International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 10 | Oct -2016 www.irjet.net p-ISSN: 2395-0072

Generate Test Cases From UML Use Case and State Chart Diagrams

Shubhangi Jagtap[1], Vishwas Gawade[2], Rahul Pawar[3], Savita Shendge[4], Prof. Pravin Avhad[5]

[1]*Savitribai Phule Pune University,*
Dept. of computer engineering, H.S.B.P.V.T.COE, Kashti, Maharashtra, India
shubhangijagtap1992@gmail.com

[2]*Savitribai Phule Pune University,*
Dept. of computer engineering, H.S.B.P.V.T.COE, Kashti, Maharashtra, India
vishwasgawade96@gmail.com

[3]*Savitribai Phule Pune University,*
Dept. of computer engineering, H.S.B.P.V.T.COE, Kashti, Maharashtra, India
pawarrahul00010@gmail.com

[4]*Savitribai Phule Pune University,*
Dept. of computer engineering, H.S.B.P.V.T.COE, Kashti, Maharashtra, India
shendgesavita801@gmail.com

[5]*Savitribai Phule Pune University,*
Professor, Dept. of computer engineering, H.S.B.P.V.T.COE, Kashti, Maharashtra, India
pravin123.avhad@gmail.com

Abstract - *Software testing is a part of software development process. However, this part is the first one to miss by software developers if there is a limited time to complete the project. Software developers often finish their software construction closed to the delivery time, they usually don't have enough time to create effective test cases for testing their programs. Creating test cases manually is a huge work for software developers in the rush hours. A tool which automatically generates test cases can help the software developers to create test cases from software designs/models in early stage of the software development (before coding).*

In this project, test cases generation technique has been proposed for UML diagrams use case and state chart. So that test data can be generated before coding, so it will be useful for the tester because, Test Engineering covers a large amount of activities to ensure that the final product achieves some quality goal.

***Key Words*: Software Testing, Test cases, Tools, Use case diagram, State chart diagram, Symbol, Generate test cases, Excel chart, Tree structure, Text box.**

1. INTRODUCTION

Software testing is an important activity to assure the quality of software. Unfortunately, software testing is very labor intensive and very expensive. It can take about 50 percent's of total cost in software developing process. Automated test data generation reduces an effort of software developers for creating test cases.

The software testers may need to spend a longer time using many test cases if the test data used are not of high quality. Therefore, a performance of executing test data is an important issue to reduce the testing time. Software testing is usually the first part of software development stages, which software developers decide to omit when there is a

limited time to deliver the software. In other word, developers may nothave enough time after they finished their coding to create test cases to test their code. Generating test cases before coding can resolve these problems. This not only helps developers to test their program when they finish coding but also controls the developers to program the software as defined in the softwarespecification

complex operation modeling etc. Main advantage of this model is its simplicity and ease of understanding the low of logic of the system. However, ending test information from use case and state chart is a formidable task.

In this work, we propose an approach for generating test cases using UML 2:0 use case diagrams and state chart diagram

We are going to use following UML Diagrams:-
1. Use Case Diagram
2. State Chart Diagram

RELATED WORK.

Software testing plays a major role in software development process because it accounts for a large part of the development cost. Moreover, manual testing technique always makes a problem. This project proposes the automatic testing technique to solve partially the testing process by generating test cases from UML Diagram Use Case and State Chart.

Firstly, we will create our own tool for drawing UML Diagram Use Case and State Chart then we transform this diagram into intermediate tree, called Testing Flow Tree. Secondly, we generate test case using the testing criteria that is the coverage of the state and transition of diagrams. Finally, we will interpret these test cases in Microsoft Office Excel (XLS) format as output.

