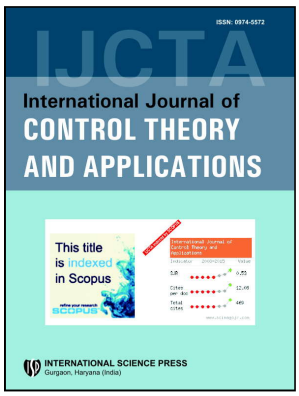

International Journal of Control Theory and Applications

ISSN : 0974-5572

Volume 10 • Number 18 • 2017

Automatic Test Suite Generation for Object Oriented programs using Metaheuristic Cuckoo Search Algorithm

Madhumita Panda[1] and Sujata dash[2]

[1] *Research Scholar, North Odisha University, India, Email: madhumita.panda3@gmail.com*
[2] *Reader Department of Computer Applications & IT North Odisha University, India,*
Email: sujata238dash@gmail.com

Abstract: Nowadays model based testing is the most widely accepted approach of object oriented testing. UML models are widely used and accepted for designing the different aspects of software systems. It is known that designing test cases for object oriented testing is a very difficult task due to the typical features of object oriented language. At the same time Search based testing is a recently evolved domain of software testing and it is gradually placing itself as a promising research area. This area mainly includes the application of nature inspired metaheuristic algorithms in software testing. Here in this paper we have proposed a model based approach to test object oriented programs using Cuckoo Search algorithm. Our approach reveals promising results in less computational time.

Keywords: Search Based Software Testing (SBST) Cuckoo Search algorithm (CO) Bio-inspired Algorithms

Object oriented testing (OOT) Model based Testing

1. INTRODUCTION

Software testing is performed by providing test inputs to a program code and then cautiously observing how that input exercises the particular program features. Manual design of the test inputs is a very costly and labor-intensive task, it needs a lot of experience and intelligence of the tester to successfully test the software. Automated software testing is the crucial requirement of current scenario to speed up the testing processes along with saving time and ensuring quality of the software under test[16-17].

Search based software testing (SBST) is a sub domain of search based software engineering(SBSE), it reformulates the testing process as a search problem and tries to find out solutions(test cases) from a large solution space using search algorithms including heuristic and meta-heuristic algorithms. It is a promising approach for the automation of software testing, where test data are carefully selected satisfying specific test adequacy criteria.The term SBSE(Search based software engineering) and SBST(Search based software testing)were coined by Harman and Jones(2001)[17], though researchers have started working in this area since 1976 i.e the application of search techniques in solving engineering problems, [8]. Many researchers have

worked in this direction using a wide range of different heuristic, metaheuristic techniques inspired from the intelligentsia applied by nature and natural organisms.

In object oriented testing area many researchers have worked in the field of model based testing and test data generation using UML models. Model based testing came to existence almost three decades before but the area started maturing only in last decade,though the area is very promising but still the research work is continuing and new models are proposed.

Model based testing involves deriving test suites from models of software systems using different UML diagrams, models of Graphical User Interfaces. These model based approach have shown some amount of success in the industries[8].

Recent studies have proved that testing methods using search based approaches are showing more promising results than traditional techniques.Most of the researchers in the area of object oriented testing have used genetic algorithms for test data generation [11]. In theobject oriented testing field very few works have been done on generation of optimized and prioritized test data using swarm based bio inspired techniques such as Cuckoo Search(CO), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC) etc.

In the previous works on structural testing it was found that Cuckoo Search algorithm gives promising results in comparison to other metaheuristic algorithms[13], thus in the proposed approach Cuckoo search algorithm is applied to generate optimized test suites from UML state activity diagrams to test object oriented programs.

The rest of the paper is organized as follows. Basic concepts for understanding the proposed approach and methodologies are elaborately described in section II.The related work in the area of search based testing as well as model based object oriented testing are described in section III. Cuckoo Search algorithm is explained in section IV. Proposed methodology and experimental results are presented in section V. Concluding remarks and future perspectives are given in section VI.

