Ilkogretim Online - Elementary Education Online, 2021; Vol 20 (Issue 3): pp. 3666-3696
http://ilkogretim-online.org
doi: 10.17051/ilkonline.2021.03.384

Generating Test Cases Automatically From Uml Diagrams:A Systematic Literature Review

Preeti Malik[1*], Varsha Mittal[1], Rohini[2]

[1]Department of Computer Science and Engineering, Graphic Era University, Dehradun.

[2]School of Management, Graphic Era Hill University, Dehradun Uttarakhand

[*] Corresponding Author: preetishivach2009@gmail.com

Abstract. Software testing is a crucial and fundamental step in the creation of software that sets the standard for software quality. However, the testing process is consuming tasks that should be automated to save a significant number of resources. Computerizing experiments is becoming the main testing technique as we go toward automated testing. The main benefit of automated testing techniques is that they hasten the delivery of services for the products to the market with low risk of failure and increase the value of the product. Testing automation would be a crucial choice if the purpose of the organization were to reduce expenses and advance innovation. The objective of this review is to examine the existing research in generating test cases using UML diagrams and to enhance the comprehension of UML chart-based testing systems.

1 Introduction

Software is the most important mean which is prompting almost all electronics and industrial organizations [1, 2]. Testing in software is a superiority phase of evolution of software. Main aim of testingis not to productivity only but also support to enhance the quality of software product from small scale to large scale. In fact, we test the software until the product is valid and verifiable. As increasingthe software complexity, the requirement of test coverage needed for generated test case increases gradually [3]. Testing is an activity where the remaining error from all the previous phases must be detected. The main focus of a testers during testing of software is that they must know about minimizing of large number of test case into manageable test set, and be able to take the calculated risk about what are important to test and what are not. The main aim of our automated testing during theminimization of test cases is to produce cost and time efficient software. In Software Development Life Cycle (SDLC), test process is the most important phase to check the Software System validation. It is mainly completed by running test and inspection of these processes. The whole Test process complete in three parts:

1. Test case generation.
2. Execution of Test case.
3. Evaluation of Test case.

The main aim of testing should be conveying advice to change and modify software if necessary. The reason behind designing of test case is to rectify the different categories of error with minimal effort and time. Software reliability and quality are mainly depended on the collection of data while testing. The advantages of testing are:

i. Improved productivity of software developer.

ii. Time

reduction. iii.

Cost reduction.

iv. Controlling on error is fast.

Testing is a major challenge in software development in terms of generating efficient test cases. Itbecomes a challenge while testing of a software, when multiple executing participant appear simultaneously in a system, as system similar to that will give different outputs depending on the occurrenceof concurrent participant. Some of the software organizations are still performing manual software testing. As the development of the test script are done manually by the test engineer, that is why the execution of test case is automated in most of the test automated tool. In comparison to that, Model Based Testing (MBT) robotized the test design for generating the test cases reflexively from SystemUnder Test (SUT) model. Unified Modeling Language (UML) is a type of MBT. UML was announced by the Object Management Group in 1997. Object-Oriented prototypes are very fast used in industries and academics. UML is the most superior and controlling modeling language used in development ofsoftware. On the basis of two category level, UML diagram are divided into twelve diagrams [4]:

A) Structural Level (package, class, object, composite, and profile diagram)

B) Behavioral Level (Activity, State machine, Use Case, Sequence, Communication, Interaction,and Timing diagram).

Structural diagram signifies the inter-relation between different component of system on differenthierarchy of abstraction while the behavior of object including changes with the time in object and their interaction with each-other comes in Behavioral diagram. UML diagrams are used for generating the test cases from model. MBT is becoming famous in both academia as well as in industry. Because of the increasing in complexity, many critical functions are performed and dependability requirement such as safety, reliability, availability, and security of the system is very crucial for the user of the system. On the basis of requirement specification, the information is preserved by the modeland it forms a basis of final implementation.

Model determines logical paths, location of program boundaries, identify reachable problem.

This paper surveys existing literature in test case generation using UML diagram. This study discusses techniques proposed by various researchers and what are the limitations of these techniques. The paper that rest part is designed as follows: Next part is dedicated to related work in same area. Section 3 includes whole review process. Section 4 discusses results and threats to validity of reviewis included in section 5. At the end section 6 conclude the review.

2 Related Work

In this section the studies that discussed existing literature in the context of test case generation usingUML diagram are presented. Many researchers have discussed about the related work which are basedon UML based test case generation. Many of the traditional techniques are taken into consideration in addition to their methods on UML diagram-based test case generation. Karambir in [5] presented a report based on the technique called test case generation. After reviewing the numerous existing techniques, they labeled the existing techniques widely into three sections (i.e.) generation of testcases using genetic algorithm, by operation of haphazard testing technique and make use of model-based testing for generation of test cases by investigating the dynamic nature of objects. At last, thewhole research surveyed that fault presented in the model can be detected by model itself during thetesting process. So, the cost of defected model can be eliminated as possible and gives efficient result. Kaur in [6] presented review by evaluating the dynamic nature of UML Diagram using evolutionary algorithm for generation of test case techniques on the basis of their hybrid approach. At last, theyintroduced a new technique to generate test case by applying sequence diagram with multipurpose. Pahwa in [7] proposed number of techniques based upon UML diagram for the generation of testcase. To make suitable execution they targeted on the best use of UML techniques for generation of test cases.

Prasanna, M., et al. in [8] reported a survey based on automatically generation of test cases using UML diagram. According to them techniques can be categorized into mainly three sections: (1)specification-based techniques (2) model-based techniques and other (3) hybrid-based techniques approaches. Thereafter, many problems are found by using the traditional techniques such as numerous test statistics may be generated by the random test cases but they fail to analyze test case to fulfill the requirement. The path for which generating the test case is described by path-oriented approach but it might be infeasible and it is found that the test data generated might be unable to traverse the input data through the path. The test case can also be generated quickly by an intelligent approach butit is found complex, that is why model-based testing is given more preference as it create pliable and effective test automation in a practical way from the first day of development.

Ingle and Mahamune [9] presented a review based upon reflexive way of generating test cases using UML diagram. The incorporation of survey depends on two levels generating the test cases based on specification and test case generation relies on model. Dias Neto., et al. [10] presented a precise review depend upon MBT method. By the analysis of selection study, it is found that the initially described papers shrink. The overall inspection shows where MBT approaches are efficient to be applied, and also shows their features and drawback. Some of the parameters are identified and Compared on the basis of their representing models, test coverage criteria, provision tools, intermediate models, level of automation, and complexity.

Khanda, M., et al. [11] presented a precise survey for parallel as well as non-concurrent system and various existing methodologies are used for generating the test cases. They presented the summarized result of work done in that field and give the features/drawbacks. They resulted that researchers get information easily about generating the test cases using UML diagram and resulted work related to the identified research field. Arvinder Kaur and Vidhi Vig presented another survey given in [12]. The scope of the survey was based on UML diagrams testing. For model-based testing to generate auto test cases many researchers use different specific diagram like Use case, class diagram, collaborationdiagram, sequence diagram etc. In this survey, researcher merged the existing work, and answer is presented through its research question about the most popular techniques used in research field of model-based testing. The discussion of the testing type used in each study being next.

