International Journal of Advanced Science and Research
ISSN: 2455-4227, Impact Factor: RJIF 5.12
www.newresearchjournal.com/science
Volume 1; Issue 5; May 2016; Page No. 46-50

Comparative analysis of testing techniques: A Review

Er Suruchi

Assistant Professor, Department of Computer Science & Engineering Gian Jyoti Group of Institutions, Shambhukalan, Rajpura, Punjab, India.

Abstract
Software testing provides a means to reduce errors, cut maintenance and overall software costs. It is aimed at evaluating the capability or usability of a program. Software testing is an important means of accessing quality of software. One of the major problems in software testing area is how to get a suitable set of test cases to test a software system. Hence, software testing research faces a collection of challenges. There are now many testing techniques available for generating test cases. This set should ensure maximum effectiveness with the least possible number of test cases. The main goal of this paper is to analysed and compare the testing technique to find out the best one to find out the error from the software. Test case design techniques can be broadly split into two main categories:

- Black box
- White box.

Black box + White box = Gray Box

Keywords: Software testing; process model; testing techniques

1. Introduction

Software testing is as old as the hills in the history of digital computers. The testing of software is an important means of assessing the software to determine its quality. Testing is an essential activity in software engineering. In the simplest terms, it amounts to observing the execution of a software system to validate whether it behaves as intended and identify potential malfunctions [1]. It provides a realistic feedback of its behavior and as such it remains the inescapable complement to other analysis techniques. The goal of testing is systematically and stepwise detection of different classes of errors within a minimum amount of time and also with a much less amount of effort [2]. Software testing is also an important component of software quality assurance (SQA), and a number of software organizations are spending up to 40% of their resources on testing. There are four main objectives of testing (Myers, Glenford J. (1979), IBM Systems Research Institute, Lecturer in Computer Science, Polytechnic Institute of New York, The Art of Software Testing, by John Wiley & Sons, Inc.):

Detection: Various errors, defects, and deficiencies are detected. System capabilities and various limitations, quality of all components, the work products, and the overall system are calculated.

Prevention
In this information to prevent or reduce the number of errors, to clarify system specifications and system performance is provided. Different ways to avoid risks and to tackle problems in the future are identified.

Demonstration
It shows how the system can be used with various acceptable risk. It also demonstrates functions with special conditions and shows how products are ready for integration or use.

Improving quality
By doing effective testing on software, errors can be minimized and thus quality of software is improved.

The basic purpose of software testing is verification, validation and error detection in order to find various errors and problems – and the aim of finding those problems is to get them fixed. Software testing is more than just error detection. Software testing is done under controlled conditions for:

Verification
To verify if system behaves as specified. It is the checking and testing of items, which includes software, for conformance and consistency of software by evaluating the results against pre-defined requirements. In verification we ask a question, are we building the product right?

Validation
In this we check the system correctness which is the process of checking that what has been specified by user and what the user actually wanted. In validation we ask a question: Are we building the right system?

Error Detection
Error Detection to detect errors. A number of testing should be done to make things go wrong to determine if what things should happen when they should not.

Most Common Software problems
Inadequate software performance, Data searches that yields incorrect results. Incorrect data edits & ineffective data edits, Incorrect coding / implementation of business rules, Incorrect calculation, Incorrect data edits and ineffective data edits, Incorrect processing of data relationship, Incorrect or

inadequate interfaces with other systems, Inadequate performance and security controls, Incorrect file handling, Inadequate support of business needs, Unreliable results or performance, Confusing or misleading data, Software usability by end users & Obsolete Software, Inconsistent processing.

2. Terminology

Mistake - A human action that produces an incorrect result.

- **Fault or Defect:** An incorrect step, process, or data definition in a program.
- **Failure:** The inability of a system or component to perform its required function within the specified performance requirement.
- **Error:** The difference between a computed, observed, or measured value and condition and the true, specified, or theoretically correct value or condition.
- **Specification:** A document that specifies in a complete, precise, Verifiable manner, the requirements, design, behavior, or other Characteristic of a system or component, and often the Procedures for determining whether these provisions have been Satisfied. We observe errors, which can often be associated with failures. But the ultimate cause of the fault is often very hard to find.

3. Objectives

A good test case is one that has a probability of finding an as yet undiscovered error.

A good test is not redundant.

1. A successful test is one that uncovers a yet undiscovered error.
2. A good test should be “best of breed”.
3. A good test should neither be too simple nor too complex.
4. To check if the system does what it is expected to do.
5. To check if the system is “Fit for purpose”.
6. To check if the system meets the requirements and be executed successfully in the Intended environment.
7. Executing a program with the intent of finding an *error*.

4. Phases of Software Testing Lifecycle

4.1 Requirements study

- Testing Cycle starts with the study of client’s requirements.
- Understanding of the requirements is very essential for testing the product.

