

A Review on Structural Software Testing Coverage Approaches

Mini Singh^[1], Dr. Devesh Katiyar^[2], Mr. Gaurav Goel^[3]
[1] Student of MCA, [2] Assist. Professor, [3] Assist. Professor
Department of Computer Science

Dr. Shakuntala Misra National Rehabilitation University, Mohaan Road, Lucknow

ABSTRACT: Testing is broadly classified into three levels: Unit Testing, Addition Testing, and System Testing. Whenever we think of developing any software we always concentrate on making the software bug free and most reliable. At this point of time Testing is used to make the software a bug free. Software Testing has been measured as the most important stage of the software development life cycle. Around 60% of resources and money are cast-off for the

Keywords: Software Testing, Software Testing Strategies, Testing Techniques, Test Cases.

1. INTRODUCTION

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

1.1 Software Development Life Cycle: SDLC, Software Growth Life Cycle is the task that is being used by the industry of software that helps to design, develop and test the high-quality software. This concept of SDLC is applied to the limit of both hardware and software shapes as we know that system is comprised of hardware

testing of software. Testing can be manual or automated. Software testing is an activity that emphasizes at assessing the competence of a program and commands that it truly meets the excellence results. There are many test cases that help in detecting the bugs so, in this paper we describe about the most commonly used test cases and testing techniques for the error detection.

only, software only and the combination of both the configurations. SDLC authorizes the set of various activities that are to be followed and designed to develop a software product effective and efficient.

SDLC authorizes the set of various activities that are to be followed and designed to develop a software product effective and efficient.

The substructure of this includes the list of steps:

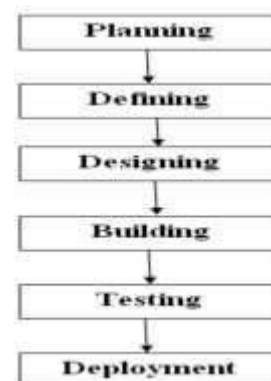


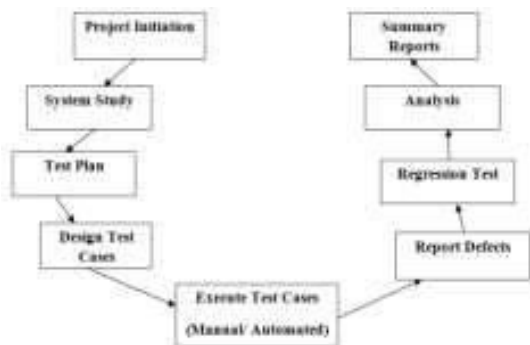
Fig. 1 Stages of SDLC

Software Testing: Software Testing plays a very important task in SDLC. It is an appraisal of the software that is against the requirements that are collected from the system and the user specifications.[2].It is defined as the process of executing the program with the purpose of finding the bugs and a procedure to test the code of computer that it does for what it is designed for. [7] According to Dale Emery and Elisabeth Hendrickson, Testing is defined as “It is a process of meeting information by making comments and comparing them to the expectations.

By whom Testing is done-Testing is being done by all those who are intricate to the software development. [8] The various professionals are indulged in testing the software: Project Manager, Software Tester, Software Developer and End Users.

When Testing should be started- The first stage of SDLC is software testing. Starts from the requirement gathering (Planning) phase to the last stage i.e. Deployment phase. [7] In waterfall model, Testing formally is being organized in the phase of testing. Testing at the incremental model is implemented at the last of every increment/iteration and the complete application is being tested at the last.

When Testing should be stopped-Testing, the software is an everlasting process. No one can process that the software is 100% bug free, instead of testing the software [7] as the Domain to the input is too large that we cannot verify every input.



