Test Case Optimization based on Specification Diagrams and Simulation Invocation Relationship

2021 Fourth International Conference on Electrical, Computer and Communication Technologies (ICECCT) | 978-1-6654-1480-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICECCT52121.2021.9616824

Mani Padmanabhan

Faculty of Computer Applications,SSL,
Vellore Institute of Technology (VIT),
Vellore, Tamil Nadu, India

**mani.p@vit.ac.in*

***Abstract*— The growing usage of software-based products is coupled with day-to-day human life. The software engineering technology can be more eye-catching among artificial intelligent-based software developers. The artificial intelligence systems such as the human machine interaction process are difficult to identify the pre-conditions during the development. Re-engineering is essential for artificial intelligent systems-based applications. Regression testing has assured the quality of products during the re-engineering process. The test cases are a core component in regression testing. Test case optimization and selection is a major activity to reduce the time and cost during regression testing. Many test case selection techniques have solved the problems in regression testing, however, the techniques seem to have much focus on reducing the number of test cases. This research proposes a test case optimization-based specification diagram. In the test, case selections are controlled by the simulation invocation relationship. The proposed simulation invocation methodology identified the simulations to be affected during the re-engineering process. The proposed optimization algorithm produced the test cases based on the fault coverage criteria. This approach had validated with three artificial intelligent-based systems during regression testing. The comparative analysis shows that the proposed approach is well suitable for re-engineering in terms of the average percentage of fault detected values.**

Keywords*— *Software Testing, Test case generation, Test Optimization, Software Validation, Simulation Invocation Relationship.

I. INTRODUCTION

Software test case optimization is a methodology that improves the effectiveness of artificial intelligence-based system software during regression testing. The major objective of the test case optimization methodology has increased the fault detection percentage [1]. The various test optimization techniques had been developed using evolutionary algorithms but the capability of solving problems is less [2]. The graph-based framework is the other most popular technique [3]. The traversal algorithm had reduced the test case generation time during the regression testing.

However, a graph-based approach is the high percentage of test cases running during the testing. The code coverage criteria have less in the existing optimization methodology. The main objective had been to minimize the test cases.

The proposed approach is a novel specification diagrams and simulation Invocation relationship-based framework that provides test case optimization techniques that attempt to achieve high fault coverage. The underlying hypothesis of this technique is that higher fault coverage tends to have a higher chance of finding faults.

Test case generation based on software design specification can be the best-optimized solution for reducing effort and cost. The efficient test cases will play a vital role in reducing the effort in the software testing life cycle. Specification diagram-based test case generation for AI systems can be used to improve system quality. In the specification-based test case generation for embedded systems, both automated and manual is limited in techniques as some situations inadvertently forget the simulation events.

The structural specification can be represented using a class diagram, component diagram, deployment diagram, object diagram, and profile diagram [4]. The UML presented only a set of notations and not a method. The notations in the UML have particular importance for modeling real–time systems. To gain this functionality, real-time systems are developed with inputs from various microprocessors and logical gates. The real-time system designs rely on hardware and software [5].

The proposed approach is a novel specification diagrams and simulation Invocation relationship-based approach that provides test case optimization techniques that attempt to achieve high fault overage. This research focuses on understanding effective solutions for modeling and analyzing the simulation Invocation relationship testing. Due to the closed relationship between simulation Invocation relationship devices and real-time software instructions, code-based testing in a simulation invocation relationship cannot cover the entire behavior interfaces in the system; this research mainly focuses on the number of test cases.

To support the investigation of research hypothesis, the methodology has presented A. Stack Simulation Allocation, Stack Simulation search algorithm, Pseudocode for simulation selection, Jaccard Similarity methodology, Minhash value Identification with validation technique will be applied and an approaches with tool support that generates test cases and saves relationships among models and abstract test cases at different levels of granularity during the re-engineering.

