International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 1 Issue 6, August - 2012

Systematic Review of Automatic Test Case Generation by UML Diagrams

Dr. Arvinder Kaur
Associate Professor
Guru Gobind Singh Indraprastha University

Vidhi Vig
Research Student
Guru Gobind Singh Indraprastha University

Abstract

Software testing is an important activity in the Software Development Life Cycle. To cut down the time and cost of manual testing and to increase the reliability of the software, researchers and practitioners have proposed various tools and techniques for automation of software testing. A great deal of research effort has been spent on finding efficient methods for different aspects of test case automation. Many researchers have proposed various techniques on automatic test case generation and it still remains an area of interest for them. This paper presents a systematic survey of the work done in the field of automatic generation of test case particularly related to UML based automated test case generation.

Keywords: Test case, UML, Automatic test case generation, MBT

Introduction

Software testing is an important activity in the Software Development Life Cycle. To cut down the time and cost of manual testing and to increase the reliability of the software, researchers and practitioners have proposed various tools and techniques for automation of software testing. A great deal of research effort has been spent on finding efficient methods for different aspects of test case automation. Many researchers have proposed various techniques on automatic test case generation and it still remains an area of interest for them.

Testing in industrial projects can be effective only when the testing effort is “affordable”; this means that the testing approach should be able to produce a test plan soon, and even when the software system is only partially modeled. Another important aspect in industrial testing is accuracy. Since inaccuracy can strongly diminish the testing utility, the best has to be done in order to enrich the testing results. As systems are increasing in complexity, more systems perform mission-critical functions, and dependability requirements such as safety, reliability, availability, and security are vital to the users of these systems. The competitive marketplace is forcing companies to define or adopt new approaches to reduce the time to market as well as the development cost of these critical systems. With the increasing complexity and size of software applications more emphasis has been placed on object oriented design strategy to reduce software cost and enhance software usability. However, object-oriented environment for design and implementation of software brings about new issues in software testing. This is because the important features of an object oriented program, such as, encapsulation, inheritance, polymorphism, dynamic binding etc. create several testing problems and bug hazards [1]

An approach for generating test cases satisfying different coverage criteria from UML state chart is described in [22]. In [25], the state charts are transformed to global finite state machine (GFSM) from which the integration test cases are generated. A number of approaches for generating test cases indirectly from the UML analysis and design models, i.e. use case diagram, sequence diagram, collaboration diagram and class diagram, are proposed in [45]. But most of them need to translate UML description into another formal description and then derive the test from the latter. [46] Introduce the approach to generate test from UML activity diagram. In [46], an UML activity diagram is formalized and transformed to a test case model, and the test case could be generated from the test case model [42].

Research questions

This survey aims at summarizing the current state of the art in automatic test case generation research by covering the questions below. The research questions aims at finding the efficient procedures for automatic test case generation in practice. The questions are:

RQ1: What are the various UML Techniques used for ATCG?
RQ2: Which is the most widely used technique?
RQ3: What are the broad areas covered by these techniques?
RQ4: What type of testing has been done?
RQ5: What are the benefits of using various techniques?
RQ6: From where are these test cases generated?
RQ7: What are the other relative techniques available?
RQ8: What are the problems with OO method of ATCG?
RQ9: What are the frequently ignored aspects during ATCG?

Sources of information

This paper presents a systematic review of the work done in the field of automatic generation of test case particularly related to UML based automated test case generation and others. In order to gain a broader perspective, various papers and journals were searched. The following six databases were covered:

- ACM Digital Library (www.portal.acm.org).
- IEEE Xplore (www.ieeexplore.ieee.org).
- ScienceDirect (www.sciencedirect.com).
- Springer LNCS (www.springer.com/lncs)
- Wiley journals {www.wiley.com)
- Google Scholar (www.googlescholar.com)

Search criteria

The initial search criteria was kept broad in order to include the articles with different uses of terminology. The key words used were <automatic> and (<test> or <test case>) and <generation>, and the database fields of title and abstract were searched. The start year was set to 1980 to ensure that most relevant research within the field would be included, and the last date for inclusion is publications within 2010.

The ultimate goal of software testing is to help designers, developers, and managers construct systems with high quality. Thus, research and development on testing aim at efficiently performing effective testing to find more errors in requirement, design and implementation. Progress toward this destination requires fundamental research, and the creation, refinement, extension, and popularization of better methods. Software has been tested as early as software has been written. The concept of testing itself evolved with time. The evolution of definition and targets of software testing has directed the research on testing techniques. Though there exist many testing techniques but the paper focus on automatic test case generation [60].

Data collection

The list of journals proceedings and conference proceedings is given below in table1.

