A COMPARATIVE STUDY OF SOFTWARE TESTING TECHNIQUES

G.Manivasagam[1], Dr.R.Gunasundari[2], B.Brindha[3]

ABSTRACT

Software Testing is a method of finding errors while executing a program. So, we look for a zero-defect software system. The main aim is to evaluate the usability of a program. Some researchers and software professionals use 50% of the time, cost and effort in testing the software. The most important task to test software is generating test cases. By using various testing tools testing is done either automatically or - manually. For generating test cases like fuzzy logic, finite state machine, neural networks, genetic algorithms, soft computing, genetic programming and evolutionary computation there are different techniques available. This paper mainly focuses on various testing techniques.

I. INTRODUCTION

Software testing is an analysis of the software quality and depending on the testing outcome the information is provided to the investors to understand its quality. It can also afford an unbiased, self-determining view about the software which allows the business to understand the risk issues and advantages in the implementation of the software. In general testing techniques never limit or control the program execution process with the intention of finding errors or defects in the software [1].

Software testing is defined as the procedure of authorizing and confirming that the software concerned has met the following criteria's [2]:

- Fulfills the necessities which guides its design and development
- It functions as anticipated,
- With the same characteristics it can be used
- Fulfills the requirement of the investors.

Depending on the methodologies used for testing the software can be applied several times during the software development process. But in conventional method requirement for the software is defined and testing efforts are also taken on coding procedure. Depending on the software development methodology the testing methodology varies. The task of the software testing extends to fixing significant errors, flaws or defects in application code.

The software testing process involves three main factors namely verification process, validation process and finding of defects

- Process of verification

During the validation whether the software accomplishes the specification of technical

[1]Assistant Professor, Department of CS, CA & IT, Karpagam Academy of Higher Education, Coimbatore, Tamil Nadu, India

[2]Assosciate Professor, Department of CS, CA & IT, Karpagam Academy of Higher Education, Coimbatore, Tamil Nadu, India

requirements is checked. Here specification means determining whether for a given input the software produces the expected output or not under various conditions.

- Process of validation

The validation metric is used to determine whether the requirement of the business is fulfilled by the software [3].

- Finding of Defects

The variation among the actual and expected output of the software is known as defect. The source of defect can be identified by tracing the cause specification, design or coding phase.

2. Software Testing Techniques

Different software testing techniques and strategies are applied in various applications. This section discusses the basic dissimilarities among various approaches to testing software [4].

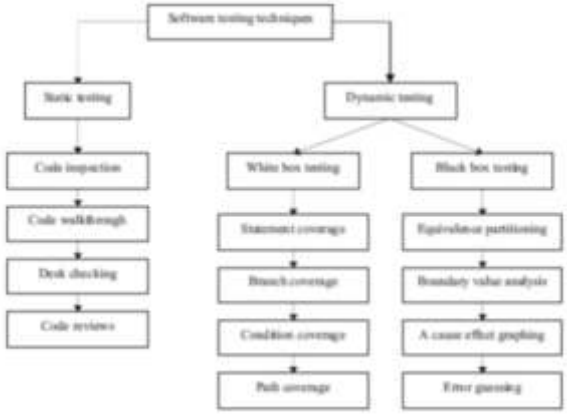

Fig 2.1 Overview of Software testing techniques

Software testing is mainly divided into two core categories. They are static testing and dynamic testing [4].The main approach of static testing is reading the program source code statement by statement and following the logical program flow visually by passing an input. This kind of testing is highly reliant on the skill of the reviewers. For visual review it uses the requirement of program and documents design. While in contrast the technique of dynamic testing executes the program which is under test.

2.1 Static Testing

Earlier most of the programmers assumed that programs were only for machine execution and not for humans, so that the only way to test a program was by justifying the output after its execution machine. This trend changed in the 1970s and static analysis- based methods were started to be applied amid code completion and starting of execution based-testing. Some of the methods of static analysis were inspection of code, walkthroughs of code, checking desk and reviewing code. The two primary methods in static analysis are inspection of codes and walkthroughs. It was done by a team of people in which these methods involved preparatory work by the participants. During the meeting errors were determined and resolved by a debugging process.

Code Inspections

It is a sequence of procedures and detection of errors for reading the code. It focuses on procedures, filling the forms, etc. Two different activities are done during inspection; they are narration and examination of code. According to the check list, history codes are read and analyzed statement by statement.

Code Walkthroughs

The starting process is same as inspection process, but the difference is that instead of just reading the program against the checklist a person is assigned as a tester. The

tester will have to test cases that represent a set of inputs and expected output for the tested module. Meeting session will be held, during the session each test case is executed by walkthrough of the logic of the program and the values of the variables.

