International Journal of Computer Applications (0975 – 8887)
Volume 184 – No.13, May 2022

A Systematic Mapping Study on UML Model based Test Case Generation and Optimization Techniques

James Maina Mburu
Department of Information Technology, Murang'a University of Technology, Kenya

John Gichuki Ndia
Department of Information Technology, Murang'a University of Technology, Kenya

ABSTRACT

Test case generation is a significant step in software testing as it ensures that, software produced is error free and is of high quality. On the other hand, optimization techniques ensure that ideal test cases are generated. In this paper, the researchers present an overview of state-of- the- art research on test case generation and optimization techniques. The study included papers from google scholar, IEEE Xplore digital library and springer that were published between 2010-2022. The results indicate that most of the studies focused on single UML models, and search techniques such as breadth first search and depth first search, while there exists very few studies on combinational UML models, and test case optimization techniques such as metaheuristic search algorithms. Further, case study remains the most popular approach for validation of test case generation and optimization techniques, but there are few studies focusing on experimental validation.

General Terms

Systematic mapping study, UML, Model based test case generation and optimization.

Keywords

Testing, Test case generation, optimization techniques, UML models.

1. INTRODUCTION

Software Testing is the one of the most important phases in software development and consumes nearly 40 to 60% of effort, time, and cost. Due to the end users urge to complete the project in short span with high quality and defects free, the testing activity must be started as early as possible to fix the bugs at early stage. Generating an effective test case plays a major role in software testing. Test generation process deals with creation of a set of testing conditions which can be used for validating the adequacy of the application [1]. Test cases are planned for every software system and as the system grows in complexity, more test cases are needed and consequently the time and effort required for appropriate testing are increased. Thus, optimizing the process of test case generation is critical.

There are different techniques used for test case generation such as model based technique which generates the test cases from the UML models, search-based test generation which uses meta heuristic techniques that direct the search towards the potential areas of input space, random approaches that generates test cases based on assumptions, Goal based test data generation approach that cover a particular section, statement or function, and specification based techniques that generate test data based on the formal requirement specifications. However, there are a myriad of challenges that have not been addressed such as test optimization using metaheuristic algorithms, use of combinational UML in test case generation and the need to evaluate techniques through experiments using large software.

Model based testing is a technique which is used for designing and modelling the artifacts of the software system. In this study, a model is a representation of the function (behavior) of software under test and a function can be in terms of input, output, action, events and many more. Software testing rely on the models as the test cases remain the same even after certain changes are made in the code. Test cases are generated from the model that defines the functionality of the software [2].

The development of automatic test case generation process assists the software testing engineer and saves more time as compared to manual testing. In addition, the cost of testing decreases with the reduction of testing time. It is important to note that most of the software's are delivered without sufficient testing, due to short duration to deliver the software product and this eventually leads to loss of revenue. By automating the test case generation process, manual efforts can be eradicated, which can result to test time savings and cost reductions of software development and maintenance [3].

UML is a general-purpose modelling language used to visualize, analyze and document the components of a system in form of a model or design. The UML diagrams are classified into two, the structural diagrams and behavioral diagrams. Structural diagrams describe the structure of the software and represent the static aspect (fixed part) of the system, while the behavioral diagrams describe the dynamic aspects (moving and changing part) of the software [2].

The goal of this study was to identify existing test case generation and optimization techniques, UML models used, methods or algorithms applied, methods of evaluation, strengths, and weaknesses of these approaches and the research gaps of the study. The study was performed using systematic mapping study (SMS) protocol presented in Section 2 and it covered hundreds of scientific publications from google scholar, IEEE digital library and springer.

The rest of the paper is structured as follows. Section 2 describes the protocol for the systematic mapping study used to find and evaluate papers in this study. The protocol is described in detail for the purpose of replicability. Finally, the section presents the research questions. In Section 3, the results of the study are presented. Potential threats to the validity of this study are discussed in Section 4, and finally in Section 5, we present our conclusions.

2. THE SYSTEMATIC MAPPING STUDY

This section describes the protocol used for the SMS. The protocol is largely based on the one used in [4], but it has been modified according to the topic of this study.

2.1 Research Questions

The research questions (RQ) are as follows:

RQ1: Which UML models are used to generate test cases?

RQ2: Which techniques are used to generate and optimize test cases?

RQ3: How are the techniques evaluated?

RQ4: What are the research gaps of studies?

2.2 Search Strategy for Primary Studies

This section presents our search strategy, which based on the systematic literature review guidelines from [5] and [6].

2.2.1 *Search terms*

Table 1 lists the search terms used when searching for original papers for this study. The search terms are derived from the research questions

Table 1. Search terms with alternate spellings

Term	Alternate spelling
Test case	Test cases
Generation	
optimization	
UML	Unified modelling language
Model	Models
Diagrams	Diagrams
Technique	Techniques
Algorithm	Algorithm

2.2.2 *Search terms*

The search terms listed in Table 1 were combined into two search strings for use in the digital libraries. These are shown in Table 2.