4.2 Test Case Design and Development

- Component Identification
- Test Specification Design
- Test Specification Review

4.3. Test Execution

- Code Review
- Test execution and evaluation
- Performance and simulation

4.4. Test Closure

- Test summary report
- Project De-brief
- Project Documentation

4.5 Test Process Analysis

Analysis done on the reports and improving the application’s performance by implementing new technology and additional features

5. Software Testing Strategies

A software testing strategy integrates various software test case design methods into a well-planned series of steps that result in successful testing of software. Software testing strategies are thus important for testing. Software testing strategy is generally developed by testing specialist, project managers and software engineer. There are four software testing strategies:

Unit testing

It is done at the lowest level. It tests the basic unit of software, which can be a module or component. Unit is the smallest module i.e. smallest set of lines of code which can be tested. Unit testing is just one of the levels of testing which contribute to make the big picture of testing a whole system. Unit testing is generally considered as a white box test class.

Integration Testing

It is done when two or more tested units are combined into a larger structure. This testing is often done on the interfaces that are between the components and the larger structure that is being constructed, if its quality property cannot be properly assessed from its components.

System Testing

It tends to test the end-to-end quality of the entire system. System test is often based on the functional and requirement specifications of the system. Non-functional quality attributes, such as security, reliability, and maintainability, are also checked.

Acceptance Testing

It is done when the complete system is handed over to the customers or users from developer side.

The aim of acceptance testing is to give assure that the system is working rather than to find errors.

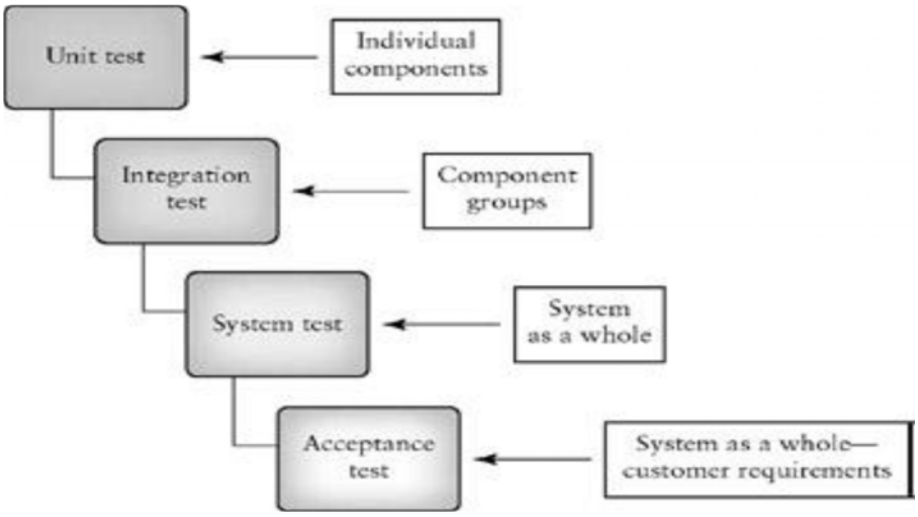

Fig 1: Levels of Testing

Table 1: Comparison of Software Testing Strategies

Testing Type	Specification	General Scope	Opacity	Who does it?
Unit	Low-level design: actual code	Classes	White box	Programmer
Integration	Low-level design: High level design	Multiple Classes	White box; black box	Programmer
System	Requirement Analysis	Whole Product Environment	Black box	Independent Tester
Acceptance	Requirement Analysis	Whole Product	Black Box	Customer

6. Software Testing Methodologies

There are following methodologies for software testing:

6.1 White Box Testing

In this testing, internal details and structure of system is made visible. Thus, it is highly efficient in detecting and resolving problems, because bugs can often be found before they cause trouble. We can thus define this method as testing software with the knowledge of its internal structure and coding [3]. White box testing is also called clear box testing, white box analysis or clear box analysis. It is a strategy for finding errors in which the tester has complete knowledge of how the program components interact. This method is rarely used practically for debugging in large systems and networks, thus used for Web services applications.

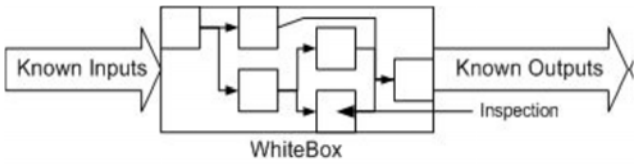

Fig 2: Represent White Box Testing

Different types of white box testing techniques are as follows:-

- Basis Path Testing
- Loop Testing
- Control Structure Testing
- Software testing methodologies

6.2 Black box testing

A black box is any device whose internal details and workings are not understood by or accessible to its user. It is testing of software based on specifications and output requirements and without any knowledge of the coding or internal structure in the program. The main aim is to test how well the system conforms to the specified requirements for the system. Black box testing has little or no knowledge to the internal logical structure of the system. Thus, it only examines the fundamental aspect of the system

