**Assignment (EXPL-414)**

**Unit-2**

**Test Case and Software Testing**

**Submitted by:**

**Abhishek Chawda**

**Semester: 7th**

**Reg.No:06-0128-2013**

**Submitted To:**

**Dr. Y.R. Ghodasara**

**College of Agricultural Information Technology,**

**Anand Agricultural University,**

**Anand-388110**

Contents

[1) What is Test Case? 1](#_Toc469907136)

[2) Test Priority 1](#_Toc469907137)

[*2.1) High Priority defects* 1](#_Toc469907138)

[*2.2) Medium priority defects* 1](#_Toc469907139)

[*2.3) Low priority defects* 1](#_Toc469907140)

[3) Typical Test Case Parameters 2](#_Toc469907141)

[4) What is Software Testing? 4](#_Toc469907142)

[5) Why Testing? 4](#_Toc469907143)

[6) Who does Testing? 5](#_Toc469907144)

[7) Beta Testing 5](#_Toc469907145)

[8) Problems with Beta Testing 7](#_Toc469907147)

[9) Black Box Testing 7](#_Toc469907148)

[*9.1) Black Box Testing Advantages* 8](#_Toc469907149)

[*9.2) Black Box Testing Disadvantages* 8](#_Toc469907150)

[*9.3) Types of Black Box Testing* 8](#_Toc469907151)

[10) White Box Testing 8](#_Toc469907152)

[*10.1) White Box Testing Advantages* 9](#_Toc469907153)

[*10.2) White Box Testing Disadvantages* 9](#_Toc469907154)

[11) Automated Testing Tools (selenium) 9](#_Toc469907155)

[*11.1) Selenium Development* 10](#_Toc469907156)

[*11.2) Birth of Selenium Grid* 10](#_Toc469907157)

[*11.3) Birth of Selenium IDE* 11](#_Toc469907158)

[*11.4) Birth of Web Driver* 11](#_Toc469907159)

[12) Internet Links Referred (\*) 11](#_Toc469907160)

**Test Case**

# What is Test Case?

A test case is a document, which has a set of test data, preconditions, expected results and post conditions, developed for a particular test scenario in order to verify compliance against a specific requirement.

Test Case acts as the starting point for the test execution, and after applying a set of input values; the application has a definitive outcome and leaves the system at some end point or also known as execution post condition.

\*[Reference: 1]

# Test Priority

Priority as the name suggests, is about prioritizing a bug according to its severity. Priority signifies the importance or urgency of fixing a defect. It is associated with scheduling, to resolve a bug. Priority status of a defect is initially set by a tester to a developer to indicate the importance of fixing that defect. If the priority status is high then that bug will be fixed first followed by lower prior ones.

Different statuses that can be given for priority are: High, Medium and Low.

*2.1) High Priority defects*

These are business critical and have to be fixed with immediate effect.

## *2.2) Medium priority defects*

These are those which have a moderate level of impact in business and have to be fixed in the current release itself.

## *2.3) Low priority defects*

These are those which do not have any impact on the business and those which are cosmetic in nature. This can be pushed to the next release also.

**Example:** Consider that our Drupal website functionality is to upload films after payment and publish it for selling. Users who visit the site has the facility to view trailer and if interested can buy it. During the testing phase I came across the following issues

Issue 1: Website failed to accept Credit Cards.

Issue 2: Order confirmation emails were not getting delivered to the consumer.

Issue 3: Submit button was observed as misaligned in one of the browsers.

* High Priority defects: Here, Issue 1 "website fails to accept credit cards" is a defect which has high impact on the revenue of the organization and so should be fixed immediately. So Issue 1 can be set as a High priority one.
* Medium priority defects: Issue 2 "Order confirmation emails are not getting delivered to the consumer" should be reported as Medium priority issue.
* Low priority defects: Issue 3 "Misalignment of submit button in a particular version of browser" should be reported as a Low priority one.