Noraida Ismail et al., (2007) [13] discusses some approaches of MBT issues arises and highlighted that the approaches are using is not fully automated. According to the systems requirement for generation of test cases, author has proposed a tool to handle these issues. The proposed process was classified into two steps: - (1) system requirements were converted into Use case diagram and (2) Using the existing approach Use case the test cases will be generated. As per the requirement of the system, the generated tests cases will be analyzed and validate because the testing phase is important in SDLC. During the process of testing, if the generation of test cases gets delayed then, there is chance of increasing error which becomes so complex to remove and debug at the end of testing. The researcher explained that as per system requirements, choosing the use case diagram is used forfurther use and modification.

Anjali Sharma and Maninder Singh [14] presented that the most valuable step in testing is generating the test cases. For generation of test cases, author proposed an algorithm that takes UML diagramas an input. The whole transaction is done on the basis of the proposed framework which will ap- plied in all existing UML diagram that includes use case, sequence diagram, class diagram, data flowdiagram etc. and translate these existing diagrams into a control flow graph. To get the output as testcases and coverage criteria, the input set is applied to the test cases generator and strike the algorithmwhich generate cost. The generated output test cases and coverage criteria are taken as input to reportthe generator tool and tester.

Salman in [15], presented a recently work that has done on literature review that is using the UMLstate diagram in Model-based testing. The outcomes are presented the systematized and complete details in tabular form. This tabular form contains the field of Author Name, tools used, Input model, method used, coverage criteria, and Intermediate Model. According to survey, it is found that most ofthe Author used Depth First Search algorithm in their techniques. The authors presented main reasonbehind the generation of test cases through UML is to validate the relationship between behavior, action state, and event. Correct Models are totally depending on the accuracy of generated test cases.The trustworthy output will not be generated, if used model are unreliable and incorrect. According toOstrand [16] they presented that input is taken as Model-based testing approach which work under the system under test for generation of system model that define the test coverage criteria. The generated test cases were produced while analysis of Model-based approach using Category Partition Method becomes completed. This survey resulted that work done on particular approach and also represent UML diagram are not covered.

Anand, S., et al. [17] presented a review based on few popular techniques for generation of test cases that contains figural execution, combinational, model-based, search-based and adaptive basedtesting. The main reason behind initialization of this survey was to present traditional techniques, updated techniques and summary of research used in automatically generation of test cases while make sure of scopes and validity.

Hooda [18] demonstrated an inspection about techniques used to generate test cases that help in reduction of test cases, prioritization, estimation and selection techniques. The aim of this survey is to help the tester to reduce the time, cost, and total effort to arrange and rank the generated test cases based on various existing techniques. Tahiliani [19] demonstrated a survey that based on generationof test cases using various existing techniques such as Use case approach and UML diagram. As the discussion shows that the complete process was not presented by any of the techniques along with declared algorithm but mentioned the advantages and limitation of both approaches.

3 Planning of Systematic Literature Review

For performing systematic literature review (SLR), the methodology adopted was recommended byKitchenham et al. [20, 21]. Some phases of the SLR have been taken by other approaches [22, 23, 24]. The phases followed for this review study has been depicted in figure 1. First and foremost, task is toformulate research questions and search string is identified. This search string is then used to search the literature resources like IEEE, ACM etc. In next phase, selection has been done on the studies we got from various databases and refine the studies by applying quality assessment criteria. And finally,data extraction and synthesis has been done.

3.1 Research Questions

The reason behind using this SLR is to study the existing well-known literature on test case generation using UML diagrams research to examine (1) what are the existing techniques used to generate test cases using UML diagrams (2) what are the limitations of existing techniques (3) what are the futuredirections and challenges in automated generation of test cases using UML diagrams? Following arethe research questions formed for this SLR:

RQ1: What are the existing techniques for automated test case generation
using UML?SQ1.1: Which of the UML diagrams is mostly used?
SQ1.2: What is the trends in using UML Diagrams versus Test Coverage Criteria?
RQ2: What are the limitations and future direction of existing techniques for automated test casegeneration using UML?

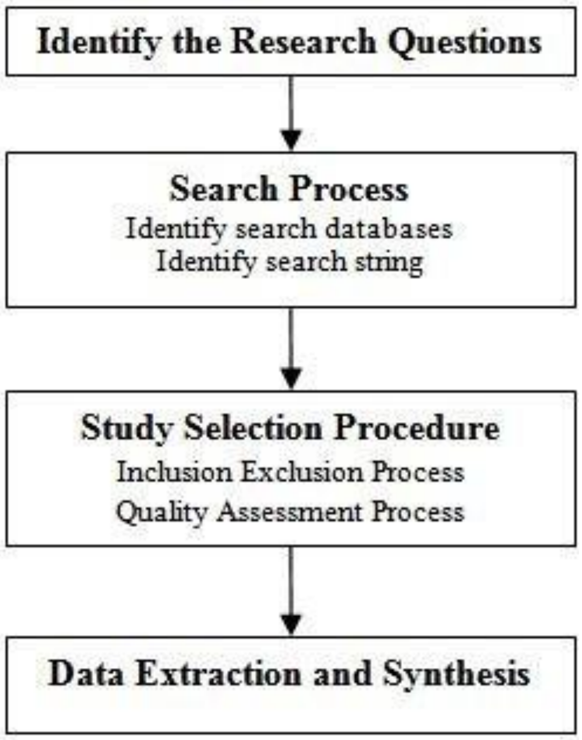
Identify the Research Questions
Search Process
Identify search databases
Identify search string
Study Selection Procedure
Inclusion Exclusion Process
Quality Assessment Process
Data Extraction and Synthesis

Fig. 1 Phases of SLR Process

3.2 Search Process

A brief description of search process used to extract the appropriate studies on test case generation using UML diagrams, research is presented in this section. The process includes identification of search databases, identification of search string and search process. We start by identifying libraries.Our selection includes:

- ACM Digital Library,
- IEEE Xplore,
- ScienceDirect,
- Springer.

