LITERATURE REVIEW OF DIFFERENT TEST CASE GENERATION APPROACHES

Nargis Akhter [1] Amar singh [2]
Research Scholar
Baddi University of Emerging Science and Technology, Baddi, India

Abstract: It has been earlier known that software testing is one of the most predominant and major phase of software development life cycle ensuring the verification and validation process of the software. Test cases should have capability to cover more test objective features i.e., achieve more software coverage. The distinct method for automatic generation of test cases find trivial set of test cases decrease labors as well as budget of software testing and produce maximum effective and operating testing of software product. This analysis provides overview of diverse techniques of automatic test cases generation. The various methods are used to generate test case like UML Sequence Diagram, Scenario Based, BPEl Business Process, UML Activity diagram, AI planner and various computing techniques etc.

***Keywords* – Software Testing, Test Cases, Automatic Test Case Generation, UML Sequence Diagram.**

I. INTRODUCTION

Software testing is an imperious and dynamic part of the software development process [1]. Software testing is very man power consuming and costly process [2]. The testing efforts are categorized in to various parts: test case generation, test execution, test evaluation [1]. Test case generation is the fundamental part of any testing process and automating it save abundant time effort as well as decreases the number of inaccuracies and mistakes. An accurately created test suit may not only uncover the errors in a software system but also helps in dropping the high cost, efforts associate with software testing [3]. Software testing plays an essential role in high quality software development. It practices the application of artificial intelligence techniques. Soft computing is a developing methodology to computing, whose purpose is to exploit easiness for inaccuracy, ambiguity and partial truth to achieve robustness, submissiveness and total low cost [1]. Soft computing is a word applied to an arena with in computer science to find out the results to computationally –hard tasks such as the solution of NP-hard problems for which a precise solution should not be derived in polynomial time. There be present a number of techniques for producing test cases like fuzzy logic, finite state machine, soft computing, genetic programing and evolutionary computation [4]. There are some techniques of test case generation that be determined by on the application like test case generation for web application, object oriented application, structured based system, UML applications etc.

UML diagrams: from system requirements we can derive test cases and we can also derive test cases from use cases [5]. Some use case models are used for production of test cases. Use case Diagram Graph (UDG) is a diagram for originating test case that demonstrates all the possible case in the SUT.

Critical Path Method: T.Y. Chen et al [6] projected a framework for producing test cases i.e. critical path method. By using this technique we generate test cases by purifying functional choices.

Code based test Generation: [A. Mathur] test cases can be produced directly from the code which is called as code based test generation. This method decreases the size of the test suite, or prioritizes tests by supporting regression testing.

Dynamic Path Testing and Evolutionary Technique: it is dynamic path testing technique that engenders test case by implementing the program with different probable test case values. Korel also offered a methodology of test case prioritization established on path testing. With the help of random testing test cases are organized according to the smallest path [7]. J. Wegener and H.Sthamer generate test cases with the help of several structural test coverage testing criteria by evolutionary approaches like Genetic algorithm, which frequently generate test cases by using Meta heuristic method.

Graph Traversal Algorithm: By following breadth first OR depth first in a graph or tree test cases are generated by traversing from parent root to child node. Once all the vertices in a path are traversed from root to leaf it is considered to be one test case. All nodes would be enclosed to make sure that all bugs in an application are covered. One flow is known as single test case.

Genetic Algorithm: the most current exploration is being done on test case generation using Genetic Algorithm. An optimization technique like GA can be used to solve various problems. It uses existence of the fittest technique, where the best solution survives. GA include two testing methods i.e. mutation and crossover testing. Mutation holds mutants which are altered function that are applied on any logic or test case to generate new test case. Crossover testing is also used to produce test cases which comprises crossing over of two diverse use cases or test cases to generate a new use case.

II. LITERATURE REVIEW

Samuel et al [8] introduced a method to generate test case automatically s based on UML state machines diagrams. This technique produces test cases for testing class as well as cluster level behaviors. This method can switch change events and transition with guards and attain transition path coverage. They indifferent the no of test cases that attain transition path coverage by testing the borders determined by simple predicates. They practice the standard UML diagrams without demanding any additional formalism explicitly developed for testing resolution. UML is an established standard by ISO as well as OMG.

Boghdady et al [1] proposed an automated technique for producing test cases from one of the well-known UML diagrams which is activity diagram. This model used an algorithm that produce a table spontaneously known as ADT (activity dependency table), and then produce a directed graph known as ADG (activity dependency graph). The ADT is in a complete form that allows the created ADG covers entire features in the activity diagram. Lastly the activity dependency graph with activity dependency table is used to create the absolute test cases. This model save time and effort likewise, increases the quality of produced test cases.