It is accomplished by developers of software and their peers or very rarely by the Testers those who are independent. Popular Tools for Unit Testing



Fig. 2 Software Testing Life Cycle

2. Software Testing Strategies:

There are various testing strategies that are being used for the purpose of testing

Fig.3 Test Strategies

2.1 Unit Testing- This type of testing is achieved at the bottom level by the developers before it is moved to the team of testing to execute the test cases. [8] It is the smallest module that can be tested and verified at the each section or lines of code. In this output of one module becomes the input of another module. If the output of any one of the module fails so then the output to which we give the input also fails. Therefore, therefore it is nevertheless better to test each component differently so that there would be less chance of fails. In this, White box testing technique is implemented.

2.1.1 When the Unit testing is accomplished: It is being accomplished prior to Integration Testing.

By whom Unit Testing is accomplished:

are: Mocha, Tape and Jasmine.

Integration Testing- Integration Testing is

performed immediately after the Unit Testing. In this all the modules are merged together to form a larger module and determine are they functioning in a proper way and then the testing is implemented on the modules.[6] Testing is done so that in case if any bug remained in the Unit Testing it can be again tested in this testing so as 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.[7] **For Example**, a unit test for database access code would not be able to talk to a real database but the integration testing would.

Testing is classified into two parts:

(I) Top-Down Testing

(II) Bottom-Up Testing

2.12 When the Integration Testing is accomplished:

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

2.13 By whom Integration Testing is accomplished:

It is accomplished by either the designers or by the Testers those who are independent. Popular Tools for Integration Testing are: Mocha, Tape and Jasmine.

System Testing- This type of testing is conducted to test the entire system. It is needed to test all the integrated components to test and verify whether it meets the requirements and the standards of quality. The basic purpose of the testing is to assess the compliance of the system within the desired specific requirements. In this, black box testing method is implemented. When the System Testing is accomplished. It is being accomplished after Integration Testing and before Acceptance Testing.

2.14 By whom System Testing is accomplished:

It is accomplished by the Testers those who are

independent.

Forexample: When the pen is fabricated, the body, cap, ink cartridge are tested differently and then **unit testing** is performed. When more than two units are organized, they all are combined and **integration testing** is performed. When the whole complete pen is consolidated then **system testing** is performed.

3. TESTING TECHNIQUES/ METHODS:

There are various methods or techniques for testing the software:

1. Black Box Testing

2. White Box Testing

3.1 BLACK BOX TESTING

In this type of testing, the internal structure/details of the data item are not known by or accessible to its user. [2] In this test cases are generated or designed from the Input / Output value only and no knowledge of design/ code is being required. The samples are only aware of knowing about what is assumed to do, and not to know how it does. These types of test can be functional or non-functional. [9] Black Box Testing is named so because as we know that in the tester's eyes it is named black box but inner side no one sees. Black Box Testing is also known as **Functional testing, Specification, Behavioral, Data Driven or Input-Output Driven.**[11] For Example, Without the recognition of the inner details of the website, we test the pages of web by the use of browser, authorize the input and then test and verify the outputs in contradiction of the outcome that is expected. [10]

There are many test cases in Black Box Testing:

I. Equivalence Class Partitioning

II. Boundary Value Analysis

III. Cause Effect Graph

IV. Comparison Testing

3.1.1 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 program's behavior is same to every input data that is belonging to the similar equivalent class. [2] 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 challenging the software with some other value that belongs to that equivalence class. [12]

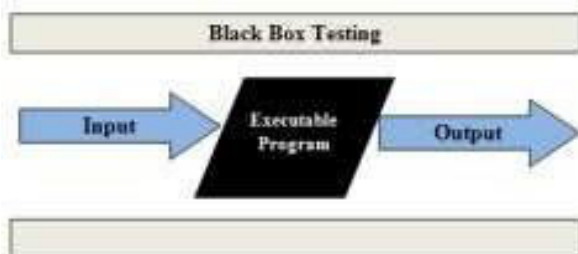


Fig.No.4 Black Box Testing

3.1.1 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 standards at the extreme end of the class.