Desk Checking

It can be viewed as a single person inspection. A person involves himself in reading a program and checking it against the error list and test data through walks. The disadvantages of this method are lack of discipline, effective and competition like team work.

Code Reviews

It is a kind of evaluation of the anonymous programs based on their quality, maintainability, usability, extensibility and clarity. These programs are rated by the group of programmer in terms of certain scale given in the review forms.

2.2 Dynamic Testing

The program is tested by executing it under a test input data and the output is checked in dynamic testing. Mostly testing refers to dynamic testing which is of two types; they are black box and white box testing.

White Box Testing

The most widely used technique is white box testing, which is also referred to as structural testing or logic coverage testing because it evaluates the program structure [2, 3, 4]. The aim of white box testing is to workout various logic structures and program flows. The testing criteria followed in white box testing are [5]

- Statement testing

Each statement in the software for testing has to be executed at least once. It can be applied to the object code directly.

- Branch testing

It is stronger than the statement coverage. In this all the possible decisions have to be worked out at least once for finding all types of outcome i.e., all the transfer of control has to be executed including coverage of statement. Anyway, some of the bugs can be identified if the statements and branches are deployed in a specific order which results in path testing.

- Condition coverage

In this case the test cases should be produced so that each condition in a decision takes all possible outputs at least once.

- Path testing

In this testing all possible path in the software to be tested is executed and it leads to an increase in the probability of detection of errors and it is much stronger than branch and statement testing. It can be defined as a conjunction of software's input and conjunction of predicates [6, 7]. Path is set to be feasible only when there exists an input for which the path is travelled or else the path is considered unfeasible.

Black Box Testing

In this black box testing, the functionalities of the software, but not its structure, are tested. There are four dissimilar kinds of black box testing. They are boundary value analysis, equivalence partitioning, error guessing and cause-effect graphing.

- Equivalence partitioning

It panels the input domains into limited set of

equivalence classes. Likewise one can assume that a characteristic value of each class is equal to the test of any other values appropriate to the respective class. It can also be applied for white box testing also.

- Boundary value analysis

The situations when the classes of test classes on the ends of input and output equivalence classes. The test cases which lie on the boundaries have higher payoff than other test cases.

- Cause-effect graphing

This method is used to translate the natural language specification to formal language which denotes the incompleteness and ambiguities in the specification. In this simpler notation is used to represent the digital logic circuit.

- Error guessing

It is an ad-hoc process in which its process is very difficult to formalize. This kind needs the assistance of the experts to identify the errors. The general ideas of error guessing are to compute a list of possible errors and then write test cases based on the list obtained.

Conclusion

There are a number of test-case generation techniques that vary from application to application, based on the specification and the programming language. The execution of testing is based on test cases. Further research is required to implement the most recent algorithm like Genetic Algorithm in combination with some other techniques like UML, traversal algorithms on Object oriented programming to optimize test case generation and improve the overall software testing process. This paper has given a brief description of about various variety of testing techniques that we are able to apply in measuring numerous quality attributes.

References :

1. Atanassov, K. Intuitionistic Fuzzy Sets. Heidelberg, Springer Physica-Verlag, 1999
2. Atanassov, K. Two Variants of Intuitionistic Fuzzy Propositional Calculus. Preprint IM-MFAIS-5-88, Sofia, 1988
3. A.Kalousis, J. Prados, M. Hilario, "Stability of Feature Selection Algorithms: a study on high dimensional spaces," Knowledge and information System, vol. 12, no. 1, pp. 95-116, 2007. Article (CrossRef Link)
4. AbinashTripathy and AnirbanMitra, "Test Case Generation Using Activity Diagram and Sequence Diagram" Proceedings of ICAdC, AISC 174, pp. 121-129. springerlink.com © Springer India 2013
5. Ahmad A. Saifan, Test Case Reduction Using Data Mining Classifier Techniques,Journal of software, Volume 11, Number 7, July 2016, 656-663
6. Ahmet Okutan and OlcayTanerY?ld?z, (2013), "A Novel Regression Method for Software Defect Prediction with Kernel Methods", ICPMRA 2013 - International Conference on Pattern Recognition Applications and Methods, pp 216-221.
7. Ahmet Okutan1 and OlcayTanerY?ld?z, (2013), "A Novel Regression Method for Software Defect Prediction with Kernel Methods", ICPMRA 2013 - International Conference on Pattern Recognition Applications and Methods, pp 216-221.