THINK INDIA JOURNAL

ISSN: 0971-1260
Vol-22-Issue-17-September-2019

Test Case Generation Techniques

Anik Acharjee[*], Dr. Amar Singh[*]

[*]School of Computer Science and Engineering, Lovely Professional University, Phagwara, Jalandhar, India

***Abstract*— Software testing is tedious and intensive task in Software Development Life Cycle. It includes verification of requirement specification, analysis, quality improvement and many more. In recent survey, it has been found that maximum amount of the time, cost and effort are devoted for Software Testing. To analyze and debug the desired software, manual testing is ideal for small-level software. To test the complex and large-sized software, automatic testing can be preferred to reduce the time, effort and cost. Various advanced techniques like genetic algorithms, evolutionary computation, soft computing etc. have been already used to generate adequate and prominent test cases. In this paper, various test case generation techniques have been reviewed and analyzed for better improvement of the software testing.**

***Index Terms*— Test case generation techniques, testing, software testing, test cases,**

I. INTRODUCTION

Testing phase plays an ideal and prominent role to validate the requirement specification, analysis, design and finally code as per the user requirements. Due to which, the size of the software is gradually increasing. In the past decades, the Line of Code (LOC) is quite high. Such an extraordinary rise in the size of the software has created various issues for the testers. In every industrial and research organization, more stress are imposed to discover the efficient automatic software testing techniques. These techniques can be used to increase the quality of the software that needs to delivery to the end-users or customers. These techniques can be utilized to manipulate test cases for software testing purpose. The test cases can be manipulated automatically by using some test tools or writing the test cases manually. In recent times, various adequate techniques are used. Basically, there are two approaches for generating test cases using such techniques. The first approach, i.e., designing test cases from requirements and design specifications is much more adopted as compared to the second approach. The second approach, i.e., design test case using code is bit complex and composite.

ISSN: 0971-1260
Vol-22-Issue-17-September-2019

In the literature, it is found that test case generation techniques can be categorized into two parts – Hard Computing based Test Case Generation Techniques and Soft Computing based Test Case Generation Techniques. The former includes Specification-based Test Case Generation, Sketch Diagram-based Test Case Generation and Source Code-based Test Case Generation techniques. The latter one includes Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO) and many more. It can be illustrate with a suitable diagram, as follows –

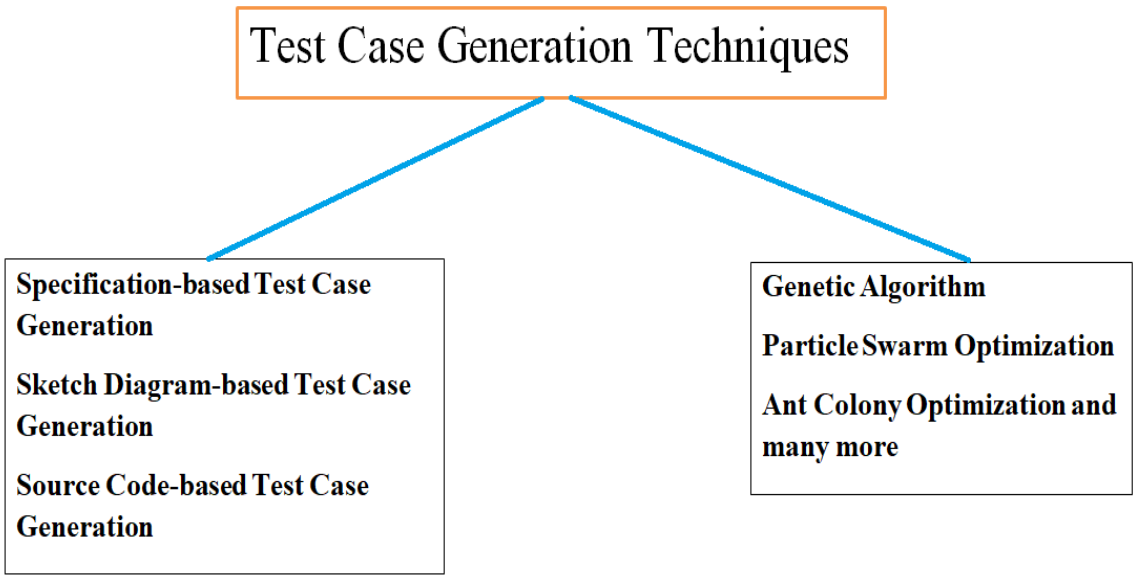

Figure - 1

In this paper, all such techniques are reviewed and analyzed properly to know and understand of how to generate the test cases automatically.

