Indian Journal of Science and Technology, *Vol 8(35), DOI: 10.17485/ijst/2015/v8i35/72881, December 2015* *ISSN (Print) : 0974-6846* *ISSN (Online) : 0974-5645*

Generation of Test Case using Automation in Software Systems – A Review

V. Maheshwari and M. Prasanna[*]

Department of School of Information Technology and Engineering, VIT University, Vellore - 632014, Tamil Nadu, India; maheshwari.v2014@vit.ac.in, prasanna.m@vit.ac.in

Abstract

Background: The main objective of this paper is to review the literature about automatic test case generation. Test Automation methods involves the following major factors for generating test case such as Unified Modelling Language (UML) diagrams, Testing types like Black Box Testing, White Box Testing, Testing techniques like model based, search based and symbolic execution, coverage criteria includes code based, fault based and function based, UML and Automation Testing tools and algorithms. **Suggestions and Conclusion:** Test Case Generation (TCG) can improve the performance of the automation in an effective way and some open research problems have been discussed for bio-inspired algorithms. This paper presents a systematic survey about Test Automation by optimisation search techniques, by novel techniques and various other approaches. The optimisation search technique provides the best solution of the problem definition. Hence test automation using optimisation approach gives an efficient test suite for the given problem model. Test case Generation are applied for various software systems, hardware systems, embedded systems, real time systems, nuclear safety systems.

Keywords: Automate Testing, Optimization Techniques, Software Testing, Test Case Generation, Unified Modelling Language

1. Introduction

Software testing provides the information about the quality of the product and the service under test. Testing techniques include program or application execution with the intent of finding defects[1]. The test case generation is one of the most relevant to improve the efficiency of the system and also it designs a set of minimal number of test cases such that it discloses several errors as early as possible. It is a most cost effective work in software development life cycle. Exhaustive testing is not possible in automation testing. In recent days, Intelligent Security and Automation System (ISAS) framework has been applied to an application[2].

According to IEEE Standards, the objective of software testing is the evaluation of a system or component through its execution, starting from initial conditions and observing the results. Testing can be done in two ways, one is manual and the other is automatic. Manual testing is a process to test the application manually by providing all possible inputs and generate the output based on that. It is most difficult and tedious work. Automatic testing is based on design models, which makes testers to free from unremarkable tasks and make them to focus on more creative tasks.

The rest of this paper is represents as follows: Section 2 describes background and elaborates better on the contributions of this survey on test case generation. Section 3 describes Mind map to start test automation, Section 4 provides test case generation related concepts. Section 5 describes about research challenges in TCG. Conclusions are presented in Section 6.

2. Background

The test case generation is widely discussed in both the software and hardware systems. There are several surveys on this area of knowledge have also been published, which are going to be described in this section.

The study presented by author is about some ATCG techniques[3] and particularly focused on specification

**Author for correspondence*

based and model based test case generation. Path oriented approaches and intelligent technique are quite complex to generate test case when compared to model based testing. In[4] elaborated a survey on the test case generation and test suites criteria.Test suites is a set of test sequence that covers the given test objective.In addition,a few open ended problems were also discussed related to test case generation techniques like Scenario based, Model based and Genetic based. In [5] presents the most notable testing techniques which are frequently used by experts like: Structural testing using Symbolic execution, model based testing, combinatorial testing, Random testing, Adaptive random testing, Search based testing.

Symbolic execution is a program analysis technique that evaluates a program's code in order to automatically create test data for the program. Symbolic variable is assigned with input arguments to execute the program rather than using concrete values. This technique is widely used for structural white box for Test case generation. Random testing is among the most fundamental and a popular testing technique. It is the only practically possible technique if the specification is tending to be imperfect and the source code is actually unavailable. Fault detection capability can be theoretically analysed. Adaptive random testing is a most effective and enhances the failure detection capability of random testing. It can achieve higher program code coverage on program structures and certain types of faults are detected. Search-based testing is used to automate the method of acquiring test information which maximises the achievement of test goals, although reducing testing costs. Mutation, crossover, Dynamic symbolic execution approaches are comes under search based testing, helps to achieve highest possible code coverage. Model-based testing can identify the uncertainty of specification of test suite and automate the test design as soon as possible, thus shortening the development life cycle, reducing development cost and improving software quality.

