficient in ensuring complete extraction of artefacts from the model files of UML diagrams. Therefore, rules that guides the identification and extraction of appropriate artefacts led to the specification of this research question.

RQ3 How can erroneous or redundant generation of test cases be avoided?

Existing methods generate test cases with many erroneous elements which can lead to generation of misleading test cases. This causes vagueness and complicate decision making processes. Therefore, methods capable of correctly traversing contents of the dependency flow tree (DFT) that stores the extracted artefacts during test case generations are required in order to efficiently produce valid test cases.

RQ4 How can coverage criteria be combined to cover all model elements?

Typically, the complexity of UML diagrams lies in the nature of their objects, messages, states, activities, classes and interactions or transitions. As a result, complex behaviours are observed when related objects passes messages with each other within a scenario. Therefore, the essence of this research question is to determine how to incorporate well-known coverage criteria into the proposed test case generation method.

RQ5 How can the quality of test cases be improved?

One of the major problems associated with existing methods is their inability to generate test cases with criteria that ensures test adequacy. A good test case should have the quality to cover more features of test objective. In other words, effectiveness of testing process relies on the quality of test cases not in the quantity of test cases. It is therefore important to generate an appropriate amount (or optimal) number of test cases to ensure quality. The aim of this research question is to propose a test case reduction method which is capable of computing or generating a small representative set of test cases that covers all testing properties of the SUT.

1.4 Research objectives

The aim of this research is to develop a systematic test case generation method with reliable mapper and extractor in order to stimulate generation of optimal test cases. To achieve this aim, the following research objectives were specifically defined:

(i) To propose an improved method that aids generation of test cases from more UML diagrams;

(ii) To propose an improved method that supports accurate extraction of artefacts from model files of UML diagrams;

(iii) To propose an improved method that enhances generation of quality test cases;

(iv) To implement the improved methods and evaluate them based on accuracy and redundancy.

1.5 Scope of the study

The scope of this research is within the confines of the following:

- The solution proposed is limited to UML diagrams. UML-based testing (UBT) is a subset of model-based testing (MBT) where test cases are derived from the diagrams used to model user’s requirements.

- The diagrams utilized include activity, class, sequence, statechart, and use cases because, they can adequately represent functional requirements. These diagrams contain artefacts drawn from the user’s requirements expressed in any of the modelling tools like ArgoUML, Rational Rose or Magic Draw but the proposed method is limited to functional requirements only.

- For this research, ArgoUML was used which supports UML 1.3, 1.4/XMI 1.0, 1.1 and 1.2. The rationale for adopting this tool for usage is because it is open source. Depending on the version, ArgoUML has the capacity of importing XMIs from another tool which makes it really convenient.

1.6 Thesis structure

The rest of this thesis consist of 6 chapters which are structured as follows:

Chapter 2 discusses review of related literature and puts the work conducted in this thesis into context. It identifies existing testing paradigms which considers the utilization of specifications or user's requirements expressed through UML diagrams to conduct testing. It analyzed the testing concepts, processes and features that are quite different from traditional testing techniques. This led to the identification of research gaps or limitations of existing methods which served as the basis for developing an improved one.

Chapter 3 mainly described the methods employed to achieve the thesis objectives. It consisted of well-crafted research framework integrated into an explicit research process with a number of knitted phases. The chapter also described the detailed design of the conducted researches which has led to the development of improved methods. In addition, it enumerated the processes involved in testing the performance of the proposed method which were used to verify the accomplishment of the research objectives.

Chapter 4 presented the design strategies for the mapper, extractor and generator. These consist of the components that constitute the design strategies with the accompanied algorithms for both structural and behavioural UML diagrams. The proposed method is customized and aimed at enhancing more diagram and test coverages during test case generation.

Chapter 5 presented the implementation strategies for the designed mapper, extractor and generator. It mainly focused on the integration of the designed methods into tool with reference to the methodological component of mapped elements, extracted artefacts and generated test cases. It also described the methodological foundation and technical aspects of the tool which included test model construction, conversion into XMI formats, mapping of XMI elements, extraction of artefacts from the XMI file, intermediate representation of the extracted artefacts and test case generation.

Chapter 6 presented the results of the proposed methods with reference to the integrated tool. The results of the proposed tool were discussed, evaluated and benchmarked with existing ones. The chapter was initiated by presenting the proposed methods based on three main issues: mapped elements, extracted artefacts and generated test cases.

Chapter 7 summarises and concludes the thesis. This chapter concludes this thesis by revisiting the original research contributions with further discussions and explored important open issues concerning areas for methodology improvement and research directions for future work.

REFERENCES

Aichernig, B. K., Ničković, D., and Tiran, S. (2015). Scalable Incremental Test-case Generation from Large Behavior Models. *Tests and Proofs* (pp. 1-18). Springer International Publishing.

Anand, S., Burke, E., Chen, T. Y., Clark, J., Cohen, M. B., Grieskamp, W., and Zhu, H. (2013). An Orchestrated Survey on Automated Software Test Case Generation. *The Journal of Systems and Software*, 86, 1978-2001.

Anbarasu, I. (2012). A Survey on Test Case Generation and Extraction of Reliable Test Cases. *International Journal of Computer Science & Applications*, *1*(10), 1-6.