2. BASIC CONCEPTS

In this section some basic concepts and notations related to object oriented paradigm are provided which will be helpful for better understanding of the approaches and methodologies used in this paper.

2.1. State machine diagram

State machine diagram is an UML 2.0 behavioral diagram used for representing the state based behavior of entities in response to the receipt of event instances.It is generally used for describing the behavior of an object, showing how an object will change in response to an event. Each reaction may be in the form of a sequence of actions, accompanied by transitions from one state to another. An event is either the receipt of a signal or the outcome of a method call. An action includes the sending of a signal or an operation call.

In the state machine diagram, a state is represented by a rectangle with rounded corners. A state is either simple or composite, A composite state is decomposed into two or more concurrent or mutually exclusive disjoint sub states or regions. A state may be subdivided into multiple compartments i.e named compartment holding the name of the state, internal activities compartment containing a list of state activities or internal actions and internal transition compartment containing a list of internal actions as well as transitions [15. A final state is a special kind of state signifying the end of action and is shown using a bull's eye [15].

An event can be categorized to any one of the following three types, signal event, change event, call event or time event. A change event occurs when the value of Boolean expression changes from true to false or false to true. A signal event is a type of send request instances communicated between objects. A signal event may cause

a response to trigger a transition. A call event represents the reception of a request to invoke a specific operation on an object. A time event specifies an instance in time using an expression.

A transition indicates a possible change from one state to another, if the change of state takes place then the transition is said to have fired.Each transition is represented by the name of an event followed by an action expression

An action represents the sending of a signal or an operation call. An optional guard, a Boolean-valued expression denotes whether a particular occurrence of the event should trigger the specified actions, if it is false, then the fact that the event has occurred is simply ignored.

Figure1, shows the state machine diagram of the ATM card verification operation. It is showing the state transition of the ATM during card verification process. It starts with the user inserting the ATM card to ATM machine, first it enters to the Card entry state, after collecting the card information, it transits to the PIN entry state and accepts the PIN. Then it invokes the verify PIN()method of bank and stays in the same verification state, then depending on the verification status it either asks for re-entering the PIN, transits to the next state or aborts the entire transaction.