In elaborated a survey on the object oriented mutation testing. Faults have been discussed separately in object oriented features. Firstly, it was studied with focus on mutation testing tools and proposed techniques such as HAZOP (Hazard and Operability Study), Class mutation. Second, it was discussed that the selected mutation testing can reduce the computational cost and avoids similar mutants in large experiments[6]. An additional interest to be focused on inherent mutation testing and practical solutions is required to resolve them. The future work is to be carried out in evolutionary techniques. In[7] provided a survey on the UML state machine diagram that has been considered for generating test cases. Their work includes generating test data for concurrent state and events using UML state machine. The future direction includes generating test case for complex state machine with the soft computing techniques.

Various symbols are used in the Table1 to mention different meanings. For instance, a ✓ is used to represent that the given factor is actually included in the survey, while + or ++ are used to highlight that the particular consideration is paid to a specific issue. On the other hand, a less detailed discussion on a given factor is denoted by a -, while ✗ is used to represent factors certainly not included in the surveys Table 1.

Table 1. Comparsion of the test case generation works with the particular survey is presented here regarding

Survey	Year	Topics Focused	Coverage Criteria	Testing Techniques	Testing Approaches	Tools	Summary
M. Prasanna et al.[3]	2005	Specific based and model based TCG	-	✓	✓	✓	✓
R. Singh et al.[4]	2014	TCG, test suite criteria	+	++	+	+	✗
S. Anand et al.[5]	2013	Technique for ATCG	++	++	++	++	++
M. Bilal Bashir et al.[6]	2012	Mutation testing , OO features	✓	++	-	✓	✓
Manuj Aggarwal et al.[7]	2012	UML state machine, concurrent states and events	++	++	✓	✓	✓
This Survey	-	Several TCG related topics	✓	✓	✓	✓	✓

3. Mind Map to Start Test Automation

A mind map is methodically built to classify the main studies of test automation. The analysis factors used for test automation are outlined in Figure 1. This section presents the factors of ATCG which are designed for test automation. It includes coverage criteria measure, UML diagrams, testing techniques, algorithms, and testing tools.

3.1 Basic Flowchart of Test Case Generation

Figure 2 explains about the test case generation process in test automation. Where, at first, the inputs are given for test execution. It performs test execution if the given inputs are valid inputs. The test results thus generated

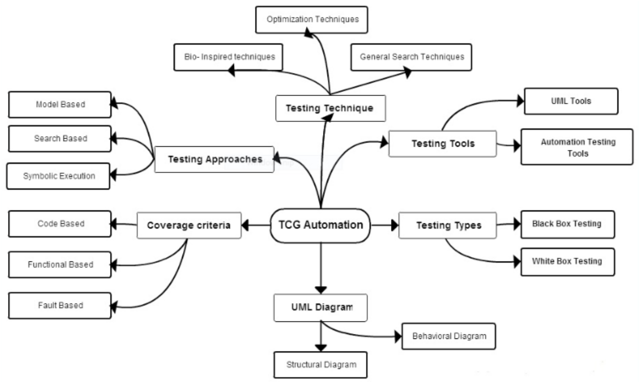

Figure 1. Factors to consider for analysing TCG.