\*[Reference: 2]

# Typical Test Case Parameters

* Test Case ID
* Test Scenario
* Test Case Description
* Test Steps
* Prerequisite
* Test Data
* Expected Result
* Test Parameters
* Actual Result
* Environment Information
* Comments

|  |  |
| --- | --- |
| **Test case Id** | Unique ID for each test case. Follow some convention to indicate types of test. |
| **Test Title/Name** | Test case title. E.g. verify login page with valid username and password. |
| **Pre-condition** | Any prerequisite that must be fulfilled before execution of this test case. List all pre-conditions in order to successfully execute this test case. |
| **Dependencies** | Mention any dependencies on other test cases or test requirement. |
| **Test Steps** | List all test execution steps in detail. Write test steps in the order in which these should be executed. Make sure to provide as much details as you can. Tip – to efficiently manage test case with lesser number of fields use this field to describe test conditions, test data and user roles for running test. |
| **Test Data** | Use of test data as an input for this test case. You can provide different data sets with exact values to be used as an input. |
| **Test Scenario** | The summary / objective of the test case. |
| **Expected Results** | What should be the system output after test execution? Describe the expected result in detail including message/error that should be displayed on screen. |
| **Actual Results** | Actual test result should be filled after test execution. Describe system behavior after test execution. |
| **Pass/Fail** | Pass or Fail. Other statuses can be 'Not Executed' if testing is not performed and 'Blocked' if testing is blocked. |

**Example:** Let us say that we need to check an input field that can accept maximum of 10 characters.

While developing the test cases for the above scenario, the test cases are documented the following way. In the below example, the first case is a pass scenario while the second case is a FAIL.

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario** | **Test Step** | **Expected Result** | **Actual Outcome** |
| Verify that the input field that can accept maximum of 10 characters | Login to application and key in 10 characters | Application should be able to accept all 10 characters. | Application accepts all 10 characters. |
| Verify that the input field that can accept maximum of 11 characters | Login to application and key in 11 characters | Application should NOT accept all 11 characters. | Application accepts all 10 characters. |

If the expected result doesn't match with the actual result, then we log a defect. The defect goes through the defect life cycle and the testers address the same after fix.

\*[Reference: 3]

**Software Testing**

# What is Software Testing?

Software Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not.

Software Testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

\*[Reference: 4]

# Why Testing?

Software Testing is necessary because we all make mistakes. Some of those mistakes are unimportant, but some of them are expensive or dangerous. We need to check everything and anything we produce because things can always go wrong – humans make mistakes all the time. Since we assume that our work may have mistakes, hence we all need to check our own work. However some mistakes come from bad assumptions and blind spots, so we might make the same mistakes when we check our own work as we made when we did it. So we may not notice the flaws in what we have done.

 Ideally, we should get someone else to check our work because another person is more likely to spot the flaws.

There are several reasons which clearly tell us as why Software Testing is important and what are the major things that we should consider while testing of any product or application.

Software testing is very important because of the following reasons:

* Software testing is really required to point out the defects and errors that were made during the development phases.
* It’s essential since it makes sure of the Customer’s reliability and their satisfaction in the application.
* It is very important to ensure the Quality of the product.  Quality product delivered to the customers helps in gaining their confidence. (Know more about Software Quality)
* Testing is necessary in order to provide the facilities to the customers like the delivery of high quality product or software application which requires lower maintenance cost and hence results into more accurate, consistent and reliable results.
* Testing is required for an effective performance of software application or product.
* It’s important to ensure that the application should not result into any failure because it can be very expensive in the future or in the later stages of the development.
* It’s required to stay in the business.

\*[Reference: 5]

# Who does Testing?

It depends on the process and the associated stakeholders of the project(s). In the IT industry, large companies have a team with responsibilities to evaluate the developed software in context of the given requirements. Moreover, developers also conduct testing which is called Unit Testing. In most cases, the following professionals are involved in testing a system within their respective capacities:

* Software Tester
* Software Developer
* Project Lead/Manager
* End User

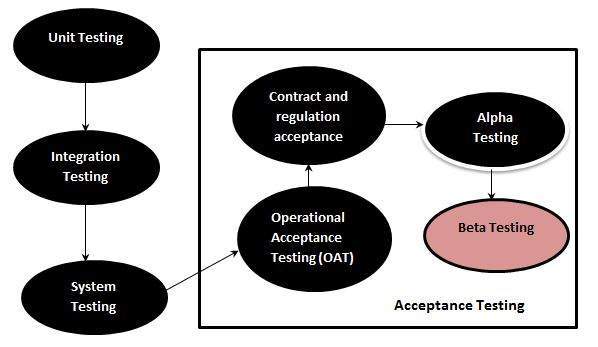
\*[Reference: 6]

# Beta Testing

Beta testing also known as user testing takes place at the end users site by the end users to validate the usability, functionality, compatibility, and reliability testing.