Andrews, A., France, R., Ghosh, S., and Craig, G., (2003). Test Adequacy Criteria for UML Design Models. *Journal of Software Testing, Verification and Reliability*, 13(2), 95-127.

Alanen, M., and Porres, I. (2005). Model Interchange using OMG Standards. *Proceedings of the 2005 IEEE Software Engineering and Advanced Applications Conference*. 30 August - 3 September. Turku, Finland. IEEE, 450-458.

Alhroob, A., Dahal, K., and Hossain, A. (2010). Automatic Test Cases Generation from Software Specifications Modules. *Journal of e-Informatica Software Engineering*, *4*(1), 109-121.

Ali, S., L. C. Briand, M. J.-u. Rehman, H. Asghar, M. Z. Z. Iqbal, and A. Nadeem (2007). A State-based Approach to Integration Testing based on UML Models. *Information Software Technology*, 49 (11),1087-1106.

Aggarwal, M., and Sabharwal, S. (2012). Test Case Generation from UML State Machine Diagram: A Survey. *Proceedings of the 2012 IEEE Conference* on *Computer and Communication Technology (ICCCT).* 23-25 November 2012. Allahabad, India: IEEE, 133-140.

Ahmed, S. U., Sahare, S. A., and Ahmed, A. (2013). Automatic Test Case Generation using Collaboration UML Diagrams. *World Journal of Science and Technology*, *3*(1), 4-6.

Arcaini, P., and Gargantini, A. (2014). Test Generation for Sequential Nets of Abstract State Machines with Information Passing. *Science of Computer Programming*, (1)94, 93-108.

Asthana, S., Tripathi, S., and Singh, S.K., (2010). A Novel Approach to Generate Test Cases Using Class and Sequence Diagrams. *Contemporary Computing*, (pp.155-167). Springer Berlin Heidelberg.

Bao-Lin, L., Zhi-shu, L., Qing, L., and Hong, C.Y., (2007). Test Case Automate Generation from UML Sequence Diagram and OCL Expression. *Proceedings of the 2007 IEEE Conference on Computational Intelligence and Security (CIS)*, 15-19 December 2007. Harbin, Heilongjiang, China: IEEE, 1048-1052.

Bandyopadhyay, A., and Ghosh, S., (2009). Test Input Generation using UML Sequence and State Machines Models. *Proceedings of the 2009 IEEE International Conference on Software Testing, Verification, and Validation*, 1-4 April 2009. Denver, CO, USA: IEEE, 121–130.

Bandyopadhyay, A., and Ghosh, S., (2008). Using UML Sequence Diagrams and State Machines for Test Input Generation, student paper, *Proceedings of the 2008 IEEE International Symposium on Software Reliability Engineering*, 10-14 Nov 2008. Seattle, Washington, USA: IEEE, 309-310.

Baudry, B., Le Traon, Y., and Sunyé, G. (2002). Testability Analysis of a UML Class Diagram. *Proceedings of the 2002 8th International Symposium on Software Metrics*, 2002, IEEE, 54-63.

Bertolino, A., amd Basanieri, F. (2000). A Practical Approach to UML-based Derivation of Integration Tests. *4th International Software Quality Week Europe and International Internet Quality Week Europe (QWE 2000)*, 92-110.

Bertolino, A., (2007). Software Testing Research: Achievements, Challenges, Dreams. *Proceedings of the 2007 IEEE International Symposium on Future of Software Engineering*. 23-25 May 2007. Minneapolis, Minnesota, USA: IEEE, 85-103.

Beizer, B. (1990). *Software Testing Techniques*. 2nd Edition, Van Nostrand Reinhold, New York, USA.

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

Biswal, B. N., Nanda, P., and Mohapatra, D. P. (2008). A Novel Approach for Scenario-based Test Case Generation. *Proceedings of the IEEE 2008 International Conference on Information Technology*.17-20 December, 2008. Bhubaneswar, India:IEEE, 244-247.

Biswal, B. N. (2010). *Test Case Generation and Optimization of Object-Oriented Software using UML Behavioral Models*. MSc Thesis. India.

Blackburn, M., Busser, R., and Nauman, A., (2004). Why Model Based Test Automation is different and what you should know to get started. *Proceedings of the IEEE 2004 International Conference on Practical Software Quality and Testing* (PSQT/PSTT), Washington, D. C. USA, 22-26.

Boberg, J., (2008). Early Fault Detection with Model-Based Testing. *Proceedings of the ACM 7th 2008 International Workshop on ERLANG (ERLANG)*, Victoria, BC, Canada, 9-20.

Booch, G., Rumbaugh, J., Jacobson, I. (2005). *The Unified Modeling Language User Guide*. 2nd Edition, Addison-Wesley.

Boghdady, P. N., Badr, N., Hashem, M., and Tolba, M. F. (2011). Test Case Generation and Test Data Extraction Techniques. *International Journal of Electrical Computer Science*,11(3), 87-94.

Buchs, D., L. Pedro, and L. Lucio (2006). Formal Test Generation from UML Models. *Dependable Systems: Software, Computing, Networks*, 1(1),145-171.

Briand, L. and Y. Labiche (2002). A UML-based Approach to System Testing. *Software and Systems Modeling* 1 (1), 10-42.

Briand, L.C., Penta, M.D., and Labiche, Y. (2004). Assessing and Improving State-Based Class Testing: A Series of Experiments. *Transactions on Software Engineering*, 30(11),770–793.