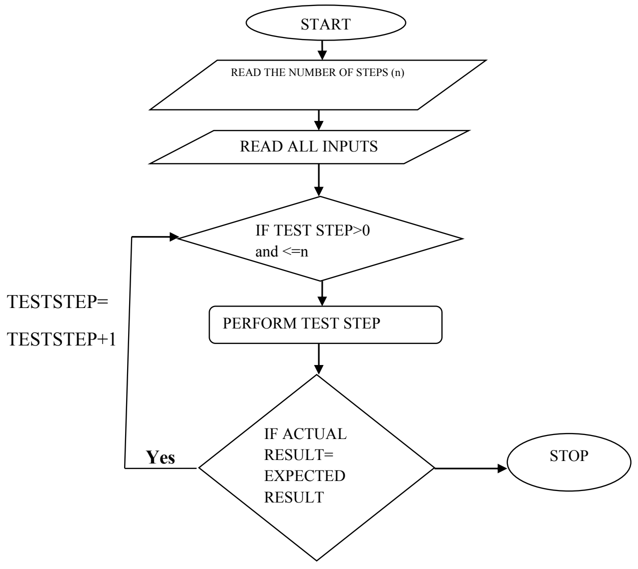

Figure 2. Represents test case generation process.

from the test execution is to be mapped with expected test results. If the results are true, the test execution iteration continues, else, the test execution is interrupted.

4. Test Case Generation-Related Concepts

In this section, fundamentals of test case generation is been presented. This particular section complements many of the ideas currently outlined in section 2, in order to understand the baseline of concepts.

4.1 Testing Techniques

Several studies introduce these search techniques. The search techniques act as an intermediate role between the test case and the problem domain. Optimization techniques are provided to obtain a best solution in a process. There are various kinds of search techniques which are carried out in most of the recent studies like Tabu search, simulated annealing, particle swarm optimization, ant colony optimization, genetic algorithm, cuckoo search, firefly and so on. These search techniques are responsible for finding out the test paths.

4.1.1 Optimisation Search Techniques

A new approach for pilot project 'Temperature Monitoring and Controlling of Nuclear Reactor Systems' (TMCNRS) to automate the test case generation by genetic algorithm. It is used to solve the complex problems like generation of efficient test suites by automated test case generator tool. To find the faults in the system, pseudo-exhaustive testing is used; it has smaller test suite size for output domain when compared to exhaustive testing of output domain. Test suites are enriched with seeded test cases, related to boundary value analysis of the input to obtain result.

In[9] proposed a genetic algorithm which is applied on tree crossover to represent possible test covers for object diagrams.Mutation analysis technique is to provide efficiency of the test case. The author derives test case using genetic algorithm and injects the faults in banking system. Through mutation score, the efficiency can be identified in unit level and also in integration. The author compared the faults of Aynur, Offutt approach and Genetic algorithm. In future, it can be worked out to other UML diagrams.

In[10] proposed a dynamic test case generation for object oriented programming classes using Grammatical

Evolution (GE); it is an evolutionary search that provides a result based on user-specified grammar. The specified grammar is obtained by translating the test case into Intermediate Test Script (ITS) format. It reduces the compiling time of source code, since it can run directly by ITS interpreter. Experiments result shows that, this method can produce high branch coverage outcome and reduces the search space of the problem when compared to Acuri's memetic method and Wappler's EvoUnit.

In[11] proposed a novel Adaptive Genetic Algorithm (ATG) to support test case generation of combination design. Combination-Index-Table (CIT) is described to process test case generation based on ATG and also to calculate the combination coverage based on t-wise strategy. Genetic Automatic Test case Generation (GATG) tool has a good user interface and performs a convenient test case design. The author compared GATG with similar tools, and shown that GATG provides average performance based on t-wise strategy. Performance and sensitivity of Adaptive Genetic Algorithm can be improved.

In[12] proposed a method for classes in object system to generate the test cases by genetic algorithm. To evaluate a suitable test case, first test cases are represented as a tree structure and encoded using objective function. The objective function is used along with GA to achieve the optimal results. The author describes the encoding and decoding of test program into data structures.

In[13] proposed SSO (Simplified Swarm Optimization) to generate optimized test suite with the use of Event Interaction Graph (EIG) for GUI functional testing. The author shows the effectiveness of SSO against other algorithms such as Pairwise Independent Combinatorial Testing (PICT), Test Vector Generator (TVG), Classification Tree Editor-eXtended Logics (CTE-XL), Intelligent Test Case Handler (ITCH) and In Parameter Order Generator (IPOG). The proposed strategy is applied in reliable artifact program to check out the correctness, feasibility and applicability So that unnecessary events will be removed that are presented in the test suite.

