Model based Test Case Prioritization

Swati B. Patil
Assistant Professor
Vishawakarma Institute of Information Technology, Pune

Rutuja Rajput
Research Scholar
Vishawakarma Institute of Information Technology, Pune

ABSTRACT
Test case prioritization involves scheduling test cases so that the performance of software testing can be improved. Thus resulting in high quality software with minimum faults and errors[12]. Therefore in this paper, model based test case prioritization using Memetic algorithm, is proposed for giving an all-purpose solution in prioritization and optimization problem which can be used efficiently. A model based testing method is introduced. This method can prioritize test case using activity diagram and genetic algorithm. The result shows that using this method with Memetic algorithm (MA) provide efficiency and accuracy for test cases [3].

Keywords
Software testing- Genetic Algorithm (Base Algorithm), Memetic Algorithm (Derived from GA) Activity Diagram, nodes and edges, objective function, fitness function, test case, SDLC, prioritization

1. INTRODUCTION
NOWADAYS many new software are being developed and also modifications are being made to the existing software for innovation. A software goes through all the phases of SDLC and through testing is been performed to increase the acceptability by the end user[2]. Maintenance of Software is an important phase in the Software Development Life Cycle and depends on testing. In recent years the needs for high quality software has been increased. The end users need software with minimum errors. Software testing is one of the major and primary techniques for achieving goals of high quality software. Software testing is performed to verify that software is build according to the customer specification or requirements. The purpose of testing is to find faults that cause software failure, probably discovering these faults as early as possible. In fact, early feedback about faults allows anticipating the costly activities of debugging and corrective maintenance, with a related economical return[1].

When the time necessary to execute all test cases is long, prioritizing them is important so as to discover most faults early might save substantial time, since bug fixing can start earlier. Hence to save time and money in testing phase we are prioritizing test cases in software development using, evolutionary algorithms like GA and MA for generating model based test cases. Hence giving an all-purpose solution in prioritization and optimization problem which can be used efficiently

Therefore in this paper, model based test case prioritization using memetic algorithm, is proposed for decreasing the time and cost of testing process. Test case prioritization aims at finding an execution order for the test cases which maximizes a given objective function. Hence, to take care of requirements change, a stack based approach for assigning weights to the nodes of activity diagram has also been proposed. In this paper, we have extended our previous work of generating test case scenarios from activity diagram by also considering the concurrent activities in nested activity diagram.

2. RELATED WORK
Code based test case prioritization techniques are classified into three groups i.e. statement level group, comparator group, and function level group. For measuring 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. Technique that extend the code-coverage test case prioritization techniques and apply test case prioritization at a system-level for both new and regression tests. The advantage of this method is that the system level test case prioritization techniques which is called the Prioritization of Requirements for Test (PORT) based techniques are been used[12].

The Prioritization of Requirements for Test (PORT) technique prioritizes system test cases based upon four factors: implementation complexity, fault proneness, requirements volatility, customer priority of the requirements. System level test case prioritization techniques improves the rate of fault detection of severe faults. PORT technique requires the team to conduct system analysis and write concrete test cases[2].

To re execute all test cases after every modification in the code is an inefficient process. A technique is to execute the modified lines of code with minimum number of test cases. The proposed test case prioritization technique organizes the test case in a test suite in an ordering such that fewer lines of code need to be re executed thus faster code coverage is attained which would lead to early detection of faults. 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 testers and developers.

3. EXISTING SYSTEM
In existing system , code based methods are used for software testing. Code based testing corresponds to the testing that is carried out on code development, code inspection, unit testing in software development process[10]. The code based testing consists of the following testing: [1]Dynamic testing- Statement coverage, Branch coverage and Path coverage. [2]Static testing- Code inspection, Code walkthrough, Code review, Code Audit.

Code coverage based prioritization techniques are formed to be used by most of the scholars. Code based methods are mainly used for regression testing. Regression testing is a kind of testing which requires maximum effort, time and cost. Regression testing is very expensive activity because it expend half the total software maintenance costs In the cost based test prioritization techniques, the source code of the system is used to prioritize the test[12].

