ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 8, August 2014

A Tool To Automate The Test Cases Of Software Using Gray Box Testing Approach

Sahida Sultana[1], Mohd Sadiq[2], Waseem Ahmad[3]

M.Tech. Scholar, Department of Computer Science and Engineering, Faculty of Engineering and Technology, AL-Falah University, Dhuj, Faridabad, Haryana, India[1]

Computer Engineering Section, University Polytechnic, Faculty of Engineering and Technology, Jamia Millia Islamia (A Central University), New Delhi, India[2]

Department of Computer Science and Engineering, Faculty of Engineering and Technology, AL-Falah University, Dhuj, Faridabad, Haryana, India[3]

Abstract: The success rate of software system depends upon the following: requirements elicitation technique, modeling, analysis, verification, validation &testing. In literature, we have identified different types of Software Testing Techniques like, black box techniques, white box techniques, and gray box techniques; and choosing gray box testing is not an easy task according to need/criteria of the software projects. In our paper, we have described and compared the three most prevalent and commonly used software testing techniques and selection of gray box approach for detecting errors, which are the combination of: white box testing, black box testing.

Keywords: Requirement Prioritization, Black Box, Grey Box, White Box.

I. INTRODUCTION

Software testing identifies defect, flows or errors in the software. In literature, we have identified various definitions of software testing. Few of them are given below: (i) testing is the process of demonstrating that errors are not present (ii) The purpose of testing is to show that a program performs its intended functions correctly. The three most important techniques that are used for finding errors are functional testing, structural testing and gray box testing [6,7]. Functional testing is also referred to as black box testing in which contents of the black box are not known. Functionality of the black box is understood on the basis of the inputs and outputs in software. There are different methods which are used in black box testing methods like boundary value analysis, robustness testing, equivalence class partitioning, and decision table testing. White box testing or structural testing is the complementary approach of functional testing or black box testing. White box testing permits us to examine the internal structure of the program. In functional testing all specifications are checked against the implementation. This type of testing includes path testing, data flow testing, and mutation testing. In white box testing there are various applications of graph theory which is used to identify the independent path in a program or software like decision to decision (DD) flow graph, Cyclomatic complexity [6] etc.

Gray box testing is the testing of software application using effective combination of white box testing, black box testing, mutation, and regression testing [2]. This testing provides a method of testing software that will be both easy to implement and understand using commercial of the shelf (COTS) software [1]. In the Gray box testing, tester is usually has knowledge of limited access of code and based on this knowledge the test cases are designed; and the software application under test treat as a black box & tester test the application from outside. Gray box software testing methodology is a ten steps process for testing computer software. The methodology starts by identifying all the inputs and output requirements to computers systems. This information is captured in the software requirements documentation. The steps are given as follows: (i) Identify inputs (ii) Identify outputs (iii) Identify major paths (iv) Identify sub-function (SF) X (v) Develop inputs for SF X (vi) Develop outputs for SF X (vii) Execute test cases for SF X (viii) Verify correct results for SF X (ix) Repeat steps from 4 to 8 for other SF X and (x) Repeat steps 7 to 8 for regression [1].

II. LITERATURE REVIEW

Most of the work in literature is based on either black box testing or white box testing for example, in 2012; Khan, Bhatia, and Sadiq [8] develop a BBTool to generate the tests cases using black box testing. In a similar study, in 2011, Khan and Sadiq [7] analyze the various black box testing techniques. In literature, authors are trying to integrate the concepts of genetic algorithms with testing, for example, In 2011 Sabharwal et al. [9] proposed a technique for optimizing static testing efficiency by identifying the critical path clusters using genetic algorithm. The testing efficiency is optimized by applying the genetic algorithm on the test data. The test case

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940
International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 8, August 2014

scenarios are derived from the source code. The information flow metric is adopted in this work for calculating the information flow complexity associated with each node of the control flow graph generated from the source code. In 2009, Mohapatra et al. [5] used genetic algorithm to optimize the test cases that are generated using the category-partition and test harness patterns. In a similar study, Vieira et al. [11] proposed a GUI Testing Using a Model-driven Approach. The authors demonstrated and evaluated their method based on use cases that was developed for testing a graphical user interface (GUI).