Briand, L.C., Cui, J., and Labiche, Y. (2003). Towards Automated Support for Deriving Test Data from UML Statecharts. *Proceedings of the IEEE/ACM 6th 2003 International Conference on the Unified Modeling Language: Modeling Languages and Applications*. 20-24 October. San Francisco, California, USA: IEEE, 249-264.

Briand, L.C., Cui, J., and Labiche, Y. (2005). Automated support for deriving test requirements from UML statecharts. *Journal of Software and Systems Modeling*, 4(4),399-423.

Bruegge B. and Dutoit A. H., (2004). *Object-Oriented Software Engineering Using UML, Patterns, and Java*. Prentice Hall, 2nd Edition.

Cavalli, A., Maag, S., Papagiannaki, S., and Verigakis, G., (2005). From UML Models to Automatic Generated Tests for the dotLRN e-learning Llatform. *Electronic Notes in Theoretical Computer Science*. 116(1),133-144.

Carvalho, G., Falcão, D., Barros, F., Sampaio, A., Mota, A., Motta, L., and Blackburn, M., (2014). NAT2TESTSCR: Test Case Generation from Natural Language Requirements Based on SCR Specifications. *Proceedings of the 28th ACM International Symposium on Applied Computing*. 2014. New York, NY, USA: ACM, 1217-1222.

Cavarra A., Crichton C., and Davies J., (2004). A Method for the Automatic Generation of Test Suites from Object Models. *Information and Software Technology*. 46(5), 309-314.

Cavarra, A. and J. Kuster-Filipe (2005). Combining Sequence Diagrams and OCL for Liveness. Electronic *Notes in Theoretical Computer Science*. 115(1), 19-38.

Cechich, A., Piattini, M., and Vallecillo, A., (2003). Component-Based Software Quality: Methods and Techniques. *Lecture Notes in Computer Science*. (pp. 2693). London: Springer-Verlag.

Chen, T. Y., Kuo, F. C., Merkel, R. G., and Tse, T. H. (2010). Adaptive Random Testing: The Art of Test Case Diversity. *Journal of Systems and Software*, *83*(1), 60-66.

Chen, T., Mishra, P., and Kalita, D., (2010). Efficient Test Case Generation for Validation of UML Activity Diagrams. *Design Automation for Embedded Systems*. 14(2), 105-130.

Chen, M., Qiu, X., Xu, W., Wang, L., Zhao, J., and Li, X. (2009). UML Activity Diagram-based Automatic Test Case Generation for Java programs. *The Computer Journal*, *52*(5), 545-556.

Conrad, M., (2004). A systematic approach to testing automotive control software. *Proceeding of Convergence Transportation Electronics Association*. 1(1):1-12.

Creswell, J.W., (2003). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. Sage Publications Inc.

Dai, Z., and Chen, M. H. (2007). *Automatic Test Generation for Database-Driven Applications*. (pp. 9-11).

Dalal, S. R., Jain, A., Karunanithi, N., Leaton, J. M., Lott, C. M., Patton, G. C., and Horowitz, B. M. (1999). Model-based testing in practice. *Proceedings of the 21st International Conference on Software Engineering*. 1999. New York, NY, USA: ACM, 285-294.

Dias-Neto, A. C., and Travassos, G. H. (2009). Model-based Testing Approaches Selection for Software Projects. *Information and Software Technology*, *51*(11), 1487-1504.

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*. 2007. New York, NY, USA: ACM, 31-36.

Dias Neto, A.C., and Travassos, G. H. (2008). Supporting the Selection of Model-based Testing Approaches for Software Projects. *Proceeding of 3rd IEEE/ACM ICSE International Workshop on Automation of Software Testing*. 10-18 May 2008. Leipzig, Germany: ACM, 21-24.

Dinh-Trong, T., S. Ghosh, and R. France (2006). A Systematic Approach to Generate Inputs to Test UML Design Models. *Proceeding of the 17th International Symposium on Software Reliability Engineering*. 95-104.

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

Dobing, B., and Parsons, J., (2006). How UML is Used. *Communications of the ACM*. 49(5):109-113.

Doungsaard, C., K. Dahal, A. Hossain, and T. Suwannasart (2007). Test Data Generation from UML State Machine Diagrams using GAs. *Proceedings of International Conference on Software Engineering Advances*. 25-31 August. 2007. Cap Esterel, France: IEEE, 47.

Doungsa-ard, C., Dahal, K., Hossain, A., and Suwannasart, T., (2008). GA-based Automatic Test Data Generation for UML State diagrams with Parallel Paths. *Advanced Design and Manufacture to Gain a Competitive Edge*. (pp.147-156). London: Springer-Verlag.

Easterbrook, S., Singer, J., Storey, M. A., and Damian, D. (2008). Selecting Empirical Methods for Software engineering Research. Guide to Advanced Empirical Software Engineering (pp. 285-311). London: Springer-Verlag.

El-Far, I.K., and Whittaker, J.A., (2001). Model-based Software Testing. John J. Marciniak (Ed), *Encyclopedia of Software Engineering*, 2nd Edition. Wiley.

Escalona, M. J., Gutierrez, J. J., Mejías, M., Aragón, G., Ramos, I., Torres, J., and Domínguez, F. J. (2011). An Overview on Test Generation from Functional Requirements. *Journal of Systems and Software*, *84*(8), 1379-1393.