In[14] proposed a combinatorial test suite by using combinatorial optimization. A new one-test-at-a-time algorithm is to generate test case and translates into pseudo Boolean optimization problem, provides a maximum coverage for each test case. A self adaptive mechanism also proposed to stop the optimization process at a proper time. The proposed algorithm works well with large constraints when compared to existing approaches. Also, common constraints are translated into normal constraints. Hence, possible improvements are been focused on stopping mechanism, also in parameter, parameter levels and covering strength.

In[15] proposed a Match technique approach for object oriented programming to reuse the test case and also to speed up the development of test case. Match algorithm describes that, match the precondition of test case and turn the outcome of match into corresponding test case. Test cases can be described with distinct items. Test case = {Test case id, Test case description, Pre-test case id, Test context, Test input, Test expected result}. More test cases can be reused for better coverage in future.

In[16] presents an approach to find test sequence using object oriented slicing technique, it breaks the cycles into slicing classes for partial testing instead of removing relationship for test stub construction. The cost of implementing the test stub is decreased. Rather than testing with test stubs, slicing classes can detect more faults and reduce the timing in testing. The future scope includes, separating mutually related composite classes such as wrapper and content class. Sequence diagram is used to determine test order by sending sequences and method calls.

4.2 Automation Testing Tools

This section discusses a number of advanced automatic testing tools which generates test cases. Numerous automation testing tools came into existence. So in order to choose appropriate testing tool, the tester should first look into available testing tools to analyze their advantages, disadvantages, and constraints in terms of speed, time, cost and productivity. Mainly, it should cover maximum test case with minimum test paths.

Automation testing tools are used to reduce the manual interaction in unskilled and repetitive tasks. To build a software system, selection of appropriate tool is very important and a developer can do it. Following are the types of UML tools which are widely available[17]. Some list of open source UML modelling tools are: Umbrello, Astade, Fujaba, Agro UML and coral. There are several modelling tools which supports UML diagrams are Metricview, MagicDraw, IBM Rational Rose, AgileJ Structure Views, JUDE, BOUML gModeler, ConceptDraw PRO, Dia, Rhapsody, Modelistic, visual thought, LucidChart, Eclipse UML, Um-studio, SmartDraw, MetaEdit+, select component Architect, Visual Paradigm for the Unified Modeling Language (VP-UML), Sequence Sketcher, EctoSetModeller, ProxyDesigner, Jvision, iUML,Embarcadero De-Scribe, WinA&D, MacA&D, HAT (Hoora Analysis Tool), object domain, Selenium, Watir, Visual UML and together, Enterprise architect tool.

In[18] proposed an automatic test suite generator tool called FPCC Test Gen. First; this tool converts the functional block diagrams to UPPAAL for FPCC transformations which are done automatically through PLCopen XML. It covers the path complete conditions for functional block diagram using white box testing. Here the author shows the highest FPCC percentage with a near optimal number of test cases in Safety injection systems. Generic methodology is applied to test the coverage criteria. It mainly focused on data flow graphs and other coverage criteria. However, the calculation of FPCC is very difficult in computer aided tools.

In[19] proposed a GenRed tool for removing the redundant test cases without executing. The author presents approaches like input on demand creation, coverage-based method selection technique and sequence-based reduction technique. They worked on open source systems and the experimental results were highlighted. Compared to Randoop tool, GenRed increases high code coverage with minimum set of test case, test inputs and method call sequences. In future, they will work on mutant object instances by random testing.

In[20] proposed a new technique called Colored Petri Net's (CPN), which is the extended version of petri nets which was used to generate test case. For Net explosion problem, the author introduced a new algorithm to convert UML Statechart to CPN; it is able to cover all instances of objects from different classes in the same hierarchy to generate test cases by CPN-tools. They formed the generalization relationship between the classes. CPN-tools are used to analyse the behaviour of banking system account and it also generates the test case for the same. The CPN tool will be further applied to generate cover association and aggregation relationships.