Beta testing adds value to the software development life cycle as it allows the "real" customer an opportunity to provide inputs into the design, functionality, and usability of a product. These inputs are not only critical to the success of the product but also an investment into future products when the gathered data is managed effectively.

The following diagram explains the fitment of Beta testing in the software development life cycle:



Unit Testing

Integration Testing

System Testing

System Integration Testing

Regression Testing

Acceptance Testing

Alpha Testing

Beta Testing

(Fig. illustrating how beta testing fits into the testing cycle)

\*[Reference: 7]

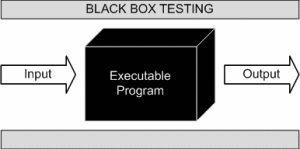
# Problems with Beta Testing

1. Beta testing often generates too much noise (that is, feedback) that is not accurate and not actionable.
2. Inconsistent participation — too much or too little — often administered with poor processes for collecting and analyzing feedback. Not all use cases get covered, so bugs slip through.
3. Good catches but insufficient information: Even when bugs are identified, the reports are often not useful because they lack sufficient information to reproduce the defect.
4. Delay: Beta testing slows the release cycle by having a dedicated phase before the production release.

\*[Reference: 8]

# Black Box Testing

**Black Box Testing**, also known as Behavioral Testing, is a software testing method in which the internal structure/ design/ implementation of the item being tested is not known to the tester.



This method is named so because the software program, in the eyes of the tester, is like a black box; inside which one cannot see. This method attempts to find errors in the following categories:

* Incorrect or missing functions
* Interface errors
* Errors in data structures or external database access
* Behavior or performance errors
* Initialization and termination errors

**Example:** A tester, without knowledge of the internal structures of a website, tests the web pages by using a browser; providing inputs (clicks, keystrokes) and verifying the outputs against the expected outcome.

## *9.1) Black Box Testing Advantages*

* Tests are done from a user’s point of view and will help in exposing discrepancies in the specifications.
* Tester need not know programming languages or how the software has been implemented.
* Tests can be conducted by a body independent from the developers, allowing for an objective perspective and the avoidance of developer-bias.
* Test cases can be designed as soon as the specifications are complete.

## *9.2) Black Box Testing Disadvantages*

* Only a small number of possible inputs can be tested and many program paths will be left untested.
* Without clear specifications, which are the situation in many projects, test cases will be difficult to design.
* Tests can be redundant if the software designer/ developer has already run a test case.
* Ever wondered why a soothsayer closes the eyes when foretelling events? So is almost the case in Black Box Testing.

