Software Testing Techniques

Sundram Trivedi[1] Shalini Sharma[2] Dr. Ramswaroop[3]
[1,2,3]Department of Information Technology
[1,2,3]Arya Institute of Engineering & Technology, Jaipur, Rajasthan, India

Abstract— Software made will never be impeccable without being tried and endorsed. This makes us find a way of testing hidden bugs at different levels by a good technique. Testing is a process where we can spot hidden bugs, code errors or any unused code in given software. Currently, many techniques being used for testing errors in software, among them most practiced techniques for testing software are black-box testing, white-box testing, and grey-box testing. In this paper, we are comparing various software testing techniques to find out their effectiveness in generating test cases and enhancing the quality of the software system.

Keywords: Software Engineering & STLC, Level of Testing, Software Testing Strategy, Testing Techniques/Methods, Conclusion & Future Work

I. Introduction

A. Software Engineering

It is defined as a discipline for developing a high-quality system that allocates with the software development of the software product that uses the clear-cut methods, techniques, sub-routines, and procedures. [1] According to IEEE's definition, Software Engineering can be defined as "The application of a systematic, well-defined, disciplined and quantifiable approach to the development, and maintenance of software and the study of these approaches that are considered as the application of engineering to software". Software Engineering is the procedure of making, testing, and documentation of the programs of the computer. [2]

B. Software Development Life Cycle

SDLC, Software Development Life Cycle is the task that is being used by the industry of software that helps to design, develop and test high-quality software. [1] This concept of SDLC is applied to the limit of both hardware and software configurations as we know that the system is comprised of hardware only, software and the combination of both the configurations. SDLC authorizes the set of varied activities that are to be followed and designed to develop a software package effective and efficient. The substructure of this includes the list of steps:

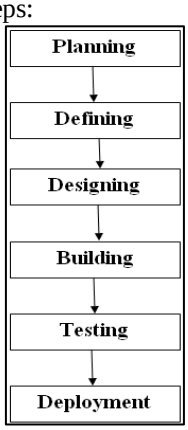

Fig. 1: Stages of SDLC

II. Levels of Testing

A. Unit Testing

Unit testing is the smallest testable part of an entire application. It is wont to provide a bit of code that has got to satisfy the wants.

B. Integration Testing

In integration testing, the code is divided into individual segments and tested as a group. The main task of integration testing is to analyze the parameters such as functional requirements, performance requirements and reliability requirements which are placed on major design items.

C. Function Testing

Functional testing can be referred to as black-box testing. In functional testing, testing is done by providing valid input and thus outcomes are observed accordingly.

D. System Testing

It is considered as a more limited type of testing, it seeks to detect any defects within the software units that are integrated.

E. Acceptance Testing

It is also known as operational acceptance testing or field acceptance testing because it runs by following the predefined acceptance test procedures to direct the user about which information is to be utilized after following bit by bit technique.

F. Regression Testing

In regression testing, the applications are tested which were previously developed and analyzes if the deviations occurred when the changes are made in existing or new programs.

III. Software Testing Strategies

Various testing strategies that are being used for testing:

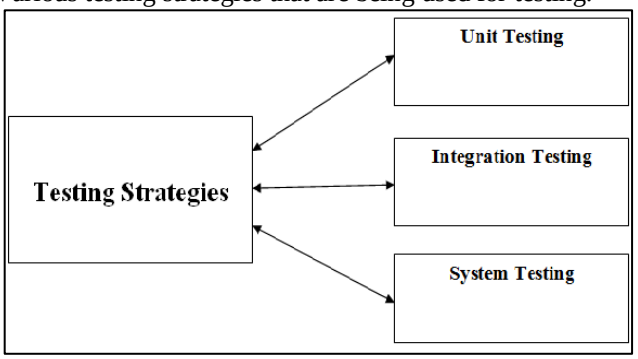

Fig. 2: Software Testing Strategies

A. Unit Testing

This sort of testing is performed at rock bottom level by the developers before it's moved to the team of testing to

execute the test cases. [5] It is the smallest module that can be tested and verified at each section or line of code. This output of 1 module becomes the input of another module. If the output of any one of the modules fails so then the output to which we give the input also fails. [1] So, therefore it is nevertheless better to test each module differently so that there would be less chance of fails. In this, the white box testing method is implemented.