Lefticaru et al [9] introduced an approach for automatic generation of test cases using state diagrams and genetic algorithms. The approach is simple, the derivation of the fitness function is straightforward and so could be easily implemented in industrial software development. Investigational indication show that the test data obtained can cover challenging path in the machine and a slightly different design of fitness function can be used for specification conformance testing.

Blanco et al [10] introduced a technique to create test cases automatically for BPEL business process with the help of Scatter Search metaheuristic technique. Transition coverage standards are used as acceptability principles. The business process is demonstrated and signified by a state graph. TCSS-LS procedure handles a set of solutions in each transition of the graph, therefore simplifying the division of the general goal in sub goals, and offers mechanism to handle the unfixed no of values of the input variables. The result obtained shows that TCSS-LS can be suitable to the test case generation of BPEL business process.

Mohi-Aldeen et al [11] proposed a Negative Selection Algorithm (NSA) that has been used to produce test cases spontaneously to justify path coverage of software. This algorithm most commonly used for benchmarking program that is triangle classifier and the tentative outcomes show the test cases are effective in time of execution and efficient in generation of test case.

Jalila et al [12] presented an approach to produce test cases from OCL (Object Constraint Language) formal specification by means of Category Partitioning Metrics (CPM). The trial results show that the proposed methodology is more effective in illuminating specification based faults. It has been perceived that OCL and CPM form and exceptional combination for performing functional testing at the primary stage to improve software quality with reduced cost.

Alsmadi et al [13] put forward a research in which they practice the concept of Genetic Algorithms to optimize the generation of test cases from application user interfaces. This is proficient through encoding the location of each control in the GUI graph to be distinctively represented and establishing the GUI controls graph. When test cases are generated the result i.e. binary sequence of its control is saved to be compared with future sequence. This is carrying out to ensure that the algorithm will generate a unique test case or path through the GUI flow graph every time.

Bird et al [14] introduced a technique to produce test case automatically for random software. The environment of such test cases make sure that they will execute to completion, and their execution is predicted at the time of generation. Self-checking test cases are produced. Predicted execution is compared with their performance at run time.

Swain et al [15] proposed innovative testing method for object oriented programs. They construct an intermediate demonstration based on the state and activity model which is known as State Activity Model (SAD). They create test cases to attain state activity

coverage of SADs. The results of this technique shows that we can detect seeded integration liabilities which is not detected in earlier methods.
Ali et al [16] presented a metaheuristic search approach which is widely used for generating test cases automatically, and thus providing solution for a more cost effective testing process. This method of test automation, frequently invented "Search Based software Testing "(SBST), has been used for a wide variability of test case generation purpose. They provide a framework which helps to originates the data collection process and also be the initial point of strategies on how (SBBT) techniques can be accessed.

K. Pal et al [17] proposed a rough –fuzzy hybridization arrangement for test case generation. Rough set theory is used to achieve dependency rules which model informative regions in the granulated feature space. Fuzzy set theory is used to characterize a pattern in terms of its membership to linguistic variables. This offer rise to proficient fuzzy granulation of the feature space. Since the rough set concept is used to acquire cases through crude rules case generation time is reduced.

Huang et al [18] presented that how to use symbolic methods to automatically produce test case for real- time systems. Activities in their test cases are categorized with symbolic timing constrictions whichever can be properly used in picking event occurrence time or be used for choosing boundary values in domain analysis. In their research test cases give description with region related coverage estimations which support high precision in detecting some timing bugs.

Memon et al [19] proposed a new method to generate test cases automatically for GUI that exploits planning, a well-developed and used method in artificial intelligence. Specified a set of operators, an initial state, a goal state, and a planner produces an arrangement of the operators that will transform the primary state to the objective state. The GUI model comprises of hierarchical scheduling operators representing the possible events in the GUI. The test designer defines the conditions and effects of the hierarchical operators, which are input into a plan-generation organization. The test designer also produces consequences that represent distinctive initial and goal states for a GUI user. The planner then produces plans representing arrangements of GUI interactions that a user might employ to reach the goal state from the initial state. They executed their test case generation system, called Planning Assisted Tester for Graphical User Interface Systems (PATHS) and experimentally evaluated its expediency and efficiency.

Lin et al [20] proposed a Genetic Algorithm that can automatically produce test case to test a particular path. This algorithm continues a specific path as a target and implements engagements of operators iteratively for test cases to develop. The generated test case can pointer the program execution to achieve the target path.

Kumar et al [21] represented a technique to form tree structure among class in design, established on their relationships. Depth First Searching (DFS) procedure is used to produce test cases from the binary tree formed. All necessities of data mining perception executed in well-known way to produce optimal number test cases. Data Mining is the practical method of identifying valid, fresh, possibly useful finally understandable patterns of data. Data mining procedures provision automatic assessment of data and tries to source out patterns and trends in the data and also determine rules from these patterns which will help the user to support review and examine conclusions in some associated business or scientific area.