4. PROPOSED METHOD

The proposed method for test case prioritization does prioritizing test cases using genetic algorithm (GA). The proposed methods are applied on a model based system (i.e. activity diagram for deriving the test case scenario in earlier phase) in designing phase. In Software Development Life Cycle(SDLC), first the design models are created. After that these models are converted into the codes at coding phase. If an error is seen during the design phase of software, it can be corrected easily before coding. Therefore, error will be prevented to propagate in software and we could prevent increasing software expense, time and cost on fixing it. We represent our problem coding in term of chromosome for GA and MA algorithms. Next we represent our cost function for evaluating the chromosomes. Finally, the structures of the algorithms are presented.

4.1 Overall Algorithm

The Genetic algorithm process is as follows[4]:

Step 1. Determine the number of chromosomes in population, generation, and mutation rate and crossover rate value.

Step 2. Generate chromosome-chromosome number of the population, and then initialization value of the genes chromosome-chromosome with a random value.

Step 3. Perform steps 4-7 until the number of generations is met.

Step 4. Evaluation of fitness value of chromosomes by calculating objective function.

Step 5. Compute Chromosomes selection and Crossover using pseudo-code.

Step 6. Perform Mutation.

Step 7. New Chromosomes (Offspring).

Step 8. Solution (Best Chromosomes).

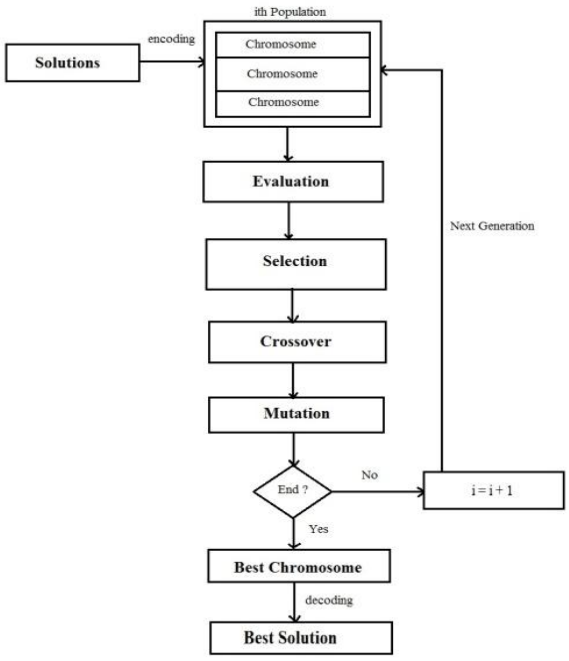

Fig 1: Genetic Algorithm Flowchart

4.2 Explanation of Base Algorithm

[1]In Initialization step we define the number of chromosomes in population and generate random value of genes. [2] In Evaluation step we compute the objective function value for each chromosome produced in initialization step. [3] Selection of the fittest chromosomes have higher probability for being selected for the next generation. Now we need to compute fitness probability using the fitness function of each chromosome. [4] For Crossover, we use one-cut point, i.e. randomly select a position in the parent chromosome then exchanging sub-chromosome. Parent chromosome which will mate is randomly selected and the number of mate Chromosomes is controlled using crossover_rate (ρc) parameters. Pseudo-code for the crossover process is as follows:

begin k← 0;

while(k<population)

do R[k] ← random(0-1);

if (R[k] < ρc)

then

select Chromosome[k] as parent;

end;

k = k + 1;

end;

end;

[5] In Mutation Number of chromosomes that have mutations in a population is determined by the mutation rate parameter. Mutation process is done by replacing the gen at random position with a new value.

4.3 Architecture of Memetic Algorithm

This section depicts the architecture for the proposed system, the models are given as input to the simulator which uses memetic algorithm to compute output. The output of the simulator is compared with the real values(Fitness value) iteration by iteration until improvement is done. Best solution is returned to final stage of architecture as calibrated model.

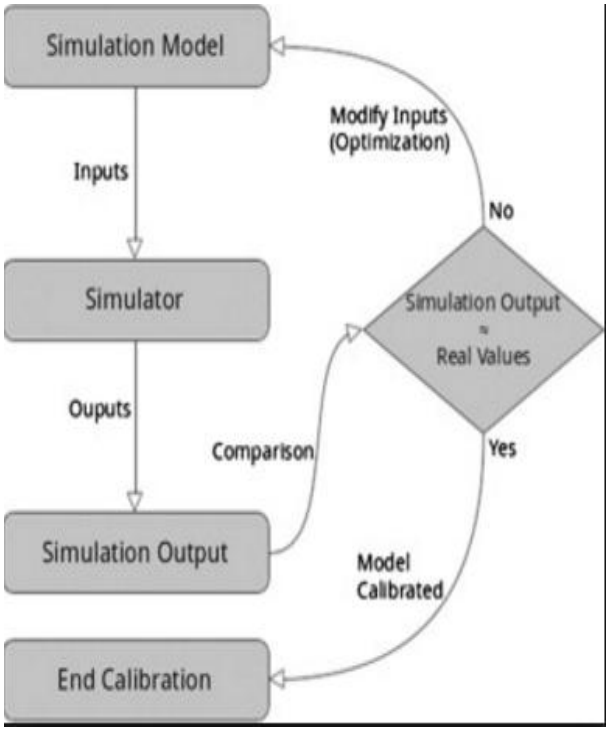

Fig 2: Architecture of Memetic Algorithm.

4.4 Working

This section depicts the details of our proposal for test case prioritization using memetic algorithm on models. The overall working of the proposed methods is given in Fig1. GA is used here as the base algorithm. This algorithm produces a population of individuals(chromosomes) which improves iteration by iteration until the termination condition is met. This feature has effect in reducing diffuseness of population.

We start from the best solution of current iteration and moves to a randomly selected solution by probability p. Otherwise better solution in the neighbourhood is selected. After termination, the input solution is replaced if the final solution of this algorithm has better performance.

GA is applied on a model-based system i.e on Activity Diagram of each use cases, which is discussed in Fig 3. For generation of test case, the Activity diagram is converted to control flow graph (CFG).weight assignment is done to find out maximum path coverage in CFG. The fitness function is calculated and the test cases are prioritized.

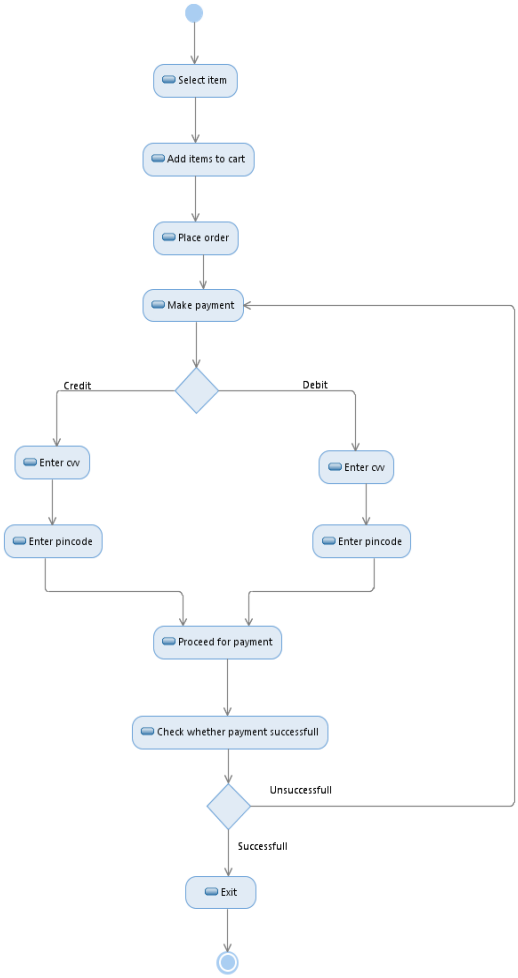

Fig 3: General Activity diagram for proposed System.

5. EXPERIMENTAL RESULTS

In this section we are going to explain the performance of the proposed methods for prioritizing test cases based on the activity diagram in the design model.

In the proposed method, GA algorithms are used to find the most critical paths in the graph. These paths must be tested first. The proposed method has some parameters. So we first need to set the parameters values. Then compute different probabilities for crossover and mutation rates in GA algorithms. After iterative random process of computation, the best solution is estimated. Finally, the algorithms are applied for 10 times and the average results are reported.

5.1 Case Study

To evaluate performance of the proposed methods, an activity diagram as a part of a design model is used here. Activity diagram shows the dynamic nature and functional view of the system. Control flow graph (CFG) is made from the activity diagram. The weights are assign to the edges. Using the fitness function the test optimization and prioritization.

The activity diagram of "Withdrawal in ATM System" given in fig 4 is used here as the case study. This diagram is converted to a CFG. Then the CFG is used as the input of the proposed method. Figure 5 shows the corresponding CFG diagram. CFG has 15 nodes.

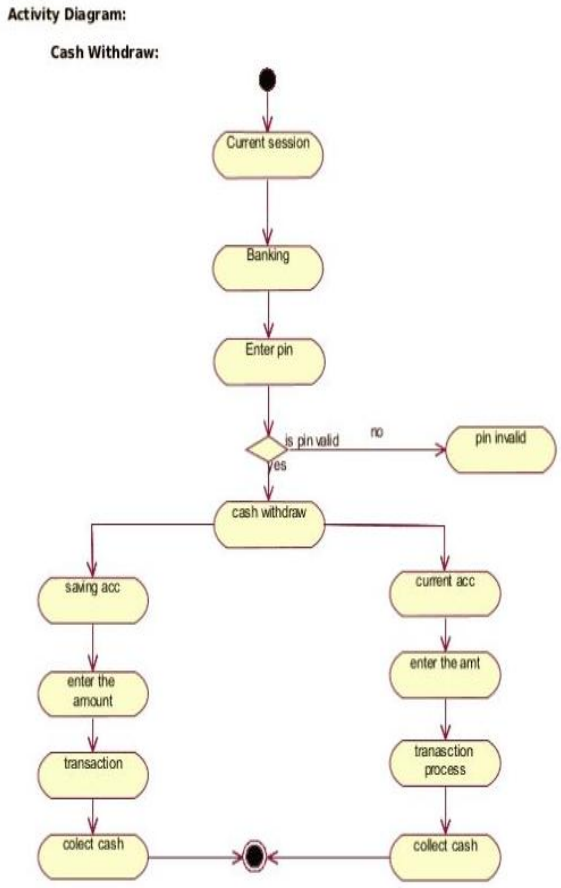

Fig 4: Activity diagram for withdrawal in ATM System.

5.2 Proposed Algorithm

Step1: Initialization: Populate the Chromosomes, formulate the objective function, for this population. The objective is minimizing the value of function f(x) where f (x) = ((a + 2b + 3c + 4d) - Smax). Here a, b, c, d are genes and integer values ranging between 0 to 14.

Step2: Now generate random values for a, b, c, d for each chromosomes.

Step3: We compute the objective function in evaluation step.

Step4: In selection step we use formula:

Fitness[chromosome] = 1/(1+f_obj[chromosome])

Compute fitness function and probabilities for each chromosome.

From the probabilities find the highest fitness value of the chromosome.

Step5: After chromosome selection, the next process is determining the position of the crossover point. This is done by generating random numbers between 1 to (length of Chromosome – 1).we get the crossover point, parents Chromosome will be cut at crossover point and its gens will be interchanged.

Step6: Perform mutation process. After finishing mutation process we complete one iteration or one generation of the genetic algorithm.

These new Chromosomes will undergo the same process as the previous generation of Chromosomes such as evaluation, selection, crossover and mutation and at the end it produce new generation of Chromosome for the next iteration[4]. This process will be repeated until a predetermined number of generations and the best solution will be obtained as the result of our proposed system as "Prioritized Test case".

The activity diagram in (Fig 4) is converted into CFG and given as input to the proposed system as below.

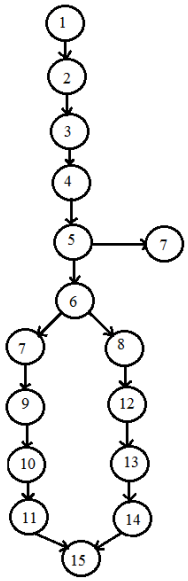

Fig 5: Control flow graph for Activity diagram.

The CFG has 15 nodes for above example corresponding to activities. The nodes are used to estimate the further values using MA steps[11].

Table1: Weight assignment for each node corresponding to CFG

Nodes	K	Size	Weight = Smax-k
1	14	15	1
2	13	14	2
3	12	13	3
4	11	12	4
5	10	11	5
6	9	10	6
7	8	9	7
8	7	8	8
9	6	7	9
10	5	6	10
12	4	5	11
13	3	4	12
14	2	3	13
15	1	2	14
	0	1	15

The weights are used in Memetic algorithm for prioritizing the test cases. Thus increasing the performance and accuracy.

5.3 Performance Analysis

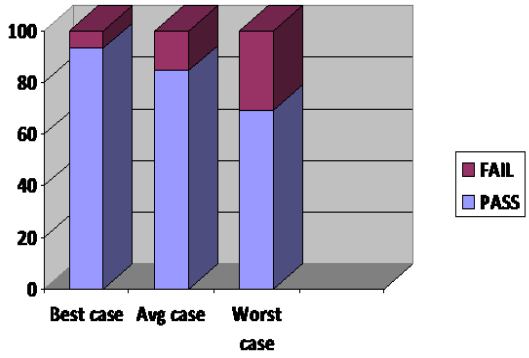

Fig 5: Performance Comparison of the Proposed Algorithm for Test Case Prioritization in a CFG Diagram

In this section, the result of the proposed methods which are carried out by model based technique is presented. The

performance of the test cases is presented in Figure 5. In this figure, the vertical axis shows the accuracy of the test cases while the horizontal axis represents the number of test cases pass or fail considering all types of loops and inputs using MA. Figure 5 shows the best result in prioritizing most important paths in the case study. The best result is obtained by model based method.

(92.30(best case),84.61(average case),69.20(worst case) approximately)

Overall Accuracy = 87.35(approximately)

Explanation: The average accuracy is calculated as 87.35 % based on the test cases. The test cases are generated considering the nested loops.

6. CONCLUSION

This paper, presents a model-based test case prioritization. This approach investigates the present methods such as code based test case prioritization with respect to reduction of time and cost and increased efficiency of system by prioritizing test cases. The result indicates that model based method is better and effective to solve problems of testing in early phase of Software Development Life Cycle (SDLC). A prioritization method is proposed for identifying the most important paths in the activity diagram that must be tested first. The activity diagram is converted to the CFG and weights are assigned to its nodes. A fitness function is used based on nodes of activity diagram. GA is used as the base method for calculating fitness function. Depending on the fitness values the test cases are prioritized.

7. FUTURE WORK

In the future research, this approach can be applied on a comparatively large application which should be open source and platform independent to review the scalability of the proposed approach. It should also be Interoperable. This system can be further extended for multiple applications as follows: Model based application and existing tools can be integrated resulting in an web based application for testing Test cases rather than standalone system. Error propogation can be avoided in early designing phase hence giving high accuracy software and many more similar applications. Thus, this system can prove to be very helpful in future.

8. REFERENCES

[1] 1st Conference on Swarm Intelligence and Evolutionary Computation (CSIEC2016), Higher Education Complex of Bam, Iran, 2016

[2] P. Mahali and A. A. Acharya, "Model Based Test Case Prioritization Using UML Activity Diagram and Evolutionary Algorithm," ed: IJCSI, 2013.

[3] Sabharwal, R. Sibal and Chayanika Sharma, "Applying gentic algorithm for prioritization of test case scenarios derived from UML diagrams," IJCSI International Journal of Computer Science Issues, vol. 8, 2011.

[4] Genetic Algorithm for Solving Simple Mathematical Equality Problem Denny Hermawanto Indonesian Institute of Sciences (LIPI), INDONESIA

[5] [1] Mitsuo Gen, Runwei Cheng, "Genetic Algorithms And Engineering Design", John Wiley & Sons, 1997.

[6] Praveen Ranjan Srivastava.(2008),"Test Case Prioritization", Journal of Theoretical and Applied Information Technology IEEE

[7] ElFar, Ibrahim K., and James A. Whittaker. "ModelBased Software Testing." Encyclopedia of Software Engineering , 2001.

[8] Dr. Velur Rajappa, Arun Biradar, Satanik Panda "Efficient software test case generation Using Genetic algorithm based Graph theory" International conference on emerging trends in Engineering and Technology, pp. 298--303, IEEE (2008).

[9] Sahil Batra , Dr.Rahul Rishi,"Improving Quality Using Testing Strategies",Journal of Global Research in Computer Science volume 2, No. 6, june 2011 R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev.

[10] B. Korel and G. Koutsogiannakis, "Experimental comparison of code- based and model-based test prioritization", in Software Testing, Verification and Validation Workshops, 2009. ICSTW'09. International Conference on, 2009, pp. 77-84 .

[11] Fatemeh Mosala Nejad "using memetic algorithm for test case prioritization in model bsed software testing." 1st Conference on Swarm Intelligence and Evolutionary Computation (CSIEC2016), Higher Education Complex of Bam, Iran, 2016.

[12] Usha Badhera1, G.N Purohit2, Debarupa Biswas3 " test case prioritization algorithm based upon modified code coverage in regression testing." International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.6, November 2012.