Black box Testing , Technique with Examples

Software Quality Engineering

Project Report

Submitted By

Fahad Ahmed (12388)
Hassan Javed(12382)

BSSE 6TH-Evening

Submitted To
Dr. Sumaira Nazir

National University of Modern Languages Islamabad
Department of Software Engineering

Abstract

This report gives a detailed look at Black Box Testing, a method for checking software without knowing how it's built inside. Black Box Testing lets testers act like regular users, checking how the system reacts to different actions, how fast it responds, and if there are any problems with how it works. The report covers why Black Box Testing is important, its good and not-so-good sides, and the various ways it's done. It also talks about how Black Box Testing can work together with White Box Testing and introduces a middle-ground approach called Grey Box Testing.

This overview aims to help people, especially those in software testing or quality assurance, understand Black Box Testing better. It explains the details of how Black Box Testing works, its strengths, and what challenges might come up. The report is meant to be a useful guide for anyone dealing with software testing, offering a clear and thorough look at Black Box Testing and how it fits into the bigger picture of creating software.

Contents

1 Introduction

Black box testing is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioral Testing [1].

2 Importance of Black Box Testing

Black Box Testing holds significance as it exercises a system comprehensively, just like end-users who are indifferent to the system's coding or architecture. Testers can simulate user activities, ensuring that the system delivers on its promises. This section highlights the importance of Black Box Testing in assessing response time, usability, and reliability issues.

3 Black Box Testing Pros and Cons

3.1 Pros

1. **No Technical Knowledge Required:**
    - Testers do not need technical knowledge, programming, or IT skills.
2. **No Need to Learn Implementation Details:**
    - Testers are not required to delve into the implementation details of the system.
3. **Crowdsourced or Outsourced Testing:**
    - Tests can be executed by crowdsourced or outsourced testers.
4. **Low Chance of False Positives:**
    - Black Box Testing has a low chance of producing false positives.
5. **Lower Complexity:**
    - Tests are less complex, focusing on modeling common user behavior.

3.2 Cons

1. **Difficult to Automate:**
    - Black Box Testing is challenging to automate.
2. **Prioritization Required:**
    - Requires prioritization; testing all user paths may be infeasible.
3. **Test Coverage Calculation Difficulty:**
    - Difficult to calculate test coverage comprehensively.

4. **Root Cause Analysis Challenges:**
    - If a test fails, understanding the root cause can be challenging.

4 Types of Black Box Testing

Black Box Testing can be applied to three main types of tests:

- Functional
- Non-functional
- Regression testing.

4.1 Functional Testing

Functional testing is a type of black box testing that focuses on validating the software against functional requirements and specifications. It ensures that the software behaves as expected in response to specific inputs. Functional testing is conducted at all levels and includes techniques like unit testing, integration testing, system testing, and acceptance testing.

4.2 Non-Functional Testing

While functional testing focuses on what the software does, non-functional testing is concerned with how the software performs. It evaluates aspects like the performance, usability, reliability, and compatibility. Black-box non-functional testing checks these criteria from the end-user's perspective. For example, a black-box performance test of a website might simulate a user session and measure the actual page load time [2].

4.3 Regression Testing

Regression Testing: It ensures that the newly added code is compatible with the existing code. In other words, a new software update has no impact on the functionality of the software. This is carried out after a system maintenance operation and upgrades [3].

5 Black Box Testing Techniques

Black Box Testing employs various techniques to ensure thorough test coverage. Each technique has its unique approach to designing test cases and uncovering potential issues. Below, we elaborate on the prominent techniques used in Black Box Testing [4]:

5.1 Equivalence Partitioning

Definition:

Equivalence Partitioning is a testing technique where testers divide possible inputs into groups or partitions, and then select one representative test case from each group for testing.

Example:

Consider a system that requires a user's age as input and responds differently based on whether the user is under 18, between 18 and 60, or over 60. Equivalence Partitioning allows testers to choose one age from each partition to ensure that the system behaves as expected for all possible scenarios.

Advantages:

- Efficiently tests a broad range of input scenarios.
- Reduces the number of test cases needed.

