A Systematic Analysis of Test Case Generation using UML Sequence and State Chart Diagrams

Iqbaldeep Kaur
Associate Professor,
Computer Science& Engineering,
Chandigarh Engineering College,
Landran, Punjab, India
iqbaldeepkaur.cu@gamil.com

Navneet Kaur
M. Tech. Research Scholar,
Computer Science & Engineering,
Chandigarh Engineering College,
Landran, Punjab, India
kaur.navneet1109@gmail.com

Dr. Amit Verma
Head of Department
Computer Science& Engineering,
Chandigarh Engineering College,
Landran, Punjab, India
Dramitverma.cu@gmail.com

Abstract— With the increasing industrialization, demand for the quality software's also increased. For quality software, there is the need of advanced technology and proper testing of software. Software testing is an important module of software development that consumes more time & cost as compare to development stage. Source code based traditional methods are mostly adapted aspects in software testing. But those methods are complex and not easy to understand for developers. So, here model based UML approaches are used for test code generation. In this paper, UML sequence & state chart diagram are considered for the systematic analysis of test case generation. Considered work described a systematic analysis on the test case generation using sequence, state chart and integrated diagrams. Based on this integration, test case generation method is also presented using genetic algorithm. Also a comparative analysis of existing concept based on case study, used tools and other key features is presented.

Keywords— UML Diagram, Test Case Generation, Sequence Diagram, State Chart Diagram, Software Development

I. INTRODUCTION

Software testing is an important aspect in software development. There is the consumption of more than 50% of the software development resources for testing [1]. Software testing involves three states of test case creation, test case execution and text case evaluation [2]. Test case execution and evaluation are easy steps but creation of test cases is the work of deep knowledge and expert skills. Software testing is important because it not involves only debugging but also maintains the software quality with validation & verification of test cases. So, for the success of software, there should be proper creation of test cases. Test cases should be minimum with maximum coverage of different software development aspects. Test case is a function of three parts input, state and output. Input is given as initial resource, state for which we need to evaluate software results and output is the result value for the desired software development state. One of the common methods to generate test cases is to make use of source code but this is traditional and complex approach for test case creation.

So, here unified modelling language is considered for the creation of test cases as UML model that reduces problem complexity with increasing software complexities [3, 4]. Unified modelling language is a widely accepted software modelling language in the field of software development. UML presents a manner to envisage the architecture of system in a diagram form. It consists of some elements as mentioned in Table 1.

Table 1: Elements of UML Architecture

• Kind of Activities/Jobs
• System's Individual Components
• Interactaction of individual components with other software components
• System working behaviour
• Interaction of different entities (components and interfaces)
• End User interface

UML consists of mainly structural and behavioural models [5] but as per viewing model, it can be categorized in user's view, structural view, behavioural view, implementation view and environmental view [6, 7]. A well elaborated UML diagram is shown in figure 1 with their respective feature diagrams. These UML views are discussed here:

A. User's View

User's view is the kind of black box view that is available to end user. It presents the information only about the system requirements but hides the other internal structure and system performance and work implementation.

B. Structural View

It defines the problem structure in terms of objects/classes and shows their relationship to implement the concept and resolve the problem.

C. Behavioral View

It defines the interaction behaviour in the system by highlighting the data flow and control among the modelled system. Behaviour view mainly represents the dynamic model of system.

D. Implementation View

It defines the internal work implementation of the system in terms of GUI presentation and database connectivity.

E. Environment View

It defines the implementation of different components on available hardware elements.

