Department of
Computer Science and Engineering

Title: Software Testing (Part II: Manual Testing) for a specific project

Integrated Design Project II

CSE 406

Green University of Bangladesh

1 Objective(s)

- To explain the purpose of manual software testing.
- To construct suitable test cases for manual software testing.

2 Problem Analysis

2.1 What is Manual Testing?

Manual software testing is when human testers check the quality of a new application without using automation tools or scripting. The purpose is to identify bugs or defects, ensure the product is error-free, and check it conforms to specified functional requirements.The process compares the behavior of a software application (or one of its components or features) with the expected behavior which was defined in the initial phases of the software development life cycle (SDLC). Manual testers design test cases or scenarios with 100 percent test coverage, and execute them one by one before verifying the results. They ensure that any reported issues are passed to the development team to fix, and then tested again. One of the fundamental principles of software testing is that 100 percent test automation is not possibleso manual testing is an essential part of your quality assurance process.

2.2 Example of manual testing

Manual testing has many real-life applications, and is especially handy for assessing usability and accessibility. For example, if you were launching an ecommerce website, you would need to check things like:

- Optimization for a range of browsers and devices
- Smooth checkout process
- Fast-loading hi-res images
- Links to social media channels

Under manual testing, testers would check the codes that drive each of these functions to ensure they work as the client intends. Manual testers are also able to comment on the look and feel of the website, evaluating it from the user's perspective.

2.3 Benefits of Manual Software Testing

Here are the benefits of using software testing-

1. **Accurate:** Automated tools are smart, but theyre not as smart as humans. There are certain things that only a real person with real-world experience can spot. So when it comes to identifying bugs and glitches in software, manual testing is more likely to catch em all.
2. **Gives human insight:** Manual software testers bring a valuable human perspective as well as accuracy, by focusing on the look and feel of a product. They can evaluate the applications visual components and highlight UI and UX issues by adopting the mindset of the end user.
3. **Adaptable:** The manual method is particularly useful in ad-hoc testing, as its easily adaptable when unplanned changes are made to software. The human input means test cases can be redesigned or tweaked with relative ease. Manual testing is also agile enough to be performed on all kinds of applications.
4. **Saves money:** Although manual testing requires skilled labor, it can actually save your company money as it doesnt need any expensive tools. Automation tools can be costly to install and will take time to set up and learn.

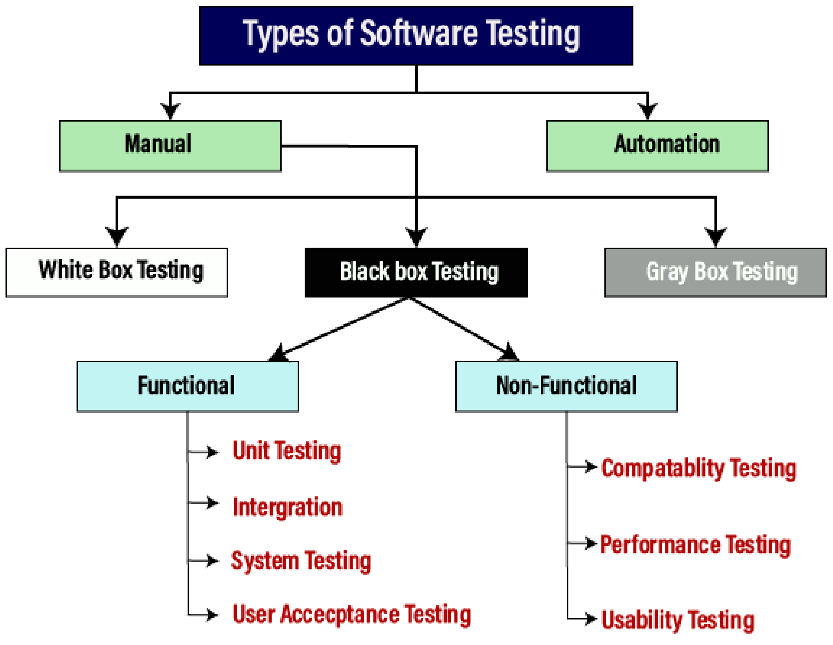


Figure 1: Different Types of Software Testing

2.4 Types of Manual Software Testing

While performing the manual testing on any application, we do not need any specific knowledge of any testing tool, rather than have a proper understanding of the product so we can easily prepare the test document. There are various types of testing available in the market, which are used to test the application or the software. These types are shown in Fig. 1.