II. LITERATURE SURVEY

Specification-based Techniques are basically used to manipulate test cases from formal requirements specification [1]. Here, specification documents are used more often. It elaborates what the system will perform without elaborating how it will work. The software engineer knows all the functionality work of the software. Such specification documents can be utilized to inherit the desired output for the test data. If testing is done at the initial stage of the software development, it can reduce the major costs of testing. Generating the test from the specification will help the test engineers to find out the problems and gradually save time and resources. Some of the drawbacks of such technique are sometimes, it is difficult to analyze the formal documents which directly or indirectly affects the project budget. More

ISSN: 0971-1260
Vol-22-Issue-17-September-2019

manual efforts are required to manipulate and generate the test cases as contrast to automatic generation processes.

Sketch diagram-based Techniques are basically used to manipulate test cases by using UML Use Case diagram [1]. Many of the sketch diagram-based techniques are used for traditional and web-based applications. Use Case can be used to generate test cases to initiate testing process [2]. The author in [2] proposed three steps – (1) manipulate use-case scenarios (2) locate at-least single test case for each scenario and (3) identify data values for testing. Nilawar et. al. have mainly tested web-based applications [3]. The author in [3] focused on black box testing. This testing allows the test engineers to derive sets of input conditions. The author highlighted that the black box testing is the most fit and desirable to test web-based applications rather than any other applications.

Source Code-based techniques basically use the control flow information which can be used to recognize the set of paths to be covered completely for testing and then manipulate the test cases for those paths [1].

The author in [4] conveyed that software testing approaches can utilize in identifying the occurrence of errors or bugs in the developed software. Here, the challenges of software testing are discussed and how software testing techniques can be evolved gradually are also discussed. Generating the effective test cases is a difficult task. Due to which, there is a need of computerized test case generation and heuristic techniques required.

K.P.Yadav et. al. in [5] focused on different types of challenges faced for generating test cases by the testing team and UML techniques can be used for such challenges. The main activity in software testing is to produce or generate test cases. Since such tasks has become the critical work, so we can apply new and adequate techniques to have better results. The author focuses that automation of test case manipulation can be done in the initial stage of Software Life cycle. This can reduce such challenges in generating test cases.

The author in [6] highlights the problem of test case generation. Previously, Particle Swarm Optimization (PSO) can be used for generating test cases. But it leads to various problems like population diversity, low accuracy etc. Due to which, a self-adaptive PSO based software testing case optimized algorithm is suggested. It overcomes the performance and quality of standard PSO.

Sheng Y et. al. in [7] highlights AI (Artificial intelligence)-based algorithm. According to author, AI-based algorithms can accomplish superior than greedy-based algorithms. The two

ISSN: 0971-1260
Vol-22-Issue-17-September-2019

different methods are – PCTG-Av , it circumvent the option of clashing test cases and PCTG-Re, it replaces the conflicting test cases.

The author in [8] discussed the cost of testing in the software development life cycle (SDLC). The cost of the testing in the maintenance phase is quite high, with compare with the total cost. The two popular techniques i.e. test case selection and the test case prioritization can be used for the optimizing testing techniques. These techniques can be incorporated with different soft-computing algorithms for better results. Modified Ant Colony Optimization (ACO) can be used for better test case generation. It will result in finding out the maximum fault in minimum time.

The author in [9] highlight about the automatic test case generation. Test cases are manipulated automatically, test errors are eliminated gradually. In order to reduce the size of test suite, combinatorial testing has been suggested. It provides high reliability compare to others.

Naveen Shaillu et. al. in [10] discussed software testing is expensive process. Generating automatic test case data, i.e., automated test case manipulation is the main stage of software testing phase. The author focuses on Parallel Big Bang-Big Crunch soft computing approach for generating automatic test case generation. It has been performed on four real world programs like Is Prime (ISPRIME), Triangle classifier (TC), Quadratic Equation Program (QUAD)and Car Brake Controller System (CBCS). It has been found the PBB-BC based approach outperforms BB-BC based approach for automatic test data generation.

The author in [11] presented efficiency of path-wise testing can be enhanced by the manipulation of the diverse-path test cases. To improve such efficiency, a new study has been proposed. Here, particle swarm optimization (PSO) algorithm with metamorphic relations (MRs) can be used. Firstly, test case is generated using PSO algorithm and secondly, fresh test cases are generated frequently using MRs between test cases.