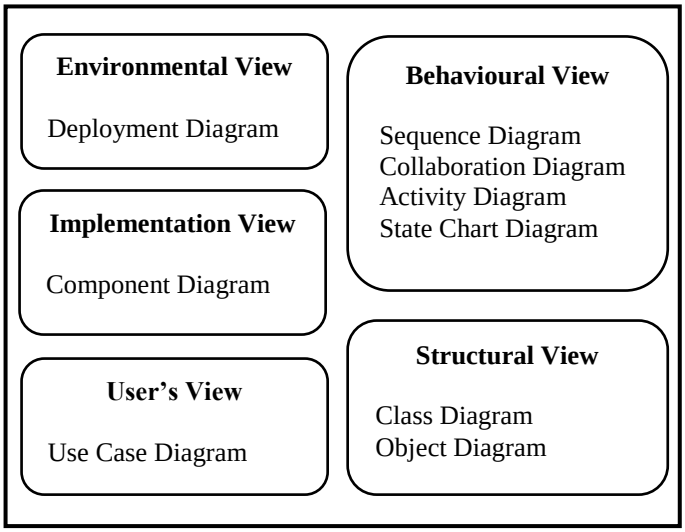

Figure 1: Views Supported in UML Diagram

In this paper, behavioural view based sequence diagram and state chart diagram are considered for test case generation.

Behavioural diagrams describe the interaction in the system by highlighting the data flow and control among the modelled system. Behavioural diagram shows the dynamic nature of the system. Some of behavioural diagrams are State Chart Diagram, Collaboration Diagram, Activity Diagram, Sequence Diagram etc. The reason for the selection of state chart and sequence diagram is the presence of unit level fault tolerance behaviour and integration level fault tolerance behaviour respectively for the UML diagrams.

- Sequence Diagram: Sequence diagram describes the process of object communication based on message sequence with an indication of object life span [8, 9]. Sequence diagrams have the ability of integration level fault tolerance. For example: Sequence diagram indicates the sequence of message flow as shown in figure 2.

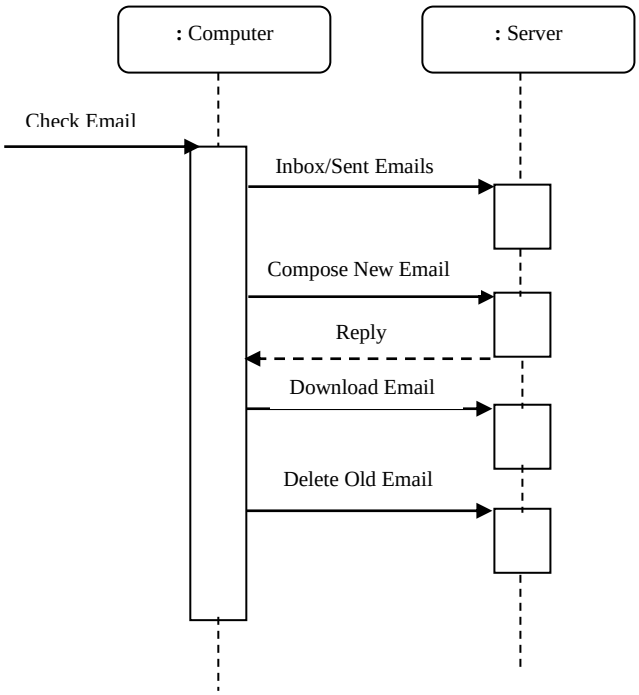

Figure 2: Sequence Diagram of Email System

- State Chart Diagram: State chart diagram illustrates about the object states and state transition in UML diagram [10, 11]. State chart diagrams have the ability to tolerate the unit level faults. It graphically represents finite state machine. For example: A finite state machine for ticket reservation system is shown in figure 3.

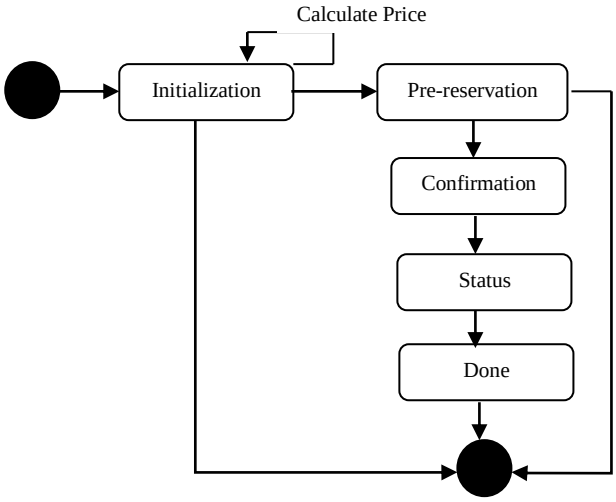

Figure 3: State Chart Diagram based Finite State Machine

Both these UML diagrams work efficiently but their integrated form can even handle the message interaction, scenario faults, message sequence faults, correctness and error handling etc. In this paper, systematic analysis is made on the sequence diagram, state chart diagram and their integrated form for software based test case generation.