II. TEST CASE GENERATION AND OPTIMIZATION

Traditional approaches to software testing are code-based. Code-Based Testing (CBT) generates test cases to test functionalities implemented in the source code, so functionalities missing from the source code will not be tested [5]. Automatic test cases generation methods of single-path are more used- In the literature listed the most common automatic test cases generation methods of single-path, such as random, dynamic method, and symbolic execution method[6]. The full path is the program of all the

possible paths in the program, if all the possible paths have been tested is called full path testing- Based on the full path of testing's advantages are test without specifying the path, because the specified path itself is more trouble test cases of full path cover all the executable path of the program, testing the feasible path in the program more comprehensively, to compute all process paths for complex structure is not desirable, so how to design a search method based on the full path has great significance for practical application- This paper designs a system framework based on SA(Source Analysis) rule automatic test case generation methods of the full path. The framework including initialization components, program analysis component, optimizing component and result from analysis component, four parts- Paper made a more detailed discussion about program analysis component and optimizing component's function and algorithm- Program analysis component has two main tasks: First, make a static analysis for test objectives, the main focus on the source of the semantic analysis and syntax analysis; the second is using SA rule to analysis the source program line by line, use a keyword to analysis structure of the program and plug the appropriate node, the purpose of plug the appropriate node is to record statement execution, get a directed graph with nonnegative weights and the objective function which has been plugged[7]. SA rules have developed a solution for constructing solution about source code's directed graph with nonnegative weights, directed graph with nonnegative weights solution can exclude loops in the source program, and simplify the path of the source program optimization component has three tasks: First, get the non-repetition path, to search for all the non-repetition path in the directed graph with nonnegative weights, and store the path to the set of non-repetition; second, put the initialized test data into source program which has been plugged to execute, get the actual execution path of test data; third, calculate the coverage of the executed path, in this work, has developed a solution using particle swarm optimization algorithm to select the set of test case If it meets the pre-set threshold, then this group of test data is the more appropriate test case of this program, if not met, continue to be adjusted using the optimization algorithm[8].

III. SPECIFICATION DIAGRAMS AND SIMULATION INVOCATION RELATIONSHIP

Software testing is significant for all software development processes. Thus, there are clear benefits in reducing the development time and improving the effectiveness of software testing by automating the process[6]. The detailed system interaction among the object needs to generate for automated testing. The familiar system diagrams i.e. behavioral and structural are subject to validation against its implementation[9]. The effective implementation of test cases based on the requirement documents and followed by specification design models. Figure 1 describes an AI-based use case diagram.

The quality of the AI system software is measured based on the testing results. The sample interaction diagram shows the S= {m1, m2,…, m7} there I represents E in the sequence graph. The nodes are to be identified through the proposed graph conversion technique. In the given sample interaction I= {n1,n2….n5}, n3,n4 are the control node and n1,n2,n5 are the message node ie,V1= {n3, n5} and V2 ={ n1,n2,n5}.

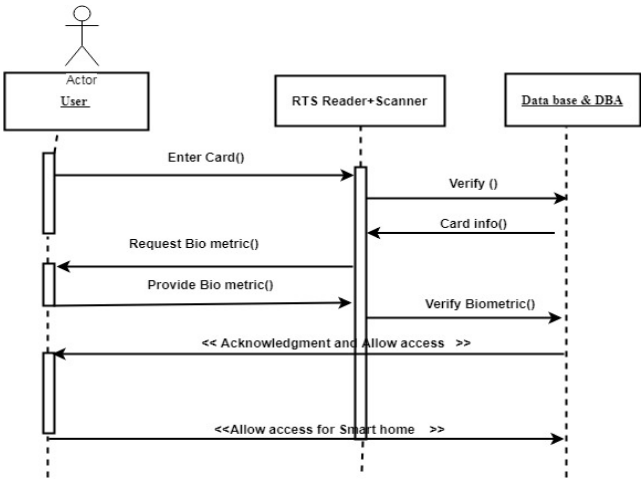

Fig .1. AI based Specification Diagram.

Testing is the process of identifying the bugs or errors as much as possible or verify the process based on the input and output[7]. The sensor-based AI system is necessary to undertake effective testing to produce reliable systems. Figure 2 describes the different types of specification diagrams.