1. PROPOSED WORK

Creating test cases manually is a huge work for software developers in the rush hours. A tool which automatically generates test cases can help the software developers to create test cases from software designs/models in early stage of the software development

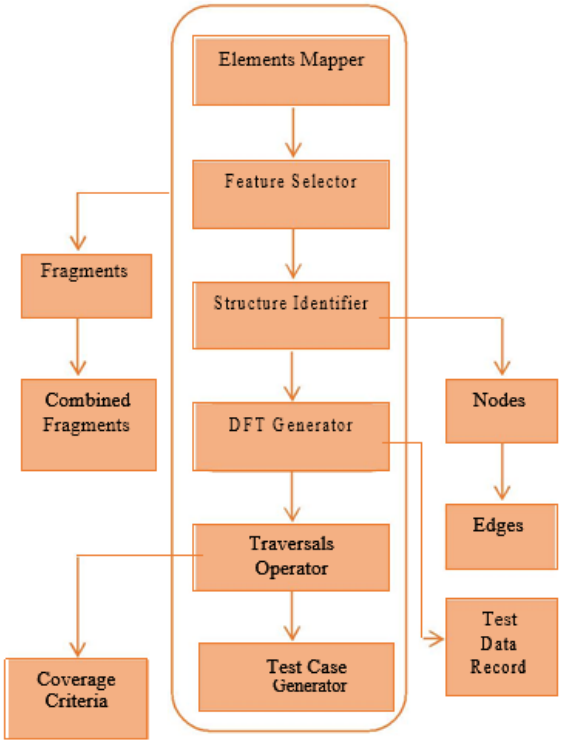

Figure 1. Proposed Test Case Generation Technique

2. Test Case Generation for Use Case Diagram with necessary information

Using use cases to generate test cases can help to launch the testing process early in the development life cycle and also help with testing methodology.

In a software development project, use cases define system software requirements. Use case development begins early on, so real use cases for key product functionality are available in early iterations. A use case fully describes a sequence of actions performed by a system to provide an observable result of value to a person or another system using the product under development.

Use cases tell the customer what to expect, the developer what to code, the tech- nical writer what to document, and the tester what to test.

Test cases are key to the process because they identify and communicate the conditions that will be implemented in test and are necessary to verify successful and acceptable implementation of the product requirements. They are all about making sure that the product fulfills the requirements of the system

3. Test Case Generation for State Chart Diagram with necessary information

Object oriented analysis and design methods offer a good framework for behavior. A UML state chart (state machine) describes the dynamics of a model element as it changes its internal state as the reaction of receiving some external stimuli. UML statE charts can describe the behavior of a classifier (a class) or a behavioral feature (a method of a class).

A state machine is a graph of states and transitions that describes the response of an object to the receipts of events. State machines are used for specifying the full dynamic behavior of a single class of objects. The diagrammatic presentation of a state machine is a state chart diagram. A state chart attached to a class specifies all behavioral aspects of the objects in that class.

State chart diagrams comprise all the possible scenarios for a given object. They emphasize the flow of control from state to state.

Structure Identifier

Since the proposed approach aims to generate test cases from any UML diagram, it becomes imperative to develop a unified structure identifier capable of identifying the nodes and edges in XMI files of UML diagrams. Therefore, if an XMI file of any UML diagram is imported, the Elements Mapper is responsible for identifying the correct diagram source of the XMI file based on the descriptive attributes of the various UML diagrams and then, correlates these attributes to the corresponding diagram based on the running procedures. The Feature Selector refines the mapped elements to aid accurate identification of nodes and edges in an XMI file. In the proposed approach, mapping is executed by considering the nomenclature of the various UML diagrams. The contents of an XMI file consists of metametamodel, comprising of the XML viewer, the element metamodel which provides the name and version of the XMI file, XMI contents which consist of the UML model and this model consists of XMI.id, UML diagram name, and Namespace.ownedElement. The requirement name and its attributes reside in the Namespace.ownedElement. Consequently, in the proposed approach, the metametamodel, XMI metamodel, model and namespace elements were used to identify the structure of XMI document across UML diagrams. In this research, the nodes connote the requirements while the attributes describing the expected functionalities of a requirement is known as an edge. Therefore, to identify the structure of an XMI file, labels of elements was used. The elements associated with the UML IDs are Nodes while the attributes of the elements are the Edges. Algorithm 1 was used to determine distinct nodes and edges of XMI files which accept XMIs as a input

1: **Input:** XMI txt.file of any UML diagrams
2: **Ouput:** Number of Nodes and Edges
3: Begin
4: for each element, visit the unique XMI IDs **do**
5: element = name 'A' visibility;
6: if element = 'public'**then** isSp
7: add decision stack indicating a root Node;
8: continue with next element;
9: **end**
10: prevPath element.descriptor.path;
11: newPath getPath (element);
12: if newPath**then**≠ prevPath
13: prevName lastSegment(prevPath);
14: newName lastSegment(newPath);
15: SubNode getGeneralizationText(child node);
16: add decision stack indicating a sub Node;
17: continue with next element;
18: sort the root and sub nodes into XMI value pair list;
19: **end**
20: if relation of nodes and sub nodes exist (element)