For Example:

Programmer may improperly use = instead of <= for a function that calculates the square root of the integer value of the limit 0-5000. P= [0, 5000] >5000

So therefore, 2 partitions required.

Boundary Value Analysis= [-1, 0, 5000, 5001]

3.1.3 Cause Effect Graph:It is 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 are that they don't consider the potential combination of input and output condition. It connects the input classes (causes) to output classes (effects) yielding a directed graph.

It utilizes 4 symbols: NOT, OR, AND, IDENTITY.

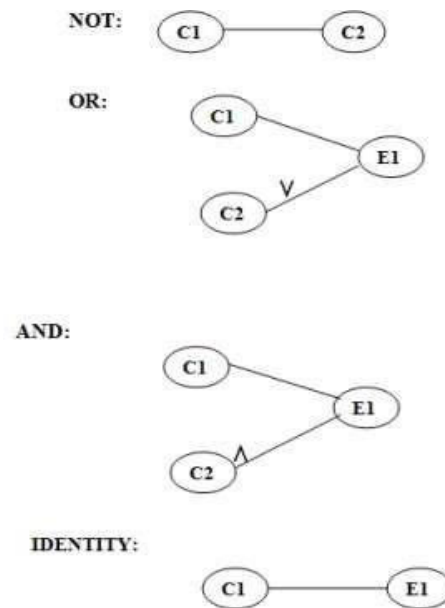


Fig.No.5 Cause Effect Graph

3.1.1 Comparison Testing: For critical applications that are required the fault tolerance, a number of independent version of the software are developed for the similar specification [6][7] if the output for each version is same then it is presumed that all the implementations are correct but if output is unique then the each version is examined to check what is responsible for the different output.

Advantages of Black Box Testing

- Testing is being performed from the view point of user's.
- Tester and Programmer both are autonomous to each other.
- Test cases can be designed immediately after the completion of specifications.
- Testers don't know about the languages of programming or how the software has been accomplished.

3.2 WHITE BOX TESTING

In this type of testing, the intramural structure/details of the data it em is known by or accessible to itsuser.In this, test cases are being made based on the code. [6] Programming very well knows about how the implementation of knowledge is significant.

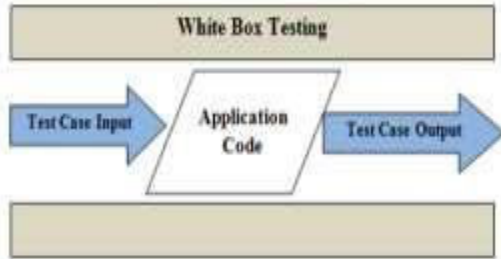


Fig. No.6 White Box Testing

WhiteBox 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. [11] White Box Testing is also known as Glass Box, Structural; Clear Box, Open Box, Logic Driven, or PathOriented. **For Example:** Basically a tester and a developer studies the code implemented of any field on a webpage, decides purposefully all the legaland the illegal inputs and verifies 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. [12]

Strategies applied to white box testing are:

Unit Testing: Within the units, the paths are tested.

Integration Testing: Between the units, the paths are tested.

SystemTesting:Between the subsystems,the paths are tested.

For white box testing, unit testing is applicable.

There are many test cases in White Box Testing:

I. Statement

II. Branch

III. Condition

IV. Path

V. Data Flow

VI. Mutation

VII. Domain and Boundary Testing

VIII. Loop Coverage Testing

IX. Logic Based

X. Fault Based

3.2.1 Statement: This is the easiest and simplest form of white box testing where there is noway to check wherea series of test cases are run in such a way that each statement is being executed at least once.The idea here is that we have no way to examine that an error existing inthe statement or not.

For Example: if (S>1 && t==0) x=9;

Therefore,2testcasesareformed,oneistrue and the other isfalse.

Branch/Edgetesting: Inthis, test cases are generated to form each branch condition presuming true and false values in turn.

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

Condition: In this test cases are designed to form each component of a composite conditional expression. [5] In this part, we verify only the part of the condition.

- a. It is the stronger testing than the branch testing.
- b. Branch testing is stronger than the statement coverage testing.
- c. Conditional coverage requires 2n test cases.

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 performed at least once.

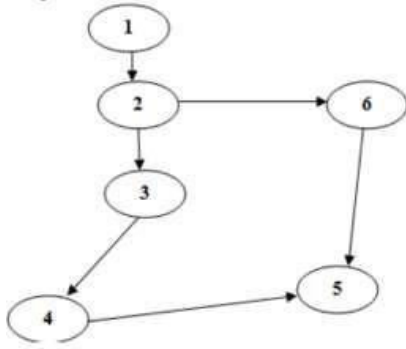


Fig No. 7 Control Flow of the Program

It describes the how the flow of control flow through the program & describes the sequence.

[15] A path that is sub path of the main path that is not considered as linear independent path.

Advantages of White Box Testing

- We need not to wait for the GUI to be implemented as testing is began at a very first stage.
- Helps in code optimization.

4. CONCLUSION

- Software testing is the basic activity of software engineering.
- It is an activity that executes the software with the aim of detecting errors or bugs init.
- This paper describes in detail about the testing techniques, strategies of testing the software.
- Important stages in the process of testing are on the methods of designing the test cases and it is impossible to find all the bugs from the software so for that we have designed the number of testing techniques that can be taken to analyze.

REFERENCES

- [1] Mao, Chengying, et al. "Adapting ant colony optimization to generate test data for software structural testing." *Swarm and Evolutionary Computation* 20 (2015):23-36.
- [2] McMinn, Phil. "Search-based software testing: The Past, present, and future." *Software testing, verification and validation workshops (icstw)*, 2011 IEEE fourth international conference on. IEEE, 2011.
- [3] Rafi, Dudekula Mohammad, et al. "Benefits and limitations of automated software testing: Systematic literature

reviewandpractionersurvey." *Proceedings of the 7th International Workshop on Automation of Software Test*. IEEE Press, 2012.

[4] Zogaj, Shkodran, Ulrich Bretschneider, and Jan Marco Leimeister. "Managing crowdsourced software testing: a case study based insight on the challenges of a crowdsourcing intermediary." *Journal of Business Economics* 84.3 (2014):375-405.

[5] Orso, Alessandro, and Gregg Rothermel. "Software testing: a research travelogue (2000–2014)." *Proceedings of the on Future of Software Engineering*. ACM, 2014.

[6] Harman, Mark, et al. "A comprehensive survey of trends in oracles for software testing." *The university of Sheffield, Department of Computer Science, Tech. Rep. CS-13-01* (2013).

[7] Garousi, Vahid, and Junji Zhi. "A survey of software testing practices in Canada." *Journal of Systems and Software* 86.5 (2013): 1354-1376.

[8] Jun, Wang, and Fanpeng Meng. "Software testing based on cloud computing." *Internet Computing & Information Services (ICICIS)*, 2011 International Conference on. IEEE, 2011.

[9] Selvam, R., and V. yKarthikeyan. "Mobile Software Testing- Automated Test Case Design Strategies." *International Journal of Computer Science and Engineering* 3.4 (2011):1450-1461.

[10] Collins, Eliane, April Dias-Neto, and Vicente F. de Lucena Jr. "Strategies for agile software testing automation: An industrial experience." *Computer Software and Applications Conference Workshops (COMPSACW)*, 2012 IEEE 36th.

[11] Irena, J.; (2008) "Software Testing Methods and Techniques", <http://tir.ipsitransactions.org/2009/January/Paper%2006.pdf>.

[12] Kaur, M; Singh, R; (2014) "A Review of Software Testing Techniques",

https://www.ripublication.com/irph/ijeev_spl/ijeev7n5_05.pdf.

[14] Kaur, K; Sharma, S; (2015) "A Survey on Software Testing",