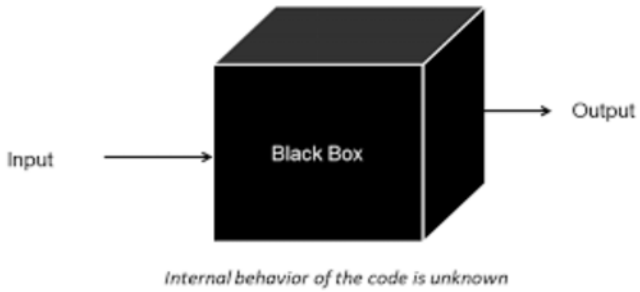

Fig 3: Represent Black Box Testing

Different types of Black box testing techniques are as follows:-

- Equivalent Partitioning
- Boundary value Analysis
- Cause-Effect Graphing Techniques
- Comparison Testing
- Mutation testing
- Fuzz Testing

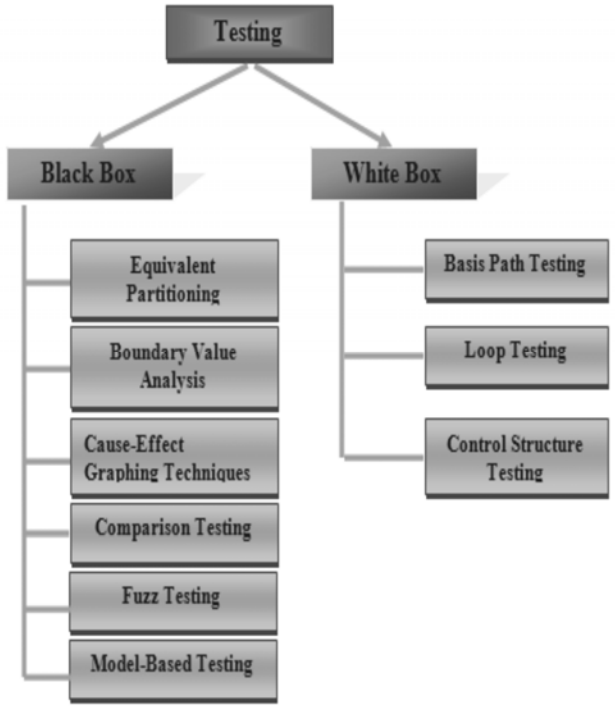

Fig 4: Classification of testing

6.3 Grey Box Testing

In recent years, a third testing method has been also considered i.e. grey box testing. It is defined as testing software and also having some knowledge of its internal logic and underlying code. It uses internal data structures and algorithms for designing the test cases more than black box testing but much less than white box testing. The aim of this testing is to search for the defects if any due to improper structure or improper usage of applications. Grey testing is also known as *translucent testing*.

Gray-box testing is well suited for Web applications. This method includes reverse engineering to determine boundary values. Grey box testing is unbiased and non-intrusive because it doesn't require that the tester have access to internal source code.

Gray-Box Testing Techniques

1. Matrix Testing
2. Regression testing
3. Pattern Testing
4. Orthogonal Array Testing

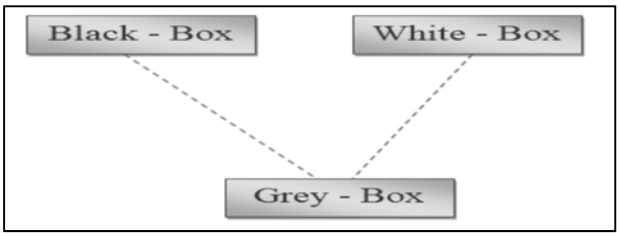

Fig 5: Grey Box Testing

Table 2: Comparison between black box testing and white box testing [11, 12]

Criteria	Black Box Testing	White Box Testing
Definition	Black Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is NOT known to the tester	White Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.
Levels Applicable To	Mainly applicable to higher levels of testing: Acceptance Testing, System Testing	Mainly applicable to lower levels of testing: Unit Testing, Integration Testing
Responsibility	Generally, independent Software Testers	Generally, Software Developers
Programming Knowledge	Not Required	Required
Implementation Knowledge	Not Required	Required
Basis for Test Cases	Requirement Specifications	Detail Design

7. Latest Research and Development in Software Testing

With the advancement in research and development on component system testing, analysis techniques and form-modeling in embedded software and the software credibility, new results and important issues on software testing keep emerging in last recent years. Some have been introduced below:

The latest results of software testing

7.1 Test-driven development

The programming is guided by testing. Before writing the code, we should write the related test cases and plans first, and program that test code, then test that develop code by testing the program and thus the cycle continues, until the development has been completed. The recent popular XP i.e. Extreme Programming mode has strongly advocated this idea of testing (Zhang Hongchun Research on New Techniques and Development Trend of Software Testing).