Biswal et al [22] gives an approach in which test cases are produced from activity diagrams, which attain test adequacy standards flawlessly. They produce test cases by evaluating the respective sequence and class diagrams of each situation, which attain maximum path coverage criteria. In this method the cost of test generation is reduced as design is reduced.

Kundu et al [23] proposed a Model base test case generation technique. Benefit of this technique is error detection at initial stages, decreasing software development time etc. Using UML 2.0 syntax they generate tests case from activity diagrams and with use case scope. Test case generation using this method is capable of detecting additional faults like synchronization faults, loop faults.

Li et al [24] presented an approach of UML activity diagrams based on Extension Theory which is used for generating test cases automatically. In order to discovery more faults in software system by minimized test cases, they design algorithm to create the Euler circuit and produces test sequences automatically by Euler circuit algorithm. Their primary result shows that test cases generated not only fulfill the specified test coverage criteria but also the number of test cases is reduced.

Linzhang et al [25] presented a method to create test cases directly from UML activity diagram using gray-box method, where the design is reprocessed to sidestep the rate of test model construction in this technique test scenarios are directly derived from the activity diagram modeling an operation. Then all the information for test case generation is take out from each test scenario.

Mayrhauser et al [26] proposed a method for test generation in which an AI planner is used for test cases from test objectives which are derived from UML class diagram. The UML class diagrams are abstract models of the system under test. The developer uses the problem description to produce a test suit that gratifies the UML- derived test objectives.

Clarke et al [27] gives an ongoing lucent technologies experiment in automated test generation from a behavioral model of the software product under test. Result shows that their new methodology can increase the effectiveness of their testing while decreasing the cost of test design and generation.

Rajappa et al [28] proposed a method created on genetic graph theory to produce test cases for software testing. They will creäte a population of all the nodes of the graph as the base population of genetic algorithm. Then by using this population they can find a couple of node the parents and perform genetic crossover and mutation on them for the getting the optimum child nodes as the output. We have to continue this process of genetic procedure until all the nodes are covered or any of the nodes, which are visited more than once, should be rejected form the inhabitants. Then follow the same process for the production of test case in the real time system.

Cartaxo et al [29] introduced a systematic process for well-designed test case generation of feature testing for mobile applications. A feature is an addition of functionality, generally with a coherent purpose that is added on top of a basic system. Feature are typically developed and tested individually from the basic system as independent modules. The process is created on model-based testing methods with test cases produced from UML sequence diagrams transformed into (LTSs) Labeled Transition Systems.

Chuaychoo et al [30] proposed a technique using genetic algorithm for generating test cases. A test case generation tool is used with (Visual Basic .NET) for supporting their method. The proposed method was applied on case studies and mutation testing was used in their experiment for performance evaluation. The outcomes show that the produced test cases are improve the search proficiency, restrain precocity, promote case coverage, and decrease the number of iterations suitable for software testing.

Javed et al [31] proposed a method which is based on sequence diagrams. Initially they model the sequence diagram and then this model is spontaneously transformed into a general unit test case model using model-to-model transformations. Later that model to text transformations are smeared on the XUnit model to produce platform specific test cases that are tangible and executable.

III. CONCLUSION

Test case generation is one of the important concerns in software testing. An appropriately generated test suite may not only locate the inaccuracies in a software system, but also help in decreasing the high cost, labors associated with software testing. In this work we surveyed various techniques of software test case generation. First we summarized traditional approaches and advanced test optimization techniques, and then we identified gaps in existing techniques**.** This review will help researchers to classify what work has been done in their respective fields. The study shows that existing methods mostly concentrate only on the combination of behavioral diagrams. New techniques need to be identified for the generation of test cases from soft computing techniques. These techniques should include features of both behavioral and structural diagrams, so that they will have maximum coverage criteria and maximum fault detection capability. In the future, we have planned to develop an efficient test path method with minimum test case size/cost, and maximum code coverage, by integrating or improving the present condition or decision coverage standard.

REFERENCES

[1] Pakinam N. Boghdady, Nagwa L. Badr, Mohamed Hashem and Mohamed F.Tolba. A Proposed Test Case Generation Technique Based on Activity Diagrams. IJET, Vol 11 No 03, 2011.
[2] Manoj Kumar, Arun Sharma and Rajesh Kumar. Optimization of Test Cases using Soft Computing Technique: A Critical Review. WSEAS, page no 440-452, 2011.
[3] Priyanka Bansal. A Critical Review on Test Case Prioritization and Optimization using Soft Computing Techniques. ICRTNB, page 74-77, 2013.
[4] Itti Hooda and Rajender Chillar. A Review: Study of Test Case Generation techniques. IJCA, Vol 107- No 16, 2014.
[5] Philip Samuel, Rajib Mall, A Novel Test Case Design Technique Using Dynamic Slicing of UML Sequence Diagrams, e-Informatica: Software Engineering Journal, Vol 2, Issue 1, 2008.
[6] J, C. Huang. An Approach to Program Testing. ACM Computing survey, page no 113-128, 1975.
[7] Korel.B,L.Tahat and M.Harman, Test Prioritization using System Models, in proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM 2005), page 559-568, September 2005.
[8] P. Samuel, R. Mall and A.K. Bothra. Automatic Test Case Generation using unified modeling language state diagrams. IET Software, page 79-93, 2008.
[9] Raluca Lefticaru and Florentin Ipate. Automatic State Based test Generation Using Genetic Algorithm. IEEE 2007, page 188-195.