5.2 Boundary Value Analysis

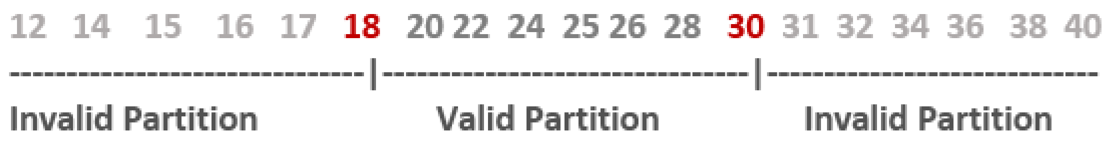
Definition:

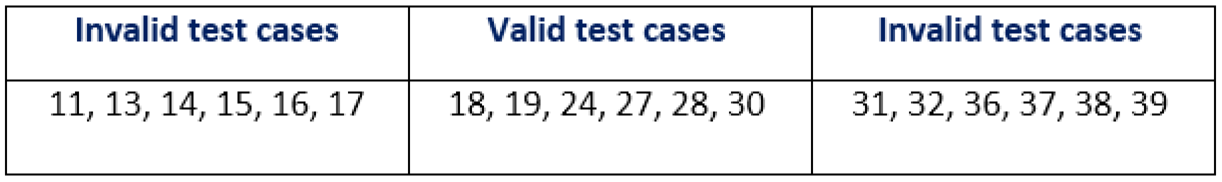
Boundary Value Analysis is a testing technique where testers identify specific boundary values where a system exhibits special responses. Test cases are then designed to evaluate the system's behavior around these boundaries.

Example:

If a system accepts values between 0 and 100, Boundary Value Analysis would focus on testing values like -1, 0, 1, 99, 100, and 101 to ensure the system handles these boundary conditions correctly.

Imagine, there is a function that accepts a number between 18 to 30, where 18 is the minimum and 30 is the maximum value of valid partition, the other values of this partition are 19, 20, 21, 22, 23, 24, 25, 26, 27, 28 and 29. The invalid partition consists of the numbers which are less than 18 such as 12, 14, 15, 16 and 17, and more than 30 such as 31, 32, 34, 36 and 40. Tester develops test cases for both valid and invalid partitions to capture the behavior of the system on different input conditions.





Invalid test cases	Valid test cases	Invalid test cases
11, 13, 14, 15, 16, 17	18, 19, 24, 27, 28, 30	31, 32, 36, 37, 38, 39

Figure 5.2.1

The software system will be passed in the test if it accepts a valid number and gives the desired output, if it is not, then it is unsuccessful. In another scenario, the software system should not accept invalid numbers, and if the entered number is invalid, then it should display error massage.

If the software which is under test, follows all the testing guidelines and specifications then it is sent to the releasing team otherwise to the development team to fix the defects [5].

Advantages:

- Often uncovers issues related to incorrect handling of boundary values.
- Helps ensure robustness around critical system limits.

5.3 Decision Table Testing

Definition:

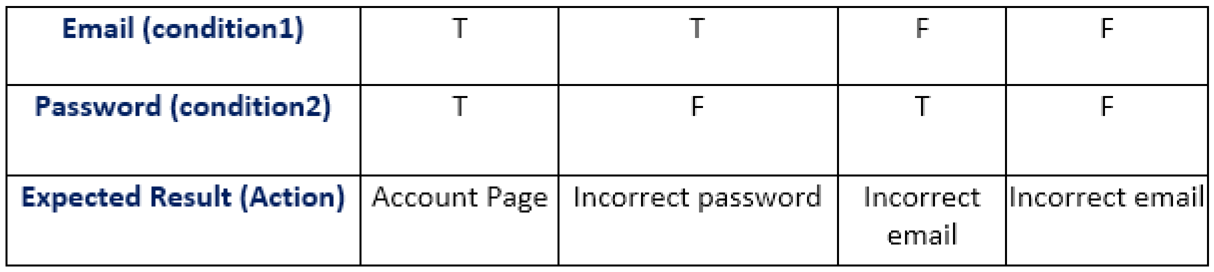
Decision Table Testing involves identifying rules and outcomes based on a set of conditions. Testers design test cases for each combination of conditions, ensuring comprehensive coverage.