Fan, X., Shu, J., Liu, L., and Liang, Q.J., (2009). Test Case Generation from UML Sub Activity and Activity diagram. *Proceeding of the IEEE 2nd International Symposium on Electronic Commerce and Security*. 2009. 244-248. IEEE.

Farooq, U., Lam, C.P., and Li, H., (2008). Towards Automated Test Sequence Generation. *Proceeding of the 19th IEEE Australian Software Engineering*. *Conference*. 26-28 March 2008. Perth, WA, Australia: IEEE, 441-450.

Fowler, M., (2004). UML Distilled: *A Brief Guide to the Standard Object Modeling Languag*e. 3rd Edition, Addison-Wesley.

Francisco, M.A. and Castro, L.M., (2012). Automatic Generation of Test Models and Properties from UML Models with OCL Constraints. *Proceedings of the 12th Workshop on OCL and Textual Modelling*. New York, NY, USA: ACM, 49-54.

Frantzen, L., and Tretmans, J. (2007). Model-based Testing of Environmental Conformance of Components. *Formal Methods for Components and Objects* (pp. 1-25). Springer Berlin Heidelberg.

Fraikin, F., and Leonhardt, T., (2002). SeDiTeC-Testing Based on Sequence Diagrams. *Proceedings of the 17th IEEE International Conference on Automated Software Engineering*. 2002. Edinburgh, UK: IEEE, 261-266.

Gantait, A., (2011). Test Case Generation and Prioritization from UML Models. *Proceedings of the 2nd IEEE International Conference on Emerging Applications of Information Technology*. 19-20 Febuary 2011. Kolkata, India: IEEE, 345-350. IEEE.

García-Domínguez, A., Medina-Bulo, I., and Marcos-Bárcena, M., (2013). An Approach for Model-Driven Design and Generation of Performance Test Cases with UML and MARTE. *Software and Data Technologies* (pp.136-150). Springer Berlin Heidelberg.

Glenford J. M., (1779). *Art of Software Testing*. John Wiley & Sons, Inc., New York, NY, USA.

Gordon, F., and Wotawa, F. (2006). Using model-checkers for mutation-based test-case generation, coverage analysis and specification analysis. *Proceedings of the IEEE International Conference* on *Software Engineering Advances*. 5 October 2006. Tahiti, French Polynesia: IEEE, 16-20.

Gulia, P., and Chillar, R. S. (2012). A New Approach to Generate and Optimize Test Cases for UML State Diagram using Genetic Algorithm. ACM *SIGSOFT Software Engineering Notes*. 37(3),1-5.

Grochtmann, M., and Grimm, K. (1993). Classification Trees for Partition Testing. *Software Testing, Verification & Reliability*3(2): 63-82.

Gross, H. G., (2005). *Component-Based Software Testing with UML*. London: Springer-Verlag.

Hametner, R., Kormann, B., Vogel-Heuser, B., Winkler, D., and Zoitl, A., (2011). Test Case Generation Approach for Industrial Automation Systems. *Proceedings of the 5th IEEE International Conference on Automation, Robotics and Applications*. 6-8 December 2011. Wellington, New Zealand: IEEE, 57-62.

Hartmann, J., Imoberdorf, C., and Meisinger, M., (2000). UML-based Integration Testing. *Proceedings of the 2000 ACM SIGSOFT International Symposium on Software Testing and Analysis*. 5 September 2000. New York, NY, USA: ACM, 60-70.

Hartmann, J., M. Vieira, H. Foster, and A. Ruder (2005). A UML-based Approach to System Testing. *Innovations in Systems and Software Engineering*. 1 (1), 12-24.

Hartman, A., Katara, M., and Olvovsky, S. (2007). Choosing a Test Modeling Language: A Survey. *2nd Proceedings of the 2006 International Haifa Verification Conference*. 23-26 October 2006. Haifa, Israel, Springer, 204-218.

Hasling, B., Goetz, H., and Beetz, K. (2008). Model Based Testing of System Requirements using UML Use case Models. *Proceedings of the 1st IEEE International Conference on Software Testing, Verification, and Validation*. 9-11 April 2008. Lillehammer, Norway: IEEE, 367-376.

Heinecke, A., Bruckmann, T., Griebe, T., and Gruhn, V., (2010). Generating Test Plans for Acceptance Tests from UML Activity Diagrams. *Proceedings of the 17th IEEE International Conference on Engineering of Computer Based Systems*. 22-26 March 2010. Oxford, England: IEEE, 57-66.

Ingle, S. E., and Mahamune, M. R. (2015). An UML Based software Automatic Test Case Generation: Survey. *International Research Journal of Engineering and Technology*. 2(1), 971-973.

Javed, A.Z., Strooper, P.A., and Watson, G.N., (2007). Automated Generation of Test Cases using Model-driven Architecture. *Proceedings of the 2nd International Workshop Automation of Software Test*ing. 20-26 May, Minneapolis, MN: IEEE, 3.

Jena, A.K., Swain, S.K., and Mohapatra, D.P. (2014). A Novel Approach for Test Case Generation from UML Activity Diagram. *Proceeding of the International Conference pn Issues and Challenges in Intelligent Computing Techniques*. 7-8 Febuary 2014, Ghaziabad, India: IEEE, 621-629.