In[21] have compared performance evaluation of automation testing tools. Here, they compared watir testing tool with selenium tool. Besides the combination of various automation tools, Selenium tool is one of the best known test suites which provide testers with different framework for different test cases. It is an open source and portable testing tool to test web applications. It provides a standardised way of writing selenium test suites. It supports test automation on diverse sets of web applications across domain. The framework permits users to perform acceptance, compatibility and functional testing for more web applications. Watir testing tool is an open source automatic testing tool. It is a library for ruby language which drives the browser. It requires programming skills. It is not a record and playback tool like selenium. They have evaluated and compared the performance of these testing tools. From their presented comparative results, it is clear that selenium test suite is better to that of water testing tool, where the selenium web driver is better choice in various conditions like use of domain specific language and framework.

4.3 Coverage Based

The main objective of Coverage criteria is to reduce the size of test suites and helps to measure the adequacy of test sets. Some commonly used coverage measurement criteria are: Code coverage, structure based and fault based.

In[22] proposed test adequacy criteria for class and interaction diagram .A systematic testing technique for testing executables forms of UML and test adequacy criteria based on UML model elements. Category partitioning technique detects the faults in test criteria where combination of test criteria has additional value in finding faults in UML diagrams.

In[23] have tested coverage criteria in web applications interaction. Coverage criteria are used in automation testing to measure how the program is exercised by a test suite. Their paper formally defines intra page interactions coverage; inter page interactions coverage criteria, page server interactions coverage criteria and scenario based page server interactions coverage criteria. Web applications are different from traditional software. Its features shows its complexity, reliability and quality which remains as a challenge to software testers.

4.4 Generation of Test Case by using Novel Technique

In[24] proposed a novel technique to generate test cases and mock classes using Pex. The author elaborates the survey on 35 third-party open-source applications; they found the branch that doesn't cover the code of supertypes, reflection and annotations. Hence the novel techniques are implemented to cover all the code. A current test case generation technique doesn't cover the branch of supertypes, reflection and annotations.

In[25] proposed a novel approach for generating test cases from UML design diagrams. The approach generates test case by the following ways. 1. Select the appropriate use case diagram and sequence diagram. 2. UML diagrams are transformed into UDG (Use case Diagram Graph) and SDG (Sequence Diagram Graph). 3. Integrating the UDG and SDG to form STG (System Testing Graph), it stores the information of test generation. 4. Finally generate test case by traversing STG. Test suite generation algorithm is used to automate traverse the STG and generates the test case according to coverage criteria. Use case diagram detects the test initialization and dependency faults where sequence diagram detects the operational faults.

In[26] proposed a novel method that automatically generates an assertion by converting from program dynamic invariants. This method includes three steps. 1. Convert invariants into assertions. 2. Test case generation algorithm based on assertions. 3. Efficiency analysis. Assertion method is another form of Invariant method. It is the specification of invariants. Conversion of invariant into its corresponding assertion is done by looking at the mapping table. The time of running assertion program is less than running invariants in Daikon (an open source tool). Testing the program invariants using Daikon tool builds more unacceptable test cases than appropriate test cases. Therefore using assertion program to determine test case validity can greatly reduce the time overhead. In comparison with invariant-directed and assertion-directed method, the author performed experiments on tested programs like maths programs and triangle, to establish the better time overhead and program coverage. Although the methods reduce the time overhead but still lexical analyzer can automatically convert common invariants into assertions. In future, they will be considering more factors in experiments.

In[27] proposed an approach of Extensic-based for ATCG from UML activity diagram. Extenics is a new methodology which deals with the contradictory problem with formalization methods and transformations. Euler circuit algorithm is constructed to generate test sequences automatically, it satisfies test coverage criteria and number of test case is decreased. The author experiments their methods in ATM system that it automatically reduces the number of test set and more faults were revealed. UMLTOTC tool is used to transform activity diagram into Euler circuit.This method covers only activity state coverage and transition coverage. Extenices will be improved in future for further development.