The rest of the paper is organized in the following manner: Section II describe the test cases in UML. Section III explains the considered concepts for test case generation, Section IV presents the test case generation model and Section V concludes the paper with some future directions.

II. MODEL BASED SOFTWARE TESTING

Software testing involves the testing of a product for different test cases. For quality test, there should be proper designing of test cases and test data [12]. Researchers are continuously working to develop the autonomous method to generate superlative test data and test cases for testing [13]. Basically, software testing is of two categories: functional testing and structural testing. Functional testing is black box testing [14] and structural testing is white box testing [15]. Another important type of test is model based testing. Model based testing concentrates on design work instead of source code. In this paper, UML model based testing is considered for test case generations. A general model based testing process is shown in figure 4. Moreover model based testing is easy to understand for developers with ease to generate test cases [16].

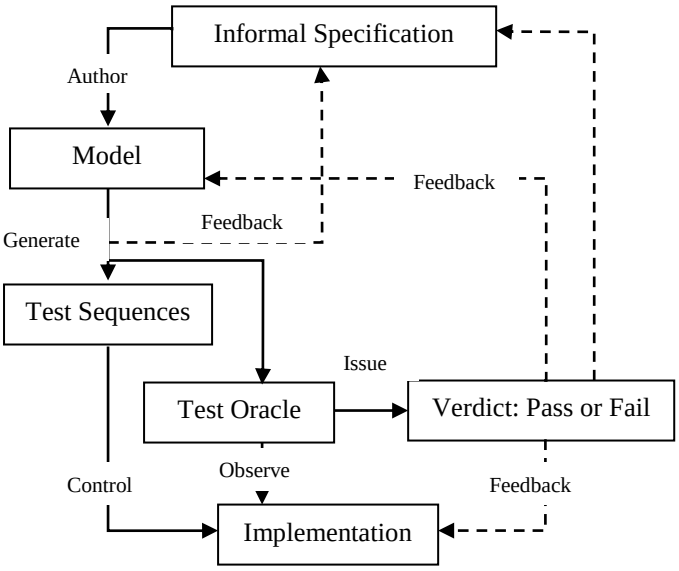

Figure 4: Model based Testing Process

III. TEST CASE GENERATION USING UML DIAGRAMS

In this section, test case generation is explained using UML sequence diagram, state chart diagram and integrated approach of sequence diagram and state chart diagram. Different authors have presented their concept for test case generation using UML diagram. Also a systematic analysis is described in Table 2 for test case generation.

In [17] authors adapted the approach of statistical probabilistic testing for the creation of test cases from UML state chart diagram. The considered approach is adapted for the object oriented software testing. For the automation of testing criteria, authors have used Rational Rose Software. For the evaluation of concept, authors have implemented proposed approach on FGS (Flight Guidance System) case study. In this study, test cases are generated for the Java based files and exhibits fault revealing power. The proposed concept supports transition coverage for the test cases.

In [18] authors generated the test cases using UML state machine diagrams. The proposed approach involves the steps of test case generation as mentioned: selection of predicates by transition of UML state diagram using DFS (Depth First Search) approach, transformation of predicates to predicate functions using minimization function and final step is to generate test cases as per predicate function. Authors have considered the example of Ice Cream Vending Machine for the evaluation of testing criteria. The proposed concept is implemented in Java with integration of MagicDraw Tool. Evaluated results exhibit transition path coverage. The considered approach can also handle transition with guards, time events and change events.

