International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

# "BLACKBOX TESTING"

***RAVI MV*[1], *PRASHANTH DR*[2]**

[1] Assistant Professor Dept of Electronics and Communication SJC Institute of Technology Chickballapur, Karnataka, India
ravimvreddy86@gmail.com
[2] Dept of Electronics and Communication SJC Institute of technology Chickballapur, Karnataka, India
Prashanthdr2002@gmail.com

ABSTRACT:

Black box testing is a software testing technique that evaluates the functionality of a system without examining its internal code or structure. It focuses on the input-output behaviour of the system, assessing how it performs specific actions from the user's perspective. This approach creates a social and critical distance between software development and testing, allowing testers to manipulate the system in ways its creators may not have considered. Black box testing is distinct from white box testing, where the tester has knowledge of the internal workings of the application code, and clear box testing, where the internal structure is fully known. It is a software testing style that can describe various test methodologies, including equivalence class partitioning, boundary value analysis, all-pairs testing, and predefined test cases. Black box testing is mainly focused on testing the functionality of the software, ensuring it meets requirements and specifications, and does not require knowledge of internal workings. It is generally used for testing software at the functional level and uses methods like equivalence partitioning, boundary value analysis, and error guessing to create test cases.

Keywords: Internal code, white box testing, functionality.

## INTRODUCTION :

Black box testing is a software testing technique where the internal workings or code structure of the system being tested are not known to the tester. With this approach, the tester doesn't have access to the internal source code of the programme and instead concentrates only on its external behaviour. Black box testing looks at a program's functionality without looking at its internal coding or structure. Based on the requirements, design characteristics, and other specifications of the software, testers develop test cases and evaluate both positive and negative scenarios to establish correct outputs. Black box testing uses a variety of methods, including equivalency partitioning, decision tables, and boundary value analysis. Testers concentrate on the software's functionality from a user's point of view, making sure it satisfies criteria, without needing to know programming as well as details. Black box testing helps identify discrepancies in functional specifications and is effective for testing in larger systems. It is also easily replicable.

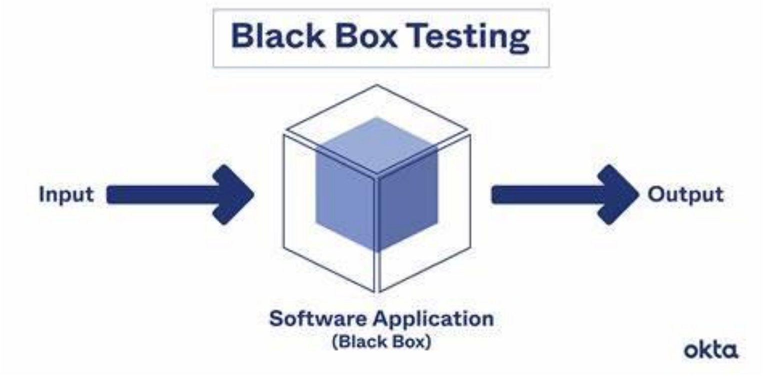

**Fig 1.1: Operation of Blackbox testing**

A fundamental method in software testing, black box testing is essential to guaranteeing the dependability and quality of software systems. Black box testing is limited to evaluating the system's behaviour and operation from the outside, as opposed to white box testing, which looks at the software's internal code structure. Black box testing essentially views the software as a "black box," with the tester only interested in the inputs that are fed into the system and the outputs that follow, keeping them ignorant of the inner workings of the programme. Black box testing's main goal is to confirm whether the programme operates as anticipated and satisfies requirements without getting bogged down in the specifics of how it was implemented. This method emulates the end-user's perspective. people, enabling testers to assess the functionality of the programme using actual user scenarios. Software development teams can reduce the risk of costly errors and ensure a smoother user experience by identifying flaws, inconsistencies, and usability concerns early in the development lifecycle by utilising black box testing approaches. Black box testing's main goal is to confirm whether the programme operates as anticipated and satisfies requirements without getting bogged down in the specifics of how it was implemented. By simulating end users' viewpoints, this method enables testers to assess the software's functionality using actual use cases. Software development teams can reduce the risk of expensive errors and faults by using black box testing approaches to find defects, inconsistencies, and usability concerns early in the development lifecycle. guaranteeing an improved customer experience.