Table 2. Search strings

No	Search String
1	Test case AND Generation AND Optimization AND (Technique OR Techniques) AND (UML OR Unified Modeling Language) AND (Model OR Models).
2	Test case AND Generation AND Optimization AND (Technique OR Techniques) AND (UML OR Unified Modeling Language) AND (Diagram OR Diagrams).

2.2.3 *Databases*

The search strings shown above were applied to the following digital libraries

- IEEE Xplore
- Google Scholar
- Springer

The first search string was used for all two databases while the second string was used to search abstracts in the IEEE Xplore database only. This was done to reduce the number of papers found.

Since the digital libraries have different possibilities for defining search strings, the strings were customized to every digital library. Duplicates were removed from the collected results.

2.2.4 *Study inclusion criteria*

- The inclusion criteria for primary studies were as follows;
- Written in English AND
- Published in a peer-reviewed journal, conference or workshop covering the subjects of software engineering
- Describing any one of the following;
    - o Models or diagrams in test case generation
    - o Techniques or methods used in test case generation and optimization.
    - o Evaluation of existing test case generation and optimization techniques.
    - o Research gaps or future work

2.2.5 *Title and abstract level screening*

The inclusion criteria were applied to publication title and abstracts. The screening results were used as a starting point for the full text screening.

2.2.6 *Full text level screening*

In this stage, the remaining papers were analyzed based on their full text. To reduce biasness, two researchers were involved in applying inclusion criteria on the full text. Here, one of the researchers screened all the papers while the other researcher screened the half of the papers due to the time constrain. The results were compared and disagreements were solved through discussion.

2.2.7 *Study quality assessment checklist and procedure*

The selected papers were assessed based on their quality in terms of contribution to test case generation and optimization.

Two researchers assessed the quality of the selected papers with one research assessing all independently, while the other researcher assessed the half of the paper. Then, thereafter results were compared and disagreement resolved through discussion between researchers.

Any paper not meeting minimum quality requirements as described below, was excluded from the set of primary studies.

Table 3 presents the checklist of the study quality assessment. For each question in the checklist a three- level, numeric scale was used [7]. The levels were True (2 points), partial (1 point) and false (0 point). If the study scored 8 points or less, it was discarded due to the lack of quality in relation to this study. The research documented the obtained score of each included/excluded study.

Table 3: Study quality assessment

NO	Question
Theoretical Contribution	
1	Is at least one of the questions addressed?
2	Was the study designed to address some of the research questions?
3	Is a problem description for the research explicitly

	provided?
4	Is the problem description supported by references of other works?
5	Are the contribution research clearly described?
6	Are there assumptions, if any, clearly stated?
7	Is there sufficient evidence to support the claims of the research?
Experimental evaluation	
8	Is the research clearly described?
9	Is prototype, simulation or empirical study presented?
10	Is the experimental set up clearly described?
11	Are the results from multiple different experiment included?
12	Are the results from multiple runs of each experiment included?
13	Are the experimental results compared with other approaches?
14	Are negative results if any presented?
15	Is the statistical significance of the results assessed?
16	Are the limitations clearly stated?
17	Are links between data, interpretation and conclusions clear?

2.2.8 *Data Extraction Strategy*

The researchers used the form shown in Table 4 to extract data from the primary studies. Two researchers extracted the information from the papers with each researcher extracting data from one third of the papers. After the data extraction, the results were double-checked by the reviewing researchers. The extracted data was then used for analysis, applying RQs from Section II-A to obtain answers.

2.2.9 *Synthesis of the Extracted Data*

The extracted data from the papers was analyzed to obtain a high-level view of the different aspects related to test case generation and optimization. The papers were categorized and collective results were extracted. The results from this phase are presented and discussed in Section 3.

3. RESULTS

In this section, researchers present the main findings of the research. They used search terms such as "test case generation" and" optimization*" that are used in several research contexts. Consequently, some findings were not related to test case generation from UML models. For example, some papers were related to structural testing or white box testing, which are not related to the topic of this paper.

As seen from table 5, the initial paper search produces a number of the study included papers that were published between 2010-2022. The researcher discarded papers not relating to test case generation and optimization from other domain e.g., structural test case generation. This study strictly covered model (behavioral) based test case generation approaches using UML

Table 4: Data extraction form

Data Item	**Value**	**Note**
General		
Data extractor name		
Data extraction Date		
Study identifier (1,2,3…)		
Bibliographic reference (Title, author, year, journal/conference/workshop		
Test case generation and optimization		
(RI) UML Models applied to generate test cases (activity, sequential, state chart, use case diagrams etc.		
(R2) methods or algorithms for test case generation and optimization		
(R3) Strengths of the identified techniques		
(R4) Weakness of the identified techniques		
(R5) Evaluation methods		
(R6) Research gaps		

Table 5: Number of papers in each phase of the paper search & screening

Phase	No. of Papers
Initial search results	348
After Title & Abstract screening	115
After full text screening	86
After quality assessment	47

After initial paper search, there were 348 papers found. After title & abstract screening, only 115 papers were included in the next phase. After full text screening, there were 86 papers which included in the next phase (Quality assessment) and finally 47 papers were selected for the study. [Most of the paper (29) were published in journals where 18 were published in conference procedures.