Kansomkeat, S. and W. Rivepiboon (2003). Automated-Generating Test Case using UML Statechart Diagrams. *Proceedings of the 2003 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on Enablement through Technology*. 296-300.

Kansomkeat, S., Offutt, J., Abdurazik, A., and Baldini, A., (2008). A Comparative Evaluation of Tests Generated from Different UML Diagrams. *Proceeding of the 9th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*. 6-8 August 2008. Phuket, Thailand: IEEE, 867–872.

Kaur, P., and Kaur, R., (2011). Approaches for Generating Test Cases Automatically to Test the Software. *International Journal of Engineering and Advanced Technology (IJEAT),* 2(3),191-193.

Kaur, A., and Vig, V. (2012). Systematic Review of Automatic Test Case Generation by UML Diagrams. *International Journal of Engineering*, *1*(6).

Khandai, M., Acharya, A.A., and Mohapatra, D.P., (2011). A Novel Approach of Test Case Generation for Concurrent Systems using UML Sequence Diagram. *Proceedings of the 3rd International Conference on Electronics Computer Technology*. 8-10 April 2011. Kanyakumari, India: IEEE, 157-161.

Kim, Y., H. Hong, D. Bae, and S. Cha (1999). Test Cases Generation from UML State Diagrams. *IET Software* 146 (4), 187-192.

Kim, H., Kang, S., Baik, J., and Ko, I., (2007). Test cases generation from UML activity diagrams. *Proceedings of the 8th International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing.* 3:556-561. IEEE.

Kosmatov, N (2006). A multi-point boundary value problem with two critical conditions. Nonlinear Analysis: *Theory, Methods & Applications*. 65, 622–633.

Kundu, D., Sarma, M., Samanta, D., and Mall, R., (2009). System Testing for Object-oriented Systems with Test Case Prioritization. *Software Testing, Verification and Reliability*, *19*(4), 297-333.

Kundu, D., Sarma, M., Samanta, D., and Mall, R., (2013). Automatic Code Generation from Unified Modelling Language Sequence Diagrams. *IET Software*, 7(1), 12-28.

Kulkarni, P., and Joglekar, Y. (2014). Generating and Analyzing Test Cases from Software Requirements using NLP and Hadoop. *International Journal of Current Engineering and Technology*. 4(6), 3934-3937.

Kyaw, A. A., and Min, M. M., (2015). An Efficient Approach for Model Based Test Path Generation. *International Journal of Information and Education Technology*. 5(10), 763-767.

Meudec, C., (2001). ATGen: Automatic Test Data Generation using Constraint Logic Programming and Symbolic Execution. *Software Testing, Verification and Reliability*, 11 (2), 81-96.

Myers, G. J., Sandler, C., and Badgett, T. (2011). *The Art of Software Testing*. John Wiley & Sons.

Mouchawrab, S., Briand, L.C., and Labiche, Y. (2007). Assessing, Comparing, and Combining Statechart-based Testing and Structural Testing: An Experiment. *Proceedings of the 1st ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. 20-21 Sept 2007. Madrid, Spain, IEEE, 41-50.

Mingsong, C., Xiaokang, Q., and Xuandong, L. (2006). Automatic Test Case Generation for UML Activity Diagrams. *Proceedings of the 2006 International Workshop on Automation of Software Testing*. 2006. New York, NY, USA. ACM, 2-8.

Nayak, A., and Samanta, D., (2011). Synthesis of Test Scenarios using UML Activity Diagrams. *Software and Systems Modeling*, 10(1), 63-89.

Nebut, C., Fleurey, F., Le Traon, Y., and Jezequel, J.M. (2006). Automatic Test Generation: A Use case Driven Approach. *IEEE Transactions on Software Engineering*, 32(3), 140-155.

Nebut, C., Fleurey, F., Traon, Y.L., and Jezequel, J.M., (2003). Requirements by Contracts allow Automated System Testing. Proceeding of the 14th International Symposium on Software Reliability Engineering. 17–20 November 2003. Denver, Colorado, USA. IEEE, 85-96.

Nogueira, S., Cartaxo, E., Torres, D., Aranha, E., and Marques, R. (2007). Model Based Test Generation: An Industrial Experience. *Proceedings of the 1st Workshop on Systematic and Automated Soft. Testing-SBBD/SBES*.

Nogueira, S., Cartaxo, E., Torres, D., Aranha, E., and Marques, R. (2007). Model Based Test Generation: A case study. *Proceedings of the 1st Brazilian Workshop on Systematic and Automated Software Testing, Recife*.

Lam, S.S.B., Raju, M.L., Ch, S., and Srivastav, P.R., (2012). Automated Generation of Independent Paths and Test Suite Optimization using Artificial Bee Colony. *Procedia Engineering*, 30(1), 191-200.

Lamancha, B. P., Polo, M., Caivano, D., Piattini, M., and Visaggio, G., (2013). Automated Generation of Test Oracles using a Model-driven Approach. *Information and Software Technology*, 55(2), 301-319.

Lédeczi, Á., Bakay, A., Maroti, M., Volgyesi, P., Nordstrom, G., Sprinkle, J., and Karsai, G. (2001). Composing Domain-Specific Design Environments. *Computer*, 34(11), 44-51.

Li, L., Li, X., He, T., and Xiong, J. (2013). Extenics-based Test Case Generation for UML Activity Diagram. *Procedia Computer Science*, 1(17):1186-1193.