Boolean AND and OR operators are used to connect keywords to formulate search string. However,the string had not been used as it is for all the databases, since some of the resources have some certain way to formulate the string. In such case, a similar string has been formed. In figure 2, the string hasbeen shown:

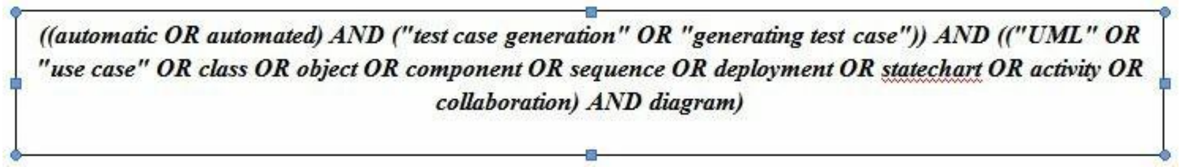
((automatic OR automated) AND ("test case generation" OR "generating test case")) AND (("UML" OR "use case" OR class OR object OR component OR sequence OR deployment OR statechart OR activity OR collaboration) AND diagram)

Fig. 2 Search String

3.3 Inclusion and Exclusion Criteria

Defining the criteria for including or excluding a particular study is one of the key activities of SLR.These criteria help the reviewers in identifying relevant studies. We also use these criteria to reducenumber of studies we got from various databases. If a study is classified in at least one exclusion criteria, it will be discarded and if a study is classified in one of the inclusions criteria, then it is included. Its criteria can be defined as follows:
Inclusion Criteria

- All studies put out in English;
- The primary study must propose a method for test case generation using UML diagrams;
- The study that answers at least one research question;
- The primary study must propose a tool/ framework development for test case generation;
- In case of two versions of same study the recent one is included.Exclusion Criteria
- Study that is not published in English language;
- Study that does not answers any of the research questions;
- In case of duplicate studies only the most updated one is included;
- The primary study is not related to test case generation using UML diagrams.

3.4 Study Selection Procedure

The selection of studies is done by last two authors. Removing duplicate is the foremost task of selection process. Then we have applied title and abstract scrutiny. We have assessed 1731 studies from a search undertaken in IEEE Xplore (36), ACM (11), Springer (849) and ScienceDirect (835).All those papers were discarded that didn't have any correlation with specific domain. In second phaseof selection process, it is necessary to review those papers again for ensuring that the papers that are selected are genuine or not for generating the test cases using UML diagram. Quality assessment is the next step that will be discussed in following sections. Finally, we got a sum of 58 primary studies(see figure 3).

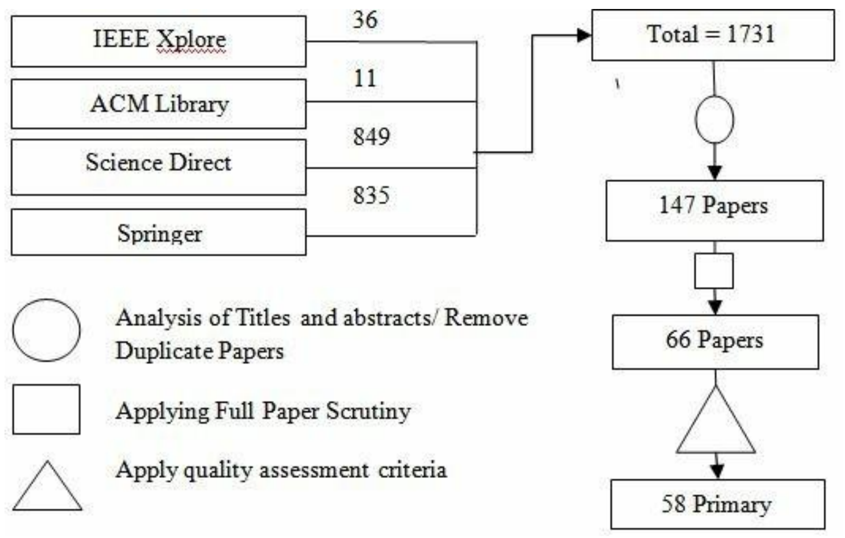
IEEE Xplore
ACM Library
Science Direct
Springer
36
11
849
835
Total = 1731
147 Papers
66 Papers
58 Primary
Analysis of Titles and abstracts/ Remove Duplicate Papers
Applying Full Paper Scrutiny
Apply quality assessment criteria

Fig. 3 Search Process

3.5 Quality Assessment Process

The quality assessment of 66 studies was done by forming six quality assessment criteria (see table 1). There are three possible answers to each question: yes, not clear or partly and no [25]. The qualityscore for the answers is 1, 0.5 and 0 respectively. Final quality score for each of the study is computedby adding up all the six scores of that particular study. The studies included are those studies havingquality score 3.5 or more on a scale of 0-6. As a result, 8 studies were found to be not relevant to ourSLR (see table 4). Finally total 58 studies are selected as primary studies (table 7). The quality scoreof 58 studies can be seen in table 6.

S. No.	Criteria
1	Is the problem clearly described?
2	Is there adequate discussion of related work?
3	Is the technique well described so that the author or others can validate it in later research?
4	Is this a significant increase of knowledge of these situations?
5	Is the technique compared with the existing one?
6	Do the study goals were achieved?

Table 1 Quality Assessment Questions [26, 20, 27]

3.6 Data Extraction

Next task is to extract relevant data from those 58 primary studies. For this purpose, a data extractionform is designed in table 2. The main purpose behind the designing of this form was to gather those information from the selected studies that is needed to answer the research questions. This design isdivided into four sections: study detail, study assessments, study description, and study findings. Firstsection holds general information of the studies like: title, author, venue of publication etc. Second section contains description of the study which includes research type, aim of the study. Third section is the most important one. It includes the detailed information about the diagram used, coverage criterion and intermediate model used for generating the test cases. And, the final study shows that the subjective evaluation of the studies that highlight their result, drawback, strength, scope for futurework and to which research questions (RQ1 or RQ2) these studies are belonging. Some of the studiesanswer more than one question.

Study Details	
Reviewer:	Name of reviewer
Paper ID:	Unique ID of primary study
authors:	Author's Details
Title:	Title of Primary Study
Publication Venue:	Journal, conference papers, bulletins, book chapters, workshops, white papers, sympo- Sium
Details of publication:	Year, Page numbers
Study Description	
Research Methodology:	Experiment, Observation, tool development, solution proposal, experience report
Aim of the study:	Domain, problem solution, conceptual framework, observation, model transformation
Detailed Assessment	
Applicability Context: Intermediate Model Used: Coverage Criterion:	Identification of the UML diagram on which method appliedWhich intermediate model is used? Coverage Criterion Used
Study Findings	
Findings and Conclusion:	Conclusion and finding of the study
Limitations:	Shortcomings of the study and area of future work
Mapping to the identified re- search question:	Under which research question the study falls?

Table 2 Data Extraction Form

4 Results and Discussion

4.1 Overview of Primary Studies

For this survey, we have found 58 primary studies. It contains 14 journal articles, 38 conference papers, 1 paper is from ACM software engineering notes, and 5 were book chapters. Primary studies found from various resources are presented in Figure 4. Figure 5 presents number of studies per yearand table 3 shows overall analysis of the studies that belongs to the research question.

4.2 Research Types

Primary studies have been divided into 6 categories; experiment, observations, problem solution, tool development, tool review, and technical report. Study [P5] is the only technical report and it is dedicated to tool development also. Figure 6 shows research type distribution of the primary studies.