1) When the Unit Testing is accomplished

It is being accomplished before Integration Testing.

2) By whom Unit Testing is accomplished

It is accomplished by developers of software and their peers or very rarely by the Tested who are independent.

B. Integration Testing

Integration Testing is performed immediately after the Unit Testing. In this, all the modules are merged to form a larger module and determine are they functioning properly and then the testing is implemented on the modules. [3] Testing is done so that in case any bug remained in the Unit Testing it can be again tested in this testing to remove all the bugs. The basic idea of integration testing is to test how different parts of the system are grouped or work together. [4] For example, a unit test for database access code wouldn't be ready to ask a true database but the mixing testing would.

Testing is classified into two parts:

– Top-Down Testing
– Bottom-Up Testing

1) When the Integration Testing is accomplished

It is being accomplished after Unit Testing and before System Testing.

2) By whom Integration Testing is accomplished

It is accomplished by either the developers or by the Tester who are independent. Mainstream Tools for Integration Testing are Mocha, Tape, and Jasmine.

C. System Testing

This type of testing is conducted to test the entire system. It is needed to check all the integrated components to check and verify whether it meets the wants and therefore the standards of quality. [2] The basic purpose of the testing is to assess the compliance of the system within the desired specific requirements. In this, the black-box testing method is implemented. [1]

1) When the System Testing is accomplished

It is being accomplished after Integration Testing and before Acceptance Testing.

2) By whom System Testing is accomplished

It is accomplished by the Testers who are independent. For example: When the pen is fabricated, the body, cap, cartridge are tested differently than unit testing is performed. When quite two units are organized, all of them are combined and integration testing is performed. When the entire complete pen is consolidated then system testing is performed.

IV. TESTING TECHNIQUES/ METHODS

There are different strategies or systems for testing the product:

– Black Box Testing
– White Box Testing

A. Black Box Testing

In this type of testing, the intramural structure/ details of the data item are not known by or accessible to its user.[10] Right now are produced or structured from the Input / Output esteem just and no information on configuration/code is being required. The testers are only conscious of knowing about what's assumed to try to and to not skills it does. These types of tests can be functional or non-functional. [6]

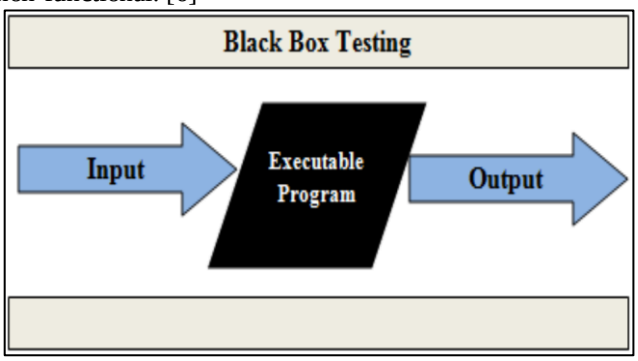

Fig. 3: Black Box Testing

Black Box Testing is named so because as we know that in the tester's eyes it is named black box but the inner side no one sees. Black Box Testing is also known as Functional testing, Specificational, Behavioral, Data-Driven or Input-Output Driven. [8]

For Example, without the recognition of the inner details of the website, we test the pages of the web by the use of the browser, authorize the input and then test and verify the outputs against the outcome that is expected. [7]

There are many test cases in recorder Testing:

1) Equivalence Class Partitioning
2) Boundary Value Analysis
3) Cause-Effect Graph

