Test Case Generation using Unified Modeling Language

Syed Asad Ali Shah
University Institute of Information Technology
PMAS-Arid Agriculture University
Rawalpindi, Pakistan
asadalishah_islamabad@yahoo.com

Syed Shafique Ali Bukhari
University Institute of Information Technology
PMAS-Arid Agriculture University
Rawalpindi, Pakistan
ali.naqvi1989@gmail.com

Mamoona Humayun
College of Computer and Information Science (CCIS)
Jouf University
Al-Jouf, Saudi Arabia
mahumayun@ju.edu.sa

NZ Jhanjhi
School of Computing & Information Technology (SoCIT)
Taylor's University
Selangor, Malaysia
noorzaman.jhanjhi@taylors.edu.my

Syed Fakhar Abbas
University Institute of Information Technology
PMAS-Arid Agriculture University
Rawalpindi, Pakistan
fakhar.dps@gmail.com

***Abstract*—Software testing is the major phase of the software development life cycle as it ensures that the software performs according to the requirements. In order to perform testing, a lot of techniques are there that can test the software after the completion of the coding phase. Model-based testing has the capability to test the software before the coding phase. It helps in saving time, cost and budget overrun as it is conducted in the initial stages. In design models, UML diagrams are most widely used in academia and industries. UML class (structural) and sequence (behavioral) diagrams are the most common diagrams being used extensively by designers as they can cover the structural and behavioral aspects of the system respectively. A survey has been conducted in this paper to evaluate our proposed framework in which we have taken both diagrams to generate test cases and it can show the test case generation process in a sequence of steps. Some major issues spotted in test case generation process include usage of intermediate form, coverage criteria, storage of results and provision of the tool. Our study aimed to address all the above issues in the proposed framework. Expert's opinion has been taken and results have been shown in a graphical and a tabular way.**

Keywords— unified modeling language, object constraint language, model-based testing, software development life cycle, object oriented programming

I. Introduction

Testing of software is said to be an important phase in the development of any software or system life cycle. as it guarantees the excellence and consistency of any software application [1]. According to research, 50% of the whole time in the development of any software is used by the testing phase [2]. The testing activities contain scheming of generating a test case that is a flow of data, executing the application with the test cases and extracting the outputs created by the execution. The test case development is identified to be the hardest phases in software testing [3]. This is why; automated test case generation is one of the major contests in the field of software testing as the manual testing process technique is error-prone and arduous [4].

The main purpose of software testing process is to verify and validate the application/system. Testing can be performed in two ways i.e. auto or manual. In the manual approach, software testing is done by manually executing the code. After performing that, the obtained output is compared with specified behavior to check and record the observations [5]. It has been observed that the manual process of testing does not deliver appropriate result, leading the system towards inefficiency and ineffectiveness. Therefore the best approach of testing the software is automatic as it gives the useful and effective coverage criteria of software testing. Here, we are enlisting some testing benefits [6]: (i) it ensures that the application works according to the 'requirements document', (ii) it provides an error free software, (iii) its goal is to attain customer happiness by fulfilling their needs, (iv) to maintain confidence in the application, (v) to assess the properties of the software/application, (vi) to show the differences between authentication and fault, (vii) it ensures the quality of the product.

Model-Based Testing is achieving attraction in academe and industrial sector as it provides significant benefits[7]. Growing market competition is interested to use new approaches for decreasing the development time and rate of the complex software [8].

UML is attracting world of software because of an industrial demanding standard for modeling object-oriented applications in software testing. There are three major reasons to test design models in OOP (1) Old techniques for software testing use the only static interpretation of code which is insufficient to test dynamic aspects of the object-oriented system. (2) Using programming for testing object-oriented software is not an easy job. (3) Generation of test cases using a model-based approach is scheduled at early stage in the SDLC that permits to carry out coding and testing simultaneously [9].