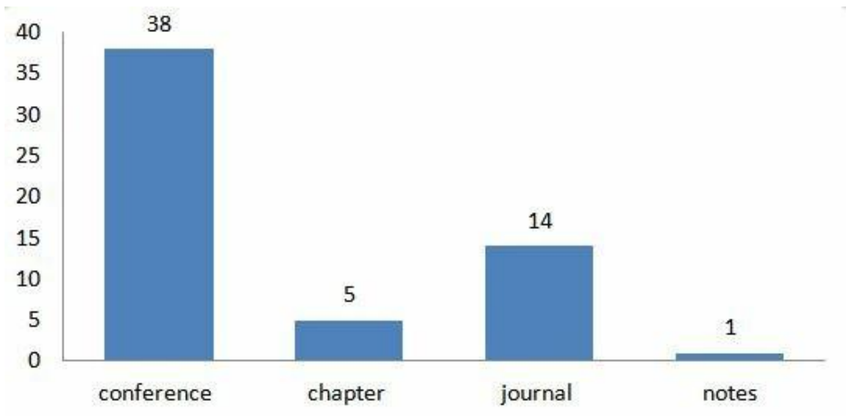

Fig. 4 Number of Studies in specific venues

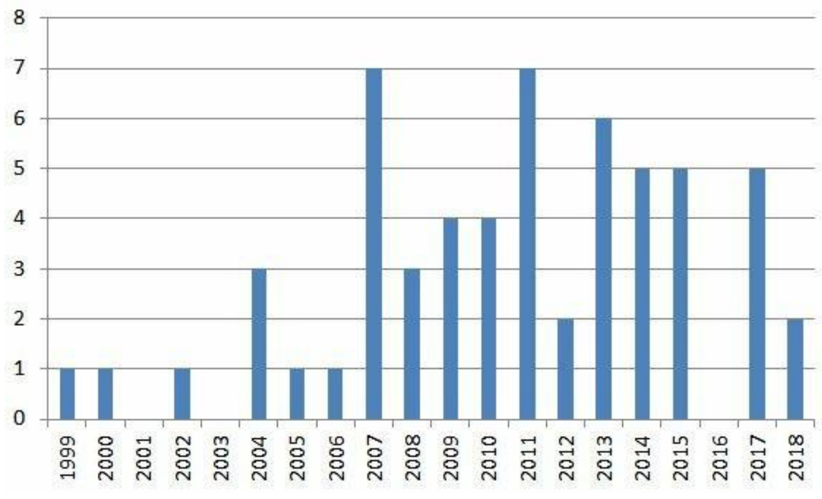

Fig. 5 Studies per year

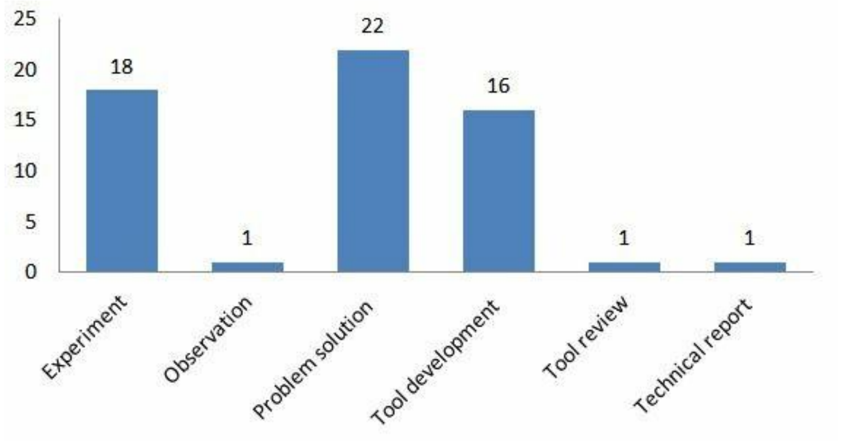

Fig. 6 Research Types

R Q	Primary Studies
1	P[1-58]
2	P[1],P[2],P[3],P[4],P[6],P[7], P[11],P[12],P[13],P[14],P[15],P[16]P[17],P[18],P[19],P[20],P[21],P[22] P[23],P[24],P[25],P[26],P[27], P[28]P[29],P[30],P[31],P[32],P[33],P[34],P[35],P[36],P[37],P[38],P[39]P[40] P[41],P[42],P[43],P[44],P[45], P[47],P[48],P[49],P[50],P[51],P[52]P[53],P[54],P[55],P[56],P[57],P[58]

Table 3 Overview of the Studies belongs to each research questions

4.3 RQ1. What are the existing test case generation techniques using UML diagrams?

To answer this question, analysis has been performed to strengthen the belongings. Table 5 depicts analysis of primary studies. The following parameters are used to compare the primary studies while analyzing generation of test cases using UML diagrams (1) Model used [19, 18]. (2) Use of Inter- mediate model or form [7, 28]. (3) Coverage criteria used. By examining the primary studies, we have following findings: (1) Appropriate coverage criteria need to be taken in an order to make test cases efficient. (2) Exact understanding of UML diagrams should be there. (3) Minimum intermediate form or no intermediate form should be used. (4) Use of no or minimum number of diagrams whilegenerating test cases.

Most used UML diagrams

Though UML diagrams have been appeared and discussed in the significant way that is mention in the traditional literature survey, as there no such protocols have been defined in the exiting literature that measures the quality and implementation of defined diagrams. For this, we analyze these 8 UMLdiagrams based on their research process mentioned in this paper. Figure 7 represents the bar graph for the UML diagram used in test case generation. Deployment and object diagram are not used for this purpose. Some of the studies used combination of two or more diagrams [P12, P13, P16, P21, P24, P28, P32, P33, P36, P41, P44, P52, P53]. Activity (39.65%), state chart (29.31%) and sequencediagrams (24.13%) are most used diagrams.

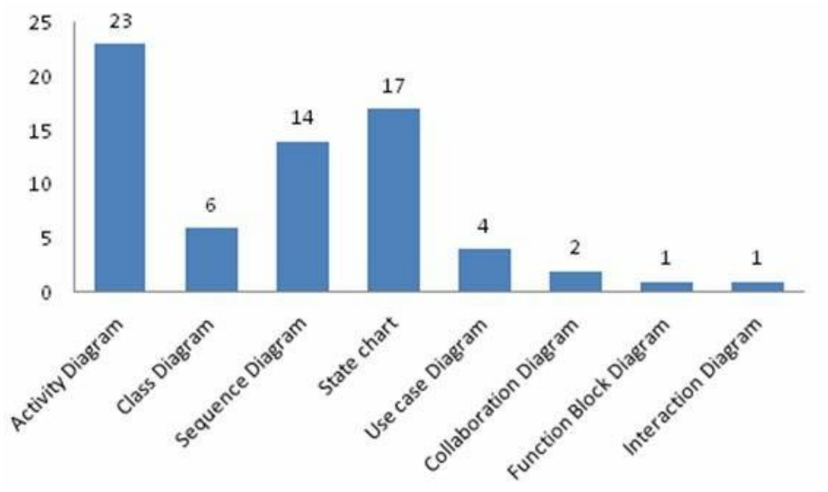

Fig. 7 Usage distribution of various UML Diagrams

UML diagram versus Test overage Criteria

In this section we have analyzed test coverage criterion with different UML diagrams (see table 5). This analysis has been shown as a bubble plot in Figure 8.

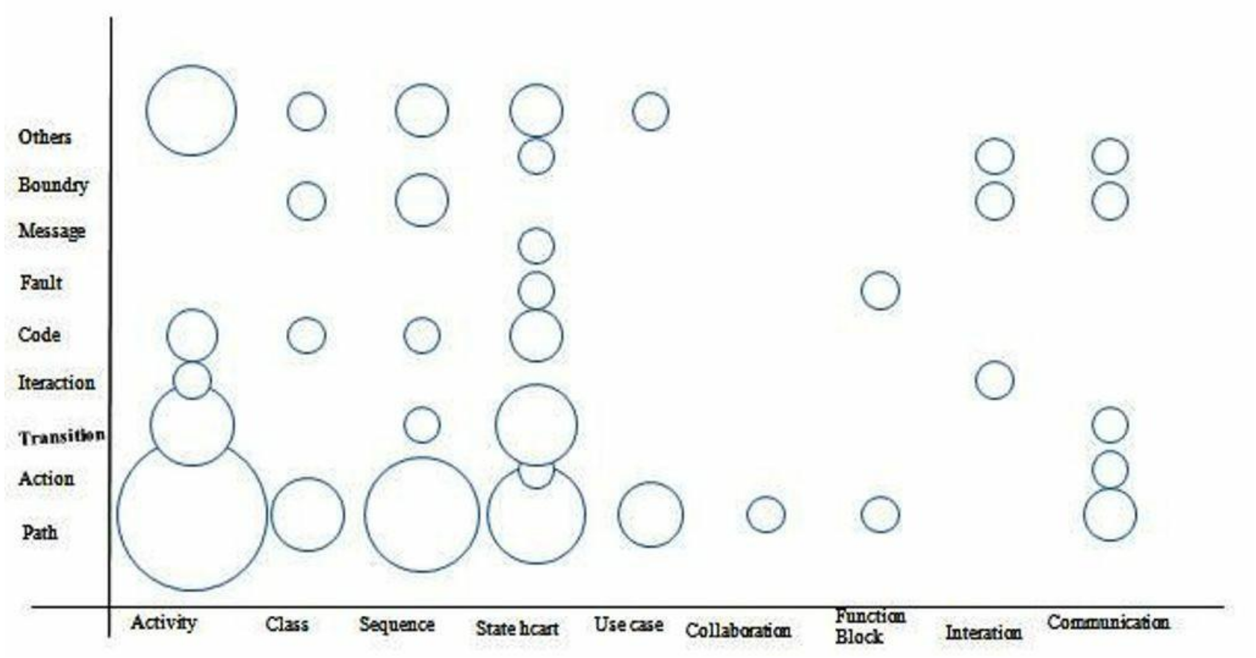

Fig. 8 Bubble Plot of UML diagram Vs. Coverage Criteria

4.4 RQ2: What are the limitations of existing techniques for automated test case generationusing UML?

4.5 RQ3: What are future direction and challenges in automated test case generation usingUML?

We have merged answer to question 2 and question 3 as they both are related. Limitations of the studies can be considered as their future work. Aim of these questions is to find gaps and challengesissues that should to be handled for effectively generating test case using UML diagrams. The identified issues that required to be addressed are numbered as follows:

1. Very small number of primary studies focused on search space saving while generating test cases.Only [P26] considered this metric. [P6, P29]
2. How to estimate the quality, correctness and completeness of the test case specification? [P27,P37, P40, P41, P52, P53, P56]
3. Manual interference should be less. [P3, P22, P25, P30]
4. Two or more diagrams can be combined [P14, P18, P19, P31, P32, P45]
5. Test Case Generation can be extrapolated so that various test coverage can be accommodated[P34, P35, P36, P39, P58]

5 Threats to Validity

This section validates the process we have applied while performing our SLR. We have validated criteria taken for consideration of inclusion and exclusion of studies and data extraction process.

Inclusion and Exclusion of studies for synthesis

The papers were first selected from the electronic databases and then selection criteria (title and abstract scrutiny) have been applied. We have found 147 papers. Full paper scrutiny is then applied thenwe found 66 papers. A summarized explanation of the selection criteria followed with the formulated questions were implemented to get rid of incorrect exclusion of the required studies. Subjectivejudgment is potential threat for inclusion and exclusion of studies. Last two authors read full text ofpapers and then reach at the conclusion. To cutoff these issues at each step the discussion has been made among authors.

Data Extraction:

A large number of researches have been done in the field of test case generation using UML diagramsand this SLR focuses on three research question (1) existing techniques,

(2) limitations, and (3) future work/ issues/problems that need to be addressed in future. The extraction form of data was structuredto analyze the information of primary studies more effectively. To reduce the hazard data extractionwas done by first two authors. The final data submitted was taken into consideration by first two authors arranged to value the study relevance and reliability.

6 Conclusion and Future Work

The aim of this study is to find out and examine the applicability of the UML diagrams in automatedtest case generation process. Two research questions formulated for this review study. 58 studies were found as primary studies for this SLR. We have identified limitation of the existing literature whichcan be future scope in this area of test case generation using UML diagrams. To estimate the quality,correctness and completeness of the test case specification can also be future work.

References

[1] Deepika, Ompal Singh, Adarsh Anand, and N. P. Singh. Testing domain dependent software reliability growth models.
International Journal of Mathematical, Engineering and Management Sciences, 2(3):140–149, 2017.

[2] Raksha Verma and Subhrta Parihar, R. S.and Das. Modeling software multi up-gradations with error generation and fault severity. International Journal of Mathematical, Engineering and Management Sciences, 3(4):429–437, 2018.

[3] Yoo-Min Choi and Dong-Jin Lim. Automatic feasible transition path generation from UML state chart diagrams using grouping genetic algorithms. Information and Software Technology, 94:38–58, 2018.

[4] Nisha Rathee and Rajender Singh Chhillar. A survey on test case generation techniques using UML diagrams. Journalof Software, 12(8):643–648, 2017.

[5] Karmbir and Kuldeep Kaur. Survey of software test case generation techniques. International Journal of Advanced Research in Computer Science and Software Engineering, 3(6):937–942, 2013.

[6] Kirandeep Kaur and Vinay Chopra. Review of automatic test case generation from UML diagram using evolutionary algorithm. International Journal of Advanced Research in Computer Science and Software Engineering, 2(11):17–20,2014.

[7] Neha Pahwa and Kamna Solanki. UML based test case generation methods: A review. IJCA Journal, 95(20):1–6, 2014.

[8] M Prasanna, S N Sivanandam, R Venkatesan, and R Sundarrajan. A survey on automatic test case generation. Aca- demic Open Internet Journal, 15:6, 2005.

[9] S. E. Ingle and M. R Mahamune. An UML based software automatic test case generation: Survey. IRJET, 02(2):971–973, 2015.

[10] Arilo C. Dias Neto, Rajesh Subramanyan, Marlon Vieira, and Guilherme H. Travassos. A survey on model-based test-ing 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) 2007, WEASELTech '07, pages 31–36. ACM, 2007.

[11] Monalisha Khanda, Arup Abhinna Acharya, and Durga Prasad Mohapatra. A survey on test case generation from UML model. IJCSIT, 2(3):1164–1171, 2011.

[12] Dr. Arvinder Kaur and Vidhi Vig. Systematic review of automatic test case generation by UML diagrams. Interna- tional Journal of Engineering Research & Technology (IJERT), 1(6):1–17, 2012.

[13] Noraida Ismail, Rosziati Ibrahim, and Noraini Ibrahim. Automatic generation of test cases from use-case diagram.

Proceedings of the International Conference on Electrical Engineering and Informatics, pages 699–702.

[14] Anjali Sharma and Maninder Singh. Generation of automated test cases using UML modeling. International Journal of Engineering Research & Technology (IJERT), 2(4):1833–1835, 2013.

[15] Yasir Dawood Salman and Nor Laily Hashim. Automatic test case generation from UML state chart diagram: A survey. In Hamzah Asyrani Sulaiman, Mohd Azlishah Othman, Mohd Fairuz Iskandar Othman, Yahaya Abd Rahim, and Naim Che Pee, editors, Advanced Computer and Communication Engineering Technology, volume 362, pages 123–134. Springer International Publishing, 2016.

[16] T. J. Ostrand and M.J. Balcer. The category-partition method for specifying and generating functional tests. Magazine Communications of the ACM, 31(6):676–686, 1988.

[17] Saswat Anand, Edmund K Burke, Tsong Yueh Chen, John Clark, Myra B. Cohen, Wolfgang Grieskamp, Mark Har- man, Mary Jean Harrold, and Phil McMinn. an orchestrated survey of methodologies for automated software test casegeneration. Journal of Systems and Software, 86(8):1978–2001, 2013.

[18] Itti Hooda and Rajendra Chhillar. A review: Study of test case generation techniques. IJCA Journal, 107(16):33–37,2014.

[19] S. Tahiliani and P. Pandit. A survey of UML based approaches to testing.

International Journal of Computational Engineering Research, 2:1396–1401, 2012.

[20] Barbara Kitchenham and Pearl Brereton. A systematic review of systematic review process research in software engineering. Information and Software Technology, 55(12):2049–2075, 2013.

[21] Barbara Kitchenham, O.Pearl Brereton, David Budgen, Mark Turner, John Bailey, and Stephen Linkman. System- atic literature reviews in software engineering a systematic literature review. Information and Software Technology, 51(1):7–15, 2009.

[22] Philip Achimugu, Ali Selamat, Roliana Ibrahim, and Mohd Nazri Mahrin. A systematic literature review of software requirements prioritization research. Information and Software Technology, 56(6):568–585, 2014.

[23] Tore Dyba, Vigdis Kampenes, and Dag I.K. Sjoberg. A systematic review of statistical power in software engineeringexperiments. Elsevier, 48(8):745–755, 2006.

[24] Tore Dyba, Torgeir Dingsoyr, and Geir K. Hanssen. Applying systematic reviews to diverse study types: An experience report. IEEE, 2007.

[25] Martin Ivarsson and Tony Gorschek. A method for evaluating rigor and industrial relevance of technology evaluations.
Empirical Software Engineering, 16(3):365–395, 2011-06.

[26] Tore Dyba and Torgeir Dingsoyr. Empirical studies of agile software development: A systematic review. Informationand Software Technology, 50(9):833–859, 2008.

[27] Roel Wieringa, Neil Maiden, and Nancy Mead. Requirements engineering paper classi?cation and evaluation criteria:A proposal and a discussion. Requirements Eng (2006), 11:102–107, 2005.

[28] Shireesh Asthana, Saurabh Tripathi, and Sandeep Kumar Singh. A novel approach to generate test cases using class and sequence diagrams. In Sanjay Ranka, Arunava Banerjee, Kanad Kishore Biswas, Sumeet Dua, Prabhat Mishra, Rajat Moona, Sheung-Hung Poon, and Cho-Li Wang, editors, Contemporary Computing, volume 95, pages 155–167.Springer Berlin Heidelberg, 2010.

7 Appendix

Table 4 Excluded Studies from Quality Assessment Step

S. No.	Title	QA 1	QA 2	QA 3	QA 4	QA 5	QA6	Total
1	UML Sequence Diagram Based Testing Using Slicing	1	0	0	0	0	1	2
2	A Business Process of Web Services Testing Method Based on UML2.0 Activity Diagram	1	0.5	0.5	0.5	0	1	3.5

3	A Novel Approach for Scenario-Based Test Case Gen- Eration	1	0	0.5	0.5	0	1	3
4	Mapping UML to Labeled Transition Systems for Test- Case Generation	1	0	1	0	0	1	3
5	Automated Test Case Generation for Object Oriented Systems Using UML Object Diagrams	1	0	0.5	0.5	0	1	3
6	Automatic Test Case Generation Using Sequence Dia- gram	0.5	0	0.5	0.5	0	0.5	2
7	Automatic Test Case Generation with State Diagram for Validating the Solar Integrated System	1	0	0.5	0	0	0.5	2
8	Agent-Based Regression Test Case Generation using Class Diagram, Use cases and Activity Diagram	0.5	0	0.5	0.5	0	0	1.5

Table 5: Analysis of Primary Studies

P ID	Apllicability	Coverage Criterion	Intermediate Model
P1	State Diagram	Code Coverage	XMI, Tree
P2	State Chart	Fault, transition coverage	OCL, MDL (Rational Rose file)
P3	State Chart	Transition	STRIPS Planning problem
P4	State Chart	Minimal arc coverage	Usage model, usage graph
P5	Activity Diagram	Path coverage	MDL
P6	State Chart	state, transition, condition, boundary, data flow	Timed automata
P7	State machine	Path coverage	XMI, composite control flow graph, adjacency matrix
P8	Activity Diagram	Path coverage	XMI, tree
P9	Sequence	Path coverage	Sequence Dependency Table

P10	use case, sequence	Path coverage	UDG, SDG, (system testing gr- pah)
P11	Sequence	Path coverage, transition cover- age	Labeled transition system
P12	Combination of collaboration diagrams and statecharts	Path coverage	State COllaboration TEst Model
P13	Sequence, class and OCL	Path coverage	Testable Aggregate Model
P14	Communication diagram	link coverage, message paths coverage as well as boundary coverage	XML
P15	State machine	Transition path coverage	NONE