## METHODOLOGY

The main procedures and approach for carrying out black box testing are outlined below:

- Perform a complete grasp of the software's requirements, specifications, and design parameters before beginning the requirement analysis process. This is the basis of your approach to black box testing.
- Planning of Tests: Create a thorough test strategy that includes the objectives, test cases, schedule, and scope of the testing. For every test case, list the input data and the desired results.
- Develop a collection of test cases that address a range of use cases and scenarios. To create efficient test cases: apply black box testing methods including state transition testing, decision table testing, equivalency partitioning, and boundary value analysis.
- Set up the testing environment: making sure to have the relevant hardware, software, and test data ready.
- Defect Reporting: Use a standardised defect reporting method to notify the development team of any discrepancies or problems found. Provide
- Comprehensive details about the issue: instructions for replicating it, and any pertinent logs or screenshots.
- Regression testing should be carried out following defect fixes to make sure that no new problems have been introduced or functionalities have been affected by the changes.
- Non-Functional Testing: Evaluate features such as performance, scalability, usability, and security using non-functional testing, contingent on the project requirements.
- Documentation for Testing: Keep thorough records of all of your test cases, tests, and defect reports. To make sure that every component of the software has received sufficient testing, monitor the test coverage.
- Test Closure: Create a test summary report after testing is finished and the programme satisfies the requirements. Provide information regarding the defect statistics, test coverage, testing procedures, and any suggested enhancements.
- Continual Improvement: Work together with the development team to identify and fix bugs, as well as to make ongoing improvements to the testing procedure.
- Without requiring knowledge of the internal code structure, this methodology guarantees a methodical and thorough approach to black box testing.
- Black box testing is a software testing approach in which the tester is blind to the internal workings, architecture, and implementation of the software being tested. Without any understanding of the core operations, the software's inputs and outputs are the only thing being considered.

The black box testing methodology is a user-centric and specification-based approach to software testing, emphasising verifying the software's functionality and behaviour without taking into account its internal implementation details.

## TECHNICAL ASPECTS

The technical aspects of black box testing involve the following key elements based on the provided sources:
Test Design Techniques: Black box testing utilizes various techniques like equivalence partitioning, boundary value analysis, decision table testing, and state transition testing to design effective test cases without knowledge of the internal code structure

Testing Objectives: Black box testing focuses on testing the functionality of the software to ensure it meets requirements and specifications, without delving into the internal code details.
Scope: Black box testing is typically used for testing software at the functional level, ensuring that the software behaves as expected from a user's perspective.
Testing Methods: Black box testing employs methods like equivalence partitioning, boundary value analysis, and error guessing to create test cases that cover various scenarios and use cases.
Test Execution Process: The test execution phase involves providing inputs to the software, comparing actual outputs with expected outputs, and documenting any discrepancies or defects discovered during testing.
Defect Reporting: In black box testing, defects encountered during testing are reported to the development team using a standardized defect reporting process, including detailed information for resolution.
Regression Testing: After fixing defects, regression testing is performed to ensure that the changes have not introduced new issues or impacted existing functionalities.
Non-Functional Testing: Depending on project requirements, non-functional testing is conducted to assess aspects like performance, scalability, usability, and security.

Test Documentation: Comprehensive documentation of test cases, test results, and defect reports is maintained throughout the testing process to track test coverage and ensure thorough testing.

These technical aspects highlight the methodologies, tools, objectives, and processes involved in black box testing, emphasizing its user-centric and requirement-focused approach to software testing.

## ADVANTAGES AND DISADVANTAGES

- The advantages of black box testing include:
- Unbiased Testing: Black box testing allows for unbiased testing as the designer and tester work independently, ensuring impartiality in the testing process.
- User-Focused Testing: It focuses on testing the software from a user's perspective, ensuring that the application meets user requirements and expectations.
- Testing from End-User Perspective: Black box testing simulates real user scenarios, helping to identify usability issues and ensuring that the software meets user needs.
- Early Detection of Interface Issues: This testing method can uncover interface-related defects, such as input validation errors and output discrepancies, at an early stage.
- Effective at Integration Testing: Black box testing verifies the interactions between different system components, making it valuable for integration testing.
- Test Case Design Flexibility: Various test case design techniques, such as equivalence partitioning and boundary value analysis, allow for effective test cover.
- Effective for Requirement Validation: Black box testing helps validate that the software meets the specified requirements.
- Suitable for Large Projects: It can be applied at different testing levels, from unit testing to acceptance testing, making it scalable for large projects.