Table 1

Journals	Number of papers found
IEEE transactions of software engineering	6
Journal of system and software	2
Software testing verification and reliability	2
ACM computing surveys	0
IBM system of journals	2
IEEE computer software and application software	2
International conference of software engineering	1
International conference of wb engineering	1
International workshop of automation on software test	4
International conference on quality software	2
IEEE conference on software maintenance	1
Proceeding of International	2

conference on UML	
Empirical software engineering	2
International conference on emerging technologies	1
IEEE in software	2
ACM SIGSOFT software engineering	3
Information and software technology	3
Others	6

The software systems need to improve their quality guarantee because of their growing complexity. A tool that assures the quality of software systems is the system testing. The system testing assures that the functionality of the system under test (SUT) satisfies its requirements. A system test case substitutes an actor and simulates its behaviour. This definition shows that the design of the system testing is based on the functional requirements of the system. Nowadays, it is usual to express the functional requirements of a web system through UML use case diagrams and text templates. Existing papers [3], [6], expose that use cases and text templates are adequate for web systems. Thus, use cases are an appropriate artifact to start the generation process of test cases for web systems.

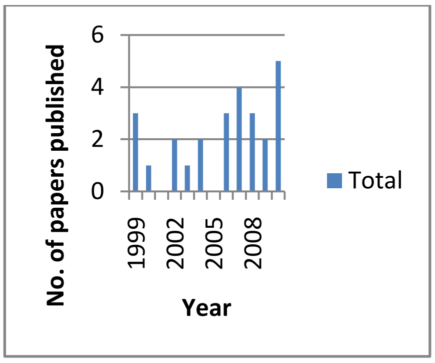
The Unified Modeling Language (UML) [47] is a visual modeling language that comprises nine types of graphics, called diagrams. Nowadays, there are many studies that are focused on test cases generation from UML specification can be found in [20, 22, 25 26].

Results

The following section reflects the results related to the research question.

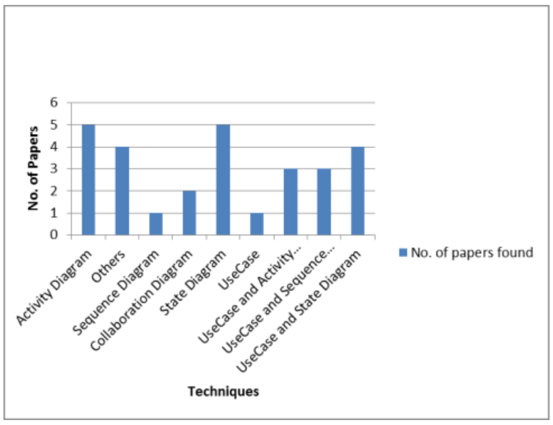
RQ1- What are the various UML Techniques used for ATCG?

There are various techniques used in automatic test case generation which includes SBST(search based software test case generation), FSM(Finite state machine), MBT(Model based testing) etc., but since the paper focus on UML based techniques, the key techniques found were: UML diagrams, State chart diagrams, sequence diagrams, activity diagrams, class diagrams, collaboration diagrams



RQ2- Which is the most widely used technique?

The most widely used techniques involve combination of various UML techniques. This can be best analyzed from the bar graph given below It can be easily noted that activity diagram and sequence diagram are the most widely used approaches so far. One of the oldest approaches for model based testing is by using Use Case and State diagram. In this approach, the models are transformed into usage models to describe both system behavior and its usage. The method is intended for integration into an iterative software development process model. Kansomkeat [41] proposed an approach using only statechart diagrams. The main advantage of this approach was the capability of automation.

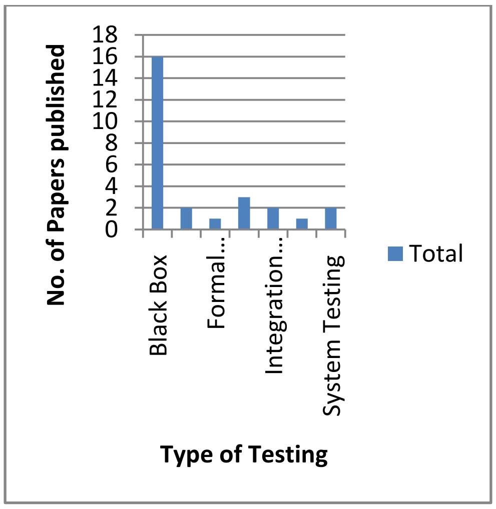


RQ3- What are the broad areas covered by these techniques?

The broad areas covered by these techniques includes Web applications, real time embedded systems, artificial intelligence planning, spreadsheets, system on chip designs and reactive systems, OO systems, SOA interacting services.

RQ4- What type of testing has been performed?