4.5 Generation of Test Case by Testing Approaches

In[28] introduced a method intended for automatic test case generation in Model Driven Engineering (MDE), which improves the generation of test systems directly from functional requirements. It derives a systematic process of test case by QVT transformations, Meta models. NDT, a model driven web approach was integrated with MDE is applied in all software projects. Where, NDT suite along with new tools is to be considered in future.

In[29] presented the quality of test by using OCL (Object Constraint Language) expression with established coverage criteria, which are more appropriate to specific-based test generation. In elevator control model, they applied a new coverage criterion i.e. Multi-Dimensional Condition Coverage (MDCC), Multi-Dimensional Decision Coverage (MDDC), Multi-Dimensional Modified Condition/Decision Coverage (MDMC/DC) and found the efficiency via mutation testing. During evaluation, ParTeG tool supports the control flow coverage criteria, but it handles only boundary and condition coverage. In future, they will work on possible combinations like boundary-based and control-flow.

In[30] presented static analysis feedback and unit test case generation in a Knit, a framework built on Kiasan symbolic execution engine. Unit leverage method supports mock object generation. Kiasan as contract checking tools that improves the quality through information of error trace. The author shown feasible branch coverage by using heap configuration and improved the coverage size and time to generate test suites in data structures.

In[31] have proposed an i-NUnit for unit test framework. A Test Data Container is been added into i-NUnit framework to separate test code and test data. The method MDA is been used to generate unit test case for i-NUnit through two model layer transformations i.e. horizontal and vertical transformation. The steps included for MDA-based automatic test case generation is, 1. Build a platform independent model for system under test. 2. Transform platform- independent model into i-NUnit model, which is been called as horizontal transformation. 3. Then i-NUnit is been transformed into test text, called as vertical transformation. The advantage of i-Nunit is to solve the redundancy problem and enhancing the test codes.

In[32,35] presented a critical analysis, possible improvements in UML behaviour diagrams and identified, analysed the issues of syntactic and semantic in use case, activity and state diagrams. They proposed an approach of capturing syntax and hidden semantics in diagrams. It links the UML diagrams and finite automata to be useful for correct and modeling of the systems. The author describes some pros, cons, limitations of the diagrams and their drawbacks in projects. The integration of approaches like UML, Automata theory and Formal methods will be useful for design and specification of software systems and facilitate the software development process.

In[33] describe test case prioritization, fault detection in regression testing. The factors of test case prioritization are explained in shortly such as traceability, completeness, fault impact of requirements, developer observed code implementation, changes in requirements and Customer-allotted priority. The better rate of fault detection is determining by proposed prioritization algorithm and also maximum priority test and minimum priority test has identified by comparing the test cases.

In[34] the challenge of an ambiguity test case will certainly minimize by proposed Adaptive Genetic Algorithm. To defeat the issues of random testing such as lengthy test case generation, identical test case for multiple times and also possible illegal inputs. Coverage metrics has used to detect the fault in test cases. The optimal inputs have generated by adaptive genetic algorithm to reduce the illegal and equivalent inputs generated by test cases. Hence, the uncertain test case is removed during random testing.

5. Research Challenges

This section presents research challenges in test case generation seen so far in terms of the testing techniques, coverage criteria and tools.

5.1 Open Ended Problems

The researchers may focus on Bio-Inspired algorithms, where still some algorithms such as Immune algorithms, probabilistic algorithms are not yet discussed. And also in physical algorithms like cultural algorithms, external optimization is not been focussed. In stochastic algorithms such as variable neighbourhood search, guided local search and iterated local search are to be discussed in future. Cloning testing methods has not been tested so far related to test case generation.

Numerous software tools are available to automate the testing. Mainly, automation tools are dependent on the execution speed. Moreover, the number of functionalities can be added to give better efficiency. Some tools are concentrating only on performance, speed, efficiency, accuracy and so research to be focussed on these parameters to develop a new tool. The generated test case from test models doesn't achieve all coverage criteria; it was particularly focused on some types like path coverage, boundary coverage and feature coverage. The more faults can be identified by coverage based method rather than by test suites.