Class diagram is considered as a useful and important diagram among other UML diagrams. It is used for different purposes at different times in the software development circle. Basically, a class diagram consists of the three major portions i.e. its name, its attribute and names of functions present in that class [10, 11]. Classes present the associations and different types of relationships with other classes[12]. Class testing is normally based on functional testing technique. In any class diagram, classes are indicated with rectangular chunks in three joined parts: (1) Uppermost part will show the name of that particular class written in bold and center aligned with its first letter in uppercase, (2) the centered rectangle keeps the attributes of that class and are aligned left with very first letter in lowercase, and (3) third and bottom most rectangle holds the name of methods of the class [13].

978-1-5386-8125-1/19/$31.00 ©2019 IEEE

This paper presents the survey of optimized test case generation using UML. After having a look on existing studies, we came to know that no specific broad survey has been conducted yet. 'Survey' methodology has been used for the studies and a broad level analysis with results of the survey has been shown.

II. Literature Review

The literature study proves that there are number of integrated methods presented by various researchers for automated test case generation using UML class diagram.

They [14, 15] have presented a novel approach for generating structural test cases using UML Class and parallel State diagrams. They have used techniques like Basic path, DD path and DU path testing approaches to generate test cases. Their approach was not compatible in the case of big and complicated class diagrams.

They [17] carried out an evaluation on different techniques for test case generation. After reviewing several approaches, they categorized them into three different groups (i.e.) generating test cases from GA (genetic algorithm), through random testing technique and lastly, using model-based testing for test case generation by examining the dynamic behavior of objects. Lastly, their study discussed that flaws in the design model can be spotted during the self-analysis of the model. Therefore, the faults can be eradicated as quickly as possible that results in lessening the cost of defect removal.

Authors [18, 19] performed a review for test case generation techniques on the basis of hybrid approach by considering the dynamic behavior of the UML diagram with the help of evolutionary algorithm. They suggested a technique to use a sequence diagram along with multi-objective GA for generation of test cases.

It [20] conducted a survey and identified different techniques used for developing test cases from use case-based approaches and UML Diagrams. In analysis, they discussed the pros and cons of both approaches but none of the techniques describes the complete process with their stated algorithm.

Authors [21] did a broad level survey to generate test cases using UML diagrams. They categorized practices on the basis of a number of events performed, a tool used and coverage criteria for test case generation.

It [22] presented a review on the basis of diverse test case generation techniques for minimization of test cases, selection, arrangement, and assessment techniques. Their main goal was to extract techniques which help the test engineers to schedule and rank the test cases to decrease the entire effort, time and cost.

Authors [23] presented a study of some most protuberant practices of automated test data generation comprising of figurative execution, combinatorial, model-based, and adaptive random and search-based testing. Their core objective was to give an initial, up-to-date and short overview of research in automatic test case generation while guaranteeing completeness and validity.

They [24, 25] carried out a literature survey on automated test case generation with the use of UML diagram. They created their literature survey based on two main categories naming specification-based test case generation and model-based test case generation.

It has been seen that approaches that used only class diagram need some kind of intermediate forms for test adequacy. Intermediate form makes test case automation difficult. Large number of steps during generating test cases also affects the automation level as a minimal step raises the level of automation. After reviewing the present methodologies, some desired parameters were set in our paper which needs to be taken under consideration during the development of a tool for test case generation.

- Appropriate ways need to be used to make automation easier.
- Precise readings of diagrams must be present.
- Less complicated or no intermediate form to be used.
- Minimum steps should be taken for test case generation.

In order to understand the previous work done by researchers, we have made a table to show the data in a more formalized way. Below is the table containing very understandable data extracted from different papers of different techniques as shown in Table I. The table has different columns including references, UML model used, testing level, intermediate form, provision of tools, coverage, and a number of activities performed in order to generate test cases.