Huang et al. [3] proposed repairing GUI test suites using a genetic algorithm. In this paper they develop a method to automatically repair GUI test suites, generating new test cases that are feasible. They use a genetic algorithm to evolve new test cases that increase our test suite's coverage while avoiding infeasible sequences. In 2007, Memon et al. [4] proposed an event flow model of GUI-based applications for testing. This paper consolidates all of the models into one scalable event-flow model and outlines algorithms to semi-automatically reverse-engineer the model from an implementation. Earlier work on model-based test-case generation, test-oracle creation, coverage evaluation, and regression testing is recast in terms of this model by defining event-space exploration strategies (ESESs) and creating an end-to-end GUI testing process. Three such ESESs are described: for checking the event-flow model, test-case generation, and test- oracle creation.

III. TESTING TECHNIQUES

Software testing identifies defects, flaws or errors in the application code that must be fixed. We can also define software testing as a process of accessing the functionality and correctness of a software through analysis. The main purpose of testing can be quality assurance, reliability estimation, validation and verification. Software testing is a fundamental component of software quality assurance and represents a review of specification, design and coding. The main objective of software testing is to affirm the quality of software system by systematically testing the software in carefully controlled circumstances, another objective is to identify the completeness and correctness of the software, and finally it uncovers undiscovered errors. [1] [2]

The three most important techniques that are used for finding errors are: [1]

1) White Box Testing Technique: It is the detailed investigation of internal logic and structure of the code. In white box testing it is necessary for a tester to have full knowledge of source code.

2) Black Box Testing Technique: It is a technique of testing without having any knowledge of the internal working of the application. It only examines the fundamental aspects of the system and has no or little relevance with the internal logical structure of the system.

3) Grey Box Testing Technique: White box + Black box = Grey box, it is a technique to test the application with limited knowledge of the internal working of an application and also has the knowledge of fundamental aspects of the system.

1. WHITE BOX TESTING TECHNIQUE

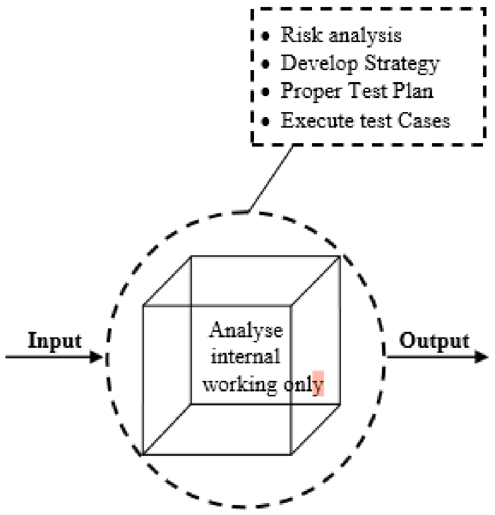

Figure 1. Represent white box testing

White box testing is a test case design method that uses the control structure of the procedural design to derive test cases. White box testing can uncover implementation errors such as poor key management by analyzing internal workings and structure of a piece of software. White box testing is applicable at integration, unit and system levels of the software testing process. In white box testing the tester needs to have a look inside the source code and find out which unit of code is behaving inappropriately. [3] Some of the advantages and disadvantages of white box testing technique are listed below: [3] [4]

Advantages

- It reveals error in hidden code by removing extra lines of code.
- Side effects are beneficial.
- Maximum coverage is attained during test scenario writing. Disadvantages
- It is very expensive as it requires a skilled tester to perform it.

- Many paths will remain untested as it is very difficult to look into every nook and corner to fin out hidden errors.
- Some of the codes omitted in the code could be missed out. Some of the synonyms of white box testing are glass box testing, clear box testing, open box testing, transparent box testing, structural testing, logic driven testing and design based testing. Some important types of white box testing techniques are briefly described below: [2]

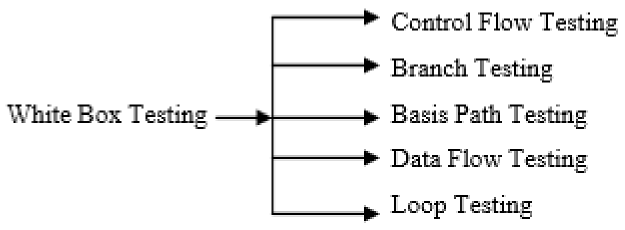

Figure 2. Represent different forms of white box testing techniques

1) Control Flow Testing: It is a structural testing strategy that uses the program control flow as a model control flow and favours more but simpler paths over fewer but complicated path.
2) Branch Testing: Branch testing has the objective to test every option (true or false) on every control statement which also includes compound decision.
3) Basis Path Testing: Basis path testing allows the test case designer to produce a logical complexity measure of procedural design and then uses this measure as an approach for outlining a basic set of execution paths.
4) Data Flow Testing: In this type of testing the control flow graph is annoted with the information about how the program variables are define and used.
5) Loop Testing: It exclusively focuses on the validity of loop construct

.

2. BLACK BOX TESTING TECHNIQUE

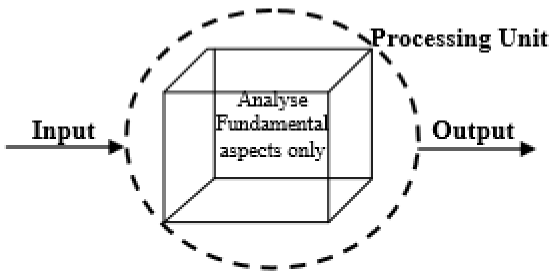

Figure 3. Represent black box testing

Black box testing treats the software as a “Black Box” – without any knowledge of internal working and it only examines the fundamental aspects of the system. While performing black box test, a tester must know the system architecture and will not have access to the source code. [5]

Some of the advantages and disadvantages of black box testing technique are listed below: [4] [5]
Advantages
- Efficient for large code segment.
- Tester perception is very simple.
- Users perspective are clearly separated from developers perspective (programmer and tester are independent of each other).
- Quicker test case development.

Disadvantages
- Only a selected number of test scenarios are actually performed. As a result, there is only limited coverage.
- Without clear specification test cases are difficult to design.
- Inefficient testing. Some of the synonyms of black box testing technique are opaque testing, functional testing, close box testing, and behavioural testing. Some important types of black box testing techniques are briefly described below: [5]

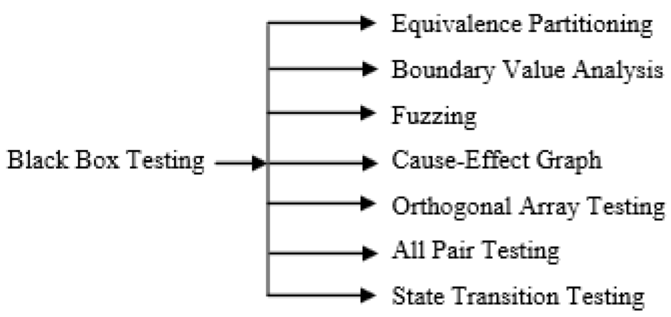

Figure 4. Represent differnet forms of black box testing techniques

1) Equivalence Partitioning: It can reduce the number of test cases, as it divides the input data of a software unit into partition of data from which test cases can be derived.

2) Boundary Value Analysis: It focuses more on testing at boundaries, or where the extreme boundary values are chosen. It includes minimum, maximum, just inside/outside boundaries, error values and typical values.
3) Fuzzing: Fuzz testing is used for finding implementation bugs, using malformed/semi-malformed data injection in an automated or semi-automated session.
4) Cause-Effect Graph: It is a testing technique, in which testing begins by creating a graph and establishing the relation between the effect and its causes. Identity, negation, logic OR and logic AND are the four basic symbols which expresses the interdependency between cause and effect.
5) Orthogonal Array Testing: OAT can be applied to problems in which the input domain is relatively small, but too large to accommodate exhaustive testing.
6) All Pair Testing: In all pair testing technique, test cases are designs to execute all possible discrete combinations of each pair of input parameters. Its main objective is to have a set of test cases that covers all the pairs.
7) State Transition Testing: This type of testing is useful for testing state machine and also for navigation of graphical user interface.

3. GREY BOX TESTING TECHNIQUE

Grey box testing technique will increase the testing coverage by allowing us to focus on all the layers of any complex system through the combination of all existing white box and black box testing.

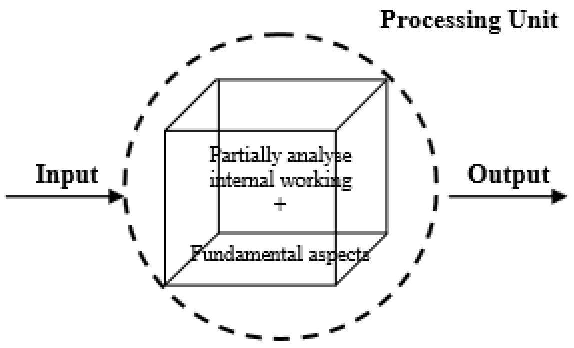

Figure 5. Represent grey box testing

In grey box testing the tester must have knowledge of internal data structures and algorithm, for the purpose of designing test cases. Examples of grey box testing technique are: [6]

- Architectural Model
- Unified Modeling language (UML)
- State Model (Finite State Machine)

In grey box testing the codes of two modules are studied (white box testing method) for the design of test cases and actual test are performed in the interfaces exposed (black box testing method).

Some of the advantages of grey box testing technique are listed below: [4] [6]

- Grey box testing provides combined benefits of white box and black box testing techniques.
- In grey box testing, the tester relies on interface definition and functional specification rather than source code.

- In grey box testing, the tester can design excellent test scenarios.
- The test is done from the user's point of view rather than designer's point of view.
- Create an intelligent test authoring.
- Unbiased testing.

Some of the disadvantages of grey box testing technique are listed below:

- Test coverage is limited as the access to source code is not available.
- If the software designer has already run a test case, the tests can be redundant. The other name of grey
- It is difficult to associate defect identification in distributed applications.
- Many program paths remain untested.

box testing is translucent testing. Different forms of grey box testing techniques are briefly described below: [6]

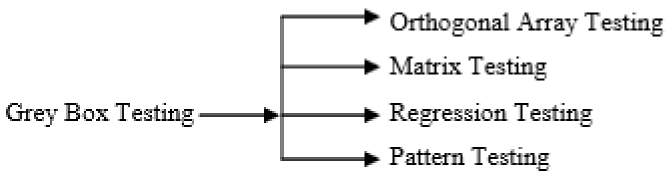

Figure 6. Represent different forms of grey box testing techniques

1) Orthogonal Array Testing: This type of testing use as subset of all possible combinations.
2) Matrix Testing: In matrix testing the status report of the project is stated.
3) Regression Testing: If new changes are made in software, regression testing implies running of test cases.
4) Pattern Testing: Pattern testing verifies the good application for its architecture and design.

IV. ANALYTIC HIERARCHY PROCESS

In 1972, T. L. Saaty proposed the analytic hierarchy process [28]. It is a multi-criteria decision (MCDM) making method. AHP helps decision maker facing a complex problem with multiple conflicting and subjective criteria [1, 28]. This process permits the hierarchical structure of the criteria or sub-criteria when allocating a weight. AHP has been widely used in various fields like banks, manufacturing systems, software evaluation, requirements prioritization [17, 18, 19], evaluation of web site performance etc. AHP involvesfollowing steps: (a) problem definition (b) pair-wise comparisons (c) compute the eigenvector of the relative importance of the criteria (d) check consistency. Once we have identified the criteria or sub-criteria according to the need of the problem or **problem definition**, then the next step is to express the decision makers opinion on only two alternatives than simultaneously on all the alternatives. On the basis of the pair wise comparison with all the alternatives, we construct the **pair-wise comparison matrix** on the basis of the following rating scale **(Judgment scale)**.