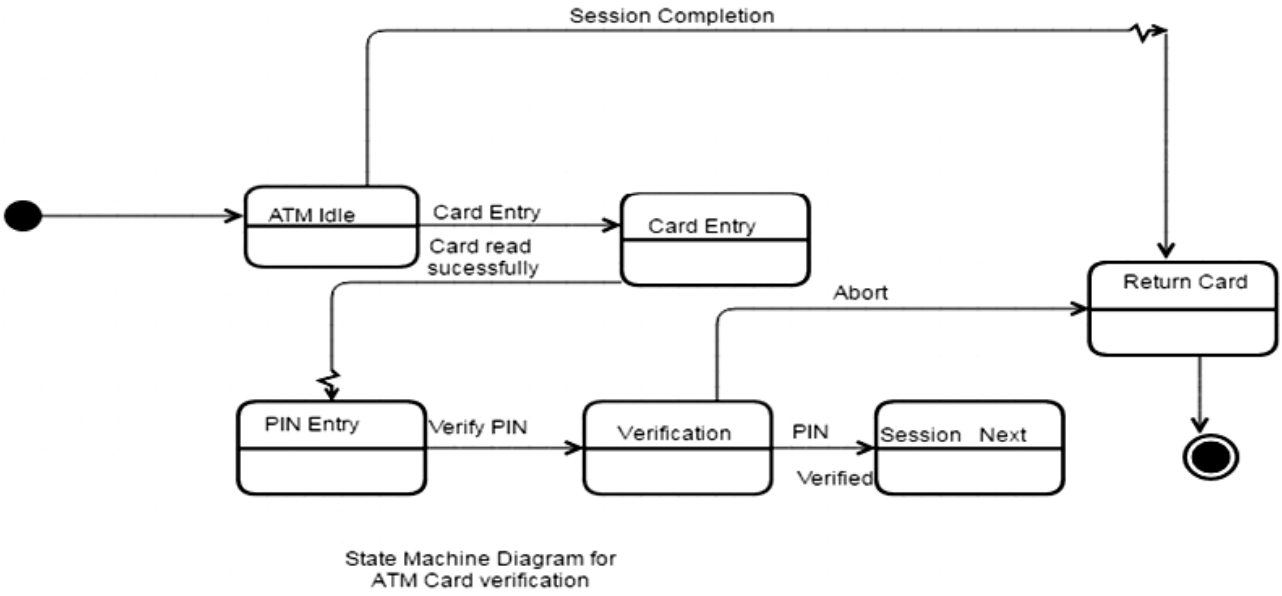

Figure 1: State Machine diagram for ATM Card verification

2.2. Activity diagram

Activity diagrams are employed in business process modeling This diagram is drawn during initial stages of requirements analysis and specification.These diagrams help help the developers to understand complex parallel processing activities.These diagrams are very similar to procedural flow charts.

The activity diagram and state machine diagram use similar notations for showing states and transitions. Rounded corner rectangles are used for showing states, arrows are used for showing transitions, branches are shown using diamond symbols.Here in case of activity diagrams a special pair of parallel lines known as forks and joins are used to show multiple transactions giving a single out put.These diagrams represent parallel activities using a special feature known as swim lanes. Swim lanes are helpful in grouping parallel activities. Using these swimlanes the activities are arranged in number of vertical segments and each segment represents the activities related to a particular class/object. The end transition shown using a bulls eye, here unlike state machine diagram more than one end transition symbols can be used [12][15].

2.3. Different types of testing

In object oriented systems in general, testing is done at three levels of abstraction, at class level similar to conventional unit testing, at cluster level similar to integration testing, and system level testing.The class level testing tests the code for each operation in a class as well as method interaction within the class.The cluster level testing which is very similar to integration testing the interactions among cooperating classes are tested.System level testing is performed on the complete system combing all clusters.

2.4. Path testing

A path of a software is a sequence of instructions, statements or a high level design that starts at an entry,junction or decision and ends at another, or possibly the same, junction, decision or exit.A path may go through several junctions,processes or decisions one or more times.Designing test cases to execute all paths is called path testing.

2.5. Basis Path Testing

A basis test path is defined as an execution path from the start state to a final state which executes any loop at most once.A basis path can be differentiated from all other basis path by at least one edge or one state activity node.Testing all basis path is called basis path testing.It fulfills the requirements of branch testing and also tests all of the independent paths that could be used to construct any arbitrary path.