P16	State, Class and OCL	Path coverage	test case tree
P17	State machine and their semantic model	Input sequence based coverage (path, state, component)	None
P18	sequence diagram	symbolic path coverage	Tree
P19	Activity Diagram	activity, transition, simple path, trail coverage	Java programs
P20	Activity Diagram	various criteria	XMI and other details
P21	Class, activity, sequence	Path coverage	Extended class, activity and sequence diagram
P22	State Chart	elementary transition path	synthesized state model
P23	Activity Diagram	Path coverage	Interaction Flow Diagram, Interaction Flow Graph
P24	Class, sequence diagram	message sequence	XMI
P25	sequence diagram	method and exception coverage	NONE
P26	Activity Diagram	Hybrid	activity dependency (AD) table and AD graph
P27	State Chart	Action Coverage	NONE
P28	use case, class, interaction	Path coverage	Graph
P29	State Diagram	Code Coverage	NONE
P30	State Diagram	Transition coverage	State Table, transtion tree
P31	Interaction diagram	slice test, meaasge path, boundary testing, basic interaction	message flow dependency graph

		coverage	
P32	Activity, sequence diagram	Path coverage	System graph
P33	Class, state, activity Diagram	Code Coverage	None
P34	Actitvity	activity path coverage	Activity Graph
P35	Activity Diagram	Path coverage	Euler's circuit
P36	sequence, state amchine	structure coverage.. Path cover- age	NONE
P37	sequence diagram	object coverage	use case contract(predicate logic)
P38	Seuence Diagram	Code Coverage	non-deterministic Automata
P39	Collaboration Diagram	Test coverage metrics	Proposed Algorithm, tree
P40	Activity Diagram	basic path coverage, simple path coverage	EADG, ADG
P41	Use Case, Activity Diagram	None	OCL ,Tree
P42	Activity Diagram	Control Flow Path Coverage	XMI, Tree
P43	Activity diagram	Path Coverage ,	FDG
P44	Sequence Diagram, State chart Diagram		Sequence Graph, State chart Graph, SYTG
P45	Activity Diagram	Path Coverage, Code Coverage	BDT, GP
P46	Sequence Diagram	message sequence path coverage	SFC, MCFG, XMI
P47	Function Block Diagram	FB-Path Complete Condition Test Coverage, FBD structural testing coverage	UPPAAL model

P48	Sequence Diagram	test path coverage	TEDCPN
P49	State based Use Case	Control Flow Path Coverage	FDR tool, CNL
P50	Class Diagram	Concerning coverage	OCL
P51	Activity Diagram	covering array	JVM
P52	Activity Diagram, Sequence Diagram	simple path flow	ADG, SDG, SYTG
P53	Communication and Activity Diagram	basic path coverage , Activity-PathCoverage	TFT, COMMACTtree
P54	Activity Diagram	Activity Coverage ,Transition Coverage ,Simple Path Coverage	DFS
P55	Activity Diagram	Transition Coverage	GGA
P56	Activity Diagram	Transition Coverage, Branch Coverage	XMI
P57	Activity Diagram	Activity coverage/ Transition coverage /Key path coverage / Interaction coverage	XMI
P58	Activity Diagram	Flow of Control	ITM

Table 6 Quality Score of Primary Studies

ID	QA1	QA2	QA3	QA4	QA5	QA6	Total	ID	QA1	QA2	QA3	QA4	QA5	QA6	Total	ID	QA1	QA2	QA3	QA4	QA5	QA6	Total
1	1	1	1	1	0	1	5	21	1	1	0.5	0.5	0	1	4	41	1	0.5	1	0.5	0	1	4
2	1	0	1	1	0	1	4	22	1	1	1	1	0	1	5	42	1	1	1	1	1	1	6
3	1	1	1	1	0	1	5	23	1	1	1	1	0	1	5	43	1	1	1	1	1	1	6
4	1	1	1	1	0	1	5	24	1	1	1	1	0	1	5	44	1	1	1	1	0	1	5
5	1	1	1	1	0	1	5	25	1	1	1	1	1	1	6	45	1	1	1	1	0	1	5
6	1	0	1	1	0	1	4	26	1	0	1	1	0	1	4	46	1	1	1	1	0	1	5
7	1	1	1	1	1	1	6	27	1	0	1	1	0	1	4	47	1	1	1	1	0	1	5
8	1	0	1	1	0	1	4	28	1	0	1	1	0	1	4	48	1	1	1	1	0	1	5
9	1	0	1	1	0	1	4	29	1	1	0.5	0.5	0	1	4	49	1	1	1	1	0	1	5
10	1	1	1	1	0	1	5	30	1	1	1	1	0	1	5	50	1	1	1	1	0	1	5
11	1	1	1	1	0	1	5	31	1	1	1	1	1	1	6	51	1	1	1	1	0	1	5
12	1	1	1	1	1	1	6	32	1	1	1	0.5	0	1	4.5	52	1	0	1	1	0	1	4
13	1	1	1	1	0	1	5	33	1	0	1	1	0	1	4	53	1	1	1	1	1	1	6
14	1	1	1	1	1	1	6	34	1	1	1	1	0	1	5	54	1	0.5	1	0.5	0	1	4
15	1	1	1	1	0	1	5	35	1	1	1	1	0	1	5	55	1	0.5	1	1	0	1	4.5
16	1	1	1	1	1	1	6	36	1	1	1	1	1	1	6	56	1	1	1	1	0	1	5
17	1	1	1	1	0	1	5	37	1	1	1	1	0	1	5	57	1	1	1	1	0	1	5
18	1	0	1	1	0	1	4		1	1	1	1	0	1	5		1	1	1	1	0	1	5
19	1	1	1	1	0	1	5		1	1	1	1	0	1	5								
20	1	0.5	1	0.5	0	1	4		1	1	1	1	0	1	5								

								38 39 40								58							

Table 7: Details of Primary Studies

S.No	Title	Author	Year
P1	Model-based automatic test case generation for automotive embedded software testing	Henny B. SipmaToms E. UribeZohar Manna	2018

P2	Generating Tests from UML Specifications	Jeff OffuttAynur Abdurazik	2003
P3	Automated Test Case Generation from Dynamic Models	Peter FrhlichJohannes Link	2000
P4	UML-Based Statistical Test Case Generation	Matthias RiebischIlka Philip- powMarco Gtze	2002
P5	Generating test cases from UML activity diagram based on Gray-box method	Wang Linzhang; Yuan Jiesong; Yu Xiaofeng; Hu Jun; Li Xuan- dong; Zheng Guoliang	2005
P6	Generation of Optimized Testsuites for UML Statecharts with Time	Tilo MckeMichaela Huhn	2004
P7	Generating and evaluating effectiveness of test sequences using state machine	A. PretschnerO. SlotoschE. AiglstorferS. Kriebel	2017
P8	Using adaptive agents to automatically generate test sce- narios from the UML activity diagrams	Dong Xu; H. Li; C. P. Lam	2005
P9	Automatic Test Case Generation from UML Sequence Di- agram	M. Sarma; D. Kundu; R. Mall	2008
P10	Automatic Test Case Generation from UML Models	M. Sarma; R. Mall	2005
P11	Test case generation by means of UML sequence diagrams and labeled transition systems	E. G. Cartaxo; F. G. O. Neto; P. D. L. Machado	2007
P12	A state-based approach to integration testing based on UML models	Ali, Shaukat; Briand, Lionel C.; Rehman, Muhammad Jaffar-ur; Asghar, Hajra; Iqbal, Muham- mad Zohaib Z.; Nadeem, Aamer	2007
P13	Testing UML designs	Pilskalns, Orest; Andrews, An-	2007