As shown in Figure 1, the subject of test case generation & optimization is trending toward greater interest over time. The year 2021 had the highest (8) papers selected for the study, 2020 had 3 papers selected, 2019, 2018 and 2016 had each 4 papers selected, the year 2017 had 5 papers selected for the study, the year 2015 and 2013, each had 6 papers selected for the study, 2014 had 3 papers selected for the study while,2012 and 2010 had each 2 papers selected for the study. Lastly, the year 2011 had 1 paper selected for the study.

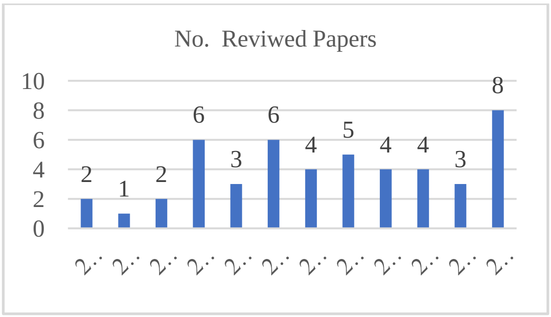


Fig 1: Reviewed papers sorted by the year of publication

3.1 UML models used in test case generation

The idea behind RQ1 was to identify UML diagrams used in producing test cases. According to the findings activity diagram is the most studied in the literature, followed by the UML combinational diagrams such as activity and sequence diagrams or state chart and sequence diagrams.

Further, to this most of the work focused on producing test cases from individual or single UML diagrams. Other types of UML diagrams were not considered by the primary studies, hence there is need to address this issue.

Figure 2 indicates that 14 primary studies covered activity diagram, 11 primary studies used combinational UML diagrams, 7 primary studies covered state chart diagram, 8 primary studies covered sequence diagram while use case diagram was covered by 2 primary studies. Finally, the collaboration diagram was covered by 3 primary study.

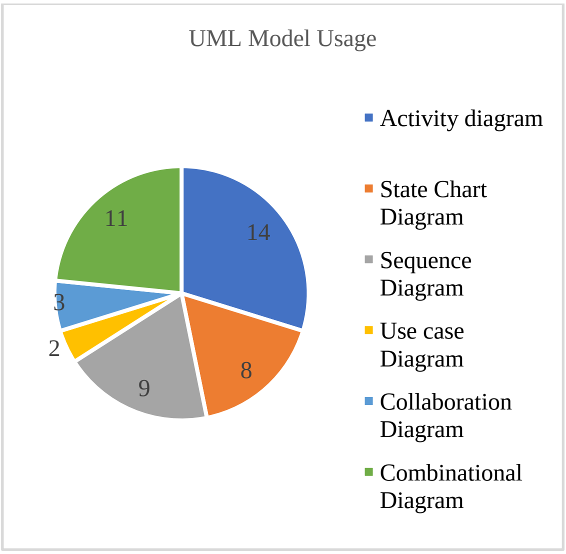


Fig 2: Number of UML Models

3.2 Techniques or methods used to generate test cases

The idea behind RQ2 was to identify the test case generation and optimization techniques with their strengths and limitations.

The results show that most of the studies focused on generating and optimizing test cases using search technique such as DFS & BFS algorithm. These algorithms have average time complexity. The most studied metaheuristic technique was genetic algorithms while other metaheuristic algorithms such as cuckoo search, bee colony, particle swarm optimization were least studied.

Table 6 show that 14 primary studies used DFS and BFS to generate and optimize test cases, 10 primary studies used genetic algorithm generate and optimize test cases, 12 primary studies applied test case generation algorithms. These techniques were proposed by the developers and did not focus on test case optimization. A hybrid of cuckoo search (CS) and bee colony (BC), bacterial foraging algorithm (BFA)-particle swarm optimization (PSO)- genetic algorithm (GA), firefly algorithm (FA)-bee colony (BC), Intelligent optimization algorithm, adaptive cuckoo search algorithm, hybrid BC algorithm, Prism and Dijkstra algorithm had each covered by 1 primary study.

Table 6: Test case generation and optimization techniques

Techniques/Method	Count	References
CS & BC Algorithm (CSBCA)	1	[1]
Hybrid BFA-PSO-GA	1	[2]
DFS & BFS, DFS	14	[3][4][5][6][7][8][9][10][11][12][13][14][15][16]
Hybrid FA-BC	1	[17]
Test Case Generation Algorithms	12	[18][19][20][21][22][15][23][24][25][26][27][28]
Genetic Algorithm	10	[23][29][30][31][32][33][34][35] [36] [37]
Intelligent optimization technique	1	[38]
Adaptive cuckoo search algorithm	1	[39]
Prism and Dijkstra algorithm	1	[40]
Hybrid BC algorithm	1	[41]

3.2.1 Strength of test case and optimization techniques

The results indicates that most of studies are able to generate test cases automatically but are not able to produce optimal test cases, only few primary studies focused on generating and optimizing test cases hence able to minimizing invalid test paths and redundant data.