The author and his team in [12] focused on regression testing. We can generate test cases by using regression test case generation for the modified code. For that, Multi-agent systems for regression test case manipulation by using standard UML models can be used.

Nargis Akhter et. al . in [13] focused on how to achieve more test coverage. Generating test case generation with more test coverage will be the effective method. Due to which, P3PGA algorithm is used. It not only improves the quality of the test data, but also make it much more efficient and effective to cover the maximum paths.

ISSN: 0971-1260
Vol-22-Issue-17-September-2019

Miranda MA et. al. in [14] focused on UML Model. The author proposes the implementation of language of use case to automate models. A specific tool has been used, termed as LUCAMTool. Tests has been performed, both in real as well as simulated environments.

Panichella A et. al. in [15] focused on how to cover multiple test branches. Existing approaches either focus on single target at a time or cluster all targets. To overcome this problem, the author proposed Dynamic Many-Objective Sorting Algorithm. This will resolve the problem of branch coverage.

Bo L et. al. in [16] highlights on generating test cases automatically. Existing test generation techniques consumes lots of time and effort. Due to which, an automatic and efficient approach is presented. This approach helps in manipulating test cases for thread-safe classes. It signifies in improving ability without spoiling bug finding capacities.

The author and his team in [17] explained about automated test suite generation (ATSG). ATSG is one of the important and crucial topics of software engineering. The author implemented a tool called mapping the effectiveness of test automation (META). This tool can be basically used effectively to select worthy ATSG techniques. These techniques can be applied to new software systems.

Liu F et. Al. in [18] focused on automatic test data generation for path coverage (ATDG-PC). This can be applied to Cloud Computing like Hadoop programmes. These programmes are difficult to find out the high-rate path coverage. This can be used to reduce test cases for path coverage, as compare to other metaheuristic algorithms.

Gupta N et. al. in [19] highlights on software testing. It can be used and enclosed as an optimization problem to re-solve problems for enormous optimizing techniques. The author also presented on test case manipulation, election, depreciation and prioritization of testing.

Yousaf N et. al. in [20] focused on testing the web applications. Testing the web applications is bit composite and stagnant process. Interaction Flow Modeling Language (IFML) can be used for UI (User-Interface) testing. Later on, the author introduced novel model-based testing approach. This can be used for IFML. This approach furnishes test cases at the initial stage of the development.

Huang H et. al. in [21] discussed about the computerized test case manipulation based on path coverage (ATCG-PC). It will reduce the cost of generating test cases. The author compared this approach with classical differential evolution (DE). It is difficult for DE to generate test cases. This achieves higher path coverage rate .

ISSN: 0971-1260
Vol-22-Issue-17-September-2019

Liu P et. al. in [22] discussed about MTTool. It is model-based test tool. It can be used for test modeling and generating test cases from the model. This tool can be used for Extended Regular Expression (ERE)-based testing.

Feyzi F et. al. in [23] focused on the Bayesian networks. The author presents a unique test data generation called Bayes-TDG, which is based upon the principles of Bayesian networks. It can be used for path-coverage ratio for a particular programme under test (PUT). The author proposed path selection strategy for generating test cases. The method is adequate and provide effective generated test suite.

Bian Y et. al. in [24] discussed on regression testing. The author introduced Epistatic Test case Segment (ETS). It can be used for multi-objective search-based regression Test Case Prioritization (MoTCP). The author proposed ETS-based pheromone update strategy. The proposed can be utilized to improve the performance of ACO. This is respect to efficiency and effectiveness for search-based MoTCP.

Mukherjee R et. al. in [25] highlights about Test Case Prioritization (TCP). TCP techniques are used in Object-Oriented Programs (OOPs), so author tried to study on prioritization of JUnit test cases. The author has examined that multi-objective Genetic Algorithm (GA) performs well as compared to single objective prioritization.

Bashir MB et. al. in [26] explored on mutation testing. The mutation testing is bit lavish as it prerequisite enormous mutants. In this scenario, different search-based approaches or algorithm identical to Genetic Algorithm (G.A) are used. It will assist to automatic test case generation to shorten the amount. The author explored that the genetic algorithm can provide optimum test cases in minimal amount of strives.