Example:

Most of us use an email account, and when you want to use an email account, for this you need to enter the email and its associated password.

If both email and password are correctly matched, the user will be directed to the email account's homepage; otherwise, it will come back to the login page with an error message specified with "Incorrect Email" or "Incorrect Password."

Now, let's see how a decision table is created for the login function in which we can log in by using email and password. Both the email and the password are the conditions, and the expected result is action.



Email (condition1)	T	T	F	F
Password (condition2)	T	F	T	F
Expected Result (Action)	Account Page	Incorrect password	Incorrect email	Incorrect email

Figure 5.3.1

In the table, there are four conditions or test cases to test the login function. In the first condition if both email and password are correct, then the user should be directed to account's Homepage.

In the second condition if the email is correct, but the password is incorrect then the function should display Incorrect Password. In the third condition if the email is incorrect, but the password is correct, then it should display Incorrect Email.

Now, in fourth and last condition both email and password are incorrect then the function should display Incorrect Email.

In this example, all possible conditions or test cases have been included, and in the same way, the testing team also includes all possible test cases so that upcoming bugs can be cured at testing level.

To find the number of all possible conditions, tester uses 2^n formula where n denotes the number of inputs; in the example there is the number of inputs is 2 (one is true and second is false).

Number of possible conditions = 2^ Number of Values of the second condition Number of possible conditions =2^2 = 4

While using the decision table technique, a tester determines the expected output, if the function produces expected output, then it is passed in testing, and if not then it is failed. Failed software is sent back to the development team to fix the defect [6].

Advantages:

- Provides a systematic approach to test combinations of conditions.
- Ensures coverage of all possible rule outcomes.

5.4 State Transition TestingDefinition:

State Transition Testing is applied in systems where significant responses are generated during transitions from one state to another. Test cases are designed to evaluate the system's behavior during these transitions.

Example:

In a login mechanism, after a specific number of failed attempts, the system may transition to a different state, such as locking the account. State Transition Testing would involve designing test cases to check the system's response during this transition.

Advantages:

- Focuses on critical points in the system where transitions occur.
- Helps identify issues related to state changes.

5.5 Error Guessing

Definition:

Error Guessing is a testing technique where testers use their intuition and experience to identify potential errors in the system. Test cases are designed to specifically test for common mistakes developers might make.

Example:

Testers might check if the system handles null values in a field, text in a numeric field, or numbers in a text-only field. Additionally, they may test for known vulnerabilities that can affect the system under test.

Advantages:

- Leverages tester expertise and intuition.
- Can uncover issues that might be overlooked by other techniques.

6 Conclusion

In conclusion, black box testing is a versatile and essential approach to ensure the quality and reliability of software applications. The combination of functional, regression, and nonfunctional testing addresses a broad spectrum of testing requirements. The techniques employed, such as error guessing, state transition testing, decision table testing, boundary value analysis, and equivalence testing, contribute to a thorough evaluation of the software's capabilities and robustness. By adopting these testing methodologies and techniques, organizations can enhance the overall quality of their software, mitigate risks, and deliver a more reliable and user-friendly product to end-users.

References

[1] T. Hamilton, "What is BLACK Box Testing? Techniques, Types & Example," 6 12 2023. [Online]. Available: https://www.guru99.com/black-box-testing.html.

[2] bright, [Online]. Available: https://brightsec.com/blog/black-box-testing-types-techniques-pros-and-cons/.

[3] GeekForGreeks, [Online]. Available: https://www.geeksforgeeks.org/software-engineering-black-box-testing/.

[4] Imperva, [Online]. Available: https://www.imperva.com/learn/application-security/black-box-testing/.

[5] java point, [Online]. Available: https://www.javatpoint.com/boundary-value-analysis-in-black-box-testing.

[6] [Online]. Available: https://www.javatpoint.com/decision-table-technique-in-black-box-testing.

[7] geekForGreeks, [Online]. Available: https://www.geeksforgeeks.org/software-engineering-black-box-testing/.