III. Proposed Approach

We have presented our proposed method in this part of the paper and the steps are given as: (1) Create sequence and class diagram, (2) Generating code of XML, (3) Parsing the code of XML, (4) Check diagram correctness, (5) Formalize data into data tables, (6) Generate test cases and (7) Save in the text file.

A. Steps of Proposed Framework

This framework shows in below Fig. 1, the complete process of our proposed methodology with the sequence of steps.

1) Modeling Sequence Diagram and Class Diagram: The first step is modeling a sequence diagram in the Visual Paradigm tool in a direction to initialize the generating test cases process. Sequence diagram of any system is taken and modeled according to the specifications. This diagram helps in showing the behavioral aspects of the system with the passing of messages between the objects.

Second step is to create the class diagram rendering to the links and classes of the system. This diagram shows the complete structure and static relationships among the classes of the system. This diagram helps in identifying the whole system structure in a graphical way.

TABLE I. Analysis Of Literature Techniques

Ref.	**UML Model used** [19] [21]	**Testing level** [19] [20] [24]	**Intermediate form** [25] [34]	**Tool** [20 [24]	**Coverage**	**No. of activities performed** [20] [21]
[11]	Class diagram, interaction	✓	×	×	✓	3

	diagram					
[12]	Class diagram	✓	×	×	✓	4
[13]	Class Diagram, State Diagram	✓	✓	×	✓	5
[14]	Class Diagram, OCL and Sequence Diagram	✓	×	×	✓	7
[15]	Class Diagram, Sequence Diagram	✓	×	✓	✓	5
[24]	Class diagram, state machine	✓	✓	×	✓	6
[26]	Class, Sequence, Use case and State chart	✓	✓	✓	×	5
[27]	Sequence, Use case and Class	✓	✓	×	×	7
[28]	Sequence, Class and Use case	✓	✓	×	✓	8
[29]	Class, Activity and Sequence	✓	×	×	✓	6
[30]	State, Class Diagrams and OCL expression	✓	✓	×	✓	5
[31]	Object, state and Class	✓	✓	×	✓	5
[31]	Activity, Class, Sequence, State chart, and Use case diagrams	×	✓	×	✓	6
[32]	Sequence, Use Case and Class Diagram	✓	✓	×	✓	5
[33, 34]	Class diagram	✓	×	×	✓	5
[35, 36]	Class and sequence	✓	×	×	×	4
[37]	Sequence and class	✓	✓	×	✓	5

2) XML Exporting Feature: One of the reasons of using Visual Paradigm is that it has built-in the diagram to XML exporting feature. With the help of this built-in feature, both diagrams are exported into XML in order to store diagram in a sequence of classes and in an extractable way. Extensible Markup Language is a markup language that describes some specific rules for encoding the documents in a particular format that is both human and machine-understandable. It is a written data presentation having strong support through Unicode for multiple human languages. In our case, complete information of both sequence and class diagrams have been exported and stored into different XML files.

3) XML Information: The key role of exporting the diagrams is to store important attributes that are further used in test case generation. For the sequence diagram and class diagram document XML exported file we get the following modules.

- Sequence Name
- Class Name
- Methods
- Attributes

C# is a multi-platform development language covering strong inputting; authoritative, declarative, practical, general, object-oriented programming standards C# is also designed for the Common Language Substructure. We use C# language to extract desired information and then stored it into C# data tables. The first table will keep the class names and their objects. The second table stores the info of classes with their methods names. The third table consists of methods names and their particular parameters.

4) Matching Properties: One of the functionalities of our framework is to check that both of the diagrams have common class names, methods, and their parameters. To accomplish this task, a code written in C# is used to match the properties of the diagrams. We stored information of both diagrams in separate data tables in different steps. The classes of sequence diagram's data table would be matched with the classes of the class diagram's table. In this way, we can identify that classes in both diagrams are same. Similarly, for methods and their parameters, we use the same code to check the similarity of both. This approach helps in identifying the correctness of the system.