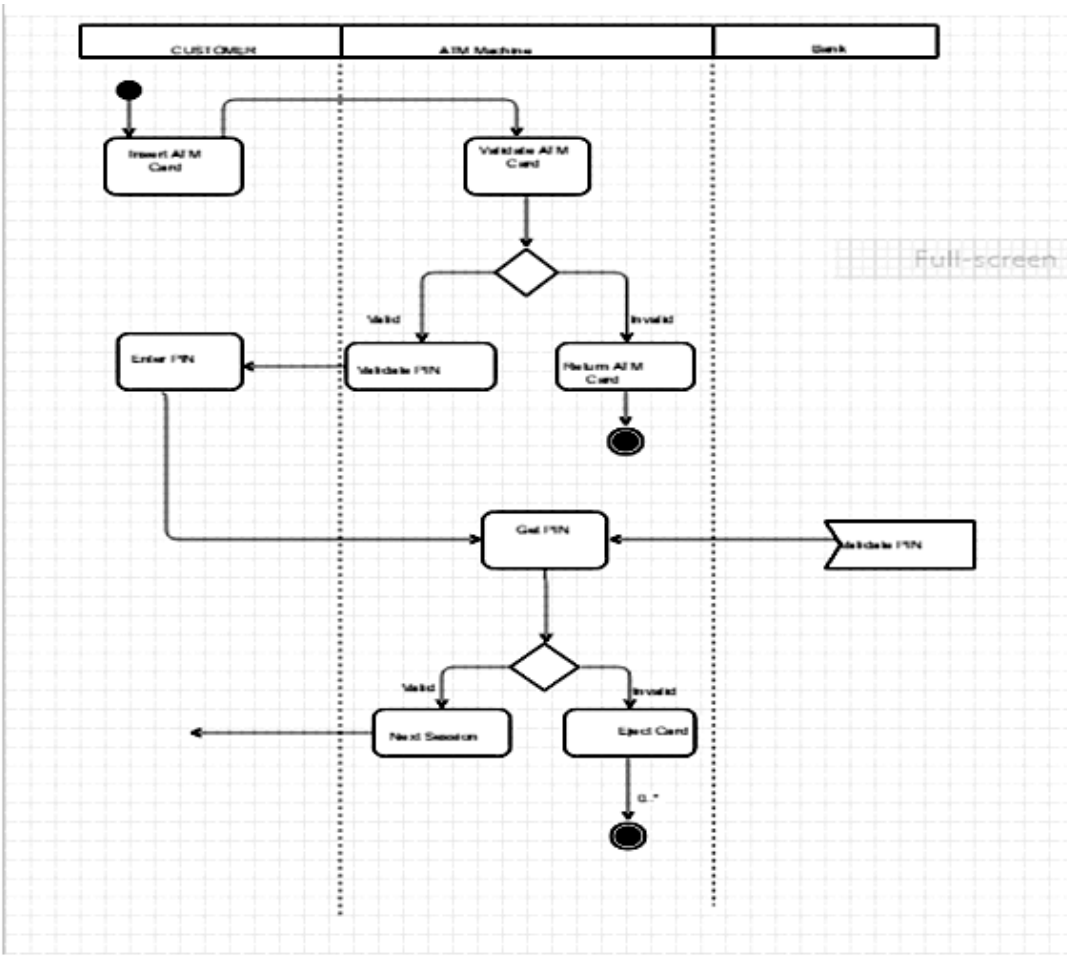

Figure 2: Activity diagram for ATM Card verification process

A basis test path is defined as an execution path from the start state to a final state which executes any loop at most once.

2.6. State activity diagram

The state activity diagram proposed by [12] has seven tuples SAD = (S; E; N; G; δ; S0; F)

where S is a finite set of state-activity nodes, N is the set of Decision nodes, E is the set of edges, G is the set of Guard conditions , ä is the transition function, S0 is the start state, F is a set of final states.

State activity diagram is an intermediate diagram which captures the information present in state machine diagram and activity diagram. It provides a detailed view of the entire control flow information handled by an use case.

The state activity diagram has the following components,

2.6.1. State activity node

The state activity node represents the state of an object during execution. Each node contains an atomic activity. The activity is shown by the initiating a node during execution time and when the activity gets terminated, an event called termination event is generated and then the outgoing edge from the node is activated. A state having no activity is assumed to be in wait state, and it waits for an external event to trigger a state transition.

2.6.2. AND-OR node

The AND-OR nodes are conditional nodes, an AND node is activated when all the outgoing edges are activated simultaneously on activation of all the incoming edges where as an OR node is activated when at least one incident edge is activated depending upon guard condition and only one of the out going edge is activated.

2.6.3. OR-Join node

When there is only one incoming edge and number of out going edges to an OR node then it is called OR-join node.

2.6.4. Decision-node

IA decision node has only one incoming edge and depending on the Boolean decision one of the outgoing edge is activated.

2.6.5. Merge node

In a merge node there are number of incoming edges and only one outgoing edge.

2.7. Test coverage

Test coverage indicates the extend to which a test adequacy criterion is able to test a given software. It is used to ensure whether the set of test cases are sufficient or adequate for testing the software for which they are designed or selected.Some of the popular and widely used coverage criteria include, basic path coverage, transition coverage, full predicate coverage etc.In this paper the sate activity diagram is used as the intermediate graphical representation and the basis path coverage criterion is the test coverage criteria.