In [19] authors have extended the work of Variable Assignment Graph by introducing the UML state machine diagrams along with sequence diagram for test case generation. Sequence diagram can extract the message paths that are essential for a system. Further, state machine diagram can generate more execution paths from message paths by system state transition. For the evaluation of considered approach, authors have applied the concept for mutation analysis on a video store system. The evaluation results shows effectiveness of approach in terms of message transition and selective test cases with reduced cost values.

In [20] authors applied minimization method to automatically generate test cases from UML state chart diagram. This test case generation process is initialized with the construction of state chart diagram for an object. Further, state chart diagram is traversed, selection of conditional predicates occurs and converted into source code. Authors have applied DFS (Depth First Search) algorithm for the selection of associated predicates. Then an initial dataset is generated based on the selected predicates and finally test cases are generated. The applied method shows the capability for action coverage,

transition pair coverage, transition coverage and state coverage. But the considered method does not concentrate on the optimization of test cases.

In [21] authors proposed ATGSD (Automatic Test Sequence Generation from Sequence Diagram) for test case generation. The proposed approach involves the steps of construction of sequence diagram, conversion of sequence diagram into sequence graph, model parsing to maintain the path traversal details, use of DFS for the selection of predicates, transformation of predicates into java source code using ModelJUnit library, test case generation using minimization approach and then test case machine & result storage. The complete process is applied for the test cases of Bank ATM. The considered approach achieves the coverage like action coverage, message path pair coverage, message path coverage and object coverage. But there is the need to optimize test cases.

In [22] authors have used state chart diagram for the generation of test cases. State chart diagram maintains the possible states of object and their transition. For the generation of test cases, authors have initially converted the state diagram into Finite State Machine (FSM). In FSM, state is represented by node and transition state is represented by arrows. For the pre & post condition of use cases, authors have used the object constraint language (OCL). The overall approach consumes less time with better software quality and optimum number of test cases. To illustrate the proposed concept, authors have discussed the application of ATM banking. The proposed concept accomplishes the state coverage, transition pair coverage and transition coverage for generated test cases.

In [23] authors presented a novel slicing based approach for test case creator using UML 2.0 sequence diagrams. These sequence diagrams are used for the construction of message dependency graph (MDG) which is further traversed for the selection of conditional predicates. Then slicing is done for each conditional predicate to examine test cases as their complexities of software. The generated test cases are efficient to detect operational faults and object interactions. The proposed concept shows 80% satisfy results for message path coverage and slice coverage. But still there is need to maintain synchronization in message dependency graph for test cases.

In [24] authors used UML diagram for test case generation especially for cluster level source code. Initially, sequence diagrams are generated using Rational Software Architecture (RSA). Then, sequence flow chart is created from state chart diagram which is further converted into Message Control Flow Graph (MCFG). From MCFG, test paths are generated using Depth First Search (DFS) algorithm. Then test cases are created from test paths and finally Genetic Algorithm (GA) is applied for the optimization of concept. The considered concept is evaluated for ATM transaction system. The effectiveness of concept is shown in terms of selective test cases and reduced time & cost values.

In [25] authors used UML state chart diagram and sequence diagram for the generation of test cases. Authors have designed a SYstem Testing Graph (SYTG) by integrating sequence graph and state chart graph which is a converted from sequence diagram and state chart diagram respectively. Rational Rose is used for UML diagram and Genetic Algorithm (GA) is used to generate test cases from SYTG based paths. GA is also used for the optimization of test cases. Both sequence diagram & state chart diagram are efficient to reveal integration level faults & unit level faults respectively. But integrated approach is more appropriate to detect scenario faults, message sequence faults, integration, pre-post condition faults, correctness and error handling etc. The considered concept only lacks for the non-autonomous behaviour of test case generation.