V. COMPARISON TESTING

A. Equivalence Class Partitioning

This type of technique partitions the program input domain into the set of equivalence classes from where we can derive the test cases. This partition is done in such a way that a program's behavior is the same as every input data that is belonging to the similar equivalent class. [10] The main idea behind the defining of the equivalent class is to test the code with only one value that belongs to the equivalence class is as better as testing the software with some other value that belongs to that equivalence class. [9]

For Example ≤ 1-500 501 and above

1) Boundary Value Analysis

It is complementary to partitioning the equivalence class instead of selecting the arbitrary input value to partition; the equivalence class chooses the values at the end of the class. [2]

For Example, a Programmer may improperly use ≤ rather than 5000. So, therefore, 2 partitions required. Boundary Value Analysis= [-1, 0, 5000, 5001]

2) Cause-Effect Graph

It is a technique of software test design that includes identifying the cases (Input conditions) and the effects (Output conditions). A weakness of the above mentioned 2 methods is that they don't consider the potential combination of input and output condition. It associates the info classes (causes) to yield classes (impacts) yielding a coordinated diagram. It utilizes 4 symbols: NOT, OR, AND, IDENTITY.

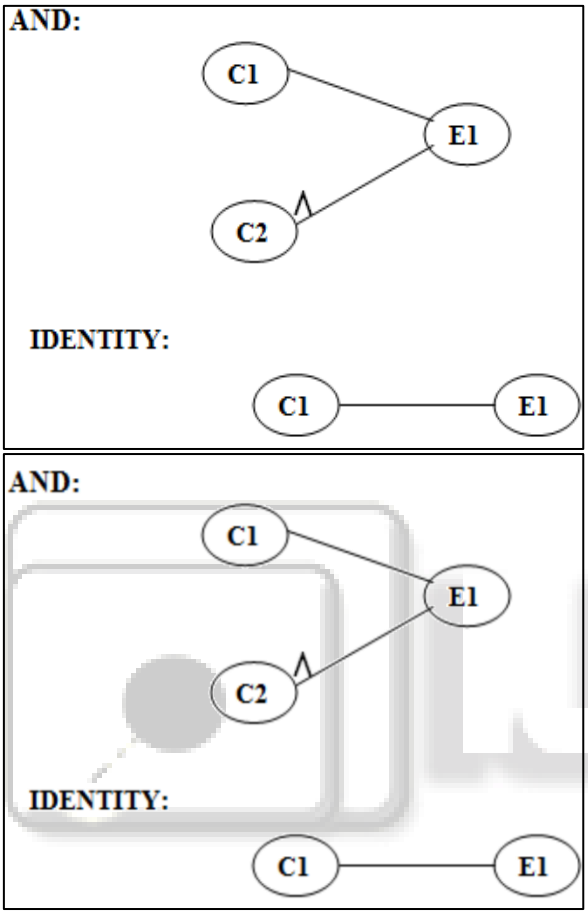

Fig. 5: Cause-Effect Graph

3) Comparison Testing

For critical applications that are required the fault tolerance, independent version of the software are developed for the similar specification [3] [4] if the output for each version is same then it is presumed that all the implementations are correct but if output is unique each version is examined to check what is responsible for the different output.

a) Advantages of Black Box Testing

- Testing is being performed from the viewpoint of the user.
- Tester and Programmer both are autonomous to every other.
- Test cases are often designed immediately after the completion of specifications.
- Testers don't know about the languages of programming or how the software has been accomplished. [11]

b) Disadvantage

- Not well suited and efficient test.
- There are chances of duplication of tests that are already tested by a programmer.
- Test cases are difficult to style without clear requirements.
- Some parts that are at the back end are not tested at all.

B. White Box Testing

In this type of testing, the intramural structure/ details of the data item are known by or accessible to its user. In this, test cases are being made based on the code.[3] Programming very well knows about how the implementation of knowledge is significant.