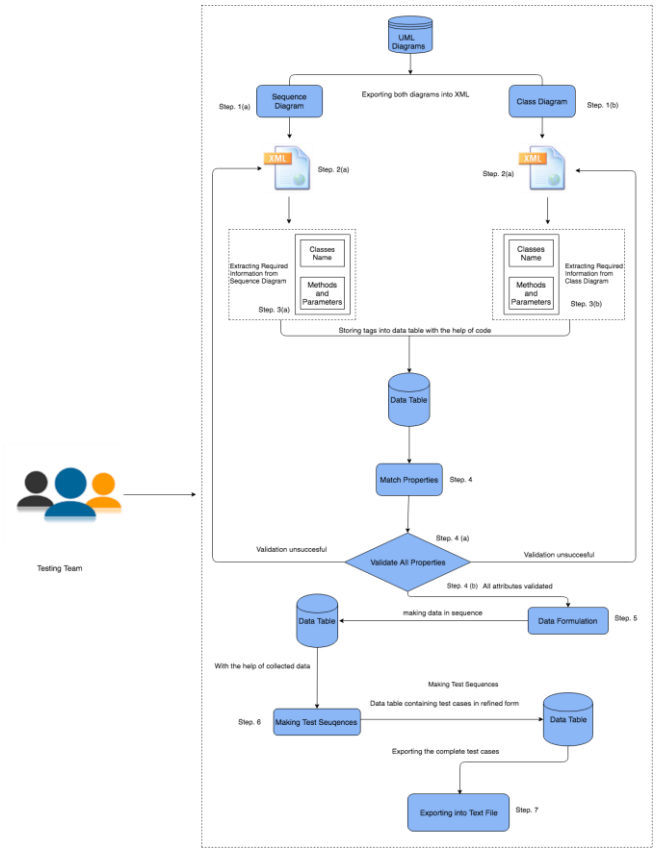

Fig. 1." Proposed framework for optimized test case generation

a) Not Valid: If the diagrams are not same, the system will show the error message that diagrams properties are not same and redirect the user to the XML files. This approach helps in identifying an error in the design phase.

b) Valid: If the diagrams properties match perfectly then it means that both of the diagrams are correct and we can proceed to the next step towards test case generation.

5) Data Formalization: The next step is formalizing data into a test case generation sequence that helps in making the sequence of data into the test case shape. To accomplish this task, the C# code is used. The code will formalize the data and store it into another data table with complete information.

6) Making Test Sequences: This step makes a complete test sequence with the whole information of the test case and stores it into a final data table.

7) Exporting into Text File: The final step is exporting complete test case information into the text document to store it permanently. We achieve this task by again using file writer property of C#. This step helps in finishing our test case generation process.

IV. Results And Discussion

For the evaluation of the proposed framework, we conducted industrial expert reviews. A questionnaire about testing activity, UML diagram correctness, result satisfaction, and the proposed framework was designed as given below. After designing the questionnaire, we sent it to the experts for evaluation. We have received 24 responses from the industry experts and then compared our proposed framework with them. The statistical results in the form of table and graph according to each question are given below.

Q1. In your organization, adequate time and effort are given to testing activity.

Q2. In your organization, MBT is often used.

Q3. The given framework is easy to understand and implement.

Q4. The framework is not using an intermediate form.

Q.5 The framework checks the correctness of UML diagrams.

Q6. The framework generates test cases in an easy and efficient way.

Q7. The presentation of the result is satisfactory.

Q8. The framework provides a proper storage mechanism for reusability of knowledge.

Q9. Completed data of test cases is extracted through XML

Q10. Data table contains enough information on test cases.

Q11. The framework is giving maximum coverage.

A. Description of Survey

The graph shows the results based on industrial expert reviews for the evaluation of the proposed framework as shown in Fig. 2. These results were calculated according to each question. The table contains statistical results of responses according to each option as given below in Table II and the discussion of results according to each question is given below.