Manual testing can be further divided into three types of testing, which are as follows:

- White Box Testing
- Black box Testing
- Gray Box Testing

2.4.1 White Box Testing:

The white box testing is done by Developer, where they check every line of a code before giving it to the Test Engineer. Since the code is visible for the Developer during the testing, that's why it is also known as White box testing. Figure 2 depicts the white box testing process.

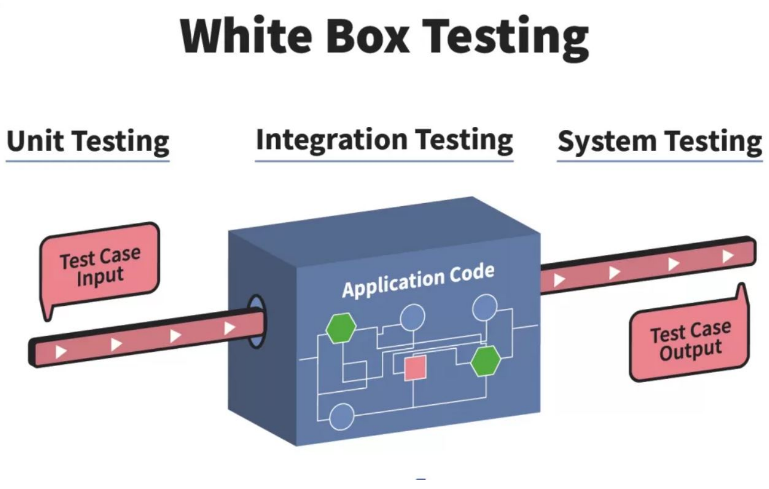


Figure 2: White Box Testing

Following are important White Box Testing Techniques:

- Statement Coverage
- Decision Coverage
- Branch Coverage
- Condition Coverage
- Multiple Condition Coverage
- Finite State Machine Coverage
- Path Coverage
- Control flow testing
- Data flow testing

White box testing can be quite complex. The complexity involved has a lot to do with the application being tested. A small application that performs a single simple operation could be white box tested in few minutes, while larger programming applications take days, weeks, and even longer to fully test. White box testing in software testing should be done on a software application as it is being developed after it is written and again after each modification.

2.4.2 Black Box Testing:

The black box testing is done by the Test Engineer, where they can check the functionality of an application or the software according to the customer /client's needs. In this, the code is not visible while performing the testing; that's why it is known as black-box testing. Figure 3 illustrates the black box testing technique.

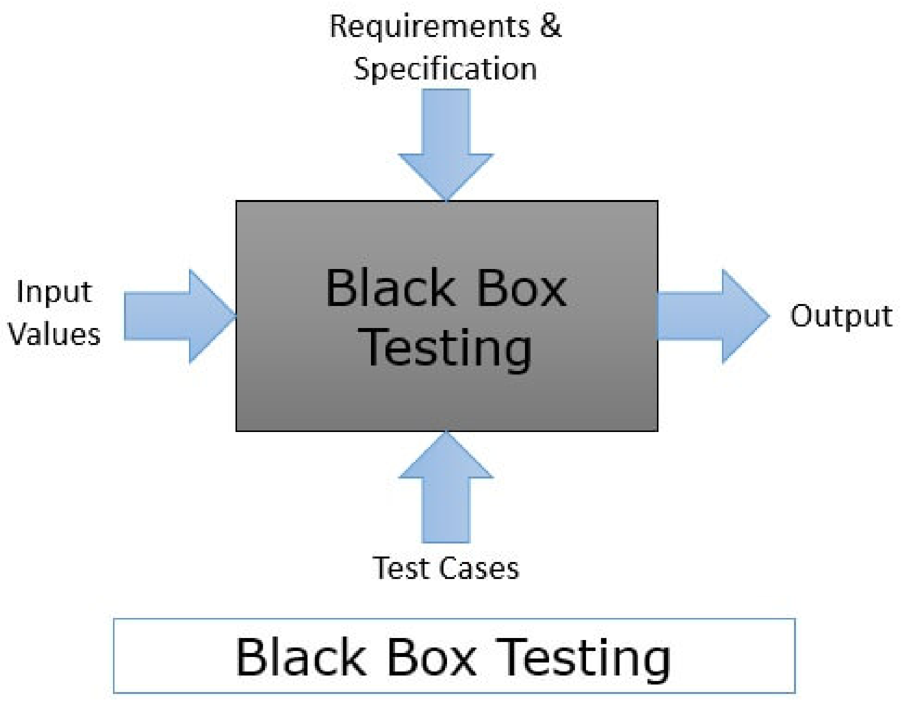


Figure 3: Black Box Testing

Black Box Testing has several subdivisions, including functional testing for requirement compliance, smoke testing to assess basic functionality, and partitioning (dividing software into groups that are expected to exhibit similar behavior).

