International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 12, Number 7 (2019), pp. 1138-1145
© International Research Publication House. http://www.irphouse.com

An Efficient Method to Generate Test Cases From UML - USE CASE DIAGRAM

Eglal M. Khalifa
University Technology Malaysia
Johor Bahru, Malaysia.

DNA Jawawi
University Technology Malaysia
Johor Bahru, Malaysia.

Haitham A. Jamil
University of Elimam Elmahdi
Kosti, White Nile, Sudan.

Safa'ai bin Deris
Universiti Malaysia Kelantan
16100 Kota Bharu. Kelantan.

Abstract

Regression testing is a process to execute a set of test cases to confirm that the performance of the software is not changed after a modification. A test case is a group of conditions and methods to verify the functionality of the software. A better test case can improve the performance of overall testing process. The manual process of generating test cases will take more time and affect the cost of testing. The available automated tools are simply executing test cases at random or depend on the user commands. The aim of the study is to generate test cases from use case diagram using a machine learning method. A metaheuristic technique is used for the automation of the process of generating test cases. The accuracy and computation time are the metrics used to evaluate the performance of the proposed method. The output of the research has shown that the performance of the proposed technique is better than existing techniques.

Keywords: Component: Metaheuristic; Test suite prioritization; Artificial Intelligence ; Test case ; Regression testing

I. INTRODUCTION

Software testing is an important phase in Software Development Life Cycle (SDLC). It is used to find faults or errors in Software[1]. The quality and performance of software can be improved through testing. Regression testing (RT) is a part of testing, which is to ensure the stability of existing functionality of an application[2][3]. The nature of RT leads to the consumption of more computation time and resources. Modern programming languages are providing flexible environment for programmers, but not for testers. Possibilities of newer defects are higher in modern programming environment[4]. The number of lines in codes are directly proportional to the number of faults; so RT is in the situation to retest whole code of software. Retest all, test selection, and test case prioritization(TCP) are the types of RT. Many tools are available to perform RT, but may create problem in software development environment. Testing team has to study the whole system before performing RT. TCP will reinforce RT to detect errors and faults in newly updated software. It will help tester to organize TC in an order according to their functionality. The high priority TC will be executed to reduce time and cost[5][6]. TC is a set of action or condition to test a system.

Unified Modelling Language (UML) is widely used in software industry to specify the key components of software. Use case diagram (UCD) is a part of UML, to represent the user interaction with the system. It is a set of actions and use cases [7].

A. Types of TC

Generally,TCs are broadly classified into effective and non effective. The following part of this section will discuss about Tcs and types of methods to generate Tcs.

1. Effective TC: A known input and expected output will be used to design an effective TC. A precondition will be applied using a known input and a post condition needs to be tested using an expected output. The code is known to work in efficient manner if the output achieved is similar to the expected output. Otherwise, there is a need of modification of the code. Positive and negative test are the two minimal TC of each requirement, which are necessary for testing all the requirements of a complete application[7][8]. The positive test is to ensure the proper functioning of software. Further, the condition in which the software does not work efficiently as per the expectations is known as negative test.

2. Non – Effective TC: There is no formal way in which the TC can be written but once the tests have been run, it is important to report the results [14]. For helping a tester to provide a solution for a complex issue, hypothetical stories are utilized generally. There is generally, no detailed description related to these scenarios. Just like a diagram within a testing scenario or a written description, these solutions can be provided. In applications which do not require any formal requirements, the non - effective TC is applied[9][10]. On the basis of normal operation of programs of the same class, it is possible to develop the TC.

RT can use partial or complete set of TCs to verify the functionality of software. Generally, the processes of generating TCs are classified into two types. The types are manual and automatic. The following figure 1.1 represents the types of methods to generate TC.

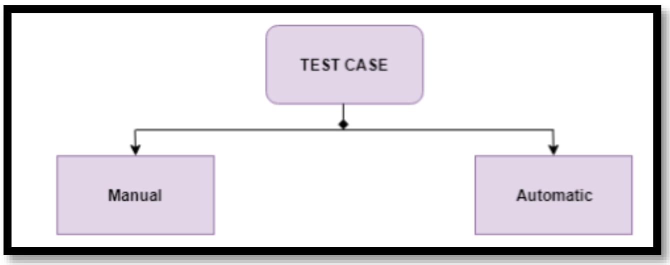


Fig 1.1 Types of Test case generation

The manual process of generating TC will take more time. The accuracy level of manual processing is high comparing to automated methods[11][12]. The computing cost of manual processing will cause more problems to software development environment. Earlier researches [13] have proved that manual generation of TCs are not effective for RT.

The automated generation of TC has showed some improvement over manual generation. A number of tools are available for the generation of TC for testing software[13]. There is no clear evidence to show the efficiency of automated tools. Artificial intelligence based automated tools are required for the process of generation of TC. Many research works are in progress to improve the efficiency of automated tools.