Jan S et. al. in [27] have generated automatic tests in Web Applications. Automatic test case can be generated in Web Applications by using XML. It is basically accomplished by automatically generating XML injection attacks.

Li H et. al. in [28] explored about the generation of test data. It is the sole of the vital factors in Software Testing. Sometimes, a well-manipulated test suite cannot be identifying the bugs or errors in the software system. It might able to reduce the cost incorporated with software testing.

Hooda I et. al. in [29] focused on generating test cases. Software testing is one of such task, which takes huge time and effort in developing the software and its quality improvement. To test the complex and composite software, testing needs to be done automatically. Different and varieties of tools are available, which can be used for such purpose. The author

ISSN: 0971-1260
Vol-22-Issue-17-September-2019

highlighted on various techniques like neural networks, fuzzy logic, soft computing, evolutionary computation and so on, which can be utilize to manipulate test cases automatically.

Windisch A et. al. in [30] tries to utilize the Genetic Algorithms (GAs) to explore and find out appropriate and similar test cases. The author tries to compare Particle Swarm Optimization (PSO) with Genetic Algorithms (GAs), with refer to the evolutionary structural testing. Various experiments have been executed whose result shows that PSO outperforms GAs.

Rodriguez J et. al. in [31] focused on GUI (Graphical User Interface) test cases. GUI test cases should be of good quality, which can be applied for application testing. Various parameters like coverage criteria and partial coverage can be used to measure the test cases (or set of test cases). The author has used Ant Colony Optimization algorithm for creating and generating test cases.

Nayak N et. al. in [32] highlights on evolutionary structural testing. Manipulating the test case automatically is sole of biggest circumstances. An approach termed evolutionary structural testing can be used computerized. It basically uses Genetic Algorithm to accomplish such activities. It has been found that Particle Swarm Optimization (PSO) surpasses the GA. Such approach can be used for data flow testing.

Arifiani S et. al. in [33] suggested Ant Colony Optimization (ACO) as one of the best search algorithm. The author suggested statistical testing technique can be applied on the Gray Box Testing using the ACO algorithm. A comparison is carried out between the UML State Machine Diagram and the source code to manipulate test case using ACO statistical testing.

Wang Z and his team in [34] focus on automatic manipulation of the software test case. This is sole of the vital and decisive parts in Software Testing. The technology used for Software Test Case Generation automatically has been proposed in this paper to improve and enhance the quality of software testing. This technology is imposed along with the advanced PSO algorithm. The result shows lower time cost and high performance in automatic manipulation of the software test case.

Khan R et. al. in [35] highlights on automatic software testing process. Various and enormous research and experiment has been carried out in automatic software testing process. The author presented a Hybrid Genetic Algorithm for manipulating test data computerized. This can be done for mutation testing.

ISSN: 0971-1260
Vol-22-Issue-17-September-2019

Khan R et. al. in [36] highlights test cases needed for the process of software testing. In the survey, it has been found that lots of money used to be invested for software process. Various test cases are utilized as input to test the software process and check for the final output. So, generating the test cases has been pointed out as one of the NP problem. So to accomplish such task, enormous nature inspired optimization algorithms are utilized. Here, the author focused on genetic algorithm for manipulating automatic test cases.

Gongge G et. al in [37] discussed about test case auto-generation model. This model can be used to generate test cases. It is possible basically on the basis of knowledge such as case studies, rules and etc.

Lin P et. al. in [38] proposed a noble algorithm called adaptive genetic algorithm (AGA). This can be utilized for test case generation. Firstly, this algorithm is used for generating test cases. Secondly, a tool called genetic automatic test case generation (GATG) is used for the task of generating test cases. Both the adaptive genetic algorithm and the automatic tool are proposed in this paper for generating test cases.

Liu A et. al. in [39] highlights on the input variables that are used in the function. Proper analysis needs to be done for function call path. Various input variables are used in the function which can be used for automatic test cases. Sometimes, different variables in test cases are used, compare to the input variables. To have efficient and accurate test cases, proper information needs to be extracted from the function call diagram and control conditions.

Chen L et. al. in [40] presented state based model called IFA (interaction finite automation). It analyses the system's behaviours, listed in use cases. The algorithm for evolution from use cases to IFA and creating test cases from IFA is also presented in this paper. This process can be done automatically.

Mirzaaghaei M et. al. in [41] focused on generating test cases during software evolution. Due to changes in the code, it is not possible to manipulate or generate the test cases properly. Here, author proposed automatic repairing and generating the test cases in the whole software evolution. This will help in expense and time consuming activity. It can be used for test suite evolution.