3. RELATED WORK

Software testing is the most widely accepted and one of the best approaches to ensure safety, security and quality of software and is a very briefly researched software engineering topic . Meanwhile from 2000 onward about fifty percent of the research publications focus on automated test input generation . Test input ge neration,

B. Korel.(1990), is not a new research direction and a considerable amount of work has been done since 1975, such as symbolic execution, Ramamoorthy, (1976), Path based testing, Howden.(1976) evolutionary testing, Michael(2001), random,B.Korel(1996)and combinatorial testing, etc.[9] keeping in view the changing hardware platforms as well as the program complexity of software.

Harman and Jones(2001)[17],coined the word search based testing for the area of software testing using metaheuristic algorithms.He presented a brief survey of the research papers and the algorithms and methodologies adopted for generating test cases for software testing.

A. Bertolino[16], A.Orso[9]in their survey very briefly analyzed all best papers published in the domain o f search based testing in last two decades and presented a detailed review of the pros and cons of each technique and methodologies.

A. Orso[9], So far many methodologies, techniques and tools have been proposed in the literature, however a few of those methods been used in real world commercial applications, [4]. Hence the reported results need further improvements. Especially the models can be combined with other approaches to get better results . Analyzing the above described features from the existing literature we are motivated to propose some new methodologies combing the best features of model based and search based testing approaches.

Model based testing came to existence almost three decades before but the area started maturing only in last decade,though the area is very promising but still the research work is continuing and new models are proposed.

Model based testing involves deriving test suites from models of software systems using different UML diagrams, models of Graphical User Interfaces. These model based approach have shown some amount of success in the industries, [8].

P. Samuel et al.[15] presented a novel method to automate the process of test data generation using UML state machine diagrams.Here they have generated test data directly from UML diagrams without using any intermediate formalism.The limitation of their approach is the generated test data is not optimal and they have suggested to use evolutionary algorithms for getting globally optimal solutions.

C. Michel(2001) used genetic algorithms for the first time for automatic test data generation.They suggested that when the program size and complexity increases it becomes difficult for randomized algorithms to give better performance than existing non random techniques.

B, Korel(1990)proposed the test data generation technique for the first time from actual program execution using function minimization and dynamic data flow analysis technique.In this approach the most difficult task was to handle pointers and arrays therefore it needs further improvement.

Mark et al. gave a detailed taxonomy of model based testing, and they provided seven dimensions to define a taxonomy where the different approaches of model based testing can be characterized. They also provided a systematic generic process of model based testing.

M. Shirole, R. Kumar et al[11] presented a survey work mainly based on test data generation and test coverage achieved using UML state chart, activity and sequence diagrams.

S. Kansomkeat et al. proposed a method for generating test sequences using UML state chart diagrams. They transform the state chart diagram to testing flow graph, then traversed these graphs from the root node to the leaf nodes to generate test cases.

4. CUCKOO SEARCH ALGORITHM

Cuckoo search (CS) is a metaheuristic search algorithm inspired from the parasitic reproductive strategy adopted by Cuckoo birds, proposed by[13], [18-19]. Cuckoo is a very fascinating bird with its attractive sweet voice and parasitic breeding pattern.

The cuckoo lays its egg in the nest of other host birds of different species. The newly hatched cuckoo chick try to mimic the pattern and behavior of host chicks, so that they could not be identified by the host birds.

The characteristic features of the Cuckoo Search algorithm is based on three basic principles adapted from the typical characteristic of Cuckoo birds, first of all one Cuckoo bird lays one egg and dumps it on a randomly selected host nest. The second principle is, the number of nests is fixed and only a few number of nests are abandoned if the host bird identifies the Cuckoo egg with a probability Pa[0,1]. The third principle indicates only a fraction of nests containing the best egg or solutions will carry over to the next generation.