B. Generating TC using Metaheuristics

Metaheuristics (Mh) is a problem independent technique, work like black boxes. It is not a greedy approach ,ability to obtain global optimum solution. It can be easily adapted to solve any kind of complex problems. A set of guidelines and strategies are developed for Mh based optimization problem. Some researchers were misunderstand the concept of Mh for only vague problems. It can be used as problem – specific by fine tuning its intrinsic parameters[14]. Some researches[14][15] have proved that Mh can be successfully implemented in field of software testing. The optimization algorithms such as constraint programming, mathematical programming, and machine learning can be combined with Mh to solve a problem[15]. The proposed method has further extended the concept of Mh to generate TC from UCD.

C. Generation of TC

TCs are used to discover faults in the system. The newly updated systems are having the possibilities of producing errors, which are not detected easily. Mh techniques can be implemented for the process of generating TCs. The technique can produce an effective system to develop TCs to find maximum faults in software. The manual process are depending on the complete code. The testing team will explore the complete code of a system to develop TC for RT. Mh technique does not need to investigate code to generate TC. It can generate TCs from UCD. UCDs are used to have complete details of software.

D. Research Questions

The objective of the research is to frame a method to generate TCs from UCD. The following Research Questions (RQ) will be answered by the proposed study.

RQ1:- How to automate the process of generating TC?

RQ2:- How to classify TCs?

The proposed study has utilized Mh to generate TC from UCD. The paper is structured as follows, The section 2 will provide details about existing literatures on TC. The section 3 will describe the methods and materials used in the research. Results and discussions are provided in the section 4. Finally, the section 5 will conclude the research.

II. REVIEW OF LITERATURE

The section will discuss the existing literatures in TC , TCP and ML methods. The focus of the proposed method is on TC, but TCP will give more information about TC. Information retrieval techniques and machine learning methods are becoming popular in the field of software testing. If TC is not effective then the detection of fault is not possible for the tester. A minor fault can cause an application to produce a wrong result, which may lead to a huge failure. The proposed study has used keywords such as “Test case”, Test case prioritization”, “Metaheuristics”, “Machine learning”, and “Test Case Generation. A systematic approach was followed in the collection of literatures. Google scholar was used to search the literatures. Year, Indexed portals, and Citation counts were the criteria applied to filter the literatures. IEEE Explore and Springer were major indexed portals to download the literatures.

Shweta mittal and Om prakash sangwan[16] have used Mh technique for the prioritization of test cases. Authors have used Artificial Bee Colony (ABC) to prioritize the test cases. Test suite prioritizations are useful to sort the test cases and perform execution according to the priority basis. RT selection, minimization, and prioritization are the different techniques used to minimize the computation time. The research has used Genetic Algorithm (GA) for ordering the test cases. Representation, operator, and fitness functions are required to implement GA. Array was used to represent the test suites. Ordered crossover, mutation, and selection were the operators used in the research. Cuscuta search was used to find the maximum number of faults with maximum probability. The minimum execution time of test cases of having same probability was chosen for each Cuscuta. The execution time was calculated according to left over starvation. The experimental results had shown that the performance of Cuscuta and GA were low comparing to ABC algorithm. The research was conducted with less number of test cases. The output of research will be varied with more number of test cases.

Arjinder singh and Sumit Sharma[17] have proposed a method to generate functional TCs from UCD. They have developed a complete user based input UCD (UBUCD) to extract TC from use cases. The initial part of the method was used to analyze the methods based on the extraction of TCs from UCD. After the development of UCD, activity diagrams were derived from UCD. The process of parsing was used to convert the activity diagram into XML file. A table called use case activity development table used to generate activity graph. A traversing technique based on depth first search was used for the generation of functional TCs. A dedicated setup was used to update the UCD after a modification in code. Authors have argued that the study has covered a maximum part of software.

Wang Linzhang et.al.[18], have developed a method to generate TCs from UCD. Gray box method was used to extract TCs from UCD. The manual process of developing TCs will consume more time and prone to errors. The automated testing methods will handle the different phases of RT. It will generate TCs, execute automatic test without human intervention and evaluate test with relevant metrics. UML diagrams are widely used to analyse and design the requirements of software. Activity diagram is used to convey the activity with more details. The practical approaches and tools for deriving TCs from UML diagram are less in number. Nodes and edges of activity diagram are used to represent processes and sequence of activities. Authors have gray box method for the extraction of TCs. The method is widely used on high level design models. Normally, UML diagram designs are used to define requirement specifications and final code. Gray box method is the extended version of white box testing. Authors have used a coverage criterion to cover the maximum part of code. The activity diagram was parsed by gray box method to derive TCs from the diagram. The derived TCs are verified with coverage criteria.