Table: 1 the Saaty rating scale

Intensity of importance	**Definition**
1	Equal importance
3	Somewhat more importance
5	Much more important
7	Very much important
9	Absolutely more important
2,4,6,8	Intermediates values (when compromise is needed)

There are several methods or algorithm for the calculation of eigenvector. In this paper, we adopt the following algorithm:

Algorithm:

Step 1: Multiplying together the entries in each row of the matrix and then take the nth root of the product.
Step 2: Compute the sum of nth root and store the result in SUM.
Step 3: The value of SUM would be used to normalize the product values and the resultant would be the eigenvector

Saaty argues that a Consistency Ratio (CR) >0.1 indicates that the judgment are at the limit of consistency, where as CR=0.9 would mean that the pair wise judgment are random and are completely untrustworthy [1, 28]

V. PROPOSED METHOD

This section presents a method for the selection of Software Testing Techniques using AHP. The proposed presented simply in the following method is:

(i) Identify the criteria
(ii) Construct the hierarchical structure of Software Testing Techniques
(iii) Construct the decision matrix
(iv) Calculate the ranking values
(v)) Selection of a Software Testing Techniques

(i) Identify the criteria

Before the selection of any Software Testing Techniques, tester should identify the criteria's the selection of an Software Testing Techniques. On the basis of our literature review, we have identified the following factors which influence the decision of choosing a software testing methodology:

(a) New or existing software
(b) Cost of requirements
(c)Test cases Identify
(d) Independent path

ii) Construct the hierarchical structure of Software Testing Techniques

As the Software Testing Techniques selection decision requires a systematic approach to help integrate different attributes or criteria into software project testing. Therefore, it is essential to break down the problem into more manageable sub-problems. As illustrated in Fig.1, the problem studied here has three level of hierarchy. The first level, i.e., the overall objective, is the selection of *an* Software Testing Techniques. Level two contains three different Software Testing Techniques, and at level three decision criteria is given.

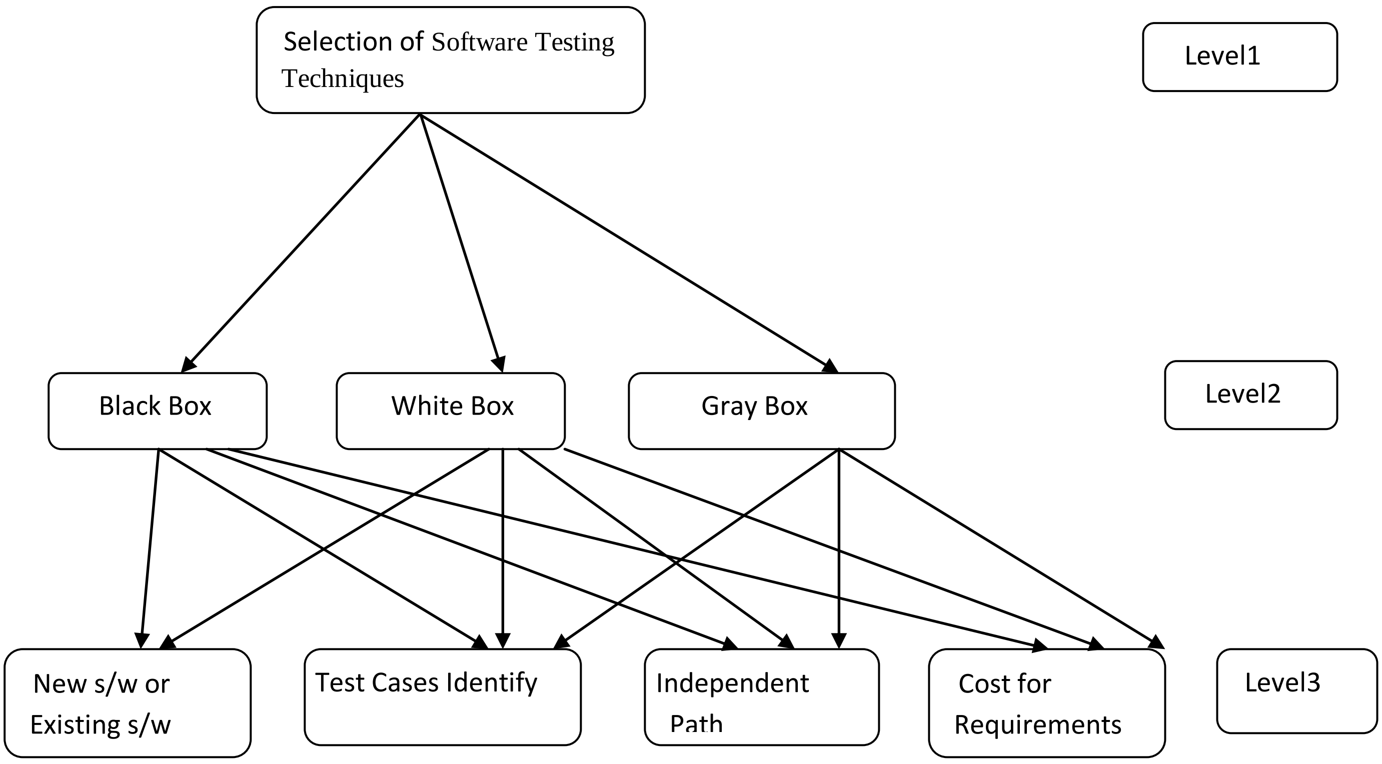

Fig.1 Hierarchical Structure of the Software Testing Techniques selection problem

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 8, August 2014

VI. CONCLUSION

This paper presents a method for the selection of Software Testing Techniques using AHP. Proposed method is a four step process, namely, (i) identify the criteria, (ii) construct the hierarchical structure of Software Testing Techniques, (iii) construct the decision matrix, and (iv) the selection of a technique. Proposed method selects the agile methods for the testing of the project. On the basis of our analysis, we identify that there is a need to improve the agile methods by intertwining of decision making approaches for the selection and prioritization of requirements. Future research agenda includes the following:

1. To improve the analysis phase of adaptive process model for agile development by applying TOPSIS method.
2. To propose a fuzzy decision making approach or the selection of Software Testing Techniques.
3. To propose a hybrid approach of Software Testing Techniques.
4. To propose a method for the selection of Software Testing Techniques using hybrid techniques like fuzzy AHP and fuzzy ANP.

REFERENCES

1. Coulter A, "Gray Box Software Testing Methodology", White paper, Version 0.8.
2. Coulter Andre, "Gray box Software testing Methodology-Embedded software testing technique", 18th IEEE Digital Avionics Systems Conference Proceedings, pp. 10.A.5-2, 1999.
3. Huang et al, "Repairing GUI test Suites using Genetic Algorithms".
4. Memon A, "An Event Flow Model of GUI based Applications for Testing", Software Testing Verification, and Reliability, Wiley Inter Science, pp. 137-157, 2007.
5. Mohapatra, Bhuyan, and Mohapatra, "Automated Test Cases Generation and Its Optimization using Genetic Algorithm and Sampling", IEEE International Conference on Information Engineering, 2009.
6. Mohd. Sadiq**,** "Application of Graph Theory to Software Engineering", South East Asian Journal of Mathematics and Mathematical Sciences, India, Vol.3, No.3, pp 53-57, 2005.
7. Mumtaz Ahmad Khan and Mohd. Sadiq, "Analysis of Black Box Software Testing Techniques: A Case Study", IEEE International Conference and Workshop on Current Trends in Information Technology, pp.1-5, December, 2011, Dubai, UAE.
8. Mumtaz Ahmad Khan, Preeti Bhatia, and Mohd. Sadiq**,** **"**BBTool: A Tool to Generate the Test Cases", International Journal of Recent Technology and Engineering, Vol. 1, Issue 2, pp. 192- 197, June 2012.
9. Sabharwal, Sibal, and Sharma, "A Genetic Algorithm based Approach for Prioritization of Test Cases Scenarios in Static Testing", IEEE International Conference of Communication and Technology, 2011.
10. Sharma, Sabharwal, and Sibal, " A Survey on Software Testing Techniques using Genetic Algorithms", IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 1, No.1, 2013.
11. Viera et al., "Automation of GUI testing Using Model – Driven Approach", ACM- AST, China, 2006.