4.1. Cuckoo search Algorithm

Initialize a population of n hostNests

h_i, i=1, 2,……., n

For all h_i do

Calculate fitness $F_i = f(h_i)$

End for

While number of objective Evaluation <MaxNumber Evaluations do

Generate a cuckoo egg(h_j) by taking

A levy flight from random nest

$F_i = f(h_i)$

Choose a random nest i

If ($f_j < F_i$) then

$h_i \leftarrow h_j$

$F_i \leftarrow F_j$

End if

Abandon a fraction ω_a of the abandoned nests

Build new nests at a new location via

Levy flights to replace nests abandoned

Evaluate fitness of new nests and

Rank all solutions

End While

5. PROPOSED METHODOLOGY

In this paper a novel approach has been proposed to generate test suite for object oriented testing using state activity diagram and metaheuristic Cuckoo search algorithm.

The proposed methodology includes following steps, First of all the UML state machine diagram and activity diagrams are constructed using UML 2.0.In the next step the intermediate state activity diagram is constructed using the state machine and state activity diagram.Then the state activity diagram is traversed using depth path search to trace the basic feasible paths. The system state, transition edges and guard conditions are stored using a stack.

After tracing the test sequences ,the guard conditions and transition edges of transition edges are traced to find out the test sequences and trace the basic paths to be covered.

Finally the Cuckoo Search algorithm with levy flight steps is applied considering each set of guard conditions for a specific transition path coverage as the solutions(eggs) for generating test suits for the coverage of all feasible basic paths present in the state activity diagram. A detailed layout of our proposed methodology is shown in figure 3.

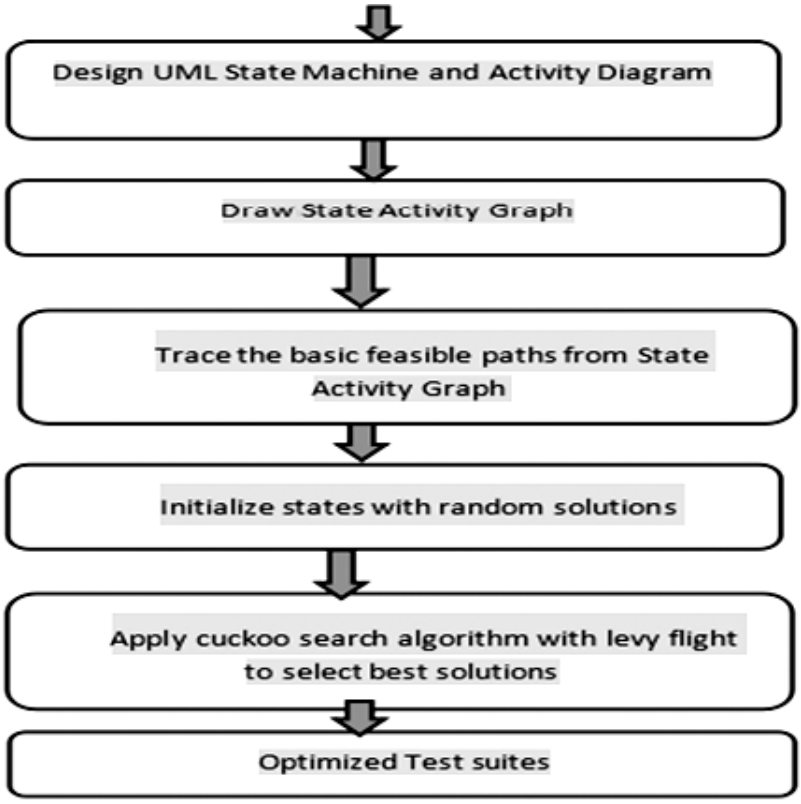

Figure 3: overall view of our proposed methodology

6. CONCLUSION

This paper presented an approach for model based testing of object oriented software using cuckoo search algorithm. Test data generation approach using State activity diagram and metaheuristic Cuckoo Search(CS) optimization algorithm has been proposed.The best part of cuckoo search algorithm is its exploration and exploitation capability to get optimal result in the search space, which has been exploited in the proposed algorithm to generate appropriate test casessatisfying path based coverage criteria. We have used the bench mark ATM transaction problem and State based Activity diagram to generate test cases satisfying path coverage criteria.we evaluated the performance of Cuckoo Search algorithm in comparison with GA. The simulation results illustrated that the Cuckoo Search(CS) algorithmis giving better coverage for all the feasible basic paths in few number of generations. The overall performance reveals that CS outperforms GA.This work can be further enhanced using other UML diagrams and nature inspired algorithms.