Khan R et. al. in [42] highlights on path testing. Automatic test case generation can be optimized by using genetic algorithm (G.A.). Various software industries invest lots of wages to test the software testing process. To reduce such huge expenses, a method has been

ISSN: 0971-1260
Vol-22-Issue-17-September-2019

proposed with genetic algorithm. It tries to check out for path coverage by using a set of inputs.

Automatic test case generation can be accomplished by using mutation testing in [43]. The paper summarized various technologies used for mutation testing for test data generation. It focuses on time, cost and code coverage.

Li K et. al. in [44] presented a model for manipulating test data by utilizing upgrade ant colony optimization and path coverage criteria. Path testing can be used for identifying bugs since it performs well for higher error coverage. Several experiments have been carried out and it has been found that this mechanism can be carried out for improving the efficiency of test data generation.

Sayyari F et. al. in [45] proposed a solution based upon ant colony optimization algorithm and model-based testing. It can be used for developing test paths faster with minimum time, cost and maximum coverage.

Biswas S et. al. in [46] proposed ant colony optimization based algorithm. This can manipulate bunch of optimal paths and prioritize paths properly. It can also generate the test data sequence. This approach guarantees full software coverage with minimal redundancy.

Arifiani S et. al. in [47] showed that ant-colony optimization (ACO) algorithm outperforms other algorithms like search algorithm, genetic algorithm etc. It can be used for generating quality test data and its firmness. This approach can be used for Gray Box testing.

Prajapati N et. al. in [48] focused on Component-Based Software Engineering (CBSE). The Ant-Colony Optimization algorithm can be utilized in such mechanism. It can be utilized for optimizing auto-generated code and prioritize optimal path. This can enhance the testing phase with less complexity. The same algorithm can be used for generating the test data for Quality Assurance (QA) based model, presented for CBSE.

III. Conclusion –

In this paper, various techniques have been reviewed. Such techniques can be classified into two parts – Hard Computing based Test Case Generation Techniques and Soft Computing based Test Case Generation Techniques.

It is found in the literature that Soft Computing based Test Case Generation techniques perform better than the Hard computing based techniques. Various soft computing based algorithms like GA, ACO etc. are used for test case generation automatically. Other soft-computing based algorithms can also be used for generating the test case generation at the initial stage of the software development.

ISSN: 0971-1260
Vol-22-Issue-17-September-2019

REFERENCES

[1] Kosindrdecha N, Daengdej J. A test case generation process and technique. J. Software Eng. 2010;4:265-87.

[2] Heumann J. Generating test cases from use cases. The rational edge. 2001 Jun;6(01).

[3] Nilawar M, Dascalu S. A UML-based approach for testing web applications. Master of Science with major in Computer Science, University of Nevada, Reno. 2003 Aug.

[4] Jain N, Porwal R. Automated Test Data Generation Applying Heuristic Approaches—A Survey. InSoftware Engineering 2019 (pp. 699-708). Springer, Singapore.

[5] K.P. Yadav, Saroj Patel, Tannu Arora. Challenges in Automatic Test Case Generation. Volume 1, 2016. International Journal of Communications.

[6] Jianqi S, Yanhong H, Ang L, Fangda C. An optimal solution for software testing case generation based on particle swarm optimization. Open Physics. 2018 Jan 1;16(1):355-63.

[7] Sheng Y, Wei C, Jiang S. Constraint test cases generation based on particle swarm optimization. International Journal of Reliability, Quality and Safety Engineering. 2017 Oct 20;24(05):1750021.

[8] Sushant Kumar and Prabhat Ranjan. ACO based test case prioritization for fault detection in maintenance phase. International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 16 (2017) pp. 5578-5586

[9] Dr. V. Chandra Prakash, Subhash Tatale, Vrushali Kondhalkar , Laxmi Bewoor. A Critical Review on Automated Test Case Generation for Conducting Combinatorial Testing Using Particle Swarm Optimization. International Journal of Engineering & Technology, 7 (3.8) (2018) 22-28

[10] Naveen Shaillu, Dr. Amar Singh, Rajesh Chaudhary. AUTOMATIC TEST DATA GENERATION BY PARALLEL BIG BANG-BIG CRUNCH. Journal of Emerging Technologies and Innovative Research (JETIR) July 2018, Volume 5, Issue 7

[11] Lv XW, Huang S, Hui ZW, Ji HJ. Test cases generation for multiple paths based on PSO algorithm with metamorphic relations. IET Software. 2018 May 1;12(4):306-17.