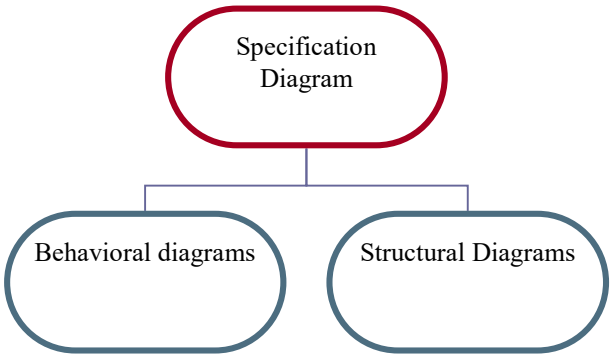

Fig. 2. Types of Specification Diagrams.

In many AI system software development projects spending 50% of the development time and effects in the error identification. Most often, a tester is given a set of tasks based on the code for verifying the event that can be performed in the IDE. Figure 3 shows the classification of software testing.

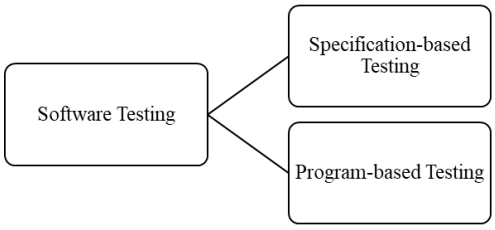

Fig. 3 Classification of Software Testing.

IV. PROPOSED METHODOLOGY

The goal of the specification diagram-based proposed methodology is to improve the effectiveness of test cases and reduce the testing cost. To solve this problem, propose a stack allocation-based framework that utilizes two sources of information (code coverage and test execution time). The approach maps the test case optimization problem to a stack allocation, in which stack data of the linked list are test cases and stack numbers between the stack data are relationships between the test cases. The following subsections describe in detail including stack allocation and search algorithm using a stack.

A. Stack Simulation Allocation

There are two elements of the stack are stack data and stack number. The code coverage becomes one of the two metrics, time is the other metric. Therefore, the stack data with the following properties the tester test ID to be the stack number, and stack simulation are generated based on the code coverage divided by test execution time.

The allocated stack simulation has to determine the relationship between the stacks. Therefore, to create links between stacks, the simulation invocation relationship during the regression teasing to define a function to measure similarity between test cases. The minhashing matrix representation of simulation invocation relationship diagram is a set represented by the column of the characteristic matrix, The minhashing value of allocated stack simulation (S1,S2…..Sn) is the number of rows to the measured similarity among test cases.

B. Stack Simulation search algorithm

The approach determines the goal of the search algorithm is to find more important test cases that have effective potential to find faults with less exaction time and more code coverage. To support this goal, our algorithm should find a stack simulation with a higher gain value, and visit the stack at least one time during the allocation.

The Jaccard similarity addresses well is that of finding textually similar documents. To find the duplicate values in the dialog flows, the minhashing value to set S and T is |S U T|/|S ∩T | that is the ratio of the size of the intersection of S and T to the size of their union. The Jaccard similarity of S and T to be represented by sim(S,T). The textual similarity has important during the testing. This proposed technique reduced the cost and time during the regression testing.

Algorithm 1 : Pseudocode for simulation selection

Pre: Specification Invocation

Post: Simulation selection

Start: Read specification stack value

For each stack (1<=link>=m) do

Simulation (stack) ← 1

End For.

For each stack (1<=link>=o) do

Sum ← 0

End For .

While connection [link] do

Link← Link+1;

if Connection Link =0 then

Sum ← sum + Link;

End if .

If Sum >0 then

Sum ← (T) // Number of Simulation

Else

Sum ← 0

End if.

End While .

End.

The linked list shows the test case prioritization was built using algorithm 1. The five stacks representing the five test cases that are labeled by the test cases that are located by their stack value.