REFERENCES

[1] M. Utting, B. Legeard, , F.Bouquet, " Chapter Two-Recent Advances in Model-Based Testing", in Advances in Computers, 101, 53-120,2016.

[2] Yenigün, Hüsnü, Cemal Yilmaz, and Andreas Ulrich,"Advances in test generation for testing software and systems", in International Journal on Software Tools for Technology Transfer 18.3 245-249, 2016.

[3] P.McMinn, , M.Harman, , G.Fraser, & G. M. Kapfhammer, "Automated search for good coverage criteria: moving from code coverage to fault coverage through search-based software engineering",In Proceedings of the 9th International Workshop on Search-Based Software Testing , (pp. 43-44), ACM, 2016.

[4] R.Elghondakly, S.Moussa, , & N. Badr, "A Comprehensive Study for Software Testing and Test Cases Generation Paradigms", in Proceedings of the International Conference on Internet of things and Cloud Computing . (p. 50), ACM, 2016.

[5] L.J. Nimpa, , & H.Lichter, "An Overview on Automated Test Data Generation" in Full-scale Software Engineering/Current Trends in Release Engineering. 31, 2016.

[6] V.Panthi, , & D. P.Mohapatra, "Generating and evaluating effectiveness of test sequences using state machine", in International Journal of System Assurance Engineering and Management, 1-11, 2016.

[7] N.Khurana, , R. S.Chhillar& U.Chhillar A Novel Technique for Generation and Optimization of Test Cases Using Use Case, Sequence, Activity Diagram and Genetic Algorithm, (2016).

[8] M.Harman, Y.Jia, & Y.Zhang, "Achievements, open problems and challenges for search based software testing". In Software Testing, Verification and Validation (ICST), 2015 IEEE 8th International Conference on (pp. 1-12). IEEE2015.

[9] A.Orso, & G.Rothermel, "Software testing: a research travelogue (2000–2014)",in Proceedings of the on Future of Software Engineering (pp. 117-132). ACM, 2014.

[10] S.Anand, E. K.Burke, T. Y.Chen, J.Clark, "An orchestrated survey of methodologies for automated software test case generation", in Journal of Systems and Software,86(8), 1978-2001, 2013.

[11] M.Shirole, &R. Kumar, "UML behavioral model based test case generation: a survey", ACM SIGSOFT Software Engineering Notes, 38(4), 1-13,2013.

[12] S.Swain, D. P. Mohapatra, and R. Mall. "Test Case Generation Based on State and Activity Models", Journal of Object Technology 9.5 : 1-27,2010.

[13] P. R.Srivastava, A. K. Singh, H.Kumhar, & M.Jain, "Optimal test sequence generation in state based testing using cuckoo search", in International Journal of Applied Evolutionary Computation (IJAEC), 3(3), 17-32, 2012.

[14] M.Harman, S. A. Mansouri,& Y.Zhang, Search-based software engineering: Trends, techniques and applications. *ACM Computing Surveys (CSUR)*, *45*(1), 11, 2012.

[15] P. Samuel, R. Mall, A.K. Bothra. Automatic test case generation using unified modeling language (UML) state diagrams, IET Software, Vol. 2, No. 2, pp. 79–93, 2008.

[16] A. Bertolino, " Software Testing Research: Achievements, Challenges, Dreams", in Future of Software Engineering, FOSE '07, IEEE, 2007.

[17] M. Harman and B. F. Jones, Search based software engineering, Information and Software Technology, 43(14),833–839, 2001.

[18] S.Walton, O.Hassan, , K.Morgan, , & M. R.Brown, "Modified cuckoo search: a new gradient free optimisation algorithm", Chaos, Solitons& Fractals, 44(9), 710-718, 2011.

[19] X. S.Yang, &S. Deb, "Engineering optimisation by cuckoo search", in International Journal of Mathematical Modelling and Numerical Optimisation,1(4), 330-343. 2010.