- **Integration Testing**
  Integration Testing is the process of testing an application with two or more integrating components. It is performed once the individual components have been unit-tested, and aims to identify problems with the interfaces and the interactions between them. The two main methods are the Bottom-Up Approach (moving steadily from the bottom module to the top module) and Top-Down Approach (the opposite).
- **System Testing**
  System Testing means testing the system as a whole, once all its components have been unit-tested and integrated. It checks that the complete application works as intended, by comparing it against the original requirements. Also called end-to-end testing, it typically involves installability testing (does the software install correctly?) and recovery testing (can the application recover from hardware crashes and network failures?).
- **Unit Testing**
  This is when the individual units or components of an applications source code are tested, to make sure each function performs as expected. It is usually carried out by developers rather than engineers, as it requires detailed knowledge of the internal program design and code. Also known as module testing or component testing, it simplifies the debugging system and helps to detect and protect against bugs in the future.

2.4.3 Gray Box Testing:

Gray box testing is a combination of white box and Black box testing which is shown in figure 4. It can be performed by a person who knew both coding and testing. And if the single person performs white box, as well as black-box testing for the application, is known as Gray box testing. To perform Gray box testing, it is not necessary that the tester has the access to the source code. A test is designed based on the knowledge of algorithm, architectures, internal states, or other high -level descriptions of the program behavior.

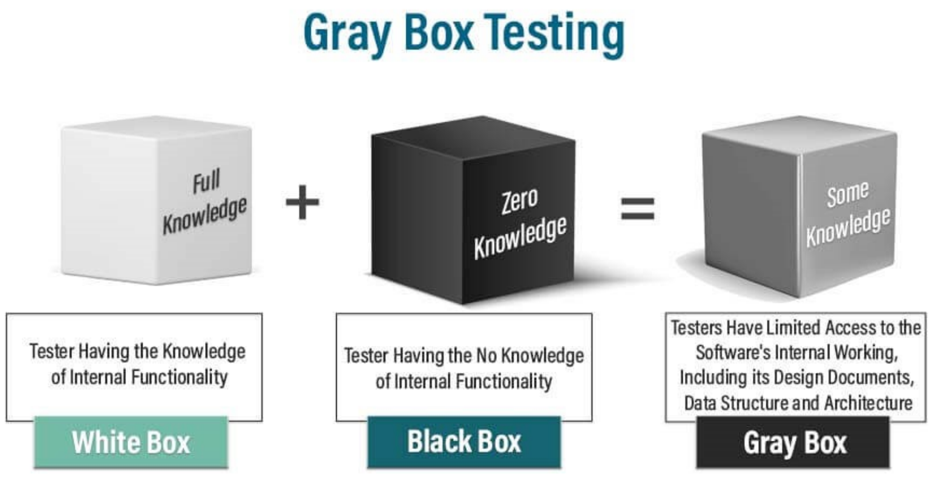

Gray Box Testing
Full Knowledge
+
Zero Knowledge
=
Some Knowledge
Tester Having the Knowledge of Internal Functionality
White Box
Tester Having the No Knowledge of Internal Functionality
Black Box
Testers Have Limited Access to the Software's Internal Working, Including its Design Documents, Data Structure and Architecture
Gray Box

Figure 4: Gray Box Testing

To perform Gray box Testing-

- It applies a straightforward technique of black box testing
- It is based on requirement test case generation, as such, it presets all the conditions before the program is tested by assertion method.

The overall cost of system defects can be reduced and prevented from passing further with Grey box testing. Grey box testing is suited more for GUI, Functional Testing, security assessment, web applications, web-services, etc. Techniques used for Grey box Testing includes:

- Matrix Testing
- Regression Testing

- OAT or Orthogonl Array Testing
- Pattern Testing

3 Methodology

Weve already seen that there are several types of manual testing, but the overall testing process is common to all of them. Sometimes this will be carried out in-house, while some businesses prefer to engage an expert company to handle testing.