Lim, S. L. (2011). *Social networks and collaborative filtering for large-scale requirements elicitation* PhD Thesis, University of New South Wales, Australia.

Linzhang, W., Jiesong, Y., Xiaofeng, Y., Jun, H., Xuandong, L., and Guoliang, Z. (2004). Generating test cases from UML activity diagram based on gray-box method. *Proceedings of the 11th Software Engineering* Conference. 2004. *Asia-Pacific*, IEEE, 284-291.

Lochau, M., and Goltz, U., (2012). Feature interaction aware test case generation for embedded control systems. *Electronic Notes in Theoretical Computer Science,* 264(3):37-52.

OMG (2005): *Unified Modeling Language Specification*, Version 2.0, Object Management Group, www.omg.org.

OMG (2007). *Meta-Object Facility 2.0.* OMG Document formal/2006-01-01.

OMG and Soley, R., (2000). *Model-Driven Architecture*. http://www.omg.org/mda.

Offutt, A., Z. Jin, and J. Pan (1999). The Dynamic Domain Reduction Procedure for Test Data Generation. *Software Practice and* Experience. 29(2):167-193.

Offutt, J., S. Liu, A. Abdurazik, and Ammann, P., (2003). Generating Test Data from State-based Specifications. *Software Testing, Verification and Reliability*, 13 (1), 25-53.

Oluwagbemi, O., and Asmuni, H., (2014). An Improved Model-Based Technique for Generating Test Scenarios from UML Class Diagrams. *Handbook of Research on Emerging Advancements and Technologies in Software Engineering*. Eds. Ghani, Imran and Wan Nasir Kadir. 434-448. IGI Global.

Tahiliani, S., and Pandit, P. A (2012). Survey of UML-Based approaches to Testing. *International Journal of Computational Engineering Research*, *2*(5), 1396-1400.

Panthi, V., and Mohapatra, D. P. (2012). Automatic Test Case Generation using Sequence Diagram. *Proceedings of International Conference on Advances in Computing*. 2012. India, Springer. 277-284.

Patel, P., and Patel, N. N. (2012). Test Case Formation using UML Activity Diagram. *World Journal of Science and Technology*, *2*(3).

Patel, P.E., and Patil, N.N., (2013). Testcases Formation Using UML Activity Diagram. *Proceedings of the International Conference on Communication Systems and Network Technologies*. 6-8 April 2013, Gwalior, India: IEEE, 884-889.

Patton R., *Software Testing*. SAMS, 2nd Edition, 2005.

Pachauri, A. (2013). Automated test data generation for branch testing using genetic algorithm: An improved approach using branch ordering, memory and elitism. *Journal of Systems and Software,* 86(5), 1191-1208.

Paulish, D. J., Kazmeier, J., and Rudorfer, A. (2009). *Software and Systems Requirements Engineering: in Practice*. New York: McGraw-Hill.

Pechtanun, K., and Kansomkeat, S., (2012). Generation of Test Case from UML Activity Diagram based on AC grammar. *Proceedings of the International Conference on Computer and Information Science*. 12-14 June 2012, Kuala Lumpur, Malaysia: IEEE, 895-899.

Pezze, M., and Young, M., (2007). Software Testing and Analysis: Process, Principles, and Techniques. John Wiley & Sons, New York City, United States.

Pilskalns, O., Andrews, A., Ghosh, S., and France, R., (2003). Rigorous Testing by Merging Structural and Behavioral UML Representations. *Proceedings of the 6th International Conference on the Unified Modeling Language.* 20–24 Oct 2003. San Francisco, CA, USA: Springer, 234-248.

Pilskalns, O., Andrews, A., Knight, A., Ghosh, S., and France, R. (2007). Testing UML Designs, *Information and Software Technology*, 49(8), 892-912.

Prasanna, M., Sivanandam, S. N., Venkatesan, R., and Sundarrajan, R. (2005). A Survey on Automatic Test Case Generation. *Academic Open Internet Journal*, 15(6).

Prasanna, M., and Chandran, K. R. (2009). Automatic Test Case Generation for UML Object Diagrams using Genetic Algorithm. *International Journal of Advanced Soft Computing Application* 1(1), 19-32.

Prasanna, M., Chandran, K. R., and Thiruvenkadam, K. (2011). Automatic Test Case Generation for UML Collaboration Diagrams. *IETE Journal of research*, 57(1),77.

Prasad, S., Jain, M., Singh, S., and Patvardhan, C. (2012). A Productive Method for Improving Test Effectiveness. *International Journal of Applied Information Systems*, 2(2), 9-17.

Pretschner, A., Slotosch, O., Aiglstorfer, E., and Kriebel, S., (2004). Model-based Testing for Teal:The Inhouse Card Case Study. *International Journal on Software Tools for Technology Transfer*, 5(2), 140-157.

Pretschner, A., Prenninger, W., Wagner, S., Kühnel, C., Baumgartner, M., Sostawa, B., Zölch, R., and Stauner, T., (2005). One Evaluation of Model-Based Testing and its Automation. *Proceedings of the 27th International Conference on Software Engineering*. 15-21 June 2005. St. Louis, MO, USA: ACM, 392-401.

Priya, S. S., and Malarchelvi, P. S. K. (2013). Test Path Generation Using UMLSequence Diagram. *International Journal*, *3*(4).