Figure 3 indicates that 72% of the primary studies focused on test automation while 30% of the studies concentrated on test optimization.

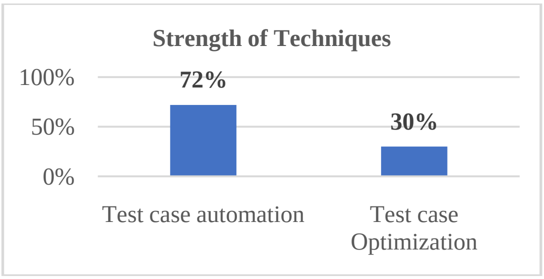


Fig 3: Strength of techniques

3.2.2 Limitations of test case and optimization techniques

The findings shows that majority of primary studies did not focus on test case optimization, automation of test case generation process, and combinational UML diagrams which increases the test coverage. Moreover, few of the studies have not been evaluated, and those that have been evaluated are using small programs and simple structures hence test cases are not effectively evaluated.

Table 7 shows that 33 primary studies had their techniques experiencing slow test case execution time, 31 studies had their test cases being generated from an individual UML diagram, then 4 primary studies had their techniques not automated. Finally, the techniques in 2 primary studies were not evaluated.

Table 7: Techniques limitation

Techniques Limitations	Count	References
Slow test case execution time	33	[18][23][4][19][15][10][31][5][20][21][6][7][12][42][38] [8][11][42][15][3][9] [40][23][36][24][13][43][14][16][27][26][25][28]
Generates test cases from an individual UML diagram	31	[39][4][19][29][31][5][20][21][32][33][12][44][45][28] [34][35][8][11][42][36][24][39][13][37][46][43][14][16][26][27][25]
Not automated	4	[18][23][29][31]
Not evaluated	2	[30][3]

3.3 Evaluation of the methods/ algorithm

The results shows that the test case generation methods were majorly evaluated using case studies. However, few of the primary studies evaluation were done by performing an experiment. Most of the papers did not conduct an experiment to compare the test case and optimization techniques with other existing methods. The few that conducted an experiment only evaluated the techniques with small programs and simple structure.

Figure 4 indicates 32 primary studies focused on case studies while 5 primary studies evaluated their results using an experiment. Only 1 study that focused on simulation method.

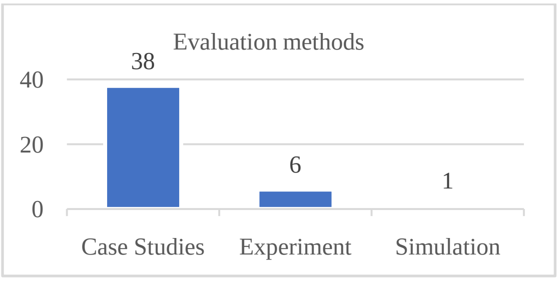


Fig 4: Evaluation methods

3.4 Research gaps

Most of the primary studies about 70% suggested the need to combine a single UML model with other behavioral UML diagrams so that the technique can be able to handle different types of errors. Secondly, 65% of primary studies suggested the need to optimize test cases to reduce execution time and redundant data, while 27% of primary studies suggested a hybrid method to improve their performance. Finally, 13% of primary studies suggested that techniques need to be automated to improve their efficiency as indicated in figure 5.

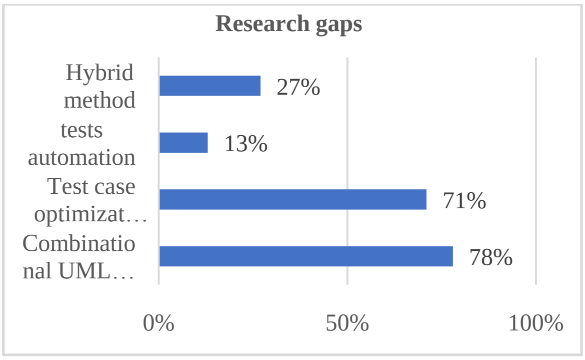


Fig 5: Research Gaps

4. THREATS TO VALIDITY

A threat to validity of this study is that only research papers from three databases i.e., IEEE Explorer, Springer and Google scholar were included. Some relevant papers from other databases may have been left out. However, the use of google scholar minimized the threat since it was able to link to papers in other databases such as ACM.