Grey box testing, black box testing, white box testing, conformance testing and scenario based testing were the major types of testing performed but out of the black box testing remained the most dominant one. Generation of test case from the code can be really cumbersome and difficult to automate while generation of test case from the requirement specifications ignores the system implementation aspect. Hence, a combination of both the techniques i.e. grey box testing is the best [51]

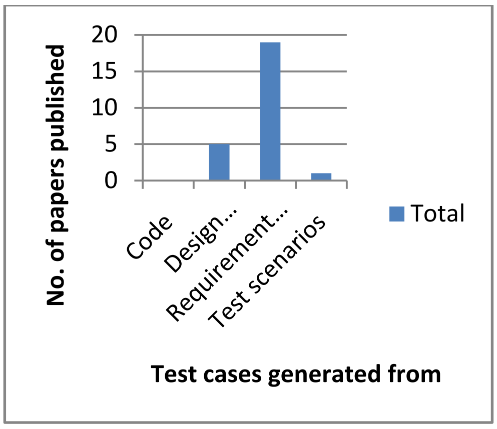


RQ5- What are the benefits of using various techniques?

Each technique has its own advantages and disadvantages, but in this section advantages of various techniques are highlighted. A UML State chart [39] covers various test criteria such as transition coverage, full predicate coverage, transition pair coverage and complete sequence coverage. It also helps in performing class level testing. Activity Diagrams on the other hand can represent both conditional and parallel activities. A fork construct is used for concurrent activities in activity diagram. Sequence diagrams [44] describe sequence of actions that generate in a system over time. It captures invocation of methods from each object and order in which it occurs. Collaboration Diagrams [67] covers the dynamic aspect of testing better than any other UML model. Therefore, it can easily represent dynamic behaviour of the system along with good graphical representation of system scope requirements

RQ6- From where are these test cases generated?

Test cases are generated either from specification of requirements or source code or design specifications. We can easily see that most of the times test cases are generated from the requirement specifications only. The easy and early availability of requirement specifications encourage its use.



RQ7- What are the other relative techniques available?

Hybrid approaches are considered a very good technique of automatic test case generation. The name is used when various UML techniques are used together at one place. The integration of use cases and UML sequence diagrams enables to obtain test scenarios that are closely related to the implementation [63]. This approach concentrates on high level system test case generation. The integration of UML sequence diagrams and UML state diagrams develops an integration diagram which serves as input for test case generation [20]. It is good for industrial application with semi-formal specifications. As it generates test case for integration tests, it enables user to test the systems even if it is partially modeled.

RQ8- What are the problems with OO method of ATCG?

Although OO method has been widely used but encapsulation and abstraction can hide some errors which are hard to find. Hence, development of integrated formal specification and testing methodology would be better. Also in OO paradigm, fault may be in requirement specification or analysis model or design model or implementation. Therefore, testing from requirements and designs are very important

RQ9- What are the frequently ignored aspects of ATCG?

Generation of test cases from UML models can address the challenges posed by object-oriented paradigms efficiently. Moreover, test cases can be generated early in the development process and thus it helps in finding out many problems in design if any even before the program is implemented. However selection of test cases from UML model is one of the most challenging tasks. Most of the studies ignore concurrent and integration testing resulting in less efficient test cases generation. Less attention is paid to ensure adequate traceability or coverage of test scenarios. Moreover, the selections of test cases are usually done randomly. The test cases generated do not address to the ever changing requirements in the process of software development. This may result in generation of less number of test cases than required. ATCG through UML Diagrams focus more on the functional behaviour and tend to ignore structural and behavioural design of the system [51]. One of the biggest drawbacks of using UML based approach is that UML hardly provides formal semantics resulting in ambiguity. Finite state machines on the other hand are considered a better solution for this problem [58].

Survey Table

The following table presents a list of papers searched along with the type of testing technique performed, type of use case diagram used, type of domain covered by the paper, type of algorithm used, case studies covered and other relevant data.

Table 2

? : Data not found in the paper, ICIT: Industrial Technology, SAICSIT: South African Institute for Computer Scientists and Information Technologists, SIGSOFT: Special Interest Group on Software Engineering, IJCA: International Journal of computer Application, WSEAS: World Scientific and Engineering Academy and Society, TSE: Transactions on software engineering, LNCS: Lecture Notes in computer science, STVR: Software, Testing, Verification and Reliability, ECOOP: European Conference on Object-Oriented Programming, IC3: International conference on contemporary computing, APSEC: Asia-Pacific Software Engineering Conference, CSSE: Computer Science and Software Engineering, ICWE: International Conference on Web Engineering, ICSTE: International Conference on Software Technology and Engineering, CMCE: International conference on Computer, Mechatronics, Control and Electronic Engineering, WEC: International Workshop on Electronic Contracting, COMPSAC: Computer software and applications conference