### *DISADVANTAGES*

- Difficulty in Creating Test Cases: Without clear functional specifications, it can be challenging for testers to create comprehensive test cases, as they have limited knowledge of the internal workings of the software.
- Inability to Detect Internal Errors: Black box testing does not reveal errors in the control structure or internal logic of the software, as it focuses solely on the inputs and outputs.

## CONCLUSION AND REFERENCES

The conclusion of black box testing is that it is an essential component of software testing that focuses on testing the functionality of a software application from an end-user's perspective. By examining the inputs and outputs of the software without knowledge of the internal code structure, black box testing helps ensure the quality, functionality, and reliability of the software. It is a practical tool for identifying functional defects, usability issues, and gaps in software features. Black box testing allows testers to verify that the software behaves as expected, helping to improve software quality, reduce the time to market, and mitigate the risk of software failures at the user's end. Overall, black box testing should be an integral part of a comprehensive software testing strategy to validate that the software meets user requirements and functions correctly.

REFERENCES :

[1] Mohd. Ehmer Khan, “Different Approaches to Black Box Testing Technique for Finding Errors,” IJSEA, Vol. 2, No. 4, pp 31-40, October 2011

[2] Khan, M. E., 2011. “Different Approaches to White Box Testing Technique for Finding Errors”. International Journal of Software Engineering and Its Applications, 5(3), p. 14.

[3] D. Shao, S. Khurshid, and D. E. Perry, “A Case for White-box Testing Using Declarative Specifications Poster Abstract,” in Testing: Academic and Industrial Conference Practice and Research Techniques - MUTATION, 2007. TAICPART-MUTATION 2007, 2007, p. 137.

[4] Ayuliana.: Testing dan Implementasi: Black Box Testing. 1–6 (2009).

[5] Mohd. Ehmer Khan, “Different Forms of Software Testing Techniques for Finding Errors,” IJCSI, Vol. 7, Issue 3, No 1, pp 11-16, May2010

[6] Mohd. Ehmer Khan, “Different Approaches to Black Box Testing Technique for Finding Errors,” IJSEA, Vol. 2, No. 4, pp 31-40, October 2011

[7] Shivkumar Hasmukhrai Trivedi, “Software Testing Techniques”, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 10, October 2012, ISSN: 2277 128X

[8] Anitha.A, “A Brief Overview of Software Testing Techniques and Metrics”, International Journal of Advanced Research in Computer and Communication Engineering,Vol. 2, Issue 12, December 2013, ISSN (Online) : 2278-102

[9]. Vineta Arnicane,” Complexity of Equivalence Class and Boundary Value Testing Methods”, Scientific Papers, University of Latvia, 2009. Vol. 751 Computer Science and Information Technologies 80–101.

[10]. Trivedi, S. H., 2012. “Software Testing Techniques”. International Journal of Advanced Research in Computer Science and applications.

[11]. Ayman Madi, O. K. Z. a. S. K., 2013. “On the Improvement of Cyclomatic Complexity Metric”. International Journal of Software Engineering and Its Applications, 7(2).

[12]. Dondeti, S. N. a. J., 2012.” Black box and white box testing. International Journal of Embedded Systems and Applications”, 2(2).
[13]. Harsh Bhasin, E. K., 2014. “Black Box Testing based on Requirement Analysis and Design Specifications”. International Journal of Computer Applications, 87(18), pp. 0975-8887.
[14]. Khan, M. E., 2011. “Different Approaches to White Box Testing Technique for Finding Errors”. International Journal of Software Engineering and Its Applications, 5(3), p. 14.
[15]. Mohd. Ehmer Khan, F. K., 2012. “A Comparative Study of White Box, Black Box and Grey Box Testing Techniques”. International Journal of Advanced Computer Science and Applications, 3(6).
[16]. Trivedi, S. H., 2012. “Software Testing Techniques. International Journal of Advanced Research in Computer” Science and Software Engineering, 2(10).
[17]. Yeresime Suresh, S. K. R., 2013. “A Genetic Algorithm based Approach for Test Data Generationin Basis Path Testing.” The International Journal of Soft Computing and Software Engineering, 3(3).
[18]. Abhijit A. Sawant, P. H. B. a. P. M. C., 2012. “Software Testing Techniques and Strategies”. International Journal of Engineering Research and Applications, 2(3), pp. 980-986.
[19]. S.Maheswari, Dr.K.Chitra “Classification Of Software Testing And Their Techniques” International Conference on Current Research in Engineering Science and Technology(ICCREST- 2016) E-ISSN :2348 – 8387.