Fernando Augusto Diniz Teixeria and Glaucia Braga e – Silva[19] have developed an approach to generate TCs from UML activity diagram. Existing automated tools have reduced the computation time but did not produce optimum accuracy in generation of TCs. Model – based testing has produced better results than code based approach. UML models were used in model based testing to derive system specification for TC design. Authors have proposed a method, named as EasyTest to generate TC from activity diagram. The approach has suggested some detection technique to prevent the unidentified defect in software. Authors have argued that activity diagram could be used to produce defect free code. EasyTest has two phases such as importing activity diagram as XML file and TC generation. The first phase is used to derive activity diagram details into a XML file. The XML files are lightweight and easily parse by any algorithm. EasyTest has the advantage of interoperability with different types of UML modelling tools. The strategy applied by EasyTest was cheaper in terms of computing time. The second phase consists of activity depending table, activity depending graph, and test path generation. Finally, TCs were applied in real time applications.

Noraida Ismail et.al.[20], have developed a method to automate the process of generation of TCs from UCD. A small delay in the software testing may maximize the number of errors. A UCD will specify the requirements of a system. The use case is used to represent the functionalities of the system. Each use case represents a small task or a function in the system. The manual conversion of use cases into a test case will take more time. Generally, use cases are developed according to the user requirement and TCs are designed by tester. A better UCD can provide meaningful information about software. A tool, Generator for Test Cases (GentTcase) was proposed to carry out UCD of a system. It has the ability to generate TCs from UCD. The tool will process all use cases using a user defined keywords. According to the keyword, TCs are generated from the use cases. A list of meta data are used to locate the use cases. The search algorithm will apply a heuristic to retrieve all TCs from the database.

Based on the literatures, the proposed study had chosen UBUCD, EasyTest, and GenTcase for the generation of TC. Theses methods are state – of – the art algorithms for the extraction of TC.The following Table 1.1 shows the details of existing methods.

Table 1.1 Selected methods for the comparison

Method	**Authors**	**Year**
GenTcase	Noraida Ismail, Rosziati Ibrahim, and Noraini Ibrahim	2007
UBUCD	Arjinder singh and Sumit sharma	2015
EasyTest	Fernando Augusto Diniz Teixeria and Glaucia Braga e Silva	2017

Limitations in existing system

1. The manual process of generation of TCs from UCD had consumed more computation time[17].

2. The number of errors and complexity was more in manual generation[17][18].

3. The performance of automated tools is not sufficient for modern programming concepts[19][20].

III. METHODS AND MATERIALS

TC is a collection of methods to find a fault in an application. Test scenarios are a collection of TCs. The proposed study has employed Mh to generate TC from UCD[18][19]. UCD is used to understand the flow of data in an application[20][21]. Let D be the UCD, A be a activity diagram, G is a activity graph, T is a token. The following expression denotes the conversion of UCD to activity graph.

$$D \rightarrow A(D) \qquad (1)$$

$$A(D) \rightarrow G[A(D)] \qquad (2)$$

G[A(D)] is an activity graph. The activity graph will be divided into tokens, which is readable for the algorithm.

$$G[A(D)] \rightarrow T(TC) \qquad (3)$$

T(TC) will be classified using bag of words technique. True positive (Tp), True negative(Tn), False positive(Fp), and False negative(Fn) are grouped into effective and non – effective Tcs. The failures will be stored as separate entities. The clusters will become key points. The key points will be clustered using Library and distance. A constant value 4 is used with the distance for the preparation of clusters. The constant value is used to adjust the P – Mh intrinsic parameters.

The following procedures are used in the research.

Procedure_Generate TC

Step 1: Input UCD of an application

Step 2: Convert UCD into activity diagram

Step 3: Convert Activity diagram into activity graph

Step 4: Collect possible keywords and store as a library to classify primary TCs

Step 5: Extract TCs as tokens from Activity graph

Step 6: Store tokens as a XML file

Step 7: Tokens are classified into TP,TN,FP, and FN.

Step 8: Majors clusters are formed as effective, non – effective and failures.

Step 9: Clusters are further clustered using library and distance with constant value 4.

Step 10: Final, TC clustered formed successfully.

Step 11: End

A. *Concept of Library*

The concept of library is used for inter clustering Mh technique. The known Tcs are stored in the library and used to identify useful Tcs. The bags of words are followed to store the key points into library. Figure 3.1 shows the concept of library.

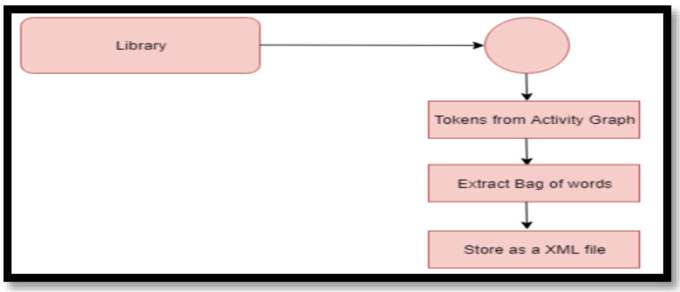


Figure 3.1 Concept of Library

B. *Collection of Dataset*

P – Mh, GenTcase, EasyTest, and UBUCD are the methods employed in the research. UCD is the main source for the algorithms. The study has collected all UCD with successful Tcs for the purpose of evaluation of algorithms.

Table 3.1 Details of UCD

Datasets	Total TC	Failures
D1	2601	712
D2	3250	845
D3	10427	1280
D4	15165	1530

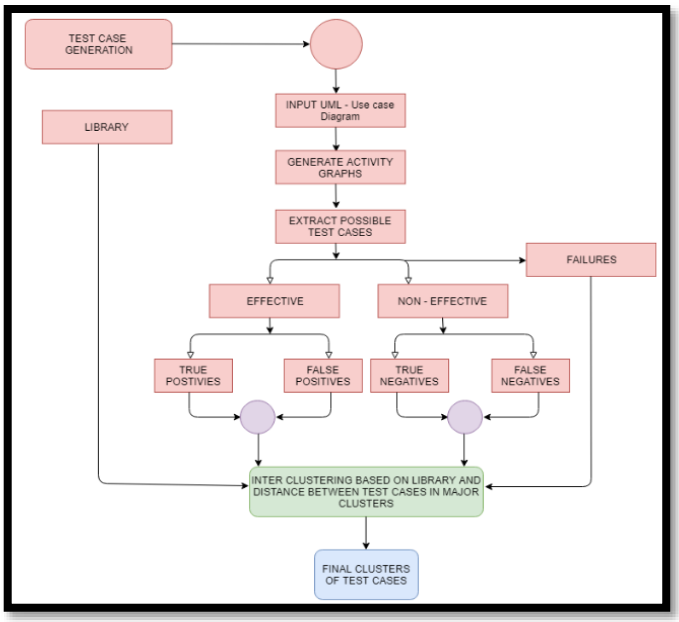

TEST CASE GENERATION
LIBRARY
INPUT UML - Use case Diagram
GENERATE ACTIVITY GRAPHS
EXTRACT POSSIBLE TEST CASES
FAILURES
EFFECTIVE
NON - EFFECTIVE
TRUE POSTIVIES
FALSE POSITIVES
TRUE NEGATIVES
FALSE NEGATIVES
INTER CLUSTERING BASED ON LIBRARY AND DISTANCE BETWEEN TEST CASES IN MAJOR CLUSTERS
FINAL CLUSTERS OF TEST CASES

Figure 3.2 Clustering process of Tcs

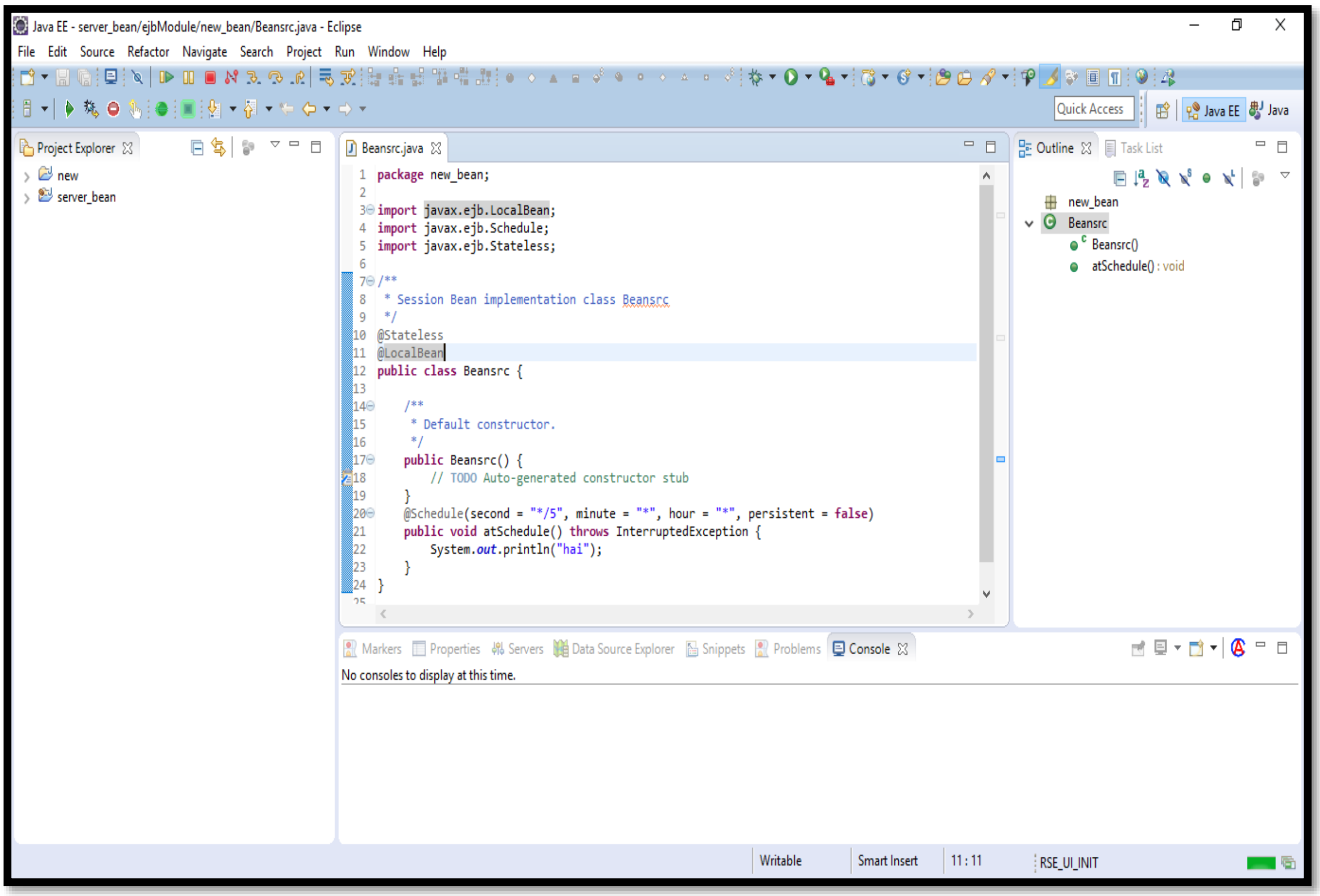
Table 3.1 shows the details of collected UCD. D1 represents UCD on online marketing application. D2 represents a mobile application, which uses to know the capital market details. D3 is the communication system in University. D4 represents banking software. Datasets were from familiar and reputed organization. Tcs were collected from their testing team. Tcs mentioned in Table 3.1 have produced faults in the system and rectified by testing team. Failures indicate that the Tc, which have failure results from testing. Figure 3.2 illustrates the process of generating Tcs from UCD.

```
Algorithm_Generate_TC
    Input:
    UCD: DF
    Library: Lb
    Output: TCs
    Start Algorithm _Generate TC
    Tp : True Positive
    Fp: False Positive
    Tn: True Negative
    Fn: False Negative
    i = 1 to No. of elements
    for each DF in DF(i)
      for j = 1 to n
      if DF == Lb(Tp)
      Tp = Tp + 1
    else if DF == Lb(Fp)
      Fp = Fp + 1
    else if DF == Lb(Tn)
```

```
  Tn = Tn + 1
else if DF == Lb(Fn)
  Fn = Fn + 1
  else
   F = F + 1
  end if
  end for
end for
cluster1 = {Tp, Fp}
cluster2 = {Tn,Fn}
cluster3 = {F}
mdist = 4
for each clusters
calculate dist between smallest point in clusters
if dist <= mdist
form TC clusters
else
Leave as single cluster
Endif
End algorithm
```

C. Implementation of Algorithm

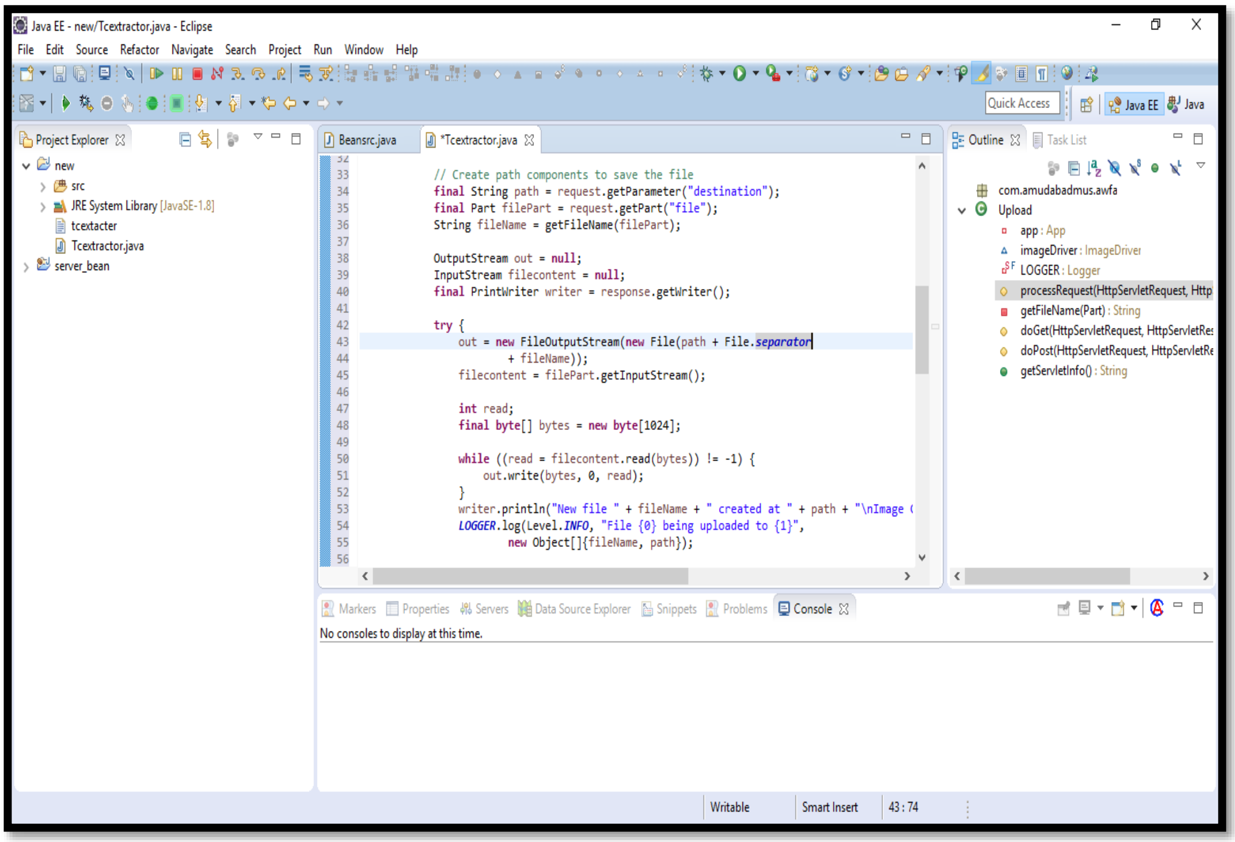
The proposed study was developed in Windows 10 professional – 64 bit operating system. Intel i7, 4th generation processor with 8 GB RAM hardware configuration was used in the development. JULIA 0.7 with Eclipse IDE was used to develop P- Mh algorithm. The reason for using JULIA 0.7 was its generic nature. The codes of UBUCD, GenTcase, and EasyTest were transferred to JULIA platform. P – Mh is a platform independent, difficult to implement for the generation of TC. The intrinsic parameters of P – Mh were altered to have inputs such as population of TC, image boundaries, and features. The population of TC was used to represent the total number of TCs. Image boundaries were used to restrict the key points of activity graph and features values for the representation of storage to store TCs. Figure 3.3 shows the scheduling process for extraction of key points from activity graph. Figure 3.4 shows the P – Mh code in Eclipse IDE. The code in Figure 3.4 indicates the extraction of texts from image.



```java
package new_bean;

import javax.ejb.LocalBean;
import javax.ejb.Schedule;
import javax.ejb.Stateless;

/**
 * Session Bean implementation class Beansrc
 */
@Stateless
@LocalBean
public class Beansrc {

    /**
     * Default constructor.
     */
    public Beansrc() {
        // TODO Auto-generated constructor stub
    }
    @Schedule(second = "*/5", minute = "*", hour = "*", persistent = false)
    public void atSchedule() throws InterruptedException {
        System.out.println("hai");
    }
}
```

```
No consoles to display at this time.
```

Figure 3.3 Scheduling of Bean to store key points into library


Java EE - new/Tcextractor.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Project Explorer
new
src
JRE System Library [JavaSE-1.8]
tcextacter
Tcextractor.java
server_bean
Beansrc.java
*Tcextractor.java
Outline
Task List
com.amudabadmus.awfa
Upload
app : App
imageDriver : ImageDriver
LOGGER : Logger
processRequest(HttpServletRequest, Http
getFileName(Part) : String
doGet(HttpServletRequest, HttpServletRes
doPost(HttpServletRequest, HttpServletRe
getServletInfo() : String
Markers
Properties
Servers
Data Source Explorer
Snippets
Problems
Console
Writable
Smart Insert
43 : 74

```java
        // Create path components to save the file
        final String path = request.getParameter("destination");
        final Part filePart = request.getPart("file");
        String fileName = getFileName(filePart);

        OutputStream out = null;
        InputStream filecontent = null;
        final PrintWriter writer = response.getWriter();

        try {
            out = new FileOutputStream(new File(path + File.separator
                    + fileName));
            filecontent = filePart.getInputStream();

            int read;
            final byte[] bytes = new byte[1024];

            while ((read = filecontent.read(bytes)) != -1) {
                out.write(bytes, 0, read);
            }
            writer.println("New file " + fileName + " created at " + path + "\nImage (
            LOGGER.log(Level.INFO, "File {0} being uploaded to {1}",
                    new Object[]{fileName, path});
```

```
No consoles to display at this time.
```

Figure 3.4 Implementation of P – Mh in JULIA – Eclipse IDE

IV. RESULTS AND DISCUSSIONS

The section discusses the details of results that are generated from the proposed research. Table 4.1 shows the details about effective and non – effective Tcs and no. of features extracted from UCD. The reason for the extraction of features is for the process of prioritization of Tcs. The existing methods were used only for the extraction of effective and non – effective Tcs. The details of accuracy are discussed in the later part of the section.

Table 4.1 Details of extracted features from P - Mh

Datasets	Total TC	Effective		Non – Effective		No. of Features
		TP	FP	TN	FN	
D1	2601	553	662	215	459	468
D2	3250	638	698	430	639	635
D3	10427	1645	2465	1358	3679	1836
D4	15165	3698	4599	2469	2869	3254

Table 4.2 shows the time consumed by each method during training phase. The training phase will teach methods to learn the environment. P – Mh is a slow learner, require more time to learn to generate Tcs from UCD. It took a maximum of 0.903 seconds for D4. GenTcase has consumed less time during the learning process. Figure 4.1 represents the related values of table 4.2.The graph has clearly shown that P- Mh has consumed more time than other methods.

Table 4.2 Training Time (in seconds)

Methods / Datasets	D1	D2	D3	D4
UBUCD	0.874	0.916	0.904	0.872
P-Mh	0.619	0.815	0.856	0.903
EasyTest	0.589	0.942	0.935	0.869
GenTcase	0.634	0.809	0.846	0.807

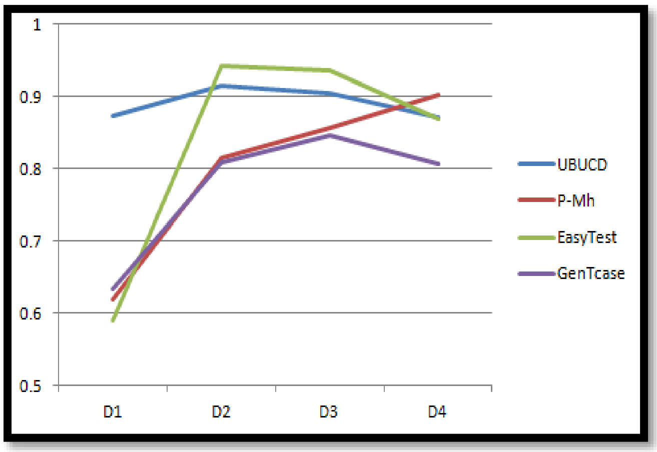


Figure 4.1 Training time (in seconds)

Table 4.3 shows the time taken by the methods during testing phase. Datasets were divided into two unique sets. The testing time shown that UBUCD has taken more time than other methods. The computation time of P – Mh is average comparing to other methods. P – Mh has taken more time for the generation of TC because of its adaptability to this specific problem. The adjustment in parameters had made the algorithm slow in the production of results. Figure 4.2 is the graphical view of table 4.3.

Table 4.3 Testing Time (in seconds)

Methods/ Datasets	D1	D2	D3	D4
UBUCD	0.348	0.304	0.394	0.415
P-Mh	0.226	0.264	0.389	0.364
EasyTest	0.257	0.289	0.378	0.349
GenTcase	0.284	0.319	0.406	0.395

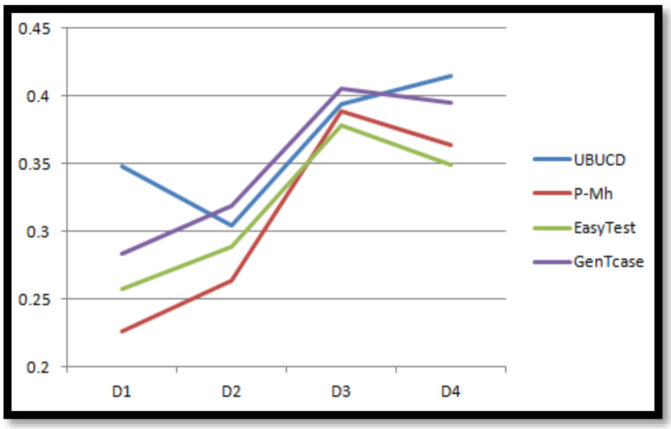


Figure 4.2 Testing time (in seconds)

Table 4.4 shows the details of accuracy of each method. P – Mh has achieved better accuracy than other methods. It has scored a maximum accuracy of 91.3 % in D4 and minimum of 84.6% in D3. The accuracy level of EasyTest is average than other methods. It has scored an minimum of 80.3% in D3 and maximum of 86.4 in D1. Figure 4.3 represents the accuracy of methods related to table 4.4. P – Mh has shown a better accuracy in generation of TCs.

Table 4.4 Details of Accuracy

Methods / Datasets	D1 (%)	D2 (%)	D3 (%)	D4 (%)
UBUCD	75.6	81.3	83.2	85.6
P-Mh	89.3	84.6	89.6	91.3
EasyTest	83.4	82.5	80.3	83.7
GenTcase	86.8	79.5	79.4	86.5

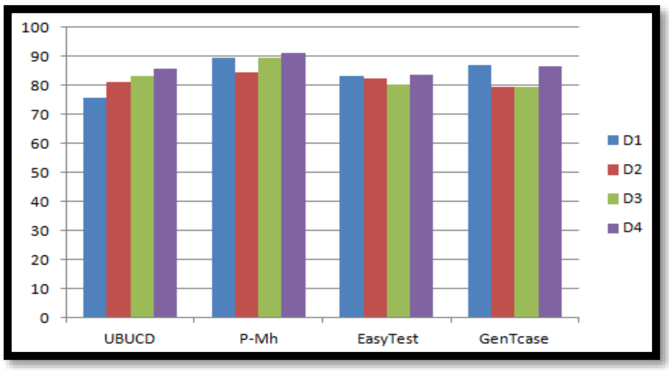


Figure 4.3 Accuracy

V. CONCLUSION

The process of generating test cases is complex and difficult. The accuracy level was slow with existing methods. Prioritizations of test cases are fully dependent on test cases. A bad test case may create a huge impact on software. The performance of the application can be improved through regression testing.