Another threat to validity is that the screening phases were performed partially by different persons. While one researcher followed the entire protocol from beginning to end, the remaining researcher had varying influence on the screening phases. These researchers may have had different views regarding paper relevancy, causing relevant papers to be excluded. In all phases where two researchers were involved, except for the data extraction phase, one researcher completed the entire phase independently, while the other two divided the workload evenly between them. Since the workload was divided, some papers may have been excluded because of differing criteria for relevance.

In the data extraction phase, each of the researchers extracted data from one third of the papers. Although each set of extracted data was double-checked by other researcher, there is a risk that some data may have been missed.

Finally, the researchers pointed out that after each phase in the protocol, consensus discussions were held and that any disagreements were resolved. Therefore, the researchers feel that any threats posed to protocol execution were minimized.

Table 8: Primary studies included, with corresponding references

ID	Reference	ID	Reference
S1	[1]	S25	[5]
S2	[2]	S26	[20]
S3	[3]	S27	[21]
S4	[22]	S28	[6]
S5	[17]	S29	[12]
S6	[40]	S30	[7]
S7	[18]	S31	[9]
S8	[23]	S32	[32]
S9	[39]	S33	[12]
S10	[4]	S34	[34]
S11	[19]	S35	[35]
S12	[29]	S36	[11]
S13	[10]	S37	[42]
S14	[30]	S38	[15]
S15	[31]	S39	[30]
S16	[44]	S40	[45]
S17	[38]	S41	[36]
S18	[24]	S42	[39]
S19	[13]	S43	[37]
S20	[46]	S44	[43]
S21	[14]	S45	[16]
S22	[25]	S46	[27]

S23	[26]	S47	[28]
S24	[41]		

5. CONCLUSIONS

The researchers have presented a systematic mapping study on test case generation and optimization. In their findings, several papers used activity diagram, state chart diagram and sequence diagrams in test case generation. Majority of studies have concentrated on generating test cases from individual UML model. Thus, the test cases produced will not be effective in fault detection since different types of faults can be detected using diverse kinds of UML models.

Many papers have used DFS and BFS algorithm in test cast generation and optimization. These algorithms have average time complexity, meaning that more time is spent in test case execution. Moreover, most of the studies are able to generate test cases automatically. However, many studies have not focused on test case generation and optimization using metaheuristic techniques hence leading to slow time execution. Use of metaheuristic techniques can be applied to produce optimal test cases and saving test execution time. Finally, several studies are evaluated using case studies. However, there are few studies focusing on experimental validation. In addition, test cases are generated from small programs and simple structures hence the need to generate test cases from large software. The said techniques are therefore, challenged for their application in test case production and optimization. Future studies should focus on generating test cases from combinational UML models using metaheuristic algorithms and evaluate their performance through experiments

6. REFERENCES

[1] P. Lakshminarayana and T. V. Sureshkumar, "Automatic Generation and Optimization of Test case using Hybrid Cuckoo Search and Bee Colony Algorithm," *J. Intell. Syst.*, vol. 30, no. 1, pp. 59–72, 2020, doi: 10.1515/jisys-2019-0051.

[2] A. T. , Et. al., "Bio Inspired Approach for Generating Test data from User Stories," *Turkish J. Comput. Math. Educ.*, vol. 12, no. 2, pp. 412–419, 2021, doi: 10.17762/turcomat.v12i2.826.

[3] Y. Tian, B. Yin, and C. Li, "A Model-based Test Cases Generation Method for Spacecraft Software," pp. 373–382, 2021, doi: 10.1109/dsa52907.2021.00057.

[4] R. Singh, "Automating the test case generation for Object Oriented Systems using Activity Diagrams," *Int. J. Eng. Comput. Sci.*, vol. 4, no. 9, pp. 14164–14171, 2015, doi: 10.18535/ijecs/v4i9.17.

[5] V. Panthi and D. P. Mohapatra, "ACO based embedded system testing using UML Activity Diagram," *IEEE Reg. 10 Annu. Int. Conf. Proceedings/TENCON*, pp. 237–242, 2017, doi: 10.1109/TENCON.2016.7847997.

[6] R. Gupta and V. Jaglan, "Test case generation for UML behavioral diagram by traversal algorithm," *Int. J. Innov. Technol. Explor. Eng.*, vol. 8, no. 10, pp. 3262–3266, 2019, doi: 10.35940/ijitee.J1190.0881019.

[7] S. P. Jain, K. S. Lalwani, N. K. Mahajan, and B. J. Gadekar, "Automatic test case generation using uml models 1," no. 6, pp. 30–34, 2014.

[8] V. Panthi and D. P. Mohapatra, "Automatic test case generation using sequence diagram," *Adv. Intell. Syst. Comput.*, vol. 174 AISC, no. June 2014, pp. 277–284, 2013, doi: 10.1007/978-81-322-0740-5_33.

[9] M. Lafi, T. Alrawashed, and A. M. Hammad, "Automated Test Cases Generation from Requirements Specification," *2021 Int. Conf. Inf. Technol. ICIT 2021 - Proc.*, pp. 852–857, 2021, doi: 10.1109/ICIT52682.2021.9491761.

[10] Meiliana, I. Septian, R. S. Alianto, Daniel, and F. L. Gaol, "Automated Test Case Generation from UML Activity Diagram and Sequence Diagram using Depth First Search Algorithm," *Procedia Comput. Sci.*, vol. 116, pp. 629–637, 2017, doi: 10.1016/j.procs.2017.10.029.

[11] M. H. J. Thanki and P. S. . Shinde, "Test Case Generation and Minimization using UML Activity Diagram in Model Driven Environment," *Int. J. Comput. Orgnanization Trends*, vol. 9, no. 1, pp. 41–44, 2014, doi: 10.14445/22492593/ijcot-v9p309.

[12] W. Rhmann and V. Saxena, "Test Case Generation from UML Sequence Diagram for Aadhaar Card Number based ATM System," *Int. J. Appl. Inf. Syst.*, vol. 11, no. 4, pp. 37–43, 2016, doi: 10.5120/ijais2016451599.

[13] A. Tripathy and A. Mitra, "Test case generation using activity diagram and sequence diagram," *Adv. Intell. Syst. Comput.*, vol. 174 AISC, pp. 121–129, 2013, doi: 10.1007/978-81-322-0740-5_16.

[14] M. Dhineshkumar and P. G. Scholar, "An Approach to Generate Test Cases from Sequence Diagram," pp. 7–11, 2014, doi: 10.1109/ICICA.2014.77.

[15] G. Kaur, "Test Case Generation Using UML Diagram," *Int. J. Emerg. Technol. Eng. Res.*, vol. 1, no. 2, pp. 23–25, 2015, [Online]. Available: www.ijeter.everscience.org.

[16] M. Khandai, "A Novel Approach of Test Case Generation for Concurrent Systems Using UML Sequence Diagram," pp. 157–161.

[17] S. S. Panigrahi, P. K. Sahoo, B. P. Sahu, A. Panigrahi, and A. K. Jena, "Model-driven automatic paths generation and test case optimization using hybrid FA-BC," *2021 Int. Conf. Emerg. Smart Comput. Informatics, ESCI 2021*, pp. 263–268, 2021, doi: 10.1109/ESCI50559.2021.9396999.

[18] S. Jagtap, V. Gawade, R. Pawar, S. Shendge, and P. Avhad, "Generate Test Cases From UML Use Case and State Chart Diagrams," *Int. Res. J. Eng. Technol.*, vol. 3, no. 10, pp. 873–881, 2016, [Online]. Available: www.irjet.net.

[19] Z. Shi, X. Zeng, T. Zhang, L. Han, and Y. Qian, "UML diagram-driven test scenarios generation based on the temporal graph grammar," *KSII Trans. Internet Inf. Syst.*, vol. 15, no. 7, pp. 2476–2495, 2021, doi: 10.3837/tiis.2021.07.010.

[20] R. Shetty, "Generation of Test Cases for Object Oriented Software using UML State Machine Diagram," *Int. J. Innov. Eng. Technol.*, vol. 8, no. 2, pp. 142–148, 2017, doi: 10.21172/ijiet.82.020.

[21] A. Monim, R. Nor, and H. Nor, "An Automated Test Case Generating Tool Using UML Activity Diagram," *Int. J. Eng. Technol.*, vol. 7, pp. 58–63, 2018, [Online]. Available: www.sciencepubco.com/index.php/IJET.

[22] S. K. Swain, D. P. Mohapatra, and R. Mall, "Test Case Generation Based on Use case and Sequence Diagram," *Int. J. Softw. Eng.*, no. JANUARY, pp. 21–52, 2010.

[23] N. Khurana and R. S. Chillar, "Test Case Generation and Optimization using UML Models and Genetic Algorithm," *Procedia Comput. Sci.*, vol. 57, pp. 996–1004, 2015, doi: 10.1016/j.procs.2015.07.502.

[24] T. Y. K. and S. H. LEE, "Combustion and Emission Characteristics of Wood Pyrolysis Oil-Butanol Blended Fuels in a Di Diesel Engine," *Int. J. …*, vol. 13, no. 2, pp. 293–300, 2012, doi: 10.1007/s12239.

[25] P. E. Patel, "Testcases Formation using UML Activity Diagram," pp. 884–889, 2013, doi: 10.1109/CSNT.2013.191.

[26] A. C. D. Iagrams, F. O. R. Uml, and B. A. T. Esting, "Jurnal Teknologi A UTOMATIC G ENERATION OF T EST C ASES FROM," vol. 13, pp. 37–48, 2015.

[27] Y. Yin, Y. Xu, W. Miao, and Y. Chen, "An Automated Test Case Generation Approach based on Activity Diagrams of SysML," vol. 13, no. 6, pp. 922–936, 2017, doi: 10.23940/ijpe.17.06.p13.922936.

[28] A. Kaur and V. Vig, "Automatic test case generation through collaboration diagram : a case study," *Int. J. Syst. Assur. Eng. Manag.*, 2017, doi: 10.1007/s13198-017-0675-8.