Rapos, E.J., and Dingel, J., (2012). Incremental Test Case Generation for UML-RT Models Using Symbolic Execution. *Proceedings of the 5th International Conference on Software Testing, Verification and Validation*. 17-21 April 2012. Montreal, QC, Canada: IEEE, 962-963.

Reales Mateo, P., and Polo Usaola, M., (2013). Automated test generation for multi-state systems. *Proceeding of the 15th Annual Conference Companion on Genetic and Evolutionary Computation*. 2002. New York, NY, USA: ACM, 211-212.

Riebisch, M., Philippow, I., and Götze, M., (2003). UML-based Statistical Test Case Generation. *Objects, Components, Architectures, Services, and Applications for a Networked World*. 394-411. Springer Berlin Heidelberg.

Robson, C. (2002). *Real World Research*. Oxford: Blackwell publishers.

Rumbaugh, J., Jacobson, I., and Booch, G., (2004). *The Unified Modeling Language Reference Manual*. 2nd Edition, Addison-Wesley Object Technology Series, Addison-Wesley.

Sabharwal, S., Singh, S. K., Sabharwal, D., and Gabrani, A. (2010). An Event-based Approach to Generate Test Scenarios. *Proceedings of the International Conference on Computer and Communication Technology*. 17-19 September 2010. Allahabad, Uttar Pradesh, India: IEEE, 551-556.

Samuel, P., Mall, R., and Sahoo, S., (2005). UML Sequence Diagram Based Testing Using Slicing. *Proceedings of the International Conference on Control, Communicaitons and Automation*. 11-13 December 2005. Chennai, India, IEEE, 176–178.

Samuel, P., Mall, R., and Bothra, A. K. (2008). Automatic Test Case Generation using Unified Modeling Language (UML) State Diagrams. *IET software*, *2*(2), 79-93.

Samuel, P., and Joseph, A.T., (2008). Test Sequence Generation from UML Sequence Diagrams. *Proceedings of the 9th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*. 6-8 August 2008. Phuket, Thailand: IEEE, 879–887.

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

Sapna, P. G., and Mohanty, H. (2010). Automated Test Scenario Selection based on Levenshtein Distance. *Distributed Computing and Internet Technology* (pp. 255-266). Springer Berlin Heidelberg.

Sapna, P. G., and Mohanty, H. (2010). Clustering Test Cases to Achieve Effective Test Selection. *Proceedings of the 1st Amrita ACM-W Celebration on Women in Computing in India*. 2010. New York, NY, USA: ACM, 15.

Sarma, M., Kundu, D., and Mall, R. (2007). Automatic Test Case Generation from UML Sequence Diagram. *Proceedings of the International Conference on Advanced Computing and Communications*. 18-21 December 2007. Guwahati, Assam, India: IEEE, 60-67.

Sawant, V., and Shah, K. (2011). Construction of Test Cases from UML Models. *Technology Systems and Management* (pp.61-68). Springer Berlin Heidelberg.

Schlick, R., Herzner, W., and Jöbstl, E., (2001). Fault-based Generation of Test Cases from UML Models Approach and Some Experiences. *Computer Safety, Reliability, and Security* (pp.270-283). Springer Berlin Heidelberg.

Schwarzl, C., and Peischl, B., (2010). Test Sequence Generation from Communicating UML Statecharts: An Industrial Application of Symbolic Transition Systems. Proceedings of the 10th International Conference on Quality Software. 14-15 July 2010. Zhangjiajie, China. IEEE, 122-131.

Schweighofer, T., and Heričko, M., (2014). Approaches for Test Case Generation from UML Diagrams. *Proceedings of the 3rd International Workshop on Software Quality Analysis, Monitoring, Improvement and Applications*. 19-22 September 2014. Lovran, Croatia: 91-98.

Seifert, D., Helke, S., and Santen, T, (2003). Test Case Generation for UML Statecharts. *Perspectives of System Informatics*, (pp.462-468). Springer Berlin Heidelberg.

Seifert, D., (2008). Conformance Testing based on UML State Machines: Automated Test Case Generation, Execution and Evaluation. [Technical Report] 2008.

Shamsoddin-Motlagh, E. (2012). A Review of Automatic Test Cases Generation. *International Journal of Computer Applications*, 57(13).

Shanthi, A. V. K., and Kumar, D. G. M. (2011). Automated Test Cases Generation For Object Oriented Software. *Indian Journal of Computer Science and Engineering*, *2*(4).

Sommerville, I., (2001). Software Engineering. Addison-Wesley.

Sokal, R. R. and Michener, C. D. (1958). A statistical method for evaluating systematic relationships. *University of Kansas Science Bulletin*, 38, 1409-1438.

Soley, R. (2000). *Model Driven Architecture*. OMG white paper, 308, 308.

Sokenou, D. and G. GmbH (2006). Generating Test Sequences from UML Sequence Diagrams and State Diagrams. *Informatik: Informatik fur Menschen* 2 (94), 236-240.

Sokenou, D., (2006). Generating Test Sequences from UML Sequence Diagrams and State Diagrams. *GI Jahrestagung* (2), 236-240.

Shirole, M., Suthar, A., and Kumar, R. (2011). Generation of Improved Test Cases from UML State Diagram using Genetic Algorithm. *Proceedings of the 4th India Software Engineering Conference*. New York, NY, USA: ACM , 125-134.