6. Conclusion

In this survey, various factors for an automate test case generation is carried out to obtain better efficiency in testing. The hype of test case generation is pumping the software industry towards a productivity and efficiency. Researchers are primarily focused on single testing levels to reduce test suites where automatic test case generation through UML diagram focus more on behavioural and tend to ignore structural design of the system. The number of techniques, algorithms, test adequacy criteria proposed for test case generation is very large, herein this paper the important concepts of automation testing have been surveyed. These studies represent the existing testing approaches and were used to additionally check the critical-based testing.

7. Acknowledgment

A special note of thanks to Vellore Institute of Technology University for providing necessary infrastructure facilities to carry out the research work and all the ones who has directly and indirectly helped us to complete this survey work.

8. References

1. Desikan S, Ramesh G. Software Testing: Principles and Practice. Canada: Pearson Education; 2006.
2. Pandey M, Rajasekhara Babu M, Manasa J, Avinash K. Mobile based automation and security systems. Indian Journal of Science and Technology. 2015 Jan; 8(S2):12–6.
3. Prasanna M, Sivanandam SN, Venkatesan R, Sundarrajan R. A survey on automatic test case generation. Academics Open Internet Journal. 2005; 15(6).
4. Singh R. Test case generation for object-oriented systems: A review. IEEE 4th International Conference Communication Systems and Network Technologies (CSNT); Bhopal. 2014 Apr 7–9. p. 981–9.
5. Anand S, Burke EK, Chen TY, Clark J, Cohen MB, Kamp GW, Harman M, Harrold MJ, McMinn P. An orchestrated