	Andreas Heinecke [57]	**Biswal [38]**	**Cartaxo [44]**	**Chang-ai Sun [46]**	**Xiuping Hou [55]**	**Supaporn Kansomkeat [62]**
Technique used	*UML Activity Diagrams*	*Activity diagram, sequence diagram and class diagram.*	*UML sequence diagrams*	*UML Activity Diagrams*	*UML Activity Diagrams*	*UML Activity Diagrams*
Type of testing	*Black Box*	*Grey-box*	*Black Box*	*Black box*	*Integration testing*	*Black box*

Algorithm used	*Applied Modified Depth-First-Search Algorithm*	*Depth first search*	*Depth First Search method*	*Yes*	*Automatic algorithm*	*No*
Tool used	*No*	*No*	*No*	*TSGen*	*No*	*Yes*
Tool validated by user	*No*	*No*	*No*	*Yes*	*No*	*No*
Area covered	*Business Domain*	*Industrial Domain*	*Telephony domain*	*Education Domain*	*Business domain*	*Business domain*
Case Study	*Traveller problem*	*ATM (Automatic Teller Machine)*	*Motorola*	*ATM*	*No*	*Payment and Car Parking*
Test cases generated from	*Specifications*	*Design models*	*Requirement specification*	*Test scenarios*	*Requirement*	*Design*
Test case selection	*?*	*?*	*?*	*TSGen*	*?*	*?*
Coverage	*All path coverage criterion*	*Path coverage*	*Complete coverage*	*concurrency coverage criteria*	*?*	*?*
Number of activities performed	*5*	*4*	*2*	*4*	*2*	*3*
Year	*2010*	*2008*	*2007*	*2009*	*2010*	*2010*

Paper published	*IEEE (WEC)*	*IEEE (ICIT)*	*IEEE (ICCSE)*	*IEEE (COMPSAC)*	*IEEE (CMCE)*	*IEEE (ICSTE)*

	Li Liuying [56]	**Monalisa Sarma [51]**	**Chen [32]**	**Cle´mentine Nebut [11]**	**Stefania Gnesi [52]**	**Wang Linzhang [58]**
Technique used	*UML state charts*	*use case and sequence diagram*	*UML use case, sequence and class level state chart models*	*Use cases*	*UML State charts*	*UML activity diagrams*
Type of testing	*Black Box*	*Black Box*	*Black Box*	*Requirement-based testing*	*Formal conformance*	*Grey Box*
Algorithm used	*No*	*Yes*	*Yes*	*Yes*	*Yes*	*DFS*
Tool used	*No*	*MagicDraw v. 10.0*	*No*	*No*	*No*	*UMLTGF*
Tool validated by user	*No*	*No*	*No*	*No*	*No*	*No*
Area covered	*Industrial domain*	*Business Domain*	*Business Domain*	*Networking Domain*	*?*	*Business Domain*
Case Study	*Recorder*	*ATM*	*No*	*ATM, FTP. VM*	*No*	*ATM*
Test cases generated from	*Requirements and design*	*Requirements*	*Requirements*	*Requirements*	*?*	*Design specifications*

Test case selection	*?*	*?*	*?*	*?*	*Random*	*Random*
Coverage	*?*	*Yes*	*Yes*	*Yes*	*?*	*Yes*
No. of activities performed	*3*	*5*	*4*	*3*	*2*	*3*
Year	*1999*	*2007*	*2007*	*2006*	*2004*	*2004*
Paper published	*IEEE* (ICIT)	*IEEE* (ICIT)	*IEEE* (ICIT)	*IEEE* (TSE)	*IEEE* (ICCSNC)	*IEEE* (APSEC)

	Xi Wang [42]	**Javier J. Gutiérrez [9]**	**Philip Samuel [39]**	**Y.G.Kim [30]**	**Shinpei ogata [53]**	**Lionel Briand [63]**
Technique used	*UML state diagrams*	*UML use cases*	*UML activity diagrams.*	*UML state diagrams*	*Use case diagrams*	*Use case, sequence, class diagram*
Type of testing	*Black box*	*Black box*	*Grey Box*	*class testing*	*Black Box*	*System Testing*
Algorithm used	*Yes*	*No*	*Yes*	*Yes*	*Yes*	*Yes*

Tool used	*Yes*	*Yes*	*Yes*	*No*	*No*	*TOTEM*
Tool validated by user	*No*	*Yes*	*No*	*No*	*No*	*Yes*
Area covered	*Web Domain*	*Web Domain*	*Business domain*	*Business Domain*	*Business Domain*	*Education domain*
Case Study	*web application for software download*	*Web application*	*ATM*	*Coffee Vending machine*	*Confectionery*	*Library system*
Test cases generated from	*Requirement s*	*Functional Requirements*	*Design specifications*	*Specification*	*Specification*	*Requirements*
Test case selection	*?*	*?*	*Random*	*?*	*?*	*?*
Coverage	*?*	*?*	*Path coverage criterion*	*Yes path, state, Transition*	*?*	*?*
No. of activities performed	*2*	*4*	*5*	*3*	*3*	*3*
Year	*2008*	*2007*	*2009*	*1999*	*2010*	*2002*
Paper published	*IEEE (CSSE)*	*ACM (ICWE)*	*ACM (SIGSOFT)*	*IEEE (ICIT)*	*WSEAS*	*Springer*