## *9.3) Types of Black Box Testing*

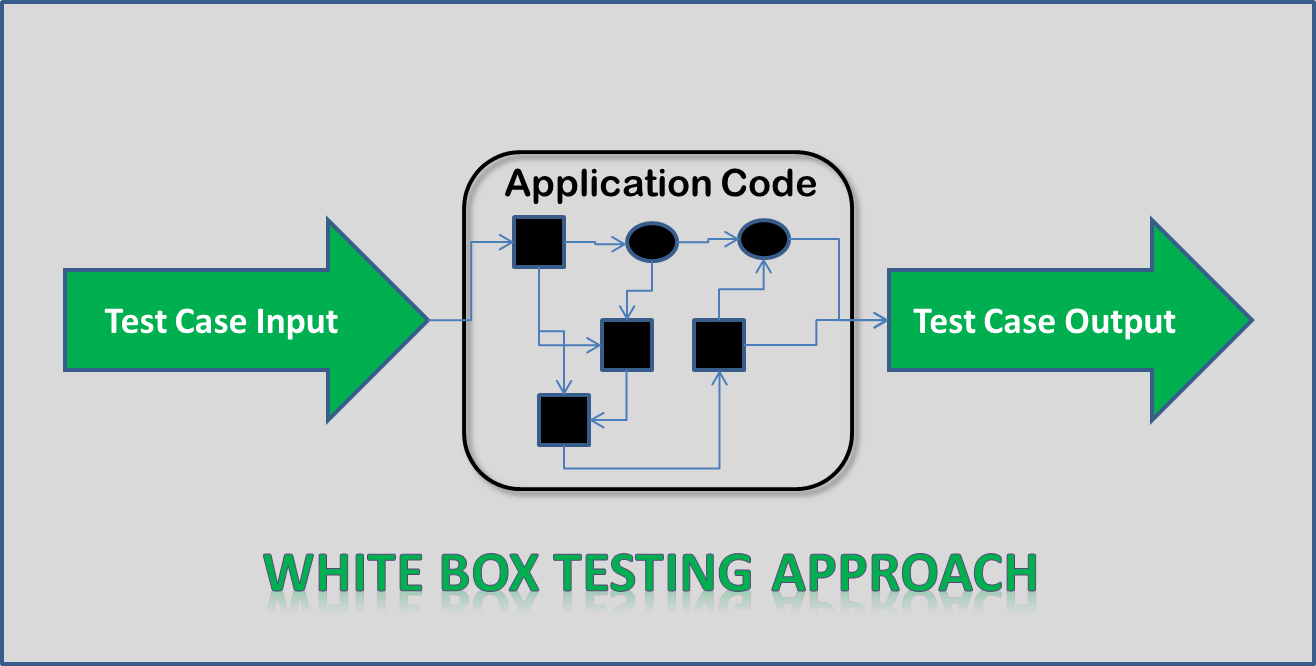
* Functional testing **-** This black box testing type is related to functional requirements of a system; it is done by software testers.
* Non-functional testing **-** This type of black box testing is not related to testing of a specific functionality, but non-functional requirements such as performance, scalability, usability.
* Regression testing **-** Regression testing is done after code fixes, upgrades or any other system maintenance to check the new code has not affected the existing code.

\*[Reference: 9]

# White Box Testing

**White Box Testing** (also known as Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing or Structural Testing) is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. Programming know-how and the implementation knowledge is essential. White box testing is testing beyond the user interface and into the nitty-gritty of a system.

This method is named so because the software program, in the eyes of the tester, is like a white/ transparent box; inside which one clearly sees.



**Example:** A tester, usually a developer as well, studies the implementation code of a certain field on a webpage, determines all legal (valid and invalid) AND illegal inputs and verifies the outputs against the expected outcomes, which is also determined by studying the implementation code.

White Box Testing is like the work of a mechanic who examines the engine to see why the car is not moving.

## *10.1) White Box Testing Advantages*

* Testing can be commenced at an earlier stage. One need not wait for the GUI to be available.
* Testing is more thorough, with the possibility of covering most paths.

## *10.2) White Box Testing Disadvantages*

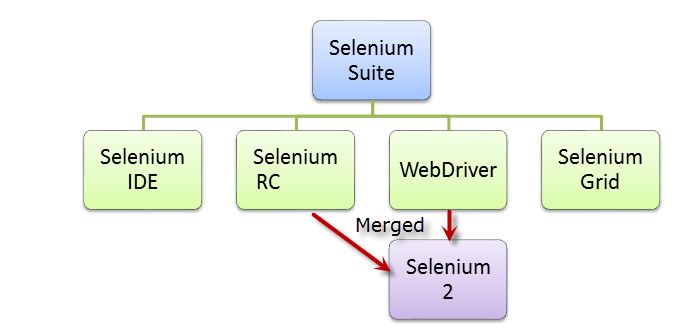
* Since tests can be very complex, highly skilled resources are required, with thorough knowledge of programming and implementation.
* Test script maintenance can be a burden if the implementation changes too frequently.
* Since this method of testing it closely tied with the application being testing, tools to cater to every kind of implementation/platform may not be readily available.

\*[Reference: 10]

# Automated Testing Tools (selenium)

* Selenium is a free (open source) automated testing suite for web applications across different browsers and platforms.
* Selenium focuses on automating web-based applications.
* Selenium is not just a single tool but a suite of software's, each catering to different testing needs of an organization.
* **It has four components:**

1. Selenium Integrated Development Environment (IDE).
2. Selenium Remote Control (RC).
3. Web Driver.
4. Selenium Grid.



## *11.1) Selenium Development*

* Selenium was created by Jason Huggins in 2004. An engineer at Thought Works, he was working on a web application that required frequent testing.
* He created a JavaScript program that would automatically control the browser's actions.
* He named this program as the "JavaScriptTestