October 2014

MODEL BASED TEST CASE PRIORITIZATION USING UML ACTIVITY DIAGRAM AND EVOLUTIONARY ALGORITHM

PRATEEVA MAHALI

School of Computer Engineering, KIIT University, Bhubaneswar, India, prateevamahali.com@gmail.com

ARUP ABHINNA ACHARYA

School of Computer Engineering, KIIT University, Bhubaneswar, India, arupacharya.kiit@gmail.com

Follow this and additional works at: <https://www.interscience.in/ijcsi>

Part of the [Computer Engineering Commons](#), [Information Security Commons](#), and the [Systems and Communications Commons](#)

Recommended Citation

MAHALI, PRATEEVA and ACHARYA, ARUP ABHINNA (2014) "MODEL BASED TEST CASE PRIORITIZATION USING UML ACTIVITY DIAGRAM AND EVOLUTIONARY ALGORITHM," *International Journal of Computer Science and Informatics*: Vol. 4 : Iss. 2 , Article 1.

DOI: 10.47893/IJCSI.2014.1177

Available at: <https://www.interscience.in/ijcsi/vol4/iss2/1>

This Article is brought to you for free and open access by the Interscience Journals at Interscience Research Network. It has been accepted for inclusion in International Journal of Computer Science and Informatics by an authorized editor of Interscience Research Network. For more information, please contact sritampatnaik@gmail.com.

MODEL BASED TEST CASE PRIORITIZATION USING UML ACTIVITY DIAGRAM AND EVOLUTIONARY ALGORITHM

PRATEEVA MAHALI¹, ARUP ABHINNA ACHARYA²

^{1,2}School of Computer Engineering, KIIT University, Bhubaneswar, India
E-mail: prateevamahali.com@gmail.com, arupacharya.kiit@gmail.com

Abstract- With the exponential growth in size and complexity of softwares, the testing activity is no more limited to testing phase of SDLC (Software Development Life Cycle). Testing process has been made iterative and incremental in Object Oriented development scenario. This leads to increase in effort and time required for testing as well as explosion in test case. But when it comes to regression testing, it has the additional issue of test case retesting which further increasing the effort and time. So a suitable prioritization technique should be used to address these issues. In this paper we had given a proposal which is based on prioritization of test cases using GA (Genetic Algorithm). This process is found to be very effective during regression testing. In this paper we found an optimized independent path having maximum critical path value, which further leads to prioritization of test cases. The three component of regression testing i.e effort, time, cost will be gradually reduce by using this approach.

Keywords- Testing, Optimization of test cases, Test case Prioritization.

I. INTRODUCTION

Nowadays the real world business is concentrating on development of innovative software or to make the development process automatic. Lot of new software are being developed and also the existing software are modified in the context of innovation. A software is developed by adopting a suitable SDLC and through testing has to be performed to increase the degree of acceptance by end user. Software maintenance is an important phase in the Software Development Life Cycle and relies on regression testing. According to IEEE [4], "Regression Testing is the process of selective retesting of a system or component to verify that modifications have not caused unintended effects that the system or component still complies with its specific requirements". Regression Testing occurs in the maintenance phase to ensure quality of the software after modification. After delivery of the updated version of software, end-users will use the product and give their feedbacks. These changes in business logic, addition of new components etc. may lead to the side effects due to dependency fault, error propagation etc. Regression testing of a product ensures that the product do not have any fault whose effect will hamper the performance and quality of the product. Simultaneously we are concerned about reducing the time, cost and resources. To reduce the cost of regression testing, there are different techniques are available.

Such techniques are [1,5]

- 1) Test case Selection
- 2) Test Suite Minimization/reduction
- 3) Test Case Prioritization

Test case prioritization is the most important. Test case prioritization means ordering of test case in such

a way that most beneficial test cases are executed first.

Prioritization of test case is required to improve[1,3]

- 1) Effectiveness of regression testing
- 2) Quality of product
- 3) For reducing the time
- 4) Estimated cost of the system

Test cases are prioritized in such a way that it increases the probability of early fault detection using various techniques. These techniques can be GA(Genetic Algorithm), RT(Random Technique), dependency structure, Traditional statement coverage, Additional statement coverage, FEP(Fault- Exposing-Potential) prioritization etc [2,6,11,12]. The rest of the paper is organized as follows: section II describe the related work. The Basic concepts are described in section III. Our proposed methodology is discussed in section IV and its analysis is discussed in section V by using a case study. Section VI contains the conclusion and future work.

II. RELATED WORK

Rothermel et al.[6] presented different techniques for code based test case prioritization, which are classified into three different groups i.e. comparator group, statement level group and function level group. To measure the effectiveness of these techniques, an experiment was conducted where seven different programs were taken. Here several dimensions like granularity were taken for test case prioritization. The main disadvantage of code based test case prioritization are it is very expensive as its execution is slow because of the execution of the actual code and code based test case prioritization may not be sensitive to the correct or incorrect

information provided by the testers or the developer. Mohapatra et al. [13] presented a technique that extend the code-coverage TCP techniques and apply test case prioritization at a system-level for both new and regression tests. Here the advantage is that the author uses a system level test case prioritization techniques which is called the Prioritization of Requirements for Test (PORT) based techniques.

The PORT technique prioritizes system test cases based upon four factors: requirements volatility, customer priority, implementation complexity, and fault proneness of the requirements. System level test case prioritization techniques are very beneficial because it improve the rate of fault detection of severe faults. PORT technique requires the team to conduct system analysis and write concrete test cases. The act of writing concrete test cases immediately after requirements specification can lead to the identification of ambiguous and unclear requirements, allowing requirements errors to be identified and rectified earlier. The PORT technique allows the engineering team to monitor the requirements covered in system test; the ability to monitor requirements covered in system test is believed to be one of the challenges faced by the industry. But here only the experiment is done for four projects developed by students in advanced graduate software testing class. So the authors have to test the scalability of the PORT method. Khandai et al.[14] proposed a method to generate test cases from combinational UML models such as Sequence Diagram (SD) and Activity Diagram (AD). In this approach AD is converted into an intermediate format known as Activity Graph (AG). After that test sequences are generated from AG by applying Activity Path Coverage Criteria. Then SD is converted into Sequence Graph (SG) and the test sequences are generated by applying All Message Path Coverage Criterion. For having better coverage and high fault detection capability the author constructed a Activity Sequence Graph (ASG) which has the combine features of AG and SG. Finally the ASG is traversed to generate the test cases.

III. BASIC CONCEPTS

Regression testing is an important activity in software maintenance process. Though it is a time, effort and cost consuming activity, it also gives a new trend or scope to fulfil stakeholder requirement [2]. That means it is the process of executing the available test suite on a changed program to assure that the unchanged portion of program remain unaffected [1,2]. If the test suites are large then it is very difficult to retest and to find out the test suite which detect maximum bug in the program. Let we have a software for regression testing which contain the test suites denoted by T and some changed program denoted by P'. Regression testing is performed to

validate P' with respect to T. Test case selection means selecting a subset of test cases from the existing test suite(from T) and execute it on the changed program(P').Test suite reduction means removing redundant test case or minimizing some test cases[2]. Test case prioritization aims at executing the test cases which have higher priority before lower priority test cases. The priority is decided based on some criteria.

In this paper we have used UML 2.0 instead of UML 1.X because UML 2.0 is more detailed or more expressive than UML 1.X, which allows to design in expressively. We can know the static and dynamic behavior of a system from UML 2.0. It contain activity diagram shows the dynamic nature of the system. So in this paper activity diagram is used as modeling diagram. Two basic questions can be formulated from the above discussion. One is why we will prioritize the test cases and the second one is to identify a suitable criteria for prioritization.

Optimization of test cases will add to the effectiveness of the prioritization. Optimization of test case can be done using Genetic Algorithm(GA)[15], Particle Swarm Optimization(PSO)[16] etc. In this paper optimization of test cases means to find out an efficient test suite with maximum path coverage. UML Activity Diagram[17] illustrate the dynamic nature of a system. It shows the flow between various activities. An activity represents an operation on some object in the system and the results in a change in the state of the system. Typically activity diagrams are used to model work flow or business processes and internal operation within the system. Regression testing is very expensive due to large number of test cases at the time of testing. So we need optimization of test cases. Optimization is used to find out feasible solution which is both efficient and effective. Genetic algorithm(GA) is one of the approaches for optimization of test cases[9,10].

IV. PROPOSED APPROACH

In this paper the author proposed an approach for prioritization of test case. At first optimization of test cases is done using an evolutionary algorithm i.e GA. The optimization is followed by prioritization leading to a better test suite for regression testing with respect to cost and time of detecting faults. In this paper the system under test is modeled using UML 2.0 Activity Diagram[17].

V. A CASE STUDY:SHOPPING MALL

This section depicts the details of our proposal for test case optimization and prioritization in regression testing. Here GA is applied on a model-based system i.e on Activity Diagram[13] of Shopping Mall

Management System(SMMS) which is discussed in Fig 1. For generation of test case from activity diagram, first it is converted to activity graph(AG). It is found that the number of test cases in AG is 24 as there are same number of independent paths which is given in Table I. To start with optimization is done to find out maximum path coverage in AG. The fitness function is given in equation 1

TABLE I
INDEPENDENT PATHS PRESENT IN AG OF UPDATED SHOPPING MALL MANAGEMENT SYSTEM

SL. NO(Path)	Test case Name	Independent path
1	T1	1→2→3→4→5→6→7→9→10→11→12→13→15→17→18→19→20→21→23→24→25
2	T2	1→2→3→4→5→6→7→9→15→17→18→19→20→21→23→24→25
3	T3	1→2→3→4→5→6→7→9→15→17→18→19→20→22→23→24→25
4	T4	1→2→3→4→5→6→7→9→10→11→12→13→15→17→18→19→20→22→23→24→25
5	T5	1→2→8→3→4→5→6→7→9→15→17→18→19→20→21→23→24→25
6	T6	1→2→8→3→4→5→6→7→9→15→17→18→19→20→22→23→24→25
7	T7	1→2→8→3→4→5→6→7→9→10→11→12→13→15→17→18→19→20→21→23→24→25
8	T8	1→2→8→3→4→5→6→7→9→10→11→12→13→15→17→18→19→20→22→23→24→25
9	T9	1→2→8→9→10→11→12→13→15→17→18→19→20→21→23→24→25
10	T10	1→2→8→9→10→11→12→13→15→17→18→19→20→22→23→24→25
11	T11	1→2→8→9→15→17→18→19→20→21→23→24→25
12	T12	1→2→8→9→15→17→18→19→20→22→23→24→25
13	T13	1→2→14→3→4→5→6→7→9→10→11→12→13→15→17→18→19→20→21→23→24→25
14	T14	1→2→14→3→4→5→6→7→9→10→11→12→13→15→17→18→19→20→22→23→24→25
15	T15	1→2→14→3→4→5→6→7→9→15→17→18→19→20→21→23→24→25
16	T16	1→2→14→3→4→4→5→6→7→9→15→17→18→20→22→23→24→25
17	T17	1→2→14→3→4→5→6→7→10→11→12→13→15→17→18→19→20→21→23→24→25
18	T18	1→2→14→3→4→5→6→7→9→10→11→12→13→15→17→18→19→20→22→23→24→25
19	T19	1→2→16→3→4→5→6→7→9→15→17→18→19→20→21→23→24→25
20	T20	1→2→16→3→4→5→6→7→9→15→17→18→19→20→22→23→24→25
21	T21	1→2→16→3→4→5→6→7→9→15→17→18→19→20→21→23→24→25
22	T22	1→2→16→3→4→5→6→7→9→15→17→18→19→20→22→23→24→25
23	T23	1→2→16→3→4→5→6→7→9→10→11→12→13→15→17→18→19→20→21→23→24→25
24	T24	1→2→16→3→4→5→6→7→9→10→11→12→13→15→17→18→19→20→21→23→24→25

$$F(x) = \sum_{i=1}^n C_{ri} \quad (1)$$

where C_i is the credit assign to each node present in the critical path of AG.

Critical path can be find out by traversing AG. While traversing the AG some credit is assigned to each node. The path having highest critical value is selected for critical path. Here we have assigned some credit to each node in the following manner.

- 1) If the node is normal activity then we assign 1 credit point.
- 2) If the node is decision node then we assign 2 credit point.
- 3) If the node is intra-depended activity then we assign 3 credit point.
- 4) If the node is inter-depended activity then we assign 4 credit point.
- 5) If the node is fork node then we assign 5 credit point.
- 6) If the node is control node then we assign 6 credit point.

In the AG for SMMS there are two decision nodes i.e node 9 and 20, five control node i.e node 8,14,15,16,17 and one fork node i.e node 2 and others are normal activity node. The normal activities are depended to each other directly and indirectly. If the

activities of one thread depends on each other then it is called as intra-dependency activity. If the activity of two different thread are indirectly depended on each other then it is called as inter-dependency activity. Table II shows the credit value assigned to each node of the AG.

TABLE II
CREDIT VALUE OF EACH NODE OF AG

NODE NO	CREDIT	NODE NO	CREDIT	NODE NO	CREDIT
1	1	10	4	19	4
2	5	11	4	20	2
3	4	12	4	21	4
4	4	13	4	22	4
5	4	14	6	23	3
6	4	15	6	24	4
7	4	16	6	25	1
8	6	17	6		
9	2	18	4

Fig. 1. UML Activity Diagram of Updated Shopping Mall Management System

The proposed algorithm for finding the optimal path by taking activity graph as input.

Algorithm : Finding Optimal Path

Input :

Activity Graph

C_{ri} ≡ Credit value for each node

r ≡ random number

C_p ≡ Crossover probability

M_p ≡ Mutation probability

Output :

Optimized test case/Optimal Path

Fig. 2. Activity Graph of Updated Shopping Mall Management System

- 1) Start
- 2) $C_{ri} = 0$ // Initialize the credit value.
- 3) Convert Activity Diagram(AD) to Activity Graph(AG)
- 4) Generate chromosome population from the independent path as given in Table III
- 5) Traverse the AG to identify path by applying DFS(Depth-First-Search)
- 6) Calculate fitness value i.e total credit value for each path.
- 7) Select two parent node(P_i, P_j).
- 8) if $r < 0.8$
 - Do crossover(P_i, P_j)
 - Child node $C_i = \text{crossover}(P_i, P_j, C_p)$
 - Child node $C_j = \text{crossover}(P_j, P_i, C_p)$
- 9) else if $r < 0.2$
 - Do mutation(P_i, P_j)
 - Child node $C_i = \text{mutation}(P_i, M_p)$
 - Child node $C_j = \text{mutation}(P_j, M_p)$
- 10) Traverse the AG to identify path by applying BFS(Breath-First-Search)
- 11) For each $N_i \in P$ //for each node belongs to the path of AG.
- 12) $C_{ri} = C_{ri} + C_{ri+1}$ // Credit of node i is added to the previous credit.

A logic is proposed to generate chromosomes from the path. Chromosomes are binary sequence of digit. If decision node is encountered then put 0 or if control node is encountered then put 1 in the path. We had generated the chromosomes from the path and fitness values are given Table III.

After optimization we found the credit value of all paths and the test case which execute the path having the highest credit value is the optimized test case. Random technique may be used in case of collision.

According to AG of SMMS, the optimized value is 83. There are 6 number of test cases i.e $T_7, T_8, T_{17}, T_{18}, T_{23}$ and T_{24} whose value is 83. This occurs because our aim of optimization is to find out the path which have maximum credit value. So we have to arrange these test cases randomly for optimization. The test sequence according to the credit value is $T_7, T_8, T_{17}, T_{18}, T_{23}, T_{24}, T_1, T_4, T_{15}, T_{16}, T_5, T_6, T_{21}, T_{22}, T_9, T_{10}, T_2, T_3, T_{11}, T_{12}, T_{13}, T_{14}, T_{19}, T_{20}$. But one question arises here that how to identify the important test cases out of many test cases.

In this paper the author has consider an UML activity diagram for SMMS which is shown in Fig 3 and the corresponding activity graph is Fig 4, is the previous version of SMMS. In due cause of time the requirement is also changing. So Fig 1 is the modified version of Fig 3. Previous version of activity diagram contain sixteen activities, so AG contains sixteen node and modified version of AD contain twenty five activities. So AD contains twenty five node. That means modified version contains extra nine node i.e nine activity. So we have to find out the difference between two graph. (1) Previous version of AG contain two decision node i.e node 3 and 11 and updated version contain two decision node i.e node 9 and 20. (2) Previous version of AG contain one control node i.e node 7 and updated version contain five control node i.e node 8,14,15,16,17.

Now our aim is to prioritize the optimized test cases i.e $T_7, T_8, T_{17}, T_{18}, T_{23}, T_{24}$. According to our proposal prioritization can be done by comparing the changed activities with original activities in the regression testing. So we have to find out changes in normal activities, control and decision nodes. Then we can find out the prioritize value by using formula given in equation 2.

Fig. 3. Activity Diagram of Shopping Mall Management System

Fig. 4. Activity Graph of Shopping Mall Management System

$$Prioritizedvalue = \frac{A_c}{A} + \frac{C_c + D_c}{C + D} \quad (2)$$

Where, A_c is changed normal activities.

A is normal activities present in the optimized test cases.

$C + D$ is the total control and decision node present in the optimized test cases.

$C_c + D_c$ is the total changed control and decision node.

TABLE III
CHROMOSOME AND FITNESS VALUE OF INDEPENDENT PATHS

SL NO(Path)	Test cases	Chromosomes	Total Credit
1	T_1	0110	77
2	T_2	0110	61
3	T_3	0110	61
4	T_4	0110	77
5	T_5	10110	67
6	T_6	10110	67
7	T_7	10110	83
8	T_8	10110	83
9	T_9	10110	64
10	T_{10}	10110	64
11	T_{11}	10110	48
12	T_{12}	10110	48
13	T_{13}	1110	46
14	T_{14}	1110	46
15	T_{15}	10110	76
16	T_{16}	10110	76
17	T_{17}	10110	83
18	T_{18}	10110	83
19	T_{19}	110	40
20	T_{20}	110	40
21	T_{21}	10110	67
22	T_{22}	10110	67
23	T_{23}	10110	83
24	T_{24}	10110	83

TABLE IV
PRIORITIZED VALUE OF OPTIMAL TEST CASES

Test Case Name	Total Activity	Normal Activity	Changes Activity	Control + Decision	Changes in Control + Decision	Prioritized Value
T_1	22	17	10	5	2	1.828
T_5	22	17	10	5	2	1.828
T_{17}	22	17	10	5	2	1.828
T_{18}	22	17	10	5	2	1.828
T_{21}	22	17	10	5	2	1.828
T_{23}	22	17	10	5	2	1.828
T_{24}	22	17	10	5	2	1.828

For this case study the prioritized value is given in Table IV.