survey of methodologies for automated software test case generation. Journal of Systems and Software. 2013 Aug; 86(8):1978–2001.
6. Bashir MB, Nadeem A. Object oriented mutation testing: A survey. International Conference on Emerging Technologies (ICET); Islamabad. 2012 Oct 8–9. p. 1–6.
7. Aggarwal M, Sabharwal S. Test case generation from UML state machine diagram: A survey. IEEE 3rd International Conference on Computer and Communication Technology (ICCCT); Allahabad. 2012 Nov 23–25. p. 133–40.
8. Vudatha CP, Jammalamadaka SK, Nalliboena S, Duvvuri BKK, Reddy LSS. Automated generation of test cases from output domain of an embedded system using genetic algorithms. IEEE 3rd International Conference Electronics Computer Technology (ICECT); Kanyakumari. 2011 Apr 8–10. p. 216–20.
9. Prasanna M, Chandran KR. Automatic test case generation for UML object diagrams using genetic algorithm. Int J Advance Soft Comput Appl. 2009; 1(1):19–32.
10. Chaiareerat J, Sophatsathit P, Lursinsap C. Test case generation for classes in objects-oriented programming using grammatical evolution. Computer Science and Convergence. Netherlands: Springer; 2012. p. 251–7.
11. Lin P, Bao X, Shu Z, Wang X, Liu J. Test case generation based on adaptive genetic algorithm. IEEE International Conference on Quality, Reliability, Risk, Maintenance and Safety Engineering (ICQR2MSE); Chengdu. 2012 June 15–18. p. 863–6.
12. Gupta NK, Rohil MK. Using genetic algorithm for unit testing of object oriented software. IEEE 1st International Conference on Emerging Trends in Engineering and Technology (ICETET'08); Nagpur, Maharashtra. 2008 July 16–18. p. 308–13.
13. Ahmed BS, Sahib MA, Potrus MY. Generating combinatorial test cases using Simplified Swarm Optimization (SSO) algorithm for automated GUI functional testing. Engineering Science and Technology: An International Journal. 2014 Dec; 17(4):218–26.
14. Zhang Z, Yan J, Zhao Y, Zhang J. Generating combinatorial test suite using combinatorial optimization. Journal of Systems and Software. 2014 Dec; 98:191–207.
15. Liu Z, Gu N, Yang G. An automated test case generation approach: Using match technique. The 5th International Conference on Computer and Information Technology; 2005 Sep 21–23. p. 922–6.
16. Jaroenpiboonkit J, Suwannasart T. Finding a test orders using object-oriented slicing technique. 14th Asia-Pacific Software Engineering Conference (APSEC); Aichi. 2007 Dec 4–7. p. 49–56.
17. Nagpurkar MS, Gurav MY. A survey on test case generation from UML based requirement analysis model. International Journal of Advancements in Research and Technology. 2013 May; 2(5):282–4.
18. YC W, Fan CF. Automatic test case generation for structural testing of function blocks diagrams. Information and Software Technology. 2014 Oct; 56(10):1360–76.
19. Jaygarl H, Lu KS, Chang CK. GenRed: A tool for generating and reducing object-oriented test cases. 34th Annual IEEE Computer Software and Applications Conference (COMPSAC); Seoul. 2010 Jul 19–23. p. 127–36.
20. Mirzaeian E, Mojaveri SG, Motameni H, Farahi A. An optimized approach to generate object oriented software test case by Colored Petri Net. IEEE 2nd International Conference on Software Technology and Engineering (ICSTE); San Juan PR. 2010 Oct 3–5. p. 251–5.
21. Angmo R, Sharma M. Performance evaluation of web based automation testing tools. IEEE 5th International Conference on the Next Generation Information Technology Summit (Confluence); Noida. 2014 Sept 25–26. p. 731–5.
22. Andrews A, France R, Ghosh S, Craig G. Test adequacy criteria for UML design models. Software Testing, Verification and Reliability. 2003; 13(2):95–127.
23. Song B, Chen S. Coverage criteria guided web application interactions testing. IEEE 3rd World Congress Software Engineering (WCSE). 2012 Nov 6–8. p. 46–50.
24. Islam M, Csallner C. Generating test cases for programs that are coded against interfaces and annotations. ACM Transactions on Software Engineering and Methodology (TOSEM). 2014 May; 23(3):21.
25. Sarma M, Mall R. Automatic test case generation from UML models. IEEE 10th International Conference on Information Technology (ICIT); Orissa. 2007 Dec 17–20. p. 196–201.
26. Zeng F, Deng C, Yuan Y. Assertion-directed test case generation. IEEE 3rd World Congress on Software Engineering (WCSE); Wuhan. 2012 Nov 6–8. p. 41–5.
27. Li L, Li X, He T, Xiong J. Extenics-based test case generation for UML activity diagram. Procedia Computer Science. 2013; 17:1186–93.
28. Gutierrez JJ, Escalona MJ, Mejias M, Ramos I, Torres J. An approach for model-driven test generation. 3rd International Conference on Research Challenges in Information Science; 2009 Apr. p. 303–12.
29. Weissleder S, Schlingloff BH. Quality of automatically generated test cases based on OCL expressions. IEEE International Conference on Software Testing, Verification, and Validation; 2008 Apr. p. 517–20.
30. Deng X, Hatcliff J, Kiasan K. Automatic test case generation and analysis feedback for open object-oriented systems. Testing: Academic and Industrial Conference Practice and Research Techniques-MUTATION (TAICPART-MUTATION); Windsor. 2007 Sept 10–14. p. 3–12.
31. Wang B, Zhu C, Sheng J. MDA-based automated generation method of test cases and supporting framework. IEEE 2nd International Conference on Computer Engineering and Technology (ICCET); Chengdu. 2010 Apr 16–18. p. 106–9.

32. Alhumaidan F, Zafar NA. Possible improvements in UML behaviour diagrams. IEEE International Conference on Computational Science and Computational Intelligence (CSCI); Las Vegas, NV. 2014 Mar 10–13. p. 173–8.
33. Chandu PMSS, Sasikala T. Implementation of regression testing of test case prioritization. Indian Journal of Science and Technology. 2015 Apr; 18(S8):290–3.
34. Koteswara RK, Raju GSVP. Developing optimal directed random testing technique to reduce interactive faults-systematic literature and design methodology. Indian Journal of Science and Technology. 2015; 8(8):715–9.
35. Jiang B, Chan WK. Input-based adaptive randomized test case prioritization: A local beam search approach. Journal of Systems and Software. 2015 Jul; 105:91–106.