	Bor-Yuan	**Didier Buchs**	**Peter**	**Shireesh**	**Francesca**	**Matthew Kaplan**

	Tsai [59]	**[60]**	**Frohlich [19]**	**Asthana [61]**	**Basanieri 20]**	**[64]**
Technique used	*State machines*	*UML class diagrams, UML collaboration diagrams, UML state charts*	*use case*	*Class and Sequence Diagrams*	*Use Case Diagrams and Sequence Diagrams*	*UML class diagram*
Type of testing	*Class Testing*	*Black Box*	*Black Box*	*Black Box*	*System and integration testing*	*Black Box*
Algorithm used	*Yes*	*Yes*	*Yes*	*Yes*	*Depth-First Search algorithm*	*Yes*
Tool used	*Yes*	*Yes*	*Yes*	*No*	*Cow Suite Tool*	*No*
Tool validated by user	*No*	*No*	*No*	*No*	*Yes*	*No*
Area covered	*Industrial Domain*	*Industrial Domain*	*Education Domain*	*Education Domain*	*Business Domain*	*Business Domain*
Case Study	*Car Alarm system*	*Mobile Phone*	*Library System*	*University*	*Course Registration System*	*Department application*
Test cases generated from	*Specifications*	*Specifications*	*Requirement*	*Specifications*	*Requirement*	*Specifications*
Test case selection	*?*	*?*	*?*	*?*	*Yes*	*?*

Coverage	*Transition path coverage*	*?*	*Yes*	*Yes*	*Functional coverage*	*?*
Number of activities performed	*5*	*4*	*2*	*3*	*5*	*4*
Year	*1999*	*2006*	*2000*	*2010*	*2002*	*2008*
Paper published	*Springer*	*Springer*	*Springer (ECOOP, LNCS)*	*Springer (IC3)*	*Springer (UML, LNCS)*	*STVR*

	Matthias Riebisch [21]	**Supaporn kansomkeat [41]**	**Santosh Kumar Swain [40]**	**Ming Song Chen [54]**	**Valdivino Santiago [23]**
Technique used	*use case models state diagrams*	*UML statechart diagrams*	UML Sequence activity diagrams	*UML activity diagrams*	Statecharts
Type of testing	*Black box*	*Black Box*	*Black box*	*Black Box*	*Black Box*
Algorithm used	*Yes*	*Yes (DFS)*	*Yes*	*Yes*	*Yes*
Tool used	*No*	*Yes*	*No*	*No*	*Condado*
Tool validated by user	*No*	*No*	*No*	*No*	*No*

Area covered	*?*	*?*	*Industrial domain*	*Industrial Domain*	*Research domain*
Case Study	*?*	*?*	*ATM*	*Online stock exchange*	*Space application software*
Test cases generated from	*Requirements*	*UML specification*	*Specification*	*Design Specification*	*Specification*
Test case selection	*?*	*?*	*?*	*Random*	*?*
Coverage	*?*	*Yes*	*Yes (path)*	*Yes (All)*	*?*
Number of activities performed	*3*	*3*	*4*	*4*	*4*
Year	*1998*	*2003*	*2010*	*2007*	*2008*
Paper published	*IEEE (ICIT)*	*ACM (SAICSIT)*	IJCA	*IEEE (TSE)*	*QSEE Project.*

	Jeff Offutt [67]	***A. Z.Javed [68]***	**M. Prasanna [70]**			
Technique used	*Collaboration Diagrams*	*Sequence Diagram*	*Collaboration Diagrams*			
Type of testing	*Dynamic testing Testing*	*Unit Testing*	*Dynamic testing*			
Algorithm used	*Yes*	*No*	*Yes*			

Tool used	*No*	*Yes*	*No*			
Tool validated by user	*No*	*Yes*	*No*			
Area covered	*Industrial Domain*	*Industrial Domain*	*Industrial Domain*			
Case Study	*Operation*	*ATM*	*Ticket System*			
Test cases generated from	*Specification s*	*LOC*	*Specifications*			
Test case selection	*?*	*?*	*?*			
Coverage	*Yes*	*Yer*	*?*			
Number of activities performed	*4*	*4*	*3*			
Year	*2000*	*2007*	*2011*			
Paper published	*Springer*	*IEEE*	*IETE*			

Conclusion

The overall objective of the study was to gather sufficient data to understand and gain deeper insights into the nature of the various testing techniques available and the possibility of improvements in them.