The 1st question deals with the amount of time given in order to perform the testing. The results clearly show that 33% of experts were strongly agreed and 54% were just agreed upon it. It has been observed that most of the organizations spare adequate time to perform testing.

The 2nd question deals with the usage of model-based testing in the organizations. Results show that 21% of the experts are strongly agreed while 63% were just agreed. It has been noticed that the testing is being performed using MBT in different organizations.

The 3rd question dealt with an understanding of the proposed framework and to implement it for testing purpose. Results show that 21% of the experts were strongly agreed while 50% were agreed. This proves that the proposed framework has been understood by most of the experts and they can go ahead with the implementation of the proposed framework in their organization.

The intention of having the 4th question was to ensure that we are not using any intermediate form in our approach. The result shows that very positive response from the experts such that 17% of the experts were strongly agreed and 58% were agreed. We can say that due to no usage of graph or tree in our approach, we did not use any intermediate form to have our approach much automated.

In our proposed approach, we have also checked the correctness of both diagrams by comparing some of their properties. To have different opinions, we have asked about the correctness in the 5th question. The result presents that 21% strongly agreed and 67% of the experts agreed upon the correctness of the diagrams.

The 6th question was about knowing that whole test case generation process difficulty level in our framework. The vision was to make it easy enough to be acceptable by everyone. The result shows that 21% of experts were strongly agreed and 58% agreed. From these appreciable results, we came to know that our framework clearly shows the test case generation process.

The intention of asking the 7th question was about the satisfaction of our results. Experts were 25% strongly agreed and 42% were agreed. From the experts, we were pleased to see that we have presented our results in a satisfactory way.

In the literature, we have seen that only a few researchers have given some permanent storage mechanism for the generated test cases. This 8th question was to ask about the storage mechanism in our framework.

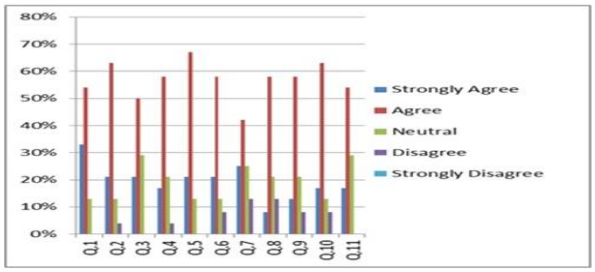

Fig. 2. Results of industrial expets review

The result presents that 8% of the experts were strongly agreed and 58% were agreed. A simple text file is used to store the generated test cases to make it easily accessible to everyone.

9th question was about the information extracted by the XML. The result shows that we got a positive response from the experts' 13% strongly agreed review and 58% were agreed. We extracted complete required information for matching and for the generation of the test cases.

10th question was related to the information stored in the data tables on temporary bases. Results clearly show that 17% of experts were strongly agreed and 63% were agreed. Data tables help us in keeping the data through which we have matched both diagrams and then test case were generated based on the data in them.

11th question was about the coverage level of our approach in the framework. The results show that 17% of experts were strongly agreed and 54% were agreed upon this question. To have maximum coverage in our approach, we covered each and every parameter of every method in each class. After having a look at the results, we can say that this approach can give good coverage within a scope.