		neliese; Knight, Andrew; Ghosh, Sudipto; France, Robert	
P14	Automatic test case generation from UML communication diagrams	Samuel, Philip; Mall, Rajib; Kanth, Pratyush	2007
P15	Automatic test case generation using unified modeling lan- guage (UML) state diagrams	P. Samuel; R. Mall; A. K. Bothra	2007
P16	Deriving Input Partitions from UML Models for Automatic Test Generation	Stephan WeilederBernd-Holger Schlingloff	2008
P17	Conformance Testing Based on UML State Machines	Dirk Seifert	2008
P18	Validation and automatic test generation on UML models: the AGATHA approach	Gustavo CabralAugusto Sam- paio	2008
P19	UML Activity Diagram-Based Automatic Test Case Gen- eration For Java Programs	M. Chen; X. Qiu; W. Xu; L. Wang; J. Zhao; X. Li	2004
P20	TSGen: A UML Activity Diagram-Based Test Scenario Generation Tool	C. Sun; B. Zhang; J. Li	2007
P21	Test Cases Generation for Embedded Real-Time Software Based on Extended UML	Y. Yongfeng; L. Bin; L. Minyan; L. Zhen	2009
P22	Automatic generation of test specifications for coverage of system state transitions	Sarma, M.; Mall, R.	2009
P23	Generating Test Plans for Acceptance Tests from UML Ac- tivity Diagrams	A. Heinecke; T. Brckmann; T. Griebe; V. Gruhn	2010
P24	A Novel Approach to Generate Test Cases Using Class and Sequence Diagrams	Shireesh AsthanaSaurabh Tri- pathiSandeep Kumar Singh	2010

P25	A Hybrid Genetic Algorithm Based Test Case Generation Using Sequence Diagrams	Mahesh ShiroleRajeev Kumar	2010
P26	An enhanced test case generation technique based on ac- tivity diagrams	P. N. Boghdady; N. L. Badr; M. A. Hashim; M. F. Tolba	2012
P27	Test case generation approach for industrial automation systems	R. Hametner; B. Kormann; B. Vogel-Heuser; D. Winkler; A. Zoitl	2012
P28	Construction of Test Cases from UML Models	Vinaya SawantKetan Shah	2011
P29	Fault-Based Generation of Test Cases from UML-Models Approach and Some Experiences	Rupert SchlickWolfgang HerznerElisabeth Jbstl	2011
P30	A Study on Test Case Generation Based on State Diagram in Modeling and Simulation Environment	Woo Yeol KimHyun Seung Son-Robert Young Chul Kim	2011
P31	Test Case Design Using Slicing of UML Interaction Dia- gram	Swain, Ranjita Kumari; Panthi, Vikas; Behera, Prafulla Kumar	2012
P32	Test Case Generation Using Activity Diagram and Se- quence Diagram	Abinash TripathyAnirban Mitra	2013
P33	Interaction Diagram Based Test Case Generation	Rohit KumarRajesh K. Bhatia	2012
P34	Testcases Formation Using UML Activity Diagram	P. E. Patel; N. N. Patil	2013
P35	Extenics-based Test Case Generation for UML Activity Diagram	Li, Liping; Li, Xingsen; He, Tao; Xiong, Jie	2013
P36	Automated Method for Software Integration Testing Based on UML Behavioral Models	Dominykas BarisasEduardas Bareia arnas Packeviius	2013
P37	An Automatic Generation Strategy for Test Cases	Dandan HeLijuan	201

	Based on Constraints	WangRuijie Liu	3
P38	Techniques and Toolset for Conformance Testing against UML Sequence Diagrams	Joo Pascoal FariaAna C. R. PaivaMrio Ventura de Castro	2013
P39	Automatic test case generation through collaboration dia- gram: a case study	Arvinder KaurVidhi Vig	2018
P40	A Graph Transformation Approach for Automatic Test Cases Generation from UML Activity Diagrams	Parosh Aziz AbdullaK. Rustan M. Leino	2015
P41	Information Systems Requirements Specification and Us- age in Test Case Generation	Neringa SipaviienKristina Smil- gytRimantas Butleris	2014
P42	Generating Test Data from a UML Activity Using the AMPL Interface for Constraint Solvers	Felix KurthSibylle Schupp- Stephan Weileder	2014
P43	Slicing-based Test Case Generation from UML Activity Diagrams	scar Snchez RamnJess Snchez CuadradoJess Garca Molina	2009
P44	Test Case Generation and Optimization using UML Mod- els and Genetic Algorithm	Khurana, Namita; Chillar, R. S.	2015
P45	A Novel Approach to Generating Test Cases with Genetic Programming	Sao KarakatiTina Schweighofer	2015
P46	Test Case Creation from UML Sequence Diagram: A Soft Computing Approach	Ajay Kumar JenaSantosh Kumar SwainDurga Prasad Mohapatra	2015
P47	Automatic test case generation for structural testing of function block diagrams	Wu, Huayao; Nie, Changhai; Kuo, Fei-Ching	2014
P48	Automated Testing of Distributed and Heterogeneous Sys- tems Based on UML Sequence Diagrams	Bruno LimaJoo Pascoal Faria	2016

P49	Test generation from state based use case models	Zuohua DingMingyue Jiang-Haibo ChenZhi JinMengchu Zhou	2014
P50	Test data generation for web application using a UML class diagram with OCL constraints	Joo Pascoal FariaAna C. R. Paiva	2011
P51	A prototype tool for generating and executing test cases from UML-based interface behavior descriptions	A. Thomas; J. Kimball	2017
P52	Automated Test Case Generation from UML Activity Dia-gram and Sequence Diagram using Depth First Search Al-gorithm	Meiliana; Septian, Irwandhi; Alianto, Ricky Setiawan; Daniel; Gaol, Ford Lumban	2017
P53	Prioritizing test scenarios from UML communication and activity diagrams	rica Ferreira de SouzaVal-divino Alexandre de Santiago JniorNandamudi Lankalapalli Vijaykumar	2014
P54	Automatic Test Case Generation for UML Activity Dia-grams	Shruti JaiswalDaya Gupta	2006
P55	Automatic feasible transition path generation from UML state chart diagrams using grouping genetic algorithms	Choi, Yoo-Min; Lim, Dong-Jin	2018
P56	EasyTest: An Approach for Automatic Test Cases Genera-tion from UML Activity Diagrams	Fernando Augusto Diniz Teix-eiraGlaucia Braga e Silva	2018
P57	Efficient test case generation for validation of UML activ-ity diagrams	Arvinder KaurVidhi Vig	2010
P58	Synthesis of test scenarios using UML activity diagrams	Ashalatha Nayak Debasis Samanta	2011