[29] P. Kaur and R. Kaur, "Approaches for Generating Test Cases Automatically to Test the Software," *Int. J. Eng. Adv. Technol.*, no. 3, pp. 2249–8958, 2013.

[30] M. Lusiana, C. Dewi, and A. Chandra, "Optimization of test case generation from uml Activity diagram and sequence diagram By using genetic algorithm," *ICIC Express Lett.*, vol. 13, no. 7, pp. 585–591, 2019, doi: 10.24507/icicel.13.07.585.

[31] V. M. Sumalatha, "An Model Based Test Case Generation Technique Using Genetic Algorithms," *Int. J. Comput. Sci. Appl.*, pp. 46–57, 2009, [Online]. Available: http://www.journalofcomputerscience.com/2012Issue/November12/V1No9Nov12P008.pdf.

[32] W. Rhmann, T. Zaidi, and V. Saxena, "Use of Genetic Approach for Test Case Prioritization from UML Activity Diagram," *Int. J. Comput. Appl.*, vol. 115, no. 4, pp. 8–12, 2015, doi: 10.5120/20137-2232.

[33] T. A. Alrawashed, A. Almomani, A. Althunibat, and A. Tamimi, "An automated approach to generate test cases from use case description model," *C. - Comput. Model. Eng. Sci.*, vol. 119, no. 3, pp. 409–425, 2019, doi: 10.32604/cmes.2019.04681.

[34] A. Jaffari, C. J. Yoo, and J. Lee, "Automatic test data generation using the activity diagram and search-based technique," *Appl. Sci.*, vol. 10, no. 10, pp. 9–13, 2020, doi: 10.3390/APP10103397.

[35] S. S. Basa, S. K. Swain, D. P. Mohapatra, C. Science, C. Science, and C. Science, "UML ACTIVITY DIAGRAM-BASED TEST CASE," vol. 5, no. 8, pp. 834–844, 2018.

[36] M. Shirole and R. Kumar, "A hybrid genetic algorithm based test case generation using sequence diagrams," *Commun. Comput. Inf. Sci.*, vol. 94 CCIS, no. PART 1, pp. 53–63, 2010, doi: 10.1007/978-3-642-14834-7_6.

[37] A. K. Jena and S. K. Swain, "Test Case Creation from UML Sequence Diagram : A Soft Computing Approach," 2012, doi: 10.1007/978-81-322-2012-1.

[38] P. Mahali, "Model based test case prioritization using UML behavioural diagrams and association rule mining," *Int. J. Syst. Assur. Eng. Manag.*, 2018, doi: 10.1007/s13198-018-0736-7.

[39] R. K. Sahoo, M. Derbali, H. Jerbi, D. van Thang, P. P. Kumar, and S. Sahoo, "Test Case Generation from UML-Diagrams Using Genetic Algorithm," *Comput. Mater. Contin.*, vol. 67, no. 2, pp. 2321–2336, 2021, doi: 10.32604/cmc.2021.013014.

[40] S. Shah, R. Shahzad, S. Bukhari, and M. Humayun, "Automated Test Case Generation Using UML Class & Sequence Diagram," *Br. J. Appl. Sci. Technol.*, vol. 15, no. 3, pp. 1–12, 2016, doi: 10.9734/bjast/2016/24860.

[41] I. J. I. Systems, S. K. Nanda, D. P. Mohapatra, and M. R. Patra, "Model Driven Test Case Optimization of UML Combinational Diagrams Using Hybrid Bee Colony Algorithm," no. June, pp. 43–54, 2017, doi: 10.5815/ijisa.2017.06.05.

[42] M. Panda, S. Dash, A. Nayyar, M. Bilal, and R. M. Mehmood, "Test suit generation for object oriented programs: A hybrid firefly and differential evolution approach," *IEEE Access*, vol. 8, pp. 179167–179188, 2020, doi: 10.1109/ACCESS.2020.3026911.

[43] M. Rocha, A. Simão, and T. Sousa, *Model-based test case generation from UML sequence diagrams using extended finite state machines*, vol. 29, no. 3. Springer US, 2021.

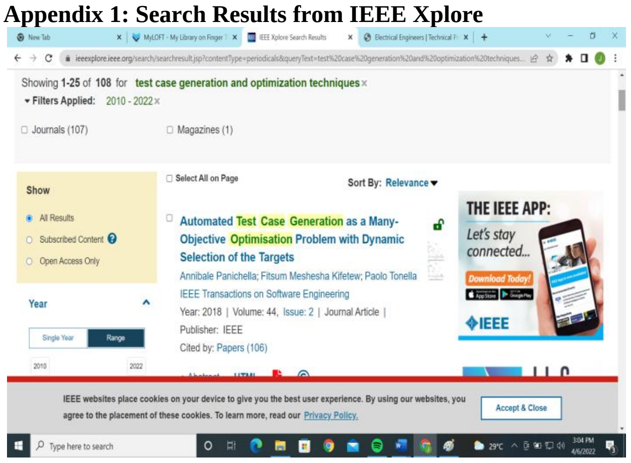
[44] S. U. Ahmed, S. A. Sahare, and A. Ahmed, "Automatic test case generation using collaboration UML diagrams," *World J. Sci. Technol.*, vol. 2, no. x, 2012.