[12] Arora PK, Bhatia R. Mobile agent-based regression test case generation using model and formal specifications. IET Software. 2017 Apr 5;12(1):30-40.

ISSN: 0971-1260
Vol-22-Issue-17-September-2019

[13] Nargis Akhter, Dr. Amar Singh, Mr. Guljar Singh. Automatic Test Case Generation by using Parallel 3 Parent Genetic Algorithm. International Journal for Research in Applied Science & Engineering Technology (IJRASET) Volume 6 Issue VII, July 2018

[14] Miranda MA, Ribeiro MG, Marques-Neto HT, Song MA. Domain-specific language for automatic generation of UML models. IET Software. 2017 Oct 6;12(2):129-35.

[15] Panichella A, Kifetew FM, Tonella P. Automated test case generation as a many-objective optimisation problem with dynamic selection of the targets. IEEE Transactions on Software Engineering. 2017 Feb 2;44(2):122-58.

[16] Bo L, Jiang S, Qian J, Wang R, Wang X. Efficient Test Case Generation for Thread-Safe Classes. IEEE Access. 2019 Feb 25;7:26984-95.

[17] Oliveira C, Aleti A, Grunske L, Smith-Miles K. Mapping the Effectiveness of Automated Test Suite Generation Techniques. IEEE Transactions on Reliability. 2018 Jun 8;67(3):771-85.

[18] Liu F, Huang H, Li X, Hao Z. Automated test data generation based on particle swarm optimisation with convergence speed controller. CAAI Transactions on Intelligence Technology. 2019 Mar 25;2(2):73-9.

[19] Gupta N, Sharma A, Pachariya MK. An insight into test case optimization: ideas and trends with future perspectives. IEEE Access. 2019 Feb 14;7:22310-27.

[20] Yousaf N, Azam F, Butt WH, Anwar MW, Rashid M. Automated Model-based Test Case Generation for Web User Interfaces (WUI) from Interaction Flow Modeling Language (IFML) Models. IEEE Access. 2019 May 20.

[21] Huang H, Liu F, Yang Z, Hao Z. Automated test case generation based on differential evolution with relationship matrix for IFOGSIM toolkit. IEEE Transactions on Industrial Informatics. 2018 Jul 18;14(11):5005-16.

[22] Liu P, Xu Z. MTTool: A Tool for Software Modeling and Test Generation. IEEE Access. 2018 Oct 1;6:56222-37.

[23] Feyzi F, Parsa S. Bayes-TDG: effective test data generation using Bayesian belief network: toward failure-detection effectiveness and maximum coverage. IET Software. 2018 Feb 13;12(3):225-35.

[24] Bian Y, Li Z, Zhao R, Gong D. Epistasis based aco for regression test case prioritization. IEEE Transactions on Emerging Topics in Computational Intelligence. 2017 May 29;1(3):213-23.

ISSN: 0971-1260
Vol-22-Issue-17-September-2019

[25] Mukherjee R, Patnaik KS. Prioritizing JUnit Test Cases Without Coverage Information: An Optimization Heuristics Based Approach. IEEE Access. 2019 Jun 12;7:78092-107.

[26] Bashir MB, Nadeem A. Improved genetic algorithm to reduce mutation testing cost. IEEE Access. 2017 Mar 3;5:3657-74.

[27] Jan S, Panichella A, Arcuri A, Briand L. Automatic generation of tests to exploit XML injection vulnerabilities in web applications. IEEE Transactions on Software Engineering. 2017 Nov 30;45(4):335-62.

[28] Li H, Lam CP. Software Test Data Generation using Ant Colony Optimization. International Journal of Computer and Information Engineering Vol : 1, No : 1, 2007.

[29] Hooda I, Chhillar R. A review: Study of test case generation techniques. International Journal of Computer Applications. 2014 Jan 1;107(16).

[30] Windisch A, Wappler S, Wegener J. Applying particle swarm optimization to software testing. InProceedings of the 9th annual conference on Genetic and evolutionary computation 2007 Jul 7 (pp. 1121-1128). ACM.

[31] Rodriguez J, Rodriguez GD. Automatic generation of GUI test cases using Ant Colony Optimization and Greedy algorithm. InCIbSE 2015 (p. 209).