The proposed method is based on metaheuristic, a machine learning method. The state of the art methods such as UBUCD, GenTcase, and EasyTest were employed and compared with the efficiency of proposed metaheuristic. The trade-off such as accuracy and time were measured for each method. The proposed method has achieved an average of 90% accuracy for all datasets. The computation time of the proposed method is more compared to the other methods. The prime focus of the research is on finding maximum faults from the system. The accuracy is the most important trade-off compared to the computation time.

The achievements of the proposed research as follows:

1. Achieved better accuracy than state of the art techniques.
2. Achieved an average computation time to generate results.

The future scope of the research is the development of a multi-objective machine learning method to prioritize the test cases.

REFERENCES

[1] Wayne Jansen, GranceTimothy,"Guidelines on Security and Privacy in Public Cloud Computing",National Institute of Standards and Technology, 2011.

[2] Mollah Muhammad Baqer, Islam KaziReazul, Islam Sikder Sunbeam," Next Generation of Computing through Cloud Computing Technology" In: Proc. Of 25th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), 2012

[3] Tianfield Huaglory,"Security issues in Cloud Computing", In : Proc of IEEE International Conference on Systems, Man, and Cybernetics, Korea, 2012

[4] Gonzalez Nelson , Miers Charles , Redígolo Fernando , CarvalhoTereza , Simplicio Marcos," A quantitative analysis of current security concerns and solutions for cloud computing", Journal of Cloud Computing: Advances, Systems and Applications Springer, 2012

[5] Zissis Dimitrios ,LekkasDimitrios," Addressing cloud computing security issues", SciVerse ScienceDirect,2012.

[6] Buyya,"Cloud Computing and emerging IT platforms: vision, hype, and relatity for deliverling computing as the 5th utility", Future Generation Computer System 25(6), 599–616, 2009.

[7] Santhosh kumar Swain, Durga Prasad Mohapatra, Rajib Mall, " Test case generation based on use case and sequence diagram", International journal of software engineering,Vol. 3, No.2, July 2010.

[8] Winkler V," Securing the Cloud: Cloud computer Security techniques and tactics", Elsevier Inc, Waltham, MA, 2012

[9] Dawoud W," Infrastructure as a service security:Challenges and solutions",In: Proc. of 7th International Conference on Informatics and Systems (INFOS), Potsdam, Germany. IEEE, 2010, pp 1–8.

[10] Paolo Tonella, Paolo Avesani, Angelo Susi(2009)" Using the Case-Based Ranking Methodology for TC Prioritization". 22nd IEEE International Conference on Software Maintenance (ICSM'06).

[11] Zheng Li, Mark Harman, and Robert M. Hierons" Search Algorithms for Regression TC Prioritization" IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 33, NO. 4, APRIL 2007.

[12] Praveen RanjanSrivastava(2008)" TC PRIORITIZATION" Journal of Theoretical and Applied Information Technology.

[13] Ruchika Malhotra, Arvinder Kaur and Yogesh Singh(June 2010)" A Regression Test Selection and PrioritizationTechnique" Journal of Information Processing Systems, Vol.6, No.2.

[14] Belinfante A., Frantzen L., Schallhart C. (2005) 14 Tools for Test Case Generation. In: Broy M., Jonsson B., Katoen JP., Leucker M., Pretschner A. (eds) Model-Based Testing of Reactive Systems. Lecture Notes in Computer Science, vol 3472. Springer, Berlin, Heidelberg

[15] Lingming Zhang," Hybrid Regression Test Selection",ACM/IEEE 40th International Conference on Software Engineering.May 27-June 3, 2018, Gothenburg, Sweden.

[16] Shweta Mittal & Om Prakash Sangwan, "Prioritizing test cases for regression techniques using metaheuristic techniques", Journal of Information and Optimization Sciences, 39:1, 39-51.

[17] Arjinder singh and Sumit sharma, " Functional test cases generation based on automated generated use case diagram", International journal of innovative research in advanced engineering, Issue 8, volume 2 (August 2015) Pages 105 - 110.

[18] Wang Linzhang, Yuan Jiesong, Yu Xiaofeng, Hu jun, Li Xuandong and Zheng Guoliang," Generating test cases from UML activity diagram based on Gray - box method, " Asia - Pacific Software Engineering Conference 2004, Pages: 284 - 291.

[19] Fernando Augusto Diniz Teixeria and Glaucia Braga e Silva, " EasyTest: An approach for automatic test cases generation from UML Activity Diagram",14th International conference on Information technology: New Generations(ITNG 2017)

[20] Noraida Ismail, Rosziati Ibrahim, Noraini Ibrahim," Automatic generation of test cases from Use - case diagram", Proceedings of the international conference on electrical engineering and informatics, Indonesia,June 2007.

[21] P. McMinn, "Search-Based Software Testing: Past, Present and Future," 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops, Berlin, 2011, pp 153-163.