[10] Raquel Blanco, José García-Fanjul, Javier Tuya. A first approach to test case generation for BPEL compositions of web services using Scatter Search. IEEE 2009, page 131-140.
[11] Shayma Mustafa Mohi-Aldeen, Radziah Mohamad and Safaai Deris. Automatic Test Case Generation for Structural Testing Using Negative Selection Algorithm. IRICT 2014, page- 270- 280.
[12] A. Jalila and Jeya Mala. OCL-Based Test Case Generation Using Category Partitioning Method. ICTACT 2015.
[13] Izzat Alsmadi. USING GENETIC ALGORITHMS FOR TEST CASE GENERATION AND SELECTION OPTIMIZATION, Electrical and Computer Engineering 1-4, 2010.
[14] D.L. Bird and C.U. Munoz. Automatic generation of random self-checking test cases. IBM SYSTEMS JOURNAL 1983.
[15] Santosh Kumar Swain, Durga Prasad Mohapatra and Rajib Mall. Test Case generation based on State and Activity Models. JOT 2010.
[16] Shaukat Ali, Lionel C. Briand and Hadi Hemmati. A Systematic Review of the Application and Empirical Investigation of Search-Based Test Case Generation. IEEE 2010.
[17] Sankar K. Pal and Pabitra Mitra. Case Generation Using Rough Sets with Fuzzy Representation. IEEE, page 293-300, 2004.

[18] Jin-Cherng Lin and Pu-Lin Yeh. Using Genetic Algorithm for Test Case Generation Using Path Testing. IEEE, page 241-246, 2000.
[19] Geng-Dian Huang, and Farn Wang. Automatic Test Case Generation with Region-Related Coverage Annotations for Real-Time Systems. Springer, page 144-158, 2005.
[20] Memon, Atef. Hierarchical GUI test case generation using automated planning. IEEE Transactions on Software Engineering, page 144-155, 2001.
[21] A.V.K Shanthi, DR.G.Mohankumar. Automated Test Case Generation for Object Oriented Software. IJCSE, page 543-546, 2011.
[22] Baikuntha Narayan Biswal, Pragyan Nanda and Durga Prasad Mohapatra. A Novel approach for Scenario-Based Test Case Generation. ICIT, page 244-247, 2008.
[23] Debasish Kundu and Debasish Samanta. A Novel Approach to Generate Test Cases from UML Activity Diagram. JOT, page 65-83, 2008.
[24] Liping Lia, Xingsen Lib, Tao Hec, Jie Xiongd. Extenics-based Test Case Generation for UML Activity Diagram. Elsevier, page 1186-1193, 2013.
[25] Wang Linzhang, Yuan Jiesong, Yu Xiaofeng, Hu Jun, Li Xuandong and Zheng Guoliang. Generating Test Cases from UML Activity Diagram based on Gray-Box method. APSEC, page 284-291, 2004.
[26] Anneliese von Mayrhauser, Senior Member, IEEE, Robert France, Member, IEEE, Michael Scheetz, and Eric Dahlman. Generating Test-Cases from an Object-Oriented Model with an Artificial-Intelligence Planning System. IEEE Transaction on reliability 49(1), page 26-31, 2000.
[27] James M. Clarke. Automated test Generation from a Behavioral Model. IEEE Press 1998.
[28] Dr. Velur Rajappa, Arun Biradar, Satanik Panda. Efficient Software Test Case Generation Using Genetic Algorithm Based Graph Theory. ICETET, page 298-303, 2008.
[29] Emanuela G. Cartaxo, Francisco G. O. Neto and Patr´ıcia D. L. Machado. Test Case Generation by means of UML Sequence Diagrams and Labeled Transition Systems. IEEE, page 1292-1297, 2007.
[30] Nuntanee Chuaychoo, Supaporn Kansomkeat. Path Coverage Test Case Generation using Genetic Algorithms. JTEC, page 115-119,2017
[31] A. Z. Javed, P. A. Strooper and G. N. Watson. Automated Generation of Test Cases Using Model-Driven Architecture. IEEE 2007