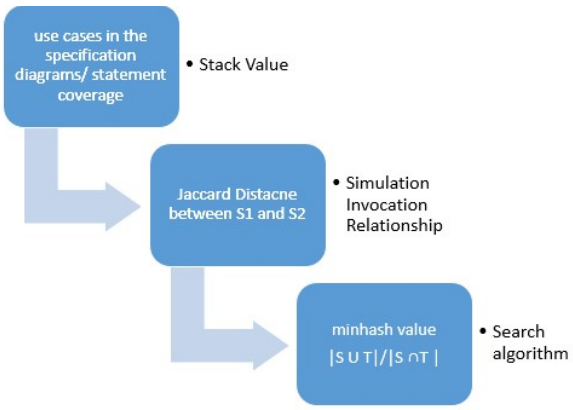

Fig. 4 Proposed Optimization Methodology.

Each stack value has been calculated from its number of specifications divided by its invocation relationship. For instance, S2 covers 5 use cases in the specification diagram and its statement coverage has 6 therefore, the value of this node is equal to 0.83 (5/6). The stack values are calculating the Jaccard Distance for each node. Finally, the search algorithm had used to produce the test case priority.

C. Jaccard Similarity

Similar items of flows are that there may be far too many pairs of items to test each pair for their degree of similarity.

To find the duplicate values in the dialog flows, the minhashing value to set S and T is |S U T|/|S ∩T | that is the ratio of the size of the intersection of S and T to the size of their union. The Jaccard similarity of S and T is to be represented by sim(S,T).

Table 1 Minhash Value from Simulation.

simulation ID	Minhash value
S_{T1}	M1→M2→M3→M4→M5
S_{T2}	M1→M2→M3→M4→M5→M6
S_{T3}	M1→M2→M3→M4→M5→M6→M7

The Jaccard similarity addresses well is that of finding image similar documents in a large human-machine interaction process. The data similarity has important during the testing. This proposed technique reduced the cost and time during the human-machine interaction process testing.

D. Minhash value Identification

The minhashing matrix representation of modeling diagram a set represented by a column of the characteristic matrix, pick a permutation of rows. The minhashing value of any sequence (S1,S2…..Sn) is the number of rows (indent or trigger), in the permuted order, in which the sequence has a 1. The 1 value has represented several trigger and 0 are the number of indent in the chatbot interaction. The matrix representation sets chosen from the dialog flows blocks figure x. The sequence diagram users sets U ={ a, b} , here S1 ={1,0}, S2={0,1}, S3 ={1,1},S4 ={0,0}. The permutation of U defines a minhashing function h that maps sets of trigger and indent in the chatbot interaction. The minhashing a set represented by a column of the characteristic matrix, pick a permutation of the rows. The minhashing value of any column is the number of the matching row, in the permuted order.

V. EXPERIMENTAL RESULTS

A. Experimental data 1

The automatic drip irrigator controller switches on the power through the remote controller. The software running on the Arduino microcontroller to start the dummy task. The dummy task check where the system battery voltage is calculated if the voltage is adequate for control the drip process then starts the system and checks the event was interrupted previously. If the voltage is inadequate, the system shuts down automatically.

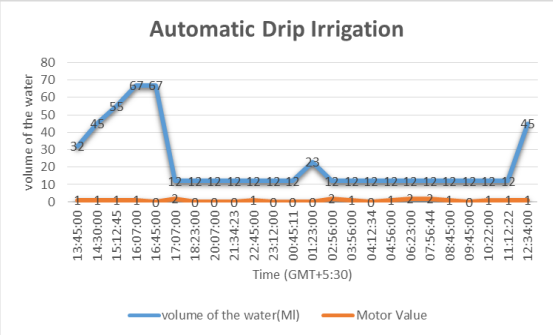

Fig. 5 Drip Value Running Process

B. Experimental data 2

A Database-Centric approach for medical guideline systems was already looked into many research communities. MySQL and PostgreSQL technologies in 2008 in the development of a European FP7 research project. In the year 2011 to 2020, most of the AI-based health sector researchers proposes a viable enabling technology for the harnessing of some of the complex issues in the incoming scenario of industrial automation where the smart features of Cyber-Physical System are going to play a leading role.