In this case study the prioritize value is equal because modification in all test case is equal. Here every test case contain at least 3 control node and 2 decision node. Therefore prioritize value is equal, so we have to prioritize with randomly. If we do some modification in the AG for next version like remove 2 control node (maybe 14, 16) then the prioritized value will differ in every test cases. Test case can be sorted according to which have maximum prioritize values. If prioritize value is equal in many test cases like in our case study, then we can apply random techniques. Therefore prioritized test suites are $T_8, T_7, T_{17}, T_{18}, T_{24}, T_{23}, T_1, T_4, T_{15}, T_{16}, T_5, T_6, T_{21}, T_{22}, T_9, T_{10}, T_2, T_3, T_{11}, T_{12}, T_{13}, T_{14}, T_{19}, T_{20}$.

VI. CONCLUSION AND FUTURE WORK

In this paper, we present a model-based test case optimization and prioritization technique for regression testing. In this approach we investigate the present methods with respect to reduction of time and cost and also to slow case the efficiency of prioritized test cases. The result indicate that our method is better effective to solve all these problems simultaneously. The case study represented in this paper is relatively small. In the future research, we plan to apply this approach on a comparatively large application to review the scalability of the proposed approach.

REFERENCES

- [1] S. Khandai, A. A. Acharya, D. P. Mohapatra, Model Based Test Case Prioritization for Regression Testing using Business Criticality Value, International Journal of Computer Applications, 2012
- [2] Bo Jiang, Zhenyu Zhang, W.K. Chan, T.H. Tse, Adaptive Random Test Case Prioritization, IEEE/ACM International Conference on Automated Software Engineering, 2009
- [3] Md. Junaid Arafeen & Hynsook Do, Adaptive Regression Testing Strategy: An Empirical study, 22nd IEEE International Symposium on Software Reliability Engineering, 2011
- [4] IEEE Glossary
- [5] Xiaobo Han, Hongwei Zeng, Honghao Gao, A Heuristic Model-Based Test Prioritization Method for Regression Testing, International Symposium on Computer, Consumer & Control, IEEE, 2012
- [6] Gregg Rothermel, Roland H. Untch, Chenyyun Chu, Mary Jean HarRold, Prioritizing Test Cases For Regression Testing, IEEE Transactions on Software Engineering, 2001
- [7] Naresh Chauhan, software Testing Principles & Practices, Oxford University Press, 2010
- [8] Rajib Mall, Fundamental of Software Engineering, PHI Learning Private Limited, 3rd edition, 2009
- [9] www.academia.edu/603194/Effective Generation of Test Cases Using Genetic Algorithms and Optimization Theory
- [10] dl.acm.org/citation.cfm?id=1764824, Quality improvement & optimization of test case : A Hybrid Genetic algorithm based approach

- [11] Rajib Mall, Siripond Roongruangsuwan, Jirapun Daengdej, Test Case Prioritization techniques, Journal of Theoretical & Applied Information technology, 2005
- [12] Shifa-e-Zehra Haidry, Tim Miller, Using Dependency structures for Prioritization of Fundamental Test suites, IEEE Transactions on Software Engineering, February 2013, vol-39, No. 2
- [13] S. Khandai, A. A. Acharya, D. P. Mohapatra, Prioritizing Test Cases Using Business Criticality Test Value, International Journal of Advanced Computer Science and Applications, 2011
- [14] M. Khandai, A. A. Acharya, D. P. Mohapatra, Test Case Generation for Concurrent System using UML Combinational Diagram, International Journal of Computer Science and Information Technologies, 2011.
- [15] Izzat Alsmadi, Using Genetic Algorithms For Test Case Generation And Selection Optimization
- [16] Luciano S. de Souza, Pericles B. C. de Miranda, Ricardo B. C. Pruden cio, Flavia de A. Barros, A Multi-Objective Particle Swarm Optimization for Test Case Selection Based on Functional Requirements Coverage and Execution Effort, 23rd IEEE International Conference on Tools with Artificial Intelligence, 2011
- [17] [www.smartdraw.com/resources/tutorials/Software Design Tutorials](http://www.smartdraw.com/resources/tutorials/Software_Design_Tutorials)