The need of new measures can be understood by first finding what the existing measures are actually capturing. The number of techniques proposed for test case generation is very large. Hence, a deeper insight on the techniques proposed is needed. We need to better understand the difference between these techniques and explore new methods to improve them. The need for new improved techniques will then arise from and be driven by, the result of such studies. Unified Modeling Language (UML) has now become a de facto standard in the field of software testing. New techniques for the generation of test case from these UML diagrams need to be explored.

Future Work

Software testing techniques are usually characterized as being black-box (or functional) or white-box (or structural) depending on whether they rely solely on a specification of the software under test (SUT) or solely on its implementation [4]. Authors also introduce the notion of grey-box testing that mixes information from the specification and the implementation to derive tests. Are these testing techniques efficient? Do they generate quality test cases? Is there any scope of improvement in them? Are there other techniques too? Listed below are few objectives which will form the base for future research:

- Survey existing test case generation techniques and areas covered by these techniques in order to draw strong conclusions from them.
- To find improvements in testing techniques.
- To explore the possibility of generation of test case generation from UML diagrams.
- To use genetic algorithm, evolutionary and optimization techniques for test case generation.
- To explore the possibility of automation of test case generation.
- To explore the possibility of use of formal methods in software testing

Hence, the next step in this field of research will involve surveying and finding new possibilities in this area. Further, possibility of automation in test case generation via UML diagrams will also be explored simultaneously. New approaches for generation of test cases for genetic algorithm, evolutionary and optimization techniques will be explored then. Hence, the next step would be to gain a deeper insight of testing techniques performed in this area.

References

[1] Heumann J. (2001) *Generating Test Cases from Use Cases*, Rational Software, IBM,

[2] Bird, D. L. Munoz, C. U. (1983) *Automatic generation of random self-checking test cases*, IBM Systems Journal

[3] Tsai, W.T. Volovik, D. Keefe, T.F. Fayad, M.E. (1988) *Automatic test case generation from relational algebra queries, Computer Software and Applications Conference.*

[4] Tsai, W.T. Volovik, D. Keefe, T.F. (1990) Automated test case generation for programs specified by relational algebra queries, Software Engineering, IEEE Transactions on software engineering

[5] Wang, C.J. Liu, M. (1993) *Automatic test case generation for Estelle*. International Conference on formal engineering methods, Network Protocols

[6] Ammann, P.E. Black, P.E. Majurski, W. (1998) *Using model checking to generate tests from specifications,* Second International Conference on *Formal Engineering Methods*

[7] Memon, A.M. Pollack, M.E. Soffa, M.L (1999) *Using a goal-driven approach to generate test cases for GUIs*, International Conference on Software Engineering

[8] Cunning, S.J. Rozenblit, J.W. (1999) *Automatic test case generation from requirements specifications for real-time embedded systems,* IEEE International

Conference on Systems, Man, and Cybernetics, vol 4, pp345-378

[9] Gutierez J., Escalona M.J. and Torres M.M. (2006) *An Approach to Generate Test Cases from Use Cases*, Proceedings of the 6th International Conference on Web Engineering, pp. 113-114.

[10] Hui L. and Hee B.K.T. (2006) *Automated Verification and Test Case Generation for Input Validation*, International Workshop on Automation on Software Test (AST'06), pp. 29-35.

[11] Nebut C, Fleurey F, Traon Y.L. and Jezequel J.M. (2006) *Automatic Test Generation: A Use Case Driven Approach*, IEEE Transactions on Software Engineeering, Vol 32, No. 3, pp. 140-155.

[12] Wee K.L., Siau C.K. and Yi S. (2004) *Automated Generation of Test Programs from Closed Specifications of Classes and Test Cases*, Proceedings of the 26th International Conference on Software Engineering (ICSE'04)

[13] Wei-Tek Tsai, Yinong Chen (2005) *WSDL-Based Automatic Test Case Generation for Web Services Testing*, IEEE International Workshop on Service-Oriented System Engineering

[14] Bor-Yuan Tsai, (2002) *An Automatic Test Case Generator Derived from State-Based Testing*, IEEE Trans. on Software Engineering, vol 7, pp781-812

[15] Chow, Tsun S. (1978, May) *Testing Software Design Modelled by Finite-State Machines*, IEEE Trans. on Software Engineering, Vol. SE-4, No. 3

[16] Turner, C. D.; Robson D. J. (1993) *The State based Testing of Object-Oriented Programs*, Conference on Software Maintenance

[17] Hoffman, D; Strooper, P., *ClassBench (1996-2003) A methodology and framework for automated class testing*, SVRC, University of Queensland; Software-Practice & Experience, Technical Report

[18] Proceedings of the Fourth International Conference on Quality Software (QSIC'04).