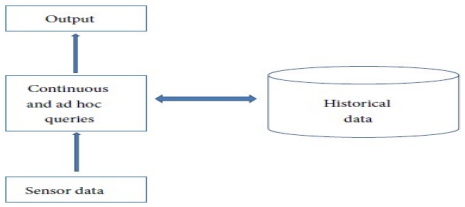

Fig. 6 Database-Centric Approach for Medical Guideline Systems.

C. Experimental data 3

In the smart home system having the identification water level system is the process of water level checking in a tank. The system is to get the signal water level low then send the signal to the house owner and switch on the pump to fill the water in the tank. The activity diagram describes the task for the switch-off process if the water level reached high, the detailed message has been share with the house owner in each process using an embedded system.

The human-machine interaction process of simulation invocation relationship is collected from the experimental data. In one of the experiment ID, there is five simulation invocation relationship in their intersection and total of 12 test cases that appear in the simulation regression testing. Thus, Sim (additional statement) = 5/12. The duplicate simulation for the human-machine interaction process has been counted to reduce the test cases during the Re-engineering software testing. The experimental results are to be compared in the following section.

This research paper investigates whether the use of stack simulation can improve the effectiveness of test case optimization techniques. Compare the four open-source programs with the associated data for identifying the effectiveness of test case optimization.

Table 1. Test Cases with LOC.

Source	#LOC	#Total Statement	#Additional Statement	#Test Cases (Code based)	#Test Cases (Proposed Approach)
Smart home security system	5656	4567	567	6789	3456
Automatic detection of field grown	1345	987	89	2345	1678
Robotic Harvesting	4567	3789	567	6789	5436
Water management system using IOT	2347	1879	345	3456	3456

The proposed approach provides the possible test cases with code coverage. The presented technique actives high statement coverage and functional coverage by implementing

the artificial intelligent systems during the re-engineering in terms of the average percentage of code coverage.

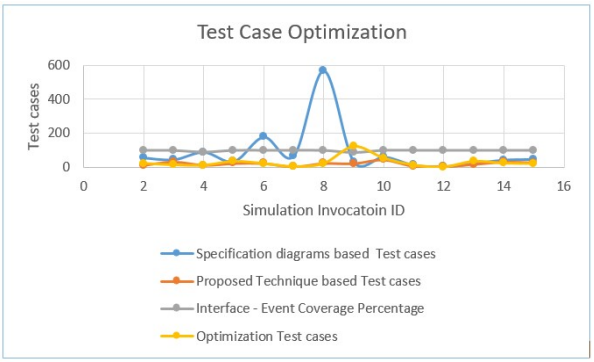

Fig. 7 Test Case Optimization and Interface Coverage.

The results obtained through experiments show that the proposed approach is feasible and capable of reducing the cost in SDLC.

$$\text{Fault coverage Percentage Percentage} = \frac{\textit{Node Covered in each generation}}{\textit{Total number of node in EFG}} * 100$$

To calculate the coverage of the entire event in the simulation invocation relationship, the paths left during the program run are identified. The event coverage percentage is compared. The experiment recorded the number of the event covered by each generation to check if the population has evolved to the desired level of coverage of all event flow. In path testing, the tester is expected to execute all paths of the test program.

Table 2. Fault Coverage Percentage.

Simulation Invocation ID	Experiment Data	Code based Test cases	Specification diagrams based Test cases	Proposed Technique based Test cases	Fault coverage Percentage
SI1	Drip irrigator controller	34	56	12	99.4
SI2	Drip irrigator controller	22	45	32	98.4
SI3	Smart Home Security System	49	89	12	89.4
SI4	Smart Home Security System	87	34	23	99.1
SI5	Medical guideline systems	134	178	23	99.3
SI6	Smart Home Security System	45	67	2	99.5
SI7	Smart Home Security System	234	568	23	99
SI9	Smart Home Security System	45	32	22	86.8
SI11	Medical guideline systems	67	61	45	99.5
SI12	Smart Home Security System	23	13	6	99.67
SI13	Medical guideline systems	8	3	4	99.45
SI14	Medical guideline systems	34	22	17	99.32
SI15	Medical guideline systems	56	43	32	99.45
SI16	Smart Home Security System	65	47	27	99.21