Swain, S. K., and Mohapatra, D. P. (2010). Test case generation from Behavioral UML Models. *International Journal of Computer Applications*, *6*(8), 5-11.

Shanthi, A. V. K., Parthiban, D., and MohanKumar, G. A Survey of UML-Based Automatic Test Cases Generation for Software Testing.

Sharma, N. K., and Saxena, D. Study of Approaches For Generating Automated Test Cases By UML Diagrams. *International Journal of Engineering Research & Technology*, 2(6), 3028-3034.

Sharma, A., and Singh, M. (2013). Generation Of Automated Test Cases Using UML Modeling. *International Journal of Engineering*, *2*(4).

Singh, S. K., Sabharwal, S., and Gupta, J. P. (2012). A Novel Approach for Deriving Test Scenarios and Test Cases from Events. *Journal of Information Processing Systems*, *8*(2), 213-240.

Spivey, J.M., (1992). The Z Notation: *A Reference Manual*, 2nd Ed.

Spivey, J. M., and Abrial, J. R. (1992). *The Z Notation* (p. 90). Hemel Hempstead: Prentice Hall.

Strauss, A., and Corbin, J., (1994). Grounded Theory Methodology. *Handbook of Qualitative Research*, 273-285.

Sun, C.A., Zhang, B., and Li, J., (2009). TSGen: A UML Activity Diagram-Based Test Scenario Generation Tool. *Proceedings of the International Conference on Computational Science and Engineering*, 2, 853-858. IEEE.

Swain, S. K., Mohapatra, D. P., and Mall, R. (2010). Test Case Generation Based on Use case and Sequence Diagram. *International Journal of Software Engineering*, 3(2), 21-52.

Thomas, J. O. and Balcer, M. J. (1988). The Category-partition Method for Specifying and Generating Functional Tests. *Communications of the ACM*, 31(6), 676-686.

Tseng, W.H., and Fan, C.F., (2013). Systematic Scenario Test Case Generation for Nuclear Safety Systems. *Information and Software Technology*, 55(2), 344-356.

Warmer, J., and Kleppe, A, (2003*). The Object Constraint Language: Getting Your Models Ready for MDA*. 2nd Edition, Addison-Wesley Professional.

Uhl, A. (2008). Model-Driven Development in the Enterprise. *IET Software*, 25(1), 46-49.

Utting, M., (2005). Model-Based Testing. *Proceedings of the Information Federation for Information Processing Working Conference*. The VSTTE Conference-Verified Software Theories, Tools, Experiments, ETH, Zurich, Switzerland, 10-13.

Utting, M., Pretschner, A., and Legeard, B., (2007). A Taxonomy of Model-based Testing. *Technical Report* 04/2006, Department of Computer Science, The University of Waikato, Hamilton, New Zealand, 17 pages. [TR online] http://www.cs.waikato.ac.nz/pubs/wp/2006/uow-cs-wp-2006-04.pdf, Accessed Wed 30 December 2014.

Utting, M., and Legeard, B., (2006). *Practical Model-Based Testing: A Tools Approach*. Morgan Kaufmann Publishers/Elsevier.

Utting, M., Pretschner, A., and Legeard, B. (2012). A Taxonomy of Model-based Testing Approaches. *Software Testing, Verification and Reliability*, *22*(5), 297-312.

Wang, Y., Bai, X., Li, J., and Huang, R. (2008). Ontology-based Test Case Generation for Testing Web Services. *8th International Symposium on Autonomous Decentralized Systems*. IEEE, 43-50.

Weißleder, S. (2010). Test Models and Coverage Criteria for Automatic Model-Based Test Generation with UML State Machines. *Doctoral thesis*, Humboldt-University Berlin, Germany.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., and Wesslén, A., (2000). *Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publishers. Boston, MA USA.

Wu, Y., Chen, M.H., and Offutt, J., (2003). UML-Based Integration Testing for Component-Based Software. *Proceedings of the 2nd International Conference on COTS-Based Software Systems*, 10-12 Feb 2003. Ottawa, Canada: Springer: 251-260.

Yue, T., Ali, S., and Briand, L., (2011). Automated Transition from Use cases to UML State Machines to Support State-based Testing. *Modelling Foundations and Applications*. 115-131. Springer Berlin Heidelberg.

Zander, J., Schieferdecker, I. and Mosterman, P.J. (2011). *Model-based testing for embedded systems*. CRC press.

Zhang, W., and Liu, S., (2013). Supporting Tool for Automatic Specification-Based Test Case Generation. *Structured Object-Oriented Formal Language and Method*, 12-25. Springer Berlin Heidelberg.

Zeng, F., Chen, A., Cao, Q., and Mao, L., (2009). Research on Method of Object-Oriented Test Cases Generation Based on UML and LTS. *Proceedings of the 1st International Conference on Information Science and Engineering*. IEEE, 5055-5058.

Zheng, W., and Bundell, G., (2007). Model-based Software Component Testing: A UML-based Approach. Proceedings of the 6th IEEE/ACIS International Conference on Computer and Information Science. 11-13 July 2007. Melbourne, Qld, Australia. IEEE, 891-899.

Zheng, W. (2012). *Model-Based Software Component Testing*. PhD Thesis. The University of Western Australia, Australia.