[19] Frohlick P. and Link J. (2004) Automated Test cases Generation from Dynamic Models, in the Proceedings of the European Conference on Object-Oriented Programming, Springer Verlag, LNCS 1850, pp. 472-491

[20] A. Bertolino, F. Basanieri, (2000) A practical approach to UML-based derivation of integration tests, in: Proceedings of the Fourth International Software Quality Week Europe and International Internet Quality Week Europe (QWE), Brussels, Belgium.

[21] Riebisch M., Philippow I, and Gotze M. (2003) UMLBased Statistical Test Case Generation, in the Proceeding of ECOOP 2003, Springer Verlag, LNCS 2591, pp. 394-411.

[22] Hartmann J., Vieira M., Foster H., Ruder A. (2005) A UML-based Approach to System Testing, Journal of Innovations System Software Engineering, Vol. 1, PP. 12-24

[23] Valdivino Santiago1, Ana Silvia Martins do Amaral1, N. L. Vijaykumar (2008) QSEE project

[24] F. Basanieri, A. Bertolino, E. Marchetti (2002) The cow suit approach to planning and deriving test suites in UML projects Proceedings of the Fifth International Conference on the UML, LNCS, 2460, Springer-Verlag GmbH, Dresden, Germany, pp. 383–397.

[25] Offutt J., Abdurazik A. (1999) Generating tests from UML specifications. Proc. 2nd Int. Conf. UML, Lecture Notes in Computer Science, Fort Collins, TX, Springer-Verlag GmbH, vol. 1723, pp. 416–429.

[26] Offutt J., LIU S., Abdurazik A. (2003) 'Generating test data from state-based specifications', Software, Testing, Verification, Reliability, Vol 13, pp. 25–53.

[27] Kansomkeat S., Rivepiboon W.(2003) 'Automated-generating test case using UML statechart diagrams'. Proc. SAICSIT 2003, ACM, pp. 296–300.

[28] Cavarra A., Crichton C., Davies J. (2004) 'A method for the automatic generation of test suites from object models', Information and Software. Technology, 46, (5), pp. 309–314.

[29] Hartmann J., Imoberdorf C., Meisinger M. (2000) UML-based integration testing, ACM SIGSOFT Software Engineering Notes, Proceedings International Symposium, Software Testing and Analysis, vol. 25.

[30] Kim Y.G., Hong H.S., Bae D.H.(1999) 'Test cases generation from UML state diagram', Proc. Software 146, (4), pp. 187–192.

[31] Zhenyu Dai, Mei-Hwa Chen (2007) Automatic Test Case Generation for Multi-tier Web Applications, 9th IEEE International Workshop on Web Site Evolution.

[32] Shengbo Chen, Huaikou Miao, Zhongsheng Qian (2007) Automatic Generating Test Cases for Testing Web Applications, International Conference on Computational Intelligence and Security Workshops.

[33] P. Samuel R. Mall A.K. Bothra (2008) Automatic test case generation using unified modeling language (UML) state diagrams, The Institution of Engineering and Technology.

[34] Yuan-Hsin Tung, Shian-Shyong Tseng, Tsung-Ju Lee, and Jui-Feng Weng (2010) *A Novel Approach to Automatic Test Case Generation for Web Applications*, 10th International Conference on Quality Software.

[35] Cao Xizhen Qian Hongbing (2010) *Research on test cases automatic generation technique based on AADL model*, 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE).

[36] Shahzad, A. Raza, S. Azam, M.N. Bilal, K. Inam-ul-Haq Shamail, S (2009) Automated optimum test case generation using web navigation graphs International Conference on Emerging Technologies.

[37] Shaoying Liu Nakajima, S. (2010) *A Decompositional Approach to Automatic Test Case Generation Based on Formal Specifications*, Fourth International Conference on Secure Software Integration and Reliability Improvement (SSIRI)

[38] Baikuntha Narayan Biswal (2008) *A Novel Approach for Scenario-Based Test Case Generation*, International Conference on Information Technology, vol 43, PP 244-247

[39] Philip Samuel Rajib Mall (2009) *Slicing-Based Test Case Generation from UML Activity Diagrams*, ACM SIGSOFT Software Engineering Notes, Volume 34 Number 6, Page 1

[40] Santosh Kumar Swain (2010) *Test Case Generation from Behavioral UML Models,* International Journal of Computer Applications, Vol 6, No.8.