This testing style is very laborious and time-consuming. The Program with loops has paths and it becomes difficult to exercise all paths for testing purposes. To overcome this problem, a subset of events based on some criteria must be selected.

$$\text{Cost Reduction Percentage} = \frac{\textit{Triggered Paths Coverage Percentage}}{\textit{Total number of event percentage}} * 100$$

The ratio of the generated triggered path set to all paths in the simulation invocation relationship shows the reduction in the cost of exercising all paths of the program for path testing, on covering a subset of target unique paths, which saves on time and effort of the tester. To find the reduction percentage of the cost in path testing, the following formula has been used. The proposed block diagram approach provides all edge coverage criteria for biomedical system development.

VI. CONCLUSION

The proposed approach for the optimization algorithm of a test case selection from the specification interaction diagram by using simulation invocation relationship yields efficient fault coverage. It also shows how to create simulation invocation relationship from specification diagram. Some concluding observations from the research are given below. The presented algorithm is very effective for identify simulation selection from specification diagrams. The methods for test case optimization-based Jaccard Similarity method achieve high fault coverage. minhas matrix representation optimized the test cases during the regression testing. The number of test cases for testing is rescued but fault coverage is high during the proposed approach applied in the experiment. The outcome of our proposed approach indicates the possible code coverage with less number of test cases for artificial intelligence systems such as the human-machine interaction process. In the future, the research to focus and expand as more data becomes available across a wider range of applications and different metrics in different areas of regression testing.

VII. REFERENCES

[1] B. Jiang and W. K. Chan, “Input-based adaptive randomized test case prioritization: A local beam search approach,” Journal of Systems and Software, vol. 105, pp. 91–106, Jul. 2015, doi: 10.1016/j.jss.2015.03.066.

[2] M. Azizi and H. Do, “Graphite: A Greedy Graph-Based Technique for Regression Test Case Prioritization,” in 2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), Memphis, TN, Oct. 2018, pp. 245–251. doi: 10.1109/ISSREW.2018.00014.

[3] P. V. Padrao Lopes, L. Hsu, M. Vilzmann, and K. Kondak, “Model-based Sensor Fault Detection in an Autonomous Solar-powered Aircraft,” Oct. 2019, pp. 247–254. doi: 10.3384/ecp19162029.

[4] H. Gong and H. Yu, “Abstract Memory Model For Complex Data Structures In Unit Testing,” in 2019 3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE), Xiamen, China, Oct. 2019, pp. 601–605. doi: 10.1109/EITCE47263.2019.9095152.

[5] Mani P, Prasanna M, “Test Case Generation for Real-Time System Software using Specification Diagram” International Journal of Intelligent Networks And Systems Society, Vol.10, No.1, PP. 166 – 175, 2017.

[6] Mani P, Prasanna M “Test case generation for mbedded system software using UML interaction diagram” Journal of Engineering Science and Technology ,Vol. 12, No. 4, pp. 860 – 874, 2017.

[7] P. Mani and M. Prasanna, “Validation of automated test cases with specification path,” Journal of Statistics and Management Systems, vol. 20, no. 4, pp. 535–542, Jul. 2017, doi: 10.1080/09720510.2017.1395173.

[8] M. Padmanabhan, “Test Path Identification for Virtual Assistants Based on a Chatbot Flow Specifications,” in

Soft Computing for Problem Solving, vol. 1057, K. N. Das, J. C. Bansal, K. Deep, A. K. Nagar, P. Pathipooranam, and R. C. Naidu, Eds. Singapore: Springer Singapore, 2020, pp. 913–925. doi: 10.1007/978-981-15-0184-5_78.

[9] G. Hou, K. Zhou, T. Qiu, X. Cao, M. Li, and J. Wang, "A novel green software evaluation model for cloud robotics," Computers & Electrical Engineering, vol. 63, pp. 139–156, Oct. 2017, doi: 10.1016/j.compeleceng.2017.08.021.

[10] C. C. Venters et al., "Software sustainability: Research and practice from a software architecture viewpoint," Journal of Systems and Software, vol. 138, pp. 174–188, Apr. 2018, doi: 10.1016/j.jss.2017.12.026.