21: then

22: order the relations in the generated list

```
23          then
24: Create an ordered list of all relations;
25: Determine the length of relations;
26: label them edges then
27: add decision stack indicating edges;
28:                 end
29: end
30: from XMI value pair list (element.attributes) then
31: Compute array of node and edges
32: end
```

Algorithm 1. Structure Identification

Dependency Flow Tree (DFT) Generator

This component is responsible for building a dependency flow tree based on the identified structure. The dependency tree is built based on the number of Nodes and Edges contained in an XMI file using Algorithm 2. It verifies that the Nodes and Edges corresponding to elements and attributes respective.

```
1:  Input: Extracted artefacts;
2:  Ouput: DFT
3:  initTgt.DFT  XMI file [L]
4:  for i = 1 to L do
5:  XMI file (createNodes.elements; edges.attributes)
6:  addElement (Nodes [edge.attributeSize])
7:  end for
8:  ModelTgt(visitList (element.descriptions, DFT)
9:  for i = 1 to L do
10: XMI file = DFT
        end for
```

The design model was constructed using ArgoUML tool which support XMI file format. It includes class diagrams, sequence diagrams, state charts and so on. The shared model approach is used for test case generation. The same model is used for extracting artefacts as well as for test case generation. A transformation tool or some adaptor transformers that is embedded in the proposed approach can be used to translate abstract test case into an executable or concrete test cases which uses certain templates or mappings to ensure completeness between the extracted artefacts and generated test cases.

```
1:  Input: Dependency flow tree (DFT)
2:  Output: Set of test cases
3:  artefactsStack=Ø
4:  decisionStack=Ø
5:  for all elements and attributes of DFT do
6:      while DFT == Nodes.Edges do
7:      artefactsStack.push(DFT.contents) whiile
8:  end for
9:   for all elementsNodes do
10:  artefactsStack.push edges
11:     if artefactsStack[top].node≠ al||loop.node|| par.node||break.node||
12:    then
13:     decisionStack.push(artefactsStack.top) = elements.attributes and label visited
14:  else
        artefcatsStack.top == Nodes.decisions
16:  then
17:       artefactsStack[top] = decisionStack[top] do
18:           output.pop {pop elements and attributes into artefactsStack}
19:       end while
20:       decisionStack.pop {Pop the top element from decisionStack}
21:  else if artefactsStack[top] == ||alt.node|| break.node || loop node
22:  then
23:  decisionStack.push(artefcatsStack.pop)
24:       for all nodes of DFT do
25:           if nodes are not Marked Visited then
26:               artefactsStack.push Nodes and Edges
27:           end if
28:      end for
29:  Print artefactstStack.decisionStack
30:  end
```

Algorithm 3. Test Case Generation

1.Ex. Use case diagram

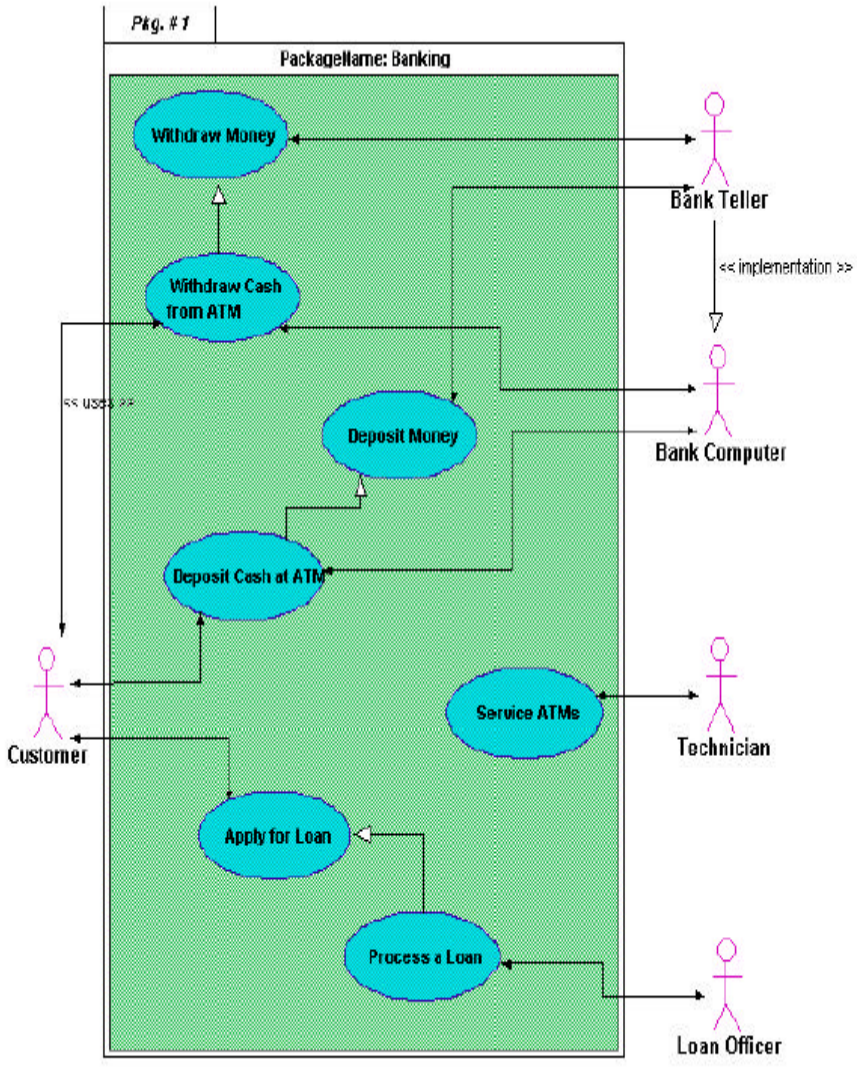

Fig2. Banking system

2.Ex. State chart diagram

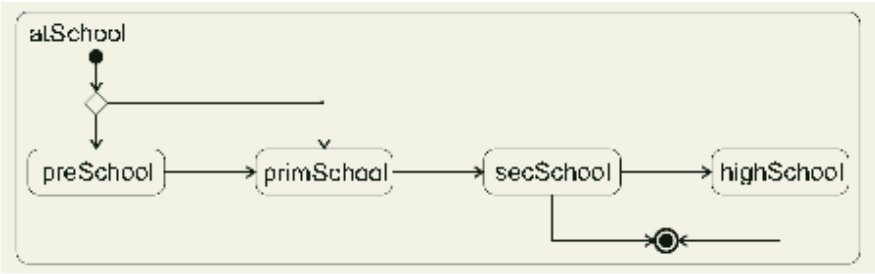

Fig3. People education status

4. ADVANTAGES

1.Automated test data generation reduces an effort of software developers for creating
Test cases.
2. Reduce the cost and reduce the time of testing.
3. Meet the customer requirement and satisfaction since the testing is applied from
Starting phase of software development process.
4. Gives assurance about quality of software processes.
5. At design time, we are generating test cases due that we are reducing 70 percent
Testing work at this stage.
6. Automated test data generation reduces an effort of software developers for creating
Test cases.
7. Reduce the testing time after coding.

5. CONCLUSION

In this paper, we have presented an approach for generating test cases from use case and state chart at use case scope. Our approach is significant due to the following reasons. First, our approach is capable to detect more faults like faults in loop,
synchronization faults than the existing approaches. Second, test case generated in our approach may help to identify location of a fault in the implementation, thus reducing testing effort. Third, our model-based test case generation approach inspires developer to improve design quality, and faults in the implementation early, and reduce software development time. Fourth, it is possible to build an automatic tool following our approach. This automatic tool will reduce cost of software development and improve quality of the software.

In the present submission, we have focused only use case and state chart of a single use case at a time. However, use case of multiple which are related to each other by various relationships such as, include, extend, generalization / specialization can be considered, which we plan to take up in our next work.

This project proposes the automatic testing technique to solve partially the testing process by generating test cases from UML Diagram Use Case and State Chart. A tool which automatically generates test cases can help the software developers to Create test cases from software designs/models in early stage of the software development (Before coding).

ACKNOWLEDGEMENT

First and foremost, we would like to thank our guide, Prof.Avhad P.S. for his guidance and support. We will forever remain grateful for the constant support and guidance extended by guide, in making this report. Through our many discussions, he helped us to form and solidify ideas. The invaluable discussions we had with him, the penetrating questions he has put to us and the constant motivation, has all led to the development of this project.
We would like to convey our sincere and heart rendering thanks to Principal Prof. Mahadik S.N. for his co-operation, valuable guidance.
Also we wish to express our sincere thanks to the Head of department, Prof. Tarte V.G. and the departmental staff members Prof.Taware C.C., Prof.Gunaware N.G., Prof.Hirave K.S., and Prof. Hiranawale S.B. for their support.

Shubhangi Jagtap1,
Vishwas Gawade2,
Rahul Pawar3,
Savita Shendge4

REFERENCES

[1]. C. Nebut, F. Fleurey and Y.L. Traon, Automatic Test Generation: A Use Case
Driven Approach, IEEE TRANSACTION ON SOFTWARE ENGINEERING Vol.32,
No.3

[2]. KIM, Y.G., HONG, H.S., CHO, S.M., BAE, D.H., AND CHA, S.D. 1999. Test
Cases generation from UML state diagrams. In IEEE Software 146(4): 187-192,
1999.

[3]. BOOCH, G., RUMBAUGH, J., AND JACOBSON, I. 1998. The Unified Modeling
Language User Guide. Object Technology Series. Addison Wesley Longman, Inc.

[4] A. C. D. Neto, R. Subramanyan, M. Vieira and G. H. Travassos, "A survey on model-based testing approaches: a systematic review", In Proceedings of the 1st ACM international workshop on Empirical assessment of software engineering languages and technologies: held in conjunction with the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE), ACM, **(2007)**, pp. 31-36.

[5] W. Zheng and G. Bundell, "Model-based software component testing: A UML-based approach", In Computer and Information Science, 2007. ICIS 2007. 6th IEEE/ACIS International Conference on IEEE, pp. 891-899.

[6] P. Kaur and R. Kaur, "Approaches for Generating Test Cases Automatically to Test the Software", International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249–8958, vol. 2, Issue 3, **(2013)** February.

[7] T. Hussain and G. Frey, "UML-based development process for IEC 61499 with automatic test-case generation", In Emerging Technologies and Factory Automation, ETFA'06. IEEE Conference on IEEE, **(2006)**, pp. 1277-1284.

[8] D. Xu and X. He, "Generation of test requirements from aspectual use cases", In Proceedings of the 3rd workshop on testing aspect-oriented programs, ACM, **(2007)**, pp. 17-22. International Journal of Software Engineering and Its Applications Vol. 9, No. 8 (2015) 104 Copyright ⓒ 2015 SERSC

[9] C. Nebut, F. Fleurey, Y. Le Traon and J. M. Jezequel, "Automatic test generation: A use case driven approach", Software Engineering, IEEE Transactions on, vol. 32, no. 3, **(2006)**, pp. 140-155.

[10] S. Ogata and S. Matsuura, "A method of automatic integration test case generation from UML-based scenario", WSEAS Trans Inf Sci., Appl., vol. 7, no. 4, **(2010)**, pp. 598-607.

[11] V. Santiago, A. S. M do Amaral, N. L. Vijaykumar, M. F. Mattiello-Francisco, E. Martins and O. C Lopes, "A practical approach for automated test case generation using statecharts", In Computer Software and Applications Conference, COMPSAC'06, 30th Annual International, IEEE, vol. 2, **(2006)**, pp. 183-188.

[12] S. R. Dalal, A. Jain, N. Karunanithi, J. M. Leaton, C. M. Lott, G. C. Patton and B. M. Horowitz, "Model-based testing in practice", In Proceedings of the 21st international conference on Software engineering, ACM, **(1999)**, pp. 285-294.

[13] A. Kaur and V. Vig, "Systematic Review of Automatic Test Case Generation by UML Diagrams", International Journal of Engineering, vol. 1, no. 6, **(2012)**.

[14] J. J. Gutiérrez, M. J. Escalona, M. Mejías and J. Torres, "Generation of test cases from functional requirements: A survey", In 4th Workshop on System Testing and Validation, **(2006)** Potsdam. Germany.

[15] P. V. R Murthy, P. C. Anitha, M. Mahesh and R. Subramanyan, "Test ready UML statechart models", In Proceedings of the international workshop on Scenarios and state machines: models, algorithms, and tools, **(2006)**, pp. 75-82.

[16] D. Sokenou, "Generating Test Sequences from UML Sequence Diagrams and State Diagrams", In GI Jahrestagung, no. 2, **(2006)**, pp. 236-240.

[17] S. Bangalore, O. Rambow and S. Whittaker, "Evaluation metrics for generation", In Proceedings of the first international conference on Natural language generation, Association for Computational Linguistics, vol. 14, **(2000)**, pp. 1-8.

[18] P. Samuel, R. Mall and P. Kanth, "Automatic test case generation from UML communication diagrams", Information and software technology, vol. 49, no. 2, **(2007)**, pp. 158-171.

[19] R. K. Swain, V. Panthi and P. K. Behera, "Test case design using slicing of UML interaction diagram", Procedia Technology, vol. 6, **(2012)**, pp. 136-144.

[20] L. Li, X. Li, T. He and J. Xiong, "Extenics-based Test Case Generation for UML Activity Diagram", Procedia Computer Science, vol. 17, **(2013)**, pp. 1186-1193.

[21] P. E. Patel and N. N. Patil, "N. N. Test cases Formation Using UML Activity Diagram", In Communication Systems and Network Technologies (CSNT), International Conference, IEEE, **(2013)**, pp. 884-889.

[22] R. Hametner, B. Kormann, B. Vogel-Heuser, D. Winkler and A. Zoitl, "Test case generation approach for industrial automation systems", In Automation, Robotics and Applications (ICARA), 5th International Conference, **(2011)**, pp. 57-62.

[23] H. H. Inbarani, P. K. N. Banu and A. T. Azar, "Feature selection using swarm-based relative reduct technique for fetal heart rate", Neural Computing and Applications, pp. 1-14.

[24] M. Sarma and R. Mall, "Automatic generation of test specifications for coverage of system state transitions", Information and Software Technology, vol. 51, no. 2, **(2009)**, pp. 418-432.

[25] L. Briand and Y. Labiche, "A UML-based approach to system testing", Software and Systems Modeling, vol. 1, no. 1, **(2002)**, pp. 10-42.

[26] A. Abdurazik and J. Offutt, "Using UML collaboration diagrams for static checking and test generation", In UML 2000-The Unified Modeling Language, Springer Berlin Heidelberg, **(2001)** pp. 383-395
.
[27] F. Zeng, Z. Chen, Q. Cao and L. Mao, "Research on Method of Object-Oriented Test Cases Generation Based on UML and LTS", In Information Science and Engineering (ICISE), 1st International Conference, IEEE, **(2009)**, pp. 5055-5058.

[28] A. García-Domínguez, I. Medina-Bulo and M. Marcos-Bárcena, "An Approach for Model-Driven Design and Generation of Performance Test Cases with UML and MARTE", In Software and Data Technologies, Springer Berlin Heidelberg, **(2013)**, pp. 136-150.

[29] M. Prasanna and K. R. Chandran, "Automated Test Case Generation for Object Oriented Systems Using UML Object Diagrams", In High Performance Architecture and Grid Computing, Springer Berlin Heidelberg, **(2011)**, pp. 417-423.
[30] V. Sawant and K. Shah, "Construction of Test Cases from UML Models", In Technology Systems and Management, Springer Berlin Heidelberg, **(2011)**, pp. 61-68.

[31] A. Nayak and D. Samanta, "Synthesis of test scenarios using UML activity diagrams", Software & Systems Modeling, vol. 10, no. 1, **(2011)**, pp. 63-89.

[32] S. Asthana, S. Tripathi and S. K. Singh, "A Novel Approach to Generate Test Cases Using Class and Sequence Diagrams", In Contemporary Computing, Springer Berlin Heidelberg, **(2010)**, pp. 155-167.

[33] O. Oluwagbemi and H. Asmuni, "An Improved Model-Based Technique for Generating Test Scenarios from UML Class Diagrams", Handbook of Research on Emerging Advancements and Technologies in Software Engineering, **(2014)**, pp. 434-448.

[34] O. Pilskalns, A. Andrews, A., Knight, S. Ghosh and R. France, "Testing UML designs", Information and Software Technology, vol. 49, no. 8, **(2007)**, pp. 892-912.
[35] B. P. Lamancha, M. Polo, D. Caivano, M. Piattini and G. Visaggio, "Automated generation of test oracles using a model-driven approach", Information and Software Technology, vol. 55, no. 2, **(2013)**, pp. 301-319.

[36] R. K. Swain, V. Panthi, D. P. Mohapatra and P. K. Behera, "Prioritizing test scenarios from UML communication and activity diagrams", Innovations in Systems and Software Engineering, **(2013)**, pp. 1-16.

[37] S. K. Swain and D. P. Mohapatra, "Test case generation from Behavioral UML Models", International Journal of Computer Applications, vol. 6, no. 8, **(2010)**, pp. 5-11.

[38] S. K. Swain, D. P. Mohapatra and R. Mall, "Test Case Generation Based on State and Activity Models", Journal of Object Technology, vol. 9, no. 5, **(2010)**, pp. 1-27.

[39] E. J. Rapos and J. Dingel, "Incremental Test Case Generation for UML-RT Models Using Symbolic Execution", In Software Testing, Verification and Validation (ICST), Fifth International Conference, IEEE, **(2012)**, pp. 962-963.

[40] X. Fan, J. Shu, L. Liu and Q. J. Liang, "Test case generation from UML sub activity and activity diagram", In Electronic Commerce and Security, ISECS'09 Second International Symposium, IEEE, vol. 2, **(2009)**, pp. 244-248.

[41] M. Prasanna and K. R. Chandran, "Automatic Test Case Generation for UML Object diagrams using Genetic Algorithm", Int. J. Advance. Soft Comput. Appl., vol. 1, no. 1, **(2009)**, pp. 19-32.

[42] P. Samuel, R. Mall and P. Kanth, "Automatic test case generation from UML communication diagrams", Information and software technology, vol. 49, no. 2, **(2007)**, pp.158-171.

[43] A. K. Jena, S. K. Swain and D. P. Mohapatra, "A novel approach for test case generation from UML activity diagram", In Issues and Challenges in Intelligent Computing Techniques (ICICT), International Conference, IEEE, **(2014)**, pp. 621-629.

[44] M. Khandai, A. A. Acharya and D. P. Mohapatra, "A novel approach of test case generation for concurrent systems using UML Sequence Diagram", In Electronics Computer Technology (ICECT), 3rd International Conference, IEEE, vol. 1, **(2011)**, pp. 157-161.

[45] P. N. Boghdady, N. L. Badr, M. Hashem and M. F. Tolba, "A Proposed Test Case Generation Technique Based on Activity Diagrams", International Journal of Engineering & Technology IJET-IJENS, vol. 11, no. 03, **(2011)**.

[46] S. S. Priya and P. S. K. Malarchelvi, "Test Path Generation Using UML Sequence Diagram", International Journal, vol. 3, no. 4, **(2013)**.

[47] D. Kundu, D. Samanta and R. Mall, "Automatic code generation from unified modelling language sequence diagrams", Software, IET, vol. 7, no. 1, **(2013)**, 12-28.

[48] W. Linzhang, Y. Jiesong, Y. Xiaofeng, H. Jun, L. Xuandong and Z. Guoliang, "Generating test cases from UML activity diagram based on gray-box method", In Software Engineering Conference, 11th Asia-Pacific, **(2004)**, pp. 284-291.

BIOGRAPHIES

Jagtap Shubhangi N.
pursuing the Bachlor Degree in Computer Science Engineering from H.S.B.P.V.T.COE,Kashti under
SPPU.
shubhangijagtap1992@gmail.com

Gawade Vishwas S.
pursuing the Bachlor Degree in Computer Science Engineering from H.S.B.P.V.T.COE,Kashti under
SPPU.
vishwasgawade96@gmail.com

Pawar Rahul R.
pursuing the Bachlor Degree in Computer Science Engineering from H.S.B.P.V.T.COE,Kashti under
SPPU.
pawarrahul00010@gmail.com

Shendge Savita G.
pursuing the Bachlor Degree in Computer Science Engineering from H.S.B.P.V.T.COE,Kashti under
SPPU.
shendgesavita801@gmail.com

Prof. Avhad Pravin S.
he has done his M. Tech(CSE) from S.V.P.M. college of engineering, JNTU, Hydrabad.
pravin123.avhad@gmail.com