[41] Kansomkeat (2010) *Generating Test Cases from UML Activity Diagrams using the Condition-Classification Tree Method*, 2nd International Conference on Software Technology and Engineering (ICSTE)

[42] Xi Wang, Liang Guo, Huaikou Miao (2008) An Approach to Transforming UML Model to FSM Model for Automatic Testing, International Conference on Computer Science and Software Engineering, IEEE, vol 34

[43] Binder R. V., Testing Object-Oriented System Models, Patterns, and Tools, Addison-Wesley, NY, 1999

[44] Emanuela G. Cartaxo, Francisco G. O. Neto and Patreıcia D. L. Machado (2007) *Test Case Generation by means of UML Sequence Diagrams and Labeled Transition Systems,*

[45] Byoungju Choi, Hoijin Yoon, Jin-Ok Jeon (1999*) A UML-based Test Model for Component Integration Test*, Workshop on Software Architecture and Component, Japan, pp63-70

[46] Zhang Mei, Liu Chao, Sun Chang-ai (2001*) Automated Test Case Generation Based on UML Activity Diagram Model*, Journal of Beijing University of Aeronautics and Astronautics(in Chinese), vol. 27 No. 4, pp 433-437

[47] Grade Booch, James Rumbaugh, Ivar Jacobson (2001) *The Unified Modeling Language User Guide*, Addison-Wesley

[48] Beizer (1995*) Black-box Testing: Techniques for functional testing of software and systems*, John Wiley & Sons, Inc, New York

[49] Paul C. Jorgrnsen (1995) *Software Testing: A Craftsman's Approach* , CRC Press Inc

[50] B. A. Kitchenham (2002) Preliminary guidelines for empirical research in software engineering,IEEE Test Selection from UML Statecharts, vol 28, pp 721-734

[51] M. Sarma, R. Mall (2009*) Automatic generation of test case specification for coverage of system state transition*, Information on software Technology, vol 51, pp 418-432

[52] Stefania Gnesi (2004*) Formal Test Case generation for UML state charts*, Proceedings of 9th International Conference on engineering complex

computer system navigating complexity, vol 34, pp 1050-4729

[53] Shinpei ogata and Saeko Matsuura (2010) A Method of Automatic Integration Test Case Generation from UML-based Scenario WSEAS transactions on information science and applications, Issue 4, Volume 7,

[54] Chen Minsong (2006) *Automatic test case generation for UML activity diagram*, AST, vol 78, pp456-478

[55] X. Hou (2010) *Integration testing system scenario generation based on UML*, International conference on computer, mechatronics, control and electronic engineering, vol 72, pp 271-273

[56] Li Liuying Qi Zhichang {1999) *Test Selection from UML Statecharts*, IEEE Test Selection from UML Statecharts, vol 99, pp 198-264

[57] Andreas Heinecke, Tobias Bruckmann, Tobias Griebe, Volker Gruhn (2010) *Generating Test Plans for Acceptance Tests from UML Activity Diagrams*, 17th IEEE International Conference and Workshops on Engineering of Computer-Based Systems, vol 14, pp 425-654

[58] Wang Linzhang, Yuan Jiesong, Yu Xiaofeng, Hu Jun, Li Xuandong and Zheng Guoliang (2004) *Generating Test Cases from UML Activity Diagram based on Gray-Box Method*, Proceedings of the 11th Asia-Pacific Software Engineering Conference (APSEC'04), pp 1362-1530

[59] Tsai, W.T. Volovik, D. Keefe, T.F. Fayad, M.E. (1988) *Automatic test case generation from relational algebra queries,* Computer Software and Applications Conference.

[60] D. Gelperin and B. Hetzel (1988) The Growth of Software Testing, Communications of the ACM, Volume 31 Issue 6, pp. 687-695

[61] Shireesh Asthana, Saurabh Tripathi, and Sandeep Kumar Singh (2010) *Novel Approach to Generate Test Cases Using Class and Sequence Diagrams*, Springer Verlag, CCIS 95, pp 155-167

[62] Supaporn Kansomkeat, Phachayanee Thiket, Jeff Offutt (2010) *Generating Test Cases from UML Activity Diagrams using the Condition Classification Tree Method*, 2nd International Conference on Software Technology and Engineering(ICSTE), vol *45, pp 456-489*

[63] Briand L. and Labiche Y. (2002) A UML-Based Approach to System Testing, in the Journal of Software and Systems Modeling, Springer Verlag, Vol. 1, pp. 10-42

[64] Matthew Kaplan, Tim Klinger, Amit M. Paradkar, Avik Sinha, Clay Williams, Cemal Yilmaz (2008*) Less is More: A Minimalistic Approach to UML Model-Based Conformance Test Generation*, International Conference on Software Testing, Verification, and Validation, IEEE, vol 69, pp 82-93

[65] Baikuntha Narayan Biswal (2008) *A Novel Approach for Scenario-Based Test Case Generation*, International Conference on Information Technology, vol 43, PP 244-247

[66] B. Beizer (1999) *Software Testing Techniques*, Second Edition, Van Nostrand Reinhold Company Limited, ISBN 0-442-20672-0

[67] Abdurazik, A. Offutt, J. Using UML Collaboration Diagrams for Static Checking and Test Generation UML 2000 - The Unified Modeling Language. Advancing the Standard. Third International Conference, York, UK, October 2000, Proceedings, Springer, 2000, vol 1939, pp 383-3.