[32] Nayak N, Mohapatra DP. Automatic test data generation for data flow testing using particle swarm optimization. InInternational Conference on Contemporary Computing 2010 Aug 9 (pp. 1-12). Springer, Berlin, Heidelberg.

[33] Arifiani S, Rochimah S. Generating test data using ant Colony Optimization (ACO) algorithm and UML state machine diagram in gray box testing approach. In2016 International Seminar on Application for Technology of Information and Communication (ISemantic) 2016 Aug 5 (pp. 217-222). IEEE.

[34] Wang Z, Liu Q. A software test case automatic generation technology based on the modified particle swarm optimization algorithm. In2018 International Conference on Virtual Reality and Intelligent Systems (ICVRIS) 2018 Aug 10 (pp. 156-159). IEEE.

[35] Khan R, Amjad M, Srivastava AK. Generation of automatic test cases with mutation analysis and hybrid genetic algorithm. In2017 3rd International Conference on Computational Intelligence & Communication Technology (CICT) 2017 Feb 9 (pp. 1-4). IEEE.

[36] Khan R, Amjad M. Automatic test case generation for unit software testing using genetic algorithm and mutation analysis. In2015 IEEE UP Section Conference on Electrical Computer and Electronics (UPCON) 2015 Dec 4 (pp. 1-5). IEEE.

ISSN: 0971-1260
Vol-22-Issue-17-September-2019

[37] Gongge G, Hao Z, Jing Y. Research on automatic generation of Test Cases. In2012 IEEE/ACIS 11th International Conference on Computer and Information Science 2012 May 30 (pp. 428-431). IEEE.

[38] Lin P, Bao X, Shu Z, Wang X, Liu J. Test case generation based on adaptive genetic algorithm. In2012 International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering 2012 Jun 15 (pp. 863-866). IEEE.

[39] Liu A, Mu Y, Liu X, Zhang Z. Analysis of Input Variable's Influence on Control Condition in Automatic Test. In2013 5th International Conference on Intelligent Human-Machine Systems and Cybernetics 2013 Aug 26 (Vol. 2, pp. 347-350). IEEE.

[40] Chen L, Li Q. Automated test case generation from use case: A model based approach. In2010 3rd International Conference on Computer Science and Information Technology 2010 Jul 9 (Vol. 1, pp. 372-377). IEEE.

[41] Mirzaaghaei M, Pastore F, Pezze M. Supporting test suite evolution through test case adaptation. In2012 IEEE Fifth International Conference on Software Testing, Verification and Validation 2012 Apr 17 (pp. 231-240). IEEE.

[42] Khan R, Amjad M, Srivastava AK. Optimization of automatic generated test cases for path testing using genetic algorithm. In2016 Second International Conference on Computational Intelligence & Communication Technology (CICT) 2016 Feb 12 (pp. 32-36). IEEE.

[43] Dave M, Agrawal R. Search based techniques and mutation analysis in automatic test case generation: A survey. In2015 IEEE International Advance Computing Conference (IACC) 2015 Jun 12 (pp. 795-799). IEEE.

[44] Li K, Zhang Z, Liu W. Automatic test data generation based on ant colony optimization. In2009 Fifth International Conference on Natural Computation 2009 Aug 14 (Vol. 6, pp. 216-220). IEEE.

[45] Sayyari F, Emadi S. Automated generation of software testing path based on ant colony. In2015 International Congress on Technology, Communication and Knowledge (ICTCK) 2015 Nov 11 (pp. 435-440). IEEE.

[46] Biswas S, Kaiser MS, Mamun SA. Applying ant colony optimization in software testing to generate prioritized optimal path and test data. In2015 International Conference on Electrical Engineering and Information Communication Technology (ICEEICT) 2015 May 21 (pp. 1-6). IEEE.

ISSN: 0971-1260
Vol-22-Issue-17-September-2019

[47] Arifiani S, Rochimah S. Generating test data using ant Colony Optimization (ACO) algorithm and UML state machine diagram in gray box testing approach. In2016 International Seminar on Application for Technology of Information and Communication (ISemantic) 2016 Aug 5 (pp. 217-222). IEEE.

[48] Prajapati N, Kumar N. Data flow based quality testing approach using ACO for component based software development. In2016 International Conference on Computing, Communication and Automation (ICCCA) 2016 Apr 29 (pp. 807-812). IEEE.