[11] B. Jiang, Z. Zhang, W. K. Chan, and T. H. Tse, "Adaptive Random Test Case Prioritization," in IEEE/ACM International Conference on Automated Software Engineering, Nov. 2009, pp. 233–244. doi: 10.1109/ASE.2009.77.

[12] R. Mall, D. Kundu, and D. Samanta, "Automatic code generation from unified modelling language sequence diagrams," IET Software, vol. 7, no. 1, pp. 12–28, Feb. 2013, doi: 10.1049/iet-sen.2011.0080.

[13] A. Nayak and D. Samanta, "Automatic Test Data Synthesis using UML Sequence Diagrams.," The Journal of Object Technology, vol. 9, no. 2, p. 115, 2010, doi: 10.5381/jot.2010.9.2.a2.

[14] R. M. Hierons, "Testing from Partial Finite State Machines without Harmonised Traces," IEEE Transactions on Software Engineering, vol. 43, no. 11, pp. 1033–1043, Nov. 2017, doi: 10.1109/TSE.2017.2652457.

[15] M. Chen, P. Mishra, and D. Kalita, "Efficient test case generation for validation of UML activity diagrams," Des Autom Embed Syst, vol. 14, no. 2, pp. 105–130, Jun. 2010, doi: 10.1007/s10617-010-9052-4.

[16] H. Wang, J. Xing, Q. Yang, P. Wang, X. Zhang, and D. Han, "Optimal control based regression test selection for service-oriented workflow applications," Journal of Systems and Software, vol. 124, pp. 274–288, Feb. 2017, doi: 10.1016/j.jss.2016.06.065.

[17] P. M. Jacob and Prasanna M, "Software architecture pattern selection model for Internet of Things based systems," IET softw., vol. 12, no. 5, pp. 390–396, Oct. 2018, doi: 10.1049/iet-sen.2017.0206.

[18] A. P. Agrawal and A. Kaur, "A Comprehensive Comparison of Ant Colony and Hybrid Particle Swarm Optimization Algorithms Through Test Case Selection," in Data Engineering and Intelligent Computing, vol. 542, S. C. Satapathy, V. Bhateja, K. S. Raju, and B. Janakiramaiah, Eds. Singapore: Springer Singapore, 2018, pp. 397–405. doi: 10.1007/978-981-10-3223-3_38.

[19] S. Ji, B. Li, and P. Zhang, "Test Case Selection for All-Uses Criterion-Based Regression Testing of Composite Service," IEEE Access, vol. 7, pp. 174438–174464, 2019, doi: 10.1109/ACCESS.2019.2957220.

[20] Mani P, "A study on transaction Specification based Software Testing for Internet of Things", in IEEE International conference on Current Trends towards Converging Technologies (ICCTCT 2018), 1-3 March 2018, Coimbatore, India

[21] Mani P, Prasanna M, "A study on functional specification based test case generation for real-time system" International Journal of Engineering and Technology, Vol.8 No.3, PP. 1801-1806, 2016.

[22] Mani P, "Sustainable Test Path Generation for Chatbots using Customized Response", International Journal of Engineering and Advanced Technology, Volume-8 Issue-6, August 2019.

[23] P. T. Devanbu and S. G. Stubblebine, "Stack and queue integrity on hostile platforms," IEEE Transactions on Software Engineering, vol. 28, no. 1, pp. 100–108, Jan. 2002, doi: 10.1109/32.979991.

[24] B. Cottenceau, L. Hardouin, and J. Trunk, "Weight-Balanced Timed Event Graphs to Model Periodic Phenomena in Manufacturing Systems," IEEE Transactions on Automation Science and Engineering, vol. 14, no. 4, pp. 1731–1742, Oct. 2017, doi: 10.1109/TASE.2017.2729894.

[25] C. Hettiarachchi, H. Do, and B. Choi, "Risk-based test case prioritization using a fuzzy expert system," Information and Software Technology, vol. 69, pp. 1–15, Jan. 2016, doi: 10.1016/j.infsof.2015.08.008.