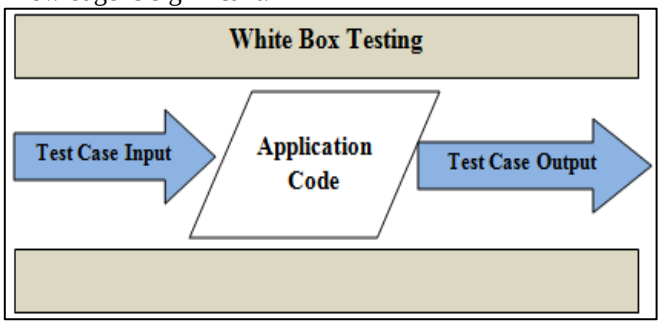

Fig. 6: White Box Testing

White Box Testing is named so because as we know that in the tester's eyes it is named white box and inner side everyone sees perfectly.[8] White Box Testing is also referred to as Glass Box, Structural; Clear Box, Open Box, Logic Driven, or Path Oriented.

For Example, a tester and a developer study the code implemented of any field on a webpage decides purposefully all the legal and the illegal inputs and verify the output for the outcome that is expected. And also decides by studying the code that is implemented. So, therefore we can say that white box testing is like the work of a mechanic who only needs to know why the car is not working correctly. [9]

Strategies applied to white box testing are:

- *Ø Unit Testing*

Within the units, the ways are tried.

- *Ø Integration Testing*

Between the units, the ways are tried.

- *Ø System Testing*

Between the subsystems, the ways are tried.

For white box testing, unit testing is material. There are many experiments in White Box Testing:

1) Statement
2) Branch
3) Condition
4) Path
5) Data Flow
6) Mutation
7) Domain and Boundary Testing
8) Loop Coverage Testing
9) Logic-Based
10) Fault Based

1) Statement

This is the easiest and simplest form of white box testing where there is no way to check where a series of test cases are run in such a way that every statement is being executed a minimum of once. [14] The idea here is that we have no way to examine an error existing in the statement or not.

For Example: if (S>1 && t==0) x=9; Therefore, 2 test cases are formed, one is true and therefore the other is fake.

2) Branch/ Edge testing

In this test cases are generated to form each branch condition presuming true and false values in turn. [12]

For Example: if (……..&&……..) // Full condition is checked whether it's true or false.

3) Condition

In this test cases are designed to form each component of a composite conditional expression. [13] In this part, we verify only the part of the condition.

a) it's the stronger testing than the branch testing.
b) Branch testing is stronger than statement coverage testing.
c) Conditional coverage requires 2n test cases.

4) Path Coverage

In this test cases are designed in such a way that all the linearly independent paths of the control flow graph of the program are executed at least once. [8]

For Example:

It describes how the flow of control flow through the program & describes the sequence. [12] A path that is a subpath of the main path that is not considered as a linear independent path.

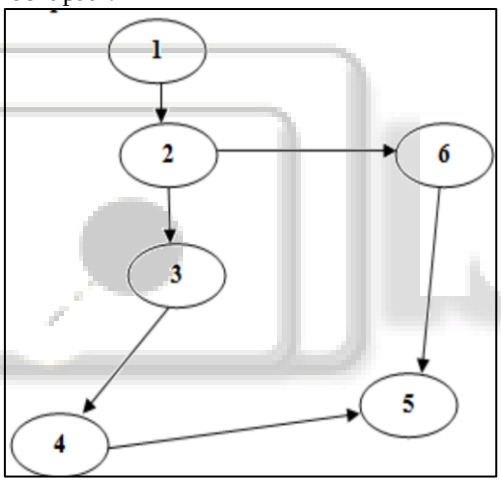

Fig. 7:

McCabe's Cyclomatic Complexity:

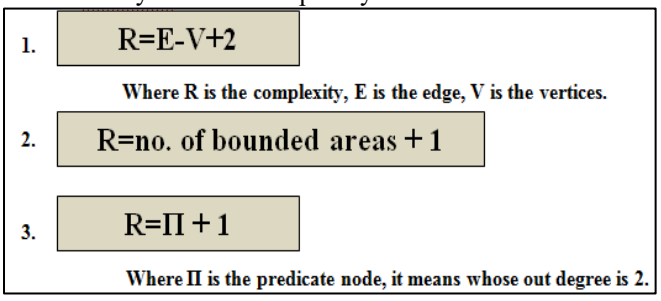

Fig. 8: McCabe's Cyclomatic Complexity

C. Advantages of White Box Testing

- We need not wait for the GUI to be implemented as testing is began at the very first stage.
- Help in code optimization.
- Disadvantage:
- Costly because a skilled tester is required.
- The cases which are omitted in code are missed out.
- It is very difficult to find out the hidden errors by looking at every bit of code. This may create problems and fail software.

VI. Conclusion & Future Work

Individuals have called for more attention to engineering principles and sounder education for software engineers for many years. I have tried to offer some concrete suggestions for how we might improve software engineering education, by identifying some skills that every software engineering student and faculty should have learned, as well as hands-on training that they should have had. I have also pointed out the following areas that the research community needs to focus on to meet the demands of the types of systems that are being built today and will increasingly be built in the future.

- Testing embedded systems
- Testing properties aside from functionality, including performance, safety, and security
- Simulation
- Industrial grade empirical studies
- Easy-to-use tools that implement testing techniques

References

[1] Yadav, P; Kumari, P; (2015) "REVIEW PAPER ON SOFTWARE TESTING", http://www.ijirt.org/vol1/paperpublished/IJIRT102003_PAPER.pdf.
[2] Worwa, K; (2016) "LOGISTICAL ASPECTS OF THE SOFTWARE TESTING PROCESS", http://www.research.logistyka-produkcja.pl/images/stories/Numer_22/10.21008j.2083-4950.2016.6.2.5.pdf.
[3] Bertolino, A; (2007) "Software Testing Research: Achievements, Challenges, Dreams", http://selab.netlab.uky.edu/homepage/sw-test-roadmap-bertolino.pdf.
[4] Chauhan, R; Singh, I; (2014) "Latest Research and Development on Software Testing Techniques and Tools", http://inpressco.com/wp-content/uploads/2014/07/Paper122368-2372.pdf.
[5] Irena, J; (2008) "Software Testing Methods and Techniques", http://tir.ipsitransactions.org/2009/January/Paper 06.pdf.
[6] Kaur, M; Singh, R; (2014) "A Review of Software Testing Techniques", https://www.ripublication.com/irph/ijeee_spl/ijeeev7n5_05.pdf.
[7] Kaur, K; Sharma, S; (2015) "A Survey on Software Testing", http://www.ijettcs.org/Volume4Issue6/IJETTCS-2015-11-21-31.pdf.
[8] Kaushik, S; Tyagi, K; (2016) "Critical Review on Test Case Generation Systems and Techniques", http://www.ijcaonline.org/research/volume133/number7/kaushik-2016-ijca-907916.pdf.
[9] Luo, L; (2015) "Software Testing Techniques Technology Maturation and Research Strategies",

http://www.cs.cmu.edu/~luluo/Courses/17939Report.pdf.
[10] Available from http://technav.ieee.org/tag/1579/software-testing
[11] Sharma, C; Sibal, R; (2013) "A Survey on Software Testing Techniques using Genetic Algorithm", https://pdfs.semanticscholar.org/3c1d/f844a948f1401a253e1aeaa453edefc60c96.pdf
[12] Tarika, B; (2014) "Review on Software Testing Techniques", http://www.ijritcc.org/download/Review on Software Testing Techniques.pdf.
[13] Bertolino, A; (2010) "Software Testing Research and Practice", http://www.cis.upenn.edu/~lee/05cis700/papers/Ber03.pdf.
[14] Singh, S; Rakshit, M; (2013) "A Review of Various Software Testing Techniques", http://www.ijreat.org/Papers 2013/Issue4/IJREATV1I4001.pdf