UML is largely used to define investigation and design stipulations by academe and industry, therefore UML models have been considered as bases for test case generation purpose [35, 37]. Among UML diagrams, the class diagram is a structural diagram that provides the static and structural configurations of the system [40]. Furthermore, the sequence diagram is a type of interaction diagram that shows the intercommunication among objects. So, both diagrams can be interrelated with each other as a lifeline in a sequence diagrams show the object of some class. As described by that the sequence from the interaction diagram is used to create a sequence of the messages in the test case, we picked complete classes from the class diagram and their respective methods and parameters from the particular sequence diagram [11]. In our approach, the sequence diagram is used only to find the methods sequence communicating between different objects of some classes. After collecting method sequences from the class diagram's XML, test cases were generated from the class diagram's XML. The complete process has described along with results. Step by step implementation process has been provided to avoid any confusion. The results show that a complete path has been covered without skipping any object. The code has started the testing from the first method of the first class until the last method of the last class. In addition, the correctness of the model has been tested by comparing both diagrams. Both conditions are checked i.e. valid and invalid showing that there is a possibility of input/output to be valid or invalid but cannot same at the same time. If we compare our approach with the previous ones, we can find that this approach has significant features. The identified gaps in literature have been tried to minimize in our approach.

V. Comparison With Other Approaches

In our proposed approach, we did not use any intermediate form like graph as using an intermediate form restricts the level of automation. Our approach directly extracts the data from XML files and stores them into data tables to generate test cases. This helps in decreasing number of activities/ steps to generate test cases.

TABLE II. Comparison With Other Techniques

Ref.	UML Model used [20] [22]	Diagrams Correctness	Intermediate form [26] [35]	Tool [21] [25]	Coverage	Test cases storage
[11]	Class diagram, interaction diagram	×	✓	×	Partially	×
[15, 38]	Class Diagram, Sequence Diagram	×	✓	✓	Partially	×
[31, 39]	Activity, Class, Sequence, State chart, and Use case diagrams	×	✓	×	Partially	×
[31, 40]	Sequence, Use Case and Class Diagram	×	✓	×	Fully	×
[33, 41]	Class diagram, sequence diagram	×	×	×	Partially	×
[34, 42]	Class diagram, Sequence diagram	×	✓	×	Fully	×
Our Model	Class & Sequence	✓	×	✓	Fully	✓

Not only the framework but also the tool has been developed that can generate the test cases efficiently according to the input. This helps researchers in finding more ways to generate test cases in a better way. No research has made a mechanism to make sure that diagrams are correct and complete as shown in comparison Table III. We are having a check in our approach that can confirm the presence of same attributes in both diagrams. Having the same classes, methods and attributes in both diagrams assure the correctness that leads to success in the further steps. The way of presenting results in our approach is quietly understandable than other approaches. Results clearly express the theme and flow of test cases. This presentation and systematic flow of our approach make the whole process easy to grasp. It also analyzed that most of the researchers adopted XMI for the process of generating test cases. We used XML to make sure that complete results can be achieved by using XML as well. Also, in the end, the generated test cases are permanently stored in the text document that no other researcher has been offered.

VI. Conclusion

Unified Modeling Language (UML) has significant features in creating the system and has attained high importance. Testing through the UML models is known as model-based testing. Testing the system before the development resists the causalities in the later stages. Automated testing techniques have priority over the manual testing due to its promptness, ease and time-saving features. This paper presented a survey based on our proposed solution to generating a test case from the UML diagrams (class and sequence). Comparison with the previous techniques has been done in this study. No intermediate form has been used to encourage the concept of automation. The class diagram was mainly considered to generating test cases from the sequence diagram by extracting the method sequence from it. The correctness of both diagrams has been tested by comparing their XML files. Our approach is tool supported and shows the result in an understandable way. The complete path has been covered without missing any

attribute tested to ensure the full coverage of the approach. In addition, the test cases were generated and permanently saved in the text files that can be used in future as well.

REFERENCES

[1]" R. Mall, "Fundamentals of software engineering", International Journal of Computer Science and Informatics, vol. 3, pp. 14-21, 2013.

[2]" R. K. Swain, V. Panthi, and P. K. Behera, "Generation of test cases using activity diagram", International Journal of Computer Science and Informatics, vol. 3, pp. 1-10, 2013.

[3]" P. Mani and M. Prasanna, "Test case generation for embedded system software using UML interaction diagram", Journal of Engineering Science and Technology, vol. 12, pp. 860-874, 2017.