7.2 Iterative and incremental testing

It has evolved from the iterative model. After the iteration is done, the system will automatically integrate some new functions until the entire system function has completed. It mainly focuses on the cumulative functions used in the regression test and each iterative test is completed in two parts: incremental test on current iterative product and the regression test on the wholly completed function of the former iterative cycle. This is one of IBM most widely used test methods.

7.3 GUI automation test

This is an automated testing framework which is based on object-oriented capture technology for GUI. About the method of generation of testing case, a type of automatic

generation method of test data which is based on ant colony algorithm. By using bit coding, a model for input domain of the software under test to ant paths of the ant colony algorithm was established. It also improved the variety of ant paths and decreased the degree of the stagnation and precocity. The idea of automatic generation of model-driven software code in MDA improved the automation degree of the software testing.

7.4 Testability of component software

On the selection of a test case, a metadata selection method to select a test case is applied. It embeds the information and case to component in order to achieve the generation of a test case, and also used the method of UML to test case meta-model, show any relevant use case meta- model, mapping between them, and the elements of the component metadata. IGA and its advantages on the generation of component software testing case were introduced. Also, an advanced method that is the IIGA, which brought the migration, parallel, self-adoption and immune operator into traditional genetic algorithm also proofed its convergence.

7.5 Embedded Software Simulation Test

Various embedded software without any significant modification by simulation of the ARM embedded system on PC were tested A test development environment for MVC-based embedded software was designed, which not only ensured the successful development of ESTDE but also improved the adaptability and repeatability of the system.

7.6 International and domestic research hotspots

The research about the use of software testing techniques and methods directing to the features of software was done. Some were those researches about software testing techniques that aim at different types of software features like embedded systems and the real-time systems, etc. (Fu Bo (2007), Automatic Generation Method of Test Data Based on Ant Colony Algorithm, Computer Engineering and Applications.

The research about some software testing techniques that direct to other new software development techniques, including the researches about software testing techniques that aim at Internet structure, Object-oriented technology, Java language and software component.

The research about automatic testing technique. It helps to improve the degree of automation in all steps of testing and thus ease the burden of test analysts, such as automatically generating test cases, automatic performance of regression tests, much more.

The research about tools and environment used in testing. Testing environment and testing tool should be developed with the techniques and methods of software testing like testing designing tool, testing planning tool, structure testing tool, testing managing tool, static analysis tool, performance and load testing tool, regression testing tool, and the improvement of interoperability and effectiveness of testing tools.

8. Conclusion

Quality is the main focus of any software engineering project. Without measuring, we cannot be sure of the level of quality in software. So the methods of measuring the quality are software testing techniques. This paper relates various types of testing technique that we can apply in measuring various quality attributes. Software testing research is the driving element of development and application. In this era of new and higher demand of software testing, it is important to constantly summarize new achievements, fresh hotspots and propose different ideas in order to promote the study on software testing system engineering, to facilitate the rapid development on software testing field and industry.

9. References

1. Effective Software Testing, by Dustin E.
2. Software testing, by Ron Patton.
3. Myers, Glenford J. IBM Systems Research Institute, Lecturer in Computer Science, Polytechnic Institute of New York, The Art of Software Testing, by John Wiley & Sons, Inc, 1979.
4. IEEE, IEEE Standard Glossary of Software Engineering Terminology, Los Alamitos, CA: IEEE Computer Society Press, 1990.
5. Zhu H. A formal analysis of the subsume relation between software test adequacy criteria. IEEE Trans. Softw. Eng, 1996; 22(4):248-255.
6. Pressman R. Software Engineering, McGraw-Hill.
7. Mohd. Ehmer Khan, Different Forms of Software Testing Techniques for Finding Errors, IJCSI, 2010 7(3-1):11-16.
8. Grey Box Testing from Wikipedia available athttp://en.wikipedia.org/wiki/Gray_box_testing
9. Mohd. Ehmer Khan, Different Forms of Software Testing Techniques for Finding Errors, IJCSI, 2010; 7(3-1):11-16.
10. Stacey D A. Software Testing Techniques Guide to the Software Engineering Body of Knowledge, Swebok - A project of the IEEE Computer Society Professional Practices Committee, 2004.
11. Mohd Ehmer Khan, Farmeena Khan, A Comparative Study of White Box, Black Box and Grey Box Testing Techniques, (IJACSA) International Journal of Advanced Computer Science and Applications, 2012, 3:6.
12. Anitha A, A Brief Overview of Software Testing Techniques and Metrics, International. Journal of Advanced Research in Computer and Communication Engineering, 2013; 2(12):2278-102.
    http://www.cs.unh.edu/~it666/reading_list/Defense/blackbox_vs_whitebox_testing.pdf