Table 2: Test Case Generation using UML Diagrams

Authors & Year	Used UML Diagram	Case Study	Tools/Platform Used	Key Features
Clevalley and Fosse (2001) [17]	State Chart Diagram	Flight Guidance System (FGS)	Rational Rose and Java Program	**1.** Fault revealing power. **2.** Supports transition coverage.
Samuel et al. (2008) [18]	State Machine Diagram	Ice Cream Vending Machine	UML behavioural test case generator (UTG), Java Platform and MagicDraw	**1.** Exhibit transition path coverage. **2.** Handle transition with guards, time events and change events.

Bandyopadhyay and Ghosh (2008) [19]	State Machine Diagram and Sequence Diagram	Mutation Analysis on a Video Store System	Java Library Functions	**1.** Effectiveness of approach in terms of message transition and selective test cases with reduced cost values.
Swain et al. (2012) [20]	State Chart Diagram	Soft Drink Vending Machine	Rational Rose, JAVA library file ModelJUnit and Net Beans IDE and	**1.** Shows the capability for action coverage, transition pair coverage, transition coverage and state coverage. **2.** Does not concentrate on the optimization of test cases.
Panthi and Mohapatra (2013) [21]	Sequence Diagram	Bank ATM Transactions	JAVA based library file ModelJUnit, Net Beans IDE and and Rational Rose	**1.** Achieves the coverage like action coverage, message path pair coverage, message path coverage and object coverage. **2.** But there is the need to optimize test cases.
Ali et al. (2014) [22]	State Chart Diagram	ATM Banking	Visual Paradigm v. 8.0 and Java based Parser	**1.** Consumes less time with better software quality and optimum number of test cases. **2.** Accomplishes the state coverage, transition pair coverage and transition coverage.
Swain et al. (2014) [23]	Sequence Diagram	Slicing based Cases	ModelJUnit, JAVA and Net Beans IDE, Swing component of Java	**1.** Efficient to detect operational faults and object interactions. **2.** Shows 80% satisfy results for message path coverage and slice coverage.
Jain et al. (2015) [24]	Sequence Diagram	ATM Transaction System	Rational Software Architecture (RSA) and Java under NetBeans IDE	**1.** Selective test cases and reduced time & cost values.
Khurana and Chillar (2015) [25]	State Chart Diagram and Sequence Diagram	Online Voting System	Rational Rose	**1.** Appropriate to detect scenario faults, message sequence faults, integration, pre-post condition faults, correctness and error handling, considered concept. **2.** Only lacks for the non-autonomous behavior of test case generation.

IV. TEST CASE GENERATION METHOD USING INTEGRATION OF STATE CHART DIAGRAM AND SEQUENCE DIAGRAM

The concept of integrated UML sequence diagram and state chart diagram to generate test cases using genetic algorithm has been proposed by Khurana and Chillar [25]. Here, authors have used unit level fault tolerance state chart diagram and integration level fault tolerance sequence diagram for test case generation. Final optimization is performed using evolutionary genetic algorithm. The steps for the test case generation are as discussed in Table 3. The work flow for the algorithm is shown in figure 5.

Table 3: Test Case Generation using Integrated Sequence and State Chart Diagram

Step 1: Consider UML Sequence and State Chart Diagram.
Step 2: Convert Sequence Diagram into Sequence Graph.
Step 3: Convert State Chart Diagram into State Chart Graph.
Step 4: Integrate both the graphs and generate System Testing Graph (SYTG).
Step 5: Generate the test cases using integrated SYTG graph.
Step 6: Optimize the test cases using Genetic Algorithm.