[45] N. Panda, A. A. Acharya, and D. P. Mohapatra, "Test scenario prioritization for object-oriented systems using UML diagram," *Int. J. Syst. Assur. Eng. Manag.*, 2019, doi: 10.1007/s13198-019-00759-z.

[46] J. K. Mandal, S. C. Satapathy, M. K. Sanyal, P. P. Sarkar, and A. Mukhopadhyay, "Information systems design and intelligent applications: Proceedings of second international conference India 2015, volume 1," *Adv. Intell. Syst. Comput.*, vol. 339, 2015, doi: 10.1007/978-81-322-2250-7.

7. APPENDIX

Appendix 1: Search Results from IEEE Xplore


New Tab | MyLOFT - My Library on Finger T | IEEE Xplore Search Results | Electrical Engineers | Technical F

ieeexplore.ieee.org/search/searchresult.jsp?contentType=periodicals&queryText=test%20case%20generation%20and%20optimization%20techniques...

Showing 1-25 of 108 for test case generation and optimization techniques ×
Filters Applied: 2010 - 2022 ×

Journals (107) Magazines (1)

Show
All Results
Subscribed Content
Open Access Only

Year
Single Year | Range
2010 2022

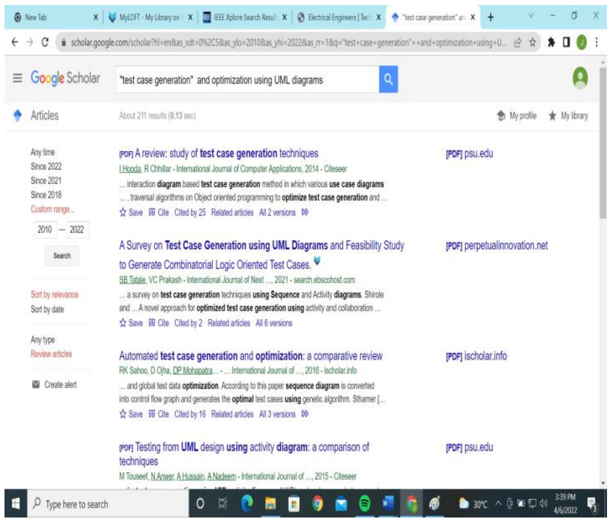
Select All on Page
Sort By: Relevance

Automated Test Case Generation as a Many-Objective Optimisation Problem with Dynamic Selection of the Targets
Annibale Panichella; Fitsum Meshesha Kifetew; Paolo Tonella
IEEE Transactions on Software Engineering
Year: 2018 | Volume: 44, Issue: 2 | Journal Article |
Publisher: IEEE
Cited by: Papers (106)

IEEE websites place cookies on your device to give you the best user experience. By using our websites, you agree to the placement of these cookies. To learn more, read our Privacy Policy.
Accept & Close

Type here to search 29°C 3:04 PM 4/6/2022

Appendix 2: Search Results from Google Scholar



"test case generation" and optimization using UML diagrams

About 211 results (0.13 sec)

[PDF] A review: study of **test case generation** techniques — [PDF] psu.edu
I Hooda, R Chhillar - International Journal of Computer Applications, 2014 - Citeseer
… interaction **diagram** based **test case generation** method in which various **use case diagrams** … , traversal algorithms on Object oriented programming to **optimize test case generation** and …
Save Cite Cited by 25 Related articles All 2 versions

A Survey on **Test Case Generation using UML Diagrams** and Feasibility Study to Generate Combinatorial Logic Oriented Test Cases. — [PDF] perpetualinnovation.net
SB Tatale, VC Prakash - International Journal of Next …, 2021 - search.ebscohost.com
… a survey on **test case generation** techniques **using Sequence** and Activity **diagrams**. Shirole and … A novel approach for **optimized test case generation using** activity and collaboration …
Save Cite Cited by 2 Related articles All 6 versions

Automated **test case generation** and **optimization**: a comparative review — [PDF] ischolar.info
RK Sahoo, D Ojha, DP Mohapatra… - … International Journal of …, 2016 - ischolar.info
… and global test data **optimization**. According to this paper **sequence diagram** is converted into control flow graph and generates the **optimal** test cases **using** genetic algorithm. Sthamer [...
Save Cite Cited by 16 Related articles All 3 versions

[PDF] Testing from **UML** design **using** activity **diagram**: a comparison of techniques — [PDF] psu.edu
M Touseef, N Anwer, A Hussain, A Nadeem - International Journal of …, 2015 - Citeseer