[4]" W. Linzhang et al., "Generating test cases from UML activity diagram based on Gray-Box method", Proceedings of the 11th Asia-Pacific Software Engineering Conference, IEEE Computer Society, pp. 284-291, 2004.

[5]" A. V. K. Shanthi and G. M. Kumar, "Automated test cases generation from UML sequence diagram", International Conference on Software and Computer Applications, vol. 41, pp. 83-89, 2012.

[6]" D. M. Rafi, K. R. K. Moses, K. Peterson, and M. V. Mantyla, "Benefits and limitations of automated software testing: systematic literature review and practitioner survey", Proceedings of the 17th International Workshop on Automation of Software Test, pp. 36-42, 2012.

[7]" S. Wang, S. Ali, A. Gotlieb, and M. Liaaen, "Automated product line test case selection: industrial case study and controlled experiment", Software and System Modeling, vol. 16, pp. 417-441, 2017.

[8]" O. Semeráth et al., "Formal validation of domain-specific languages with derived features and well-formedness constraints", Software & Systems Modeling, vol. 16, pp. 357-392, 2017.

[9]" S. G. Shukla and G. S. Chandel, "A systematic approach to generate test cases using UML activity diagrams", IRACST- International Journal of Research in Management and Technology, vol. 2, pp. 469-475, 2012.

[10]" M. Shirole and R. Kumar, "UML behavioral model-based test case generation: a survey", SIGSOFT Software Engineering Notes, ACM, vol. 38, pp. 1-13, 2013.

[11]" S. A. A. Shah et al., "A review of class based test case generation techniques", Journal of Software, vol. 11, pp. 464-480, 2016.

[12]" Y. Wang and M. Zheng, "Test case generation from UML model", 45th Annual Midwest Instruction and Computing Symposium, pp. 1-8, 2012.

[13]" M. Prasanna, K. R. Chandran, and D. B. Suberi, "Automatic test case generation for UML class diagram using data flow approach", Academia Education, pp. 1-7, 2011.

[14]" R. Anbunathan and A. Basu, "Dataflow test case generation from UML class diagrams", IEEE International Conference Computational Intelligence and Computing Research (ICCIC), pp. 1-9, 2013.

[15]" A. Alhroob, K. Dahal, and A. Hossain, "Automatic test cases generation from software specifications modules", Journal of e-Informatica Software Engineering, vol. 4, pp. 109-121, 2010.

[16]" K. K. Karambir, "Survey of software test case generation techniques", International Journal of Advanced Research in Computer Science and Software Engineering; vol. 3, pp. 937-942, 2013.

[17]" K. Kaur and V. Chopra, "Review of automatic test case generation from UML diagram using evolutionary algorithm", International Journal of Inventive Engineering and Sciences, vol. 2, pp.17-20, 2014.

[18]" S. Tahiliani and P. Pandit, "A survey of UML-based approach to testing", International Journal of Computational Engineering Research, vol. 2, pp.1396-1401, 2012.

[19]" A. Kaur and V. Vig, "Systematic review of automatic test case generation by UML diagrams", International Journal of Engineering Research & Technology, vol. 1, pp.1-17, 2012.

[20]" D. S. Maylawati, W. Darmalaksana, and M. A. Ramdhani, "Systematic design of expert system using unified modelling language", IOP Conf. Series: Materials Science and Engineering, vol. 288, pp. 1-7, 2018.

[21]" I. Hooda and R. Chhillar, "A review: study of test case generation techniques", International Journal of Computer Applications, vol. 107, pp. 33-37, 2014.

[22]" S. Anand et al., "An orchestrated survey of methodologies for an automated software test case generation", Journal of Systems and Software, vol. 86, pp. 1978–2001, 2013.

[23]" S. E. Ingle and M. R. Mahamune, "An uml based software automatic test case generation: survey", International Research Journal of Engineering and Technology, vol. 2, pp. 971-973, 2015.