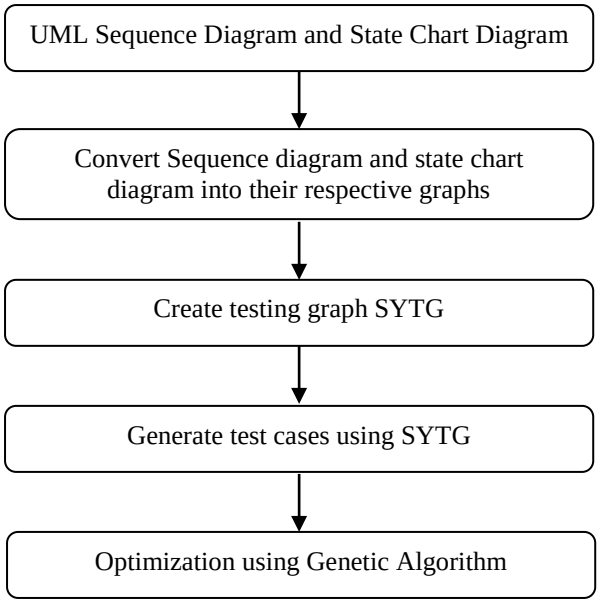

Figure 5: Work flow for Test case generation

V. CONCLUSION

Software testing is an important aspect of software development life cycle as it checks for the matching of software demand and their fulfil functionalities. The traditional methods of software testing lacks system due to manual source code based testing. In this paper, UML model based test cases generation methods are considered which are efficient approaches even for high software complexities. The overall objective of the paper is to make a comparative analysis of the existing methods for test case generation using UML sequence diagram, state chart diagram and their integrated approach. Different methods uses different tools for test case generation and supports different coverage like action coverage, transition pair coverage, transition coverage and state coverage etc. Each test case generation method is suitable for different case studies. So, there is the need of some efficient method with advanced computing so that testing can be completed with reduced cost and time. Also need of optimization in generated test cases so that there should be some specific test cases for the testing in software development life cycle.

REFERENCES

[1]. Myers, Glenford J., Corey Sandler, and Tom Badgett. *The art of software testing*. John Wiley & Sons, ISBN: 9781118031964, 2011.

[2]. Hailpern, Brent, and Padmanabhan Santhanam. "Software debugging, testing, and verification." *IBM Systems Journal* , ISSN: 0018-8670, vol no. 41, issue no. 1, pp. 4-12, 2002.

[3]. Mellor, Stephen J., Marc Balcer, and Ivar Foreword By-Jacoboson. *Executable UML: A foundation for model-driven architectures*. Addison-Wesley Longman Publishing Co., Inc., ISBN: 0201748045, 2002.

[4]. Kovse, Jernej, and Theo Härder. "Generic XMI-based UML model transformations." In *International Conference on Object-Oriented Information Systems*, Springer Berlin Heidelberg, ISBN: 978-3-540-44087-1 , series vol. 2425, pp. 192-198, 2002.

[5]. Offutt, Jeff, and Aynur Abdurazik. "Generating tests from UML specifications." In *International Conference on the Unified Modeling Language*, Springer Berlin Heidelberg, ISBN: 978-3-540-66712-4 , Series vol. 1723, pp. 416-429., 2003.

[6]. Booch, Grady. *The unified modeling language user guide*. Pearson Education India, ISBN: 978-81-317-1582-6, 2005.

[7]. Idani, Akram, and Yves Ledru. "Dynamic graphical UML views from formal B specifications." *Information and Software Technology* , ISSN: 0950-5849 , vol. no. 48, issue no. 3, pp. 154-169, 2006.

[8]. Sarma, Monalisa, Debasish Kundu, and Rajib Mall. "Automatic test case generation from UML sequence diagram." In *Advanced Computing and Communications, 2007. ADCOM 2007. International Conference on*, ISBN: 0-7695-3059-1 , pp. 60-67. IEEE, 2007.

[9]. Li, Xiaoshan, Zhiming Liu, and He Jifeng. "A formal semantics of UML sequence diagram." In *Software Engineering Conference, 2004. Proceedings. 2004 Australian*, ISBN : 0-7695-2089-8 , pp. 168-177. IEEE, 2004.

[10]. Bernardi, Simona, Susanna Donatelli, and José Merseguer. "From UML sequence diagrams and statecharts to analysable petri net models." In*Proceedings of the 3rd international workshop on Software and performance*, ISBN: 1-58113-563-7, pp. 35-45. ACM, 2002.

[11]. Whittle, Jon, and Johann Schumann. "Generating statechart designs from scenarios." In *Software Engineering, 2000. Proceedings of the 2000 International Conference on*, ISBN: 1-58113-206-9, pp. 314-323. IEEE, 2000.

[12]. El-Far, Ibrahim K., and James A. Whittaker. "Model-Based Software Testing."*Encyclopedia of Software Engineering* ISBN: 9780471028956, 2001.

[13]. Rubin, Jeffrey, and Dana Chisnell. *Handbook of usability testing: how to plan, design and conduct effective tests*. John Wiley & Sons, ISBN: 978-81-265-1690-2, 2008.

[14]. Beizer, Boris. *Black-box testing: techniques for functional testing of software and systems*. John Wiley & Sons, Inc., ISBN: 0-471-12094-4 , 1995.

[15]. Haley, Allen, and Stuart Zweben. "Development and application of a white box approach to integration testing." *Journal of Systems and Software* ISSN: 0164-1212 , vol. 4, issue no. 4, pp. 309-315, 1984.

[16]. Dalal, Siddhartha R., Ashish Jain, Nachimuthu Karunanithi, J. M. Leaton, Christopher M. Lott, Gardner C. Patton, and Bruce M. Horowitz. "Model-based testing in practice." In *Proceedings of the 21st international conference on Software engineering*, ISBN: 1-58113-074-0 , pp. 285-294. ACM, 1999.

[17]. Chevalley, Philippe, and Pascale Thévenod-Fosse. "Automated generation of statistical test cases from UML state diagrams." In *Computer Software and Applications Conference, 2001. COMPSAC 2001. 25th Annual International*, ISBN: 0-7695-1372-7 , pp. 205-214. IEEE, 2001.

[18]. Samuel, Philip, R. Mall, and Ajay Kumar Bothra. "Automatic test case generation using unified modeling language (UML) state diagrams." *IET software*,ISSN : 1751-8806, vol 2, issue no. 2, pp. 79-93, 2008.

[19]. Bandyopadhyay, Aritra, and Sudipto Ghosh. "Using UML Sequence Diagrams and State Machines for Test Input Generation." In *2008 19th International Symposium on Software Reliability Engineering (ISSRE)*, ISSN: 1071-9458 , pp. 309-310. IEEE, 2008.

[20]. Swain, Ranjita, Vikas Panthi, Prafulla Kumar Behera, and Durga Prasad Mohapatra. "Automatic test case generation from UML state chart diagram."*International Journal of Computer*

Applications ISSN: 0975 – 8887, vol. 42, issue no. 7 pp.- 26-36, 2012.

[21]. Panthi, Vikas, and Durga Prasad Mohapatra. "Automatic test case generation using sequence diagram." In *Proceedings of International Conference on Advances in Computing*, ISBN: 978-81-322-0739-9 , pp. 277-284. Springer India, 2013.

[22]. Ali, Md Azaharuddin, Khasim Shaik, and Shreyansh Kumar. "Test case generation using UML state diagram and OCL expression." *International Journal of Computer Applications* ISSN: 0975 – 8887, vol 95, issue no. 12 (2014).

[23]. Swain, Ranjita Kumari, Vikas Panthi, Prafulla Kumar Behera, and Durga Prasad Mohapatra. "Slicing-based test case generation using UML 2.0 sequence diagram." *International Journal of Computational Intelligence Studies 2*, ISSN: 1755-4977, vol. 3, issue no. 2-3 pp. 221-250, 2014.

[24]. Jena, Ajay Kumar, Santosh Kumar Swain, and Durga Prasad Mohapatra. "Test case creation from UML sequence diagram: a soft computing approach." In *Intelligent Computing, Communication and Devices*, ISBN: 978-81-322-2011-4 , pp. 117-126. Springer India, 2015.

[25]. Khurana, Namita, and R. S. Chillar. "Test Case Generation and Optimization using UML Models and Genetic Algorithm." *Procedia Computer Science*, ISSN: 1877-0509, vol. 57, pp. 996-1004, 2015.