There are roughly six basic stages in performing manual testing:

1. **Understand requirements**
   The first thing testers need to do is to fully understand the projects requirements. What does the client expect from the application? What problem is it aiming to solve for end-users? Testers must analyze all requirement documents in order to recognize the expected behavior of the software and exactly what needs to be tested.
2. **Prepare test cases**
   Once the requirements are understood, testers can draft test cases to cover various scenarios, such as what happens when a user enters an invalid password or how the software would cope with a crash. These test plans will set out a sequence for testing functionality and usability across the entire application, measured against expected results.
3. **Review test cases**
   Its helpful to review the draft test cases with team leaders and the client, to check that they will cover all bases and make any amendments before commencing the execution. This will save time in the long run.
4. **Execute test cases**
   Manual testing can now be carried out, using any of the techniques listed in the previous section. As well as finding bugs, the aim is to identify potential pain points for users and loopholes that could be exploited by hackers. Testers execute the test cases one by one, sometimes using bug-tracking tools like Jira.
5. **Report bugs**
   When bugs are identified, the testing team will pass the metrics to the development team in the form of a test report. This contains details on how many defects or bugs were found, how many test cases failed, and which need to be re-run.
6. **Test again**
   Once the development team has fixed the bugs, the software is handed back to the testers. They carry out the test cases again to check that the problem is resolved.

4 Implementation

4.1 Manual Testing

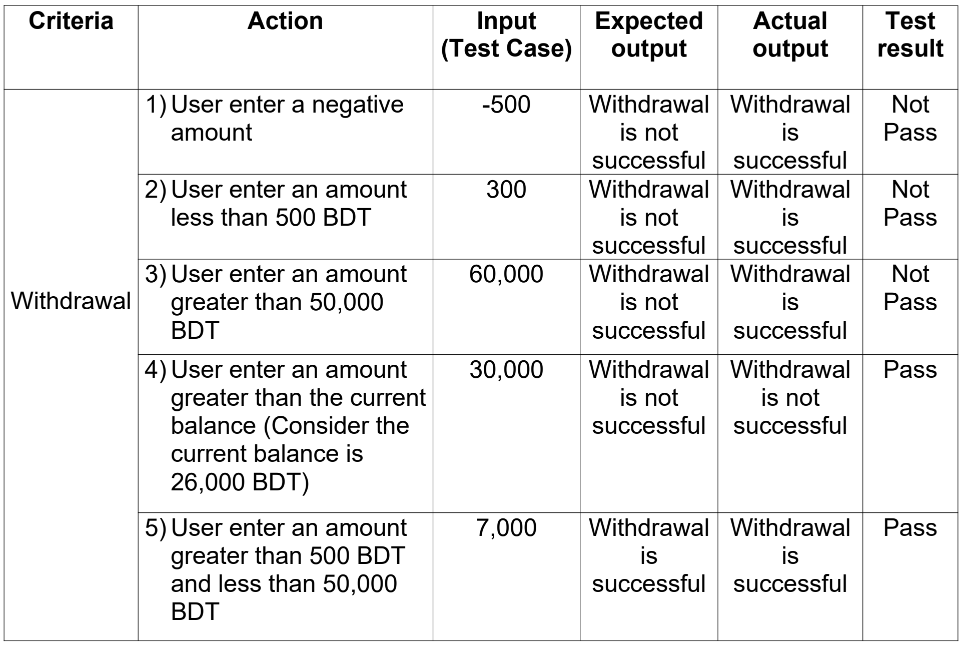
A Banking ATM System may have many use-cases or functions. For example, Balance Checking, Cash Withdrawal, Account Statement, Transaction and so on. In Fig. 5). the manual test cases are shown for Withdrawal' function of a Banking ATM System.

4.2 Performance Evaluation

We can evaluate the testing performance by the following formula,

$$TestingPerformance(\%) = \frac{SuccessfulTestCases}{TotalNumberofTestCases} \times 100 \tag{1}$$

- For **Manual Testing** the Testing Performance is $= (2/5) * 100 = 40\%$



Criteria	Action	Input (Test Case)	Expected output	Actual output	Test result
Withdrawal	1) User enter a negative amount	-500	Withdrawal is not successful	Withdrawal is successful	Not Pass
	2) User enter an amount less than 500 BDT	300	Withdrawal is not successful	Withdrawal is successful	Not Pass
	3) User enter an amount greater than 50,000 BDT	60,000	Withdrawal is not successful	Withdrawal is successful	Not Pass
	4) User enter an amount greater than the current balance (Consider the current balance is 26,000 BDT)	30,000	Withdrawal is not successful	Withdrawal is not successful	Pass
	5) User enter an amount greater than 500 BDT and less than 50,000 BDT	7,000	Withdrawal is successful	Withdrawal is successful	Pass

Figure 5: Manual Testing: Test Cases for Withdrawal Function of a Banking ATM System.

5 Lab Task (Please implement yourself and show the output to the instructor)

- Generate test cases for Balance Checking, Balance Transfer and Request Statement of a Banking ATM System for Manual Testing.
- Implement the test cases and evaluate performances for Balance Checking, Balance Transfer and Request Statement of a Banking ATM System for Manual Testing.

6 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.