[24]" A. C. D. Neto, R. Subramanyan, M. Vieira, and G. H. Travassos, "A survey on model-based testing approaches: a systematic review", Proceedings of the 1st ACM International Workshop on Empirical Assessment of Software Engineering Languages and Technologies, pp. 31-36, 2007.

[25]" N. Pahwa and K. Solanki, "UML based test case generation methods: a review", International Journal of Computer Applications vol. 95, pp. 1-6, 2014.

[26]" A. Verma and M. Dutta, "Automated test case generation using uml diagrams based on behavior", International Journal of Innovations in Engineering and Technology, vol, 4, pp. 31-39, 2014.

[27]" V. Sawant and K. Shah, "Automatic generation of test cases from uml models", Proceedings of International Conference on Technology Systems and Management, vol. 2, pp. 7-10, 2011.

[28]" A. Sharma and M. Singh, "Generation of automated test cases using uml modeling", International Journal of Engineering Research & Technology, vol. 2, pp. 1833-1835, 2013.

[29]" B. N. Biswal, P. Nanda, and D. P. Mohapatra, "A novel approach for scenario-based test case generation", International Conference on Information Technology, IEEE Computer Society, pp.244-247, 2008.

[30]" H. Khalil and Y. Labiche, "State-based tests suites automatic generation tool (Stage-1)", IEEE 41st Annual Computer Software and Applications Conference, pp. 357-363, 2017.

[31]" K. W. Shin and D. J. Lim, "Model-based automatic test case generation for automotive embedded software testing", International Journal of Automotive Technology, vol. 19, pp. 107-119, 2018.

[32]" O. Oluwagbemi and H. Asmuni, "An approach for automatic generation of test cases from UML diagrams", International Journal of Software Engineering and Its Applications, vol. 9, pp. 87-106, 2015.

[33]" M. Sarma, D. Kundu, and R. Mall, "Automatic test case generation from uml sequence diagram", International Conference on Advanced Computing and Communications, IEEE, pp. 60-67, 2007.

[34]" A. V. K. Shanthi, "Automated test cases generation for object-oriented software", Journal of Computer Science and Engineering, vol. 2, pp. 543-546, 2011.

[35]" S. Asthana, S. Tripathi, and S. K. Singh, "A novel approach to generate test cases using class and sequence diagrams", In: Contemporary Computing. Springer-Verlag, vol. 95, pp. 155-167, 2010.

[36]" T. T. Dinh-Trong, S. Ghosh, and R. B. France, "A systematic approach to generate inputs to test uml design models", 17th International Symposium on Software Reliability Engineering (ISSRE), IEEE Computer Society, pp-95-104, 2006.

[37]" M. Shirole and R. Kumar, "UML behavioral model-based test case generation: a survey", SIGSOFT Software Engineering Notes, ACM, vol. 38, pp. 1-13, 2013.

[38]" S. S. Priya and P. D. Sheba, "Test case generation from uml models- a survey", International Journal of Emerging Technology and Advanced Engineering, vol. 3, pp. 449-459, 2013.

[39]" A. Kaur and V. Vig, "Automatic test case generation through collaboration diagram: a case study", International Journal of System Assurance Engineering and Management, vol. 9, pp.362-376, 2018.

[40]" A. Sullivan, K. Wang, R. N. Zaeem, and S. Khurshid, "Automated test generation and mutation testing for alloy", 10th IEEE International Conference on Software Testing, Verification and Validation, pp. 264-275, 2018.

[41]" M. Shirole and R. Kumar, "UML behavioral model-based test case generation: a survey", SIGSOFT Software Engineering Notes, ACM, vol. 38, pp. 1-13, 2013.

[42]" S. S. Priya and P. D. Sheba, "Test case generation from uml models- a survey", International Journal of Emerging Technology and Advanced Engineering, vol. 3, pp. 449-459, 2013.