Generate Optimized GUI Test Sequence using GA and Fuzzy Logic

Suman
M.Tech Student
Dept. Of Computer Science And Applications
MDU Rohtak, Haryana, India

R.S. Chhillar, Ph.D
Head Of Department
Dept. Of Computer Science And Applications
MDU Rohtak, Haryana, India

ABSTRACT
A software system is composed of multiple modules. These modules and procedures are integrated via some user friendly environment such as graphical interface. End User is very much interested in the effective working of this graphical interface. To satisfy the user in terms of available graphical options, it is required to optimize the GUI Testing. GUI Testing plan includes two main stages. First is to identify the cost of individual testing in terms of time or effort. Once the cost is identified, the next work is to generate the optimal test sequence so that the cost effective test sequence is identified. In this present work, a genetic approach based solution is provided to generate the optimal test sequence. The work is implemented in matlab environment. The obtained results show the effective cost estimation and sequence generation.

Keywords: **GUI Testing, Optimized Test Sequence, Cost Effective, Genetic**

1. INTRODUCTION
One of the major aspects of software development is to provide the user satisfaction. Along with developing a reliable and secure software system, it is also required to provide the software options to end user in an easy way. GUI Testing basically to analyze the software system in terms of its representation and graphical options. These software systems can be desktop application, mobile applications or the web sites. The graphical interface is been defined for each even associated with the user. Such kind of testing is called model based testing that requires analyzing the software system under different factors. These factors include the behavioral coverage, better representation and generation of automated events over the system. GUI testing basically performs the analysis to study the automated behavior of these tools so that the optimized results will be derived by the user. Today most of the available languages provides such interfaces to provide the graphical interfacing to application.

To provide the effective testing for such application environment, it is required to identify the test cost and identify the optimal test sequence. During the test plan stage, the GUI testing is performed by considering the view of end user. As the test plan is generated, with each test case, the possibility of test case occurrence and its criticality is identified. Based on these vectors the particular test cost is identified. Here figure 1 is showing the different factors that identify the test cost for a software system. To optimize the GUI Testing it is required to optimize the sequence of test case generation. The selection of test sequence also depends on the criticality of the module or that is resolved by the particular test case. To estimate test cost, it is required to analyze the GUI Testing respective to these all vectors..

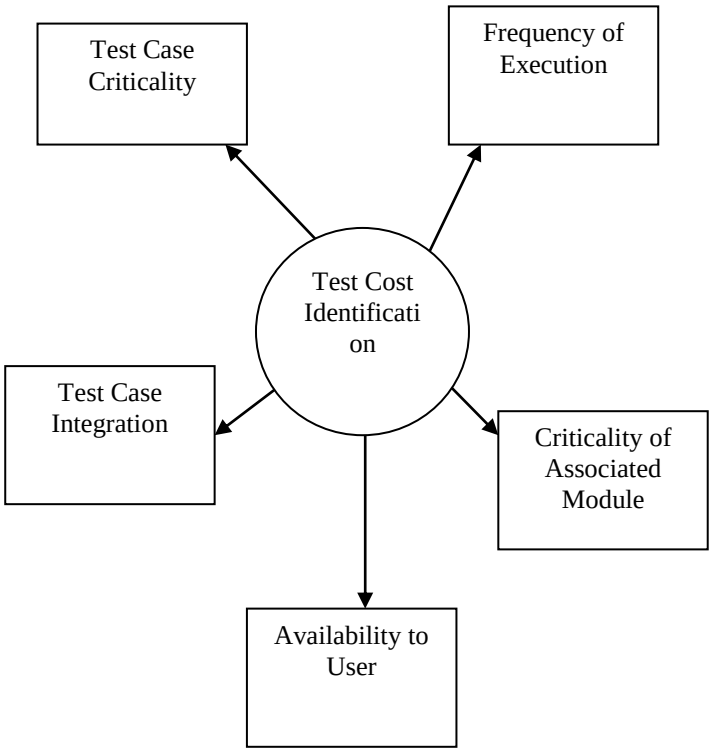


Figure 1: Factors for Test Cost Identification

Based on these vectors the classification of the test cases is done and based on this classification, the particular test cost is assigned to the test cases.Genetic algorithm can be used for finding out optimized test sequence. GA searches for optimal test sequence that satisfies predefined criteria. This criteria is represented through fitness function which is calculated using fuzzy logic.

A Graphical system that represents the software system, itself is the collection of interconnected components. These components are defined to identify the input level integration between the software modules and the graphical interface. To perform GUI testing, the designer has to generate and traverse all the possible paths. The analysis of these paths is done under test case criticality vector. A test path having the non critical components will be considered as the most effective path and having the least cost for implementing the particular test sequence. GUI testing needs to analyze the different widgets and gajets connected with the main graphical interface. To elaborate such situation in software system it is required to represent the software system under different phases. In this work, a genetic approach is defined to provide the coverage to

all the graphical components and software modules without repeating the component itself. The work here defined to generate the optimal sequence.
In this section, the basic steps required for the GUI test sequence generation is been discussed. Later on in second section, the literature associated with test cases identification is been discussed. In section 3, the research methodology adopted in this work is been discussed. In section IV, the results driven from this work are defined. Finally the conclusion is listed here.

2. RELATED WORK

Lot of work is already presented by different programmers to perform GUI Testing and to generate the optimal test sequence. Some of the work defined by earlier researchers is discussed in this section. Bentley [5] has presented an effective software system tool for generating the powerful application. The testing is here defined as the primary operation to analyze the software system and to generate the optimal sequence. Author defined a work to analyze the software system respective to all the methodologies, approaches and tools associated with software system. The testing includes the tools and terminology analysis so that the adaptation to the software system will be done. Author defined a professional way to overcome such situation so that the software generation risk will be minimize. Author defined the application respective to the business user and provides the final product analysis so that the software system and software frustration will be reduced. Author defined a work to reduce the problems that a user faces due to poor graphical design. Author focused on work to reduce the user efforts to work on the application. Kruse et al. [6] has presented another work on the generation of optimal test sequence. This sequence generation is defined under the classification tree analysis. Author defined the dependency analysis along with rule generation. Here the dependency is analyzed bi directionally, i.e. with the previous as well as next modules. Author presented the coverage analysis on these modules and generates the test sequence so that the classification tree for the software system will be generated. Ma et al. [8] has presented a work to improve the communication rate in case of fault occurrence in the software system. Author analyzes the system for regression testing and identifies the chances of fault occurrence over the system. Author performed the fault analysis to generate the fault free sequence for test case generation. Author also performed a comparative study to the system under the application and experiment analysis so that the cost of the prioritization gets minimum. Author defined the prioritization based approach for the coverage method generation and sequence setup. Author identify all the possibility and performing the variety of possibility analysis for generation of optimal sequence and to generate the traditional prioritization for the system coverage generation. The fault analysis is here performed for optimizing the system as well as the sequence generation process. Elbaum et al. [10] has defined a study on eight different system under different test cases so that the analysis will be done for finding of the test sequence. Author defined the case prioritization technique under component analysis and component interaction analysis. Author support the search under strategic analysis so that the particular test case optimization will be done and the new system will be generation for whole system. Author presented the instance and threshold approach for selection of the particular test case or the sequence under cost benefit analysis. Author defined the metrics analysis approach for optimal sequence generation and to reduce the failure rate over the testing process. Author defined a classification tree based approach for likelihood analysis so that the improved system will be obtained.

3. PROPOSED METHOD

In this present work, a genetic based work is defined to generate the optimize test case sequence for GUI Testing. The work is here under the component criticality vector. Here the all modules are defined as the associated procedures and each procedure is connected with some component present on the interface graphically. In this work, the graphical interface is provided in an integrated way to obtain the optimal test sequence and to analyze the system effectively over the system. The main work associated is performed according to steps given below:

Step 1: Describe the software system in terms of procedures and associated graphical component.

Step 2: Identify the criticality of each component and associated module

Step 3: Identify the test cases of software component.

Step 4: Identify the cost of execution of particular test case using criticality vector

Step 5: Identify all the possible paths for execution

Step 6 : Apply genetic with fuzzy to identify optimal test sequence

These are the main steps we will follow in this present work. The sequence generation is here been defined using genetic approach.

To analyze the individual graphical component and associated procedure, the module criticality analysis approach is defined. Higher the criticality of software module , higher the cost will be of particular test case. Based on these vectors, the overall cost of sequence generation is defined. The genetic approach and fuzzy logic used for this work is discussed in this section

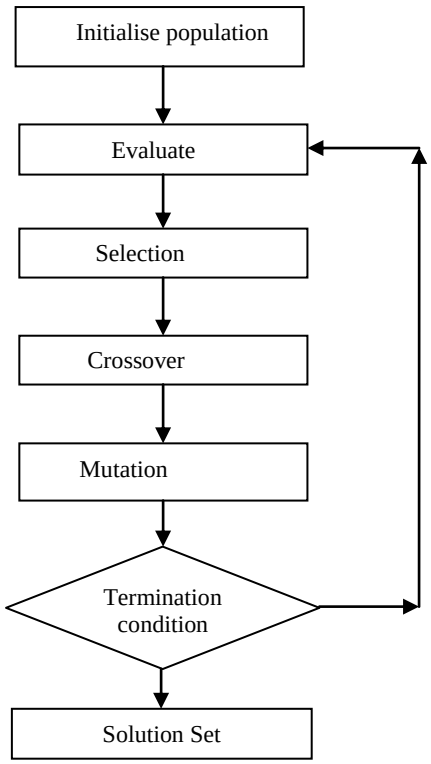
Genetics

Genetic is defined as the evolutionary algorithm that uses the evolution procedure for optimizing the problem solution. This algorithm basically defined under some rules and constraints. The process is defined for maximum number of generation and with each generation a feasible solution is identified. If the solution is under the cost based rule specification, the solution is considered. As all the possible feasible solutions are analyzed, the optimal solution is identified. The generation process is defined to generate the possible solution. The stages of genetic process include the Encoding mechanism, Evaluation, Selection, Crossover, Mutation and Decoding process.

The encoding mechanism is here defined to test sequence path so that the rule over the sequence will be applied. As the possible generations are identified, it works as the population set for genetic process. Along with this process the fitness rule is defined to accept the particular test sequence or to reject it. Once the population set is defined and the fitness rule is identified, the genetic procedure is implemented for defined number of iterations. With each iteration, two test sequences are selected and on it the cross over operation is applied for generation of new child. Next mutation process is applied to modify the sequence under business rule. This whole process is repeated till the optimal solution is not recognized.

A basic algorithm for a GA is as follows:

Initialize (population)
Evaluate (population)
While (stopping condition not satisfied) do
{
Selection (population)
Crossover (population)
Mutate (population)
Evaluate (population)
}
The algorithm will iterate until the population has evolved to form a solution to the problem or until a maximum number of iterations have taken place.The whole process of genetic algorithm is shown in figure 2 also. In this work this genetic approach is applied for the generation of optimal test sequence for GUI Testing. The reliability of the genetic process is defined in its fitness function that actually works as the controller function to decide the acceptability of a generated sequence as the feasible solution or not. In this work, the fitness function is defined to estimate the cost of the generated test sequence. This sequence cost depends on the individual test cost and the priority of the test cost. This test cost and priority depends on the criticality vector. More critical the particular test case is, higher the cost and priority will be. The work is here defined the generation of crossover stage where the selection procedure of new test case from existing test cases is defined.



Fuzzy System

Fuzzy logic is a form of many-valued logic, it deals with reasoning that is approximate rather than fixed and exact.[14] Fuzzy logic variables may have a truth value that ranges in degree between 0 and 1. It includes 0 and 1 as extreme states for acceptance and rejection and also includes various states in between, for example result of comparison between two things could be in decimals e.g. 0.8 etc. Fuzzy logic provides a more efficient and resourceful way to solve control systems (e.g. Temperature controller).

A fuzzy set actually defines the degree of membership between 0 and 1. If the data value is 1 then, the object is completely accepted and if the object value is 0, then object is completed rejected. If other cases,the acceptance and rejection is partial. The numbers are assigned to the object degree under the membership definition. As the mathematical definition, a set is defined as a collection of things that belongs to some definition. Here the rule is defined either the object belong to the set or not.

To analyze the priority of the test case, a fuzzy system is integrated in the fitness function. Here fuzzy is applied as a rule with three main vectors called high, medium and low. Here high represents the high criticality of test case, that itself represents the higher test cost. In same way, medium priority represents the medium criticality and cost and low priority represents the low criticality and cost. The fitness function here estimated the test cost aggregative for the complete sequence. The fuzzy system description used in this work at fitness rule stage is described here under.
Three Fuzzy rules as Fuzzy low, Fuzzy medium and Fuzzy high used in this work to represent the prioritization vector. These fuzzy rules are used for the priority of a particular test and based on it cost is assigned to a particular test case. Here factor is probabilistic value of node and a, b and c are defined value for fuzzy rules. These rules are defined as follow:

Rules for Fuzzy Low

1) if(factor < a) then value=1
2) if(factor >=a && factor < b) then value=((b -factor)) / (b-a)
3) else value=0

Rules for Fuzzy medium

1) if(factor ==b) then value=1
2) if(factor >=a && factor < b) then value=((factor-a)) / (b-a)
3) if(factor >=b && factor < c) then value=((c-factor)) / (c-b)
4) else value=0

Rules for Fuzzy high

1) if(factor >=b) then value=1
2) if(factor >=a && factor < b) then value=((factor-a)) / (b-a)
3) else value=0

The degree of membership is shown in figure 3.
The test cases are classified as low, medium, high using these rules and priorities are assigned based upon membership function calculated using fuzzy logic.

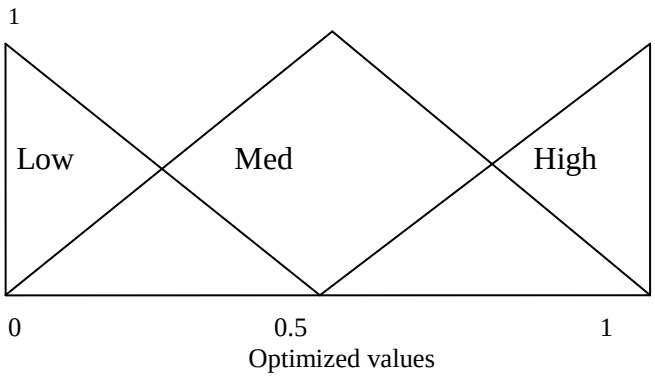


Figure 3 : membership using fuzzy logic

4. RESULTS

The presented work is implemented in matlab environment. The analysis is here been done on a example software system with 10 different components integrated with separate modules. Each component is defined with specification of the test cases with them. After the analysis of these modules, the test case criticality is identified for identification of test cost. Once the test cost is identified, it work as the initial dataset for genetic process. The random encoding mechanism is here applied to generate the possible sequence. This possible sequence set work as the population set for genetic process. Based on this, complete genetic process is implemented and optimized test sequence and the optimized cost sequence is identified. The result is here tested for a test case with different number of test cases and the cost analysis is here done. This analysis result is shown in figure 3.

Here figure 4 is showing the analysis of test cost respective to number of test cases. As the test cases increases, the cost of test sequence generation also increases.

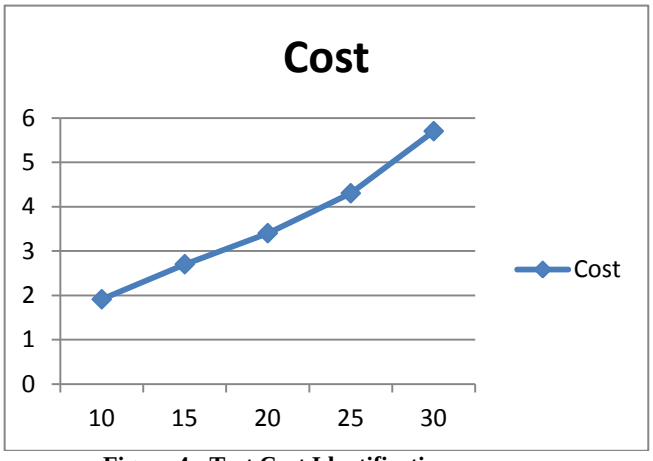


Figure 4 : Test Cost Identification

5. CONCLUSION

In this present work, a genetic based work is defined for generation of optimal test sequence for GUI testing. The work is tested on different test case sets and the cost analysis is here performed. In this present work genetic algorithm with fuzzy logic is used. Here fuzzy logic is used to find fitness function, and hence optimized test sequence is determined.

6. REFERENCES

[1] D. Richardson, O. O'Malley and C. Title [1989], "Approaches to specification-based testing".

[2] Akira K. Onoma, Wei-Tek Tsai, Mustafa H. Poonawala, and Hiroshi Suganuma [1998],"Regression Testing in an Industrial Environment".

[3] Elbaum, Malishevsky, Rothermel [2002],"Test case prioritization: a family of empirical studies".

[4] David Leon, Andy Podgurski [2003],"A Comparison of Coverage-Based and Distribution-Based Techniques for Filtering and Prioritizing Test Cases".

[5] Mark Last, Shay Eyal, and Abraham Kandel proposed a new [2005],"Effective Black-Box Testing with Genetic Algorithms".

[6] Xiaofang Zhang, Changhai Nie, Baowen Xu, Bo Qu [2007], "Test Case Prioritization based on Varying Testing Requirement Priorities and Test Case Costs".

[7] Gaurav Duggal, Bharti Suri [2008],"UNDERSTANDING REGRESSION TESTING TECHNIQUES".

[8] Gul Q, Tang B [2010]," Optimal Regression Testing based on Selective Coverage of Test Requirements".

[9] Tao C. [2010]," An Approach to Regression Test Selection Based on Hierarchical Slicing Technique".

[10] Chen L., Wang Z., Xu L. [2010],"Test Case Prioritization for Web Service Regression Testing".

[11] Bo Yang, Ji Wu, Chao Liu, Luo Xu [2010], "A Regression Testing Method for Composite Web Service".

[12] Emelie Engstrom[2010], "Regression Test Selection and Product Line System Testing".

[13] Wei Jin and Alessandro Orso [2010], "Automated Behavioral Regression Testing".

[14] http://en.wikipedia.org/wiki/Fuzzy_logic

[15] FULLY AUTOMATED GUI TESTING AND COVERAGE ANALYSIS USING GENETIC ALGORITHMS, International Journal of Innovative Computing, Information and Control ICIC International c 2011 ISSN 1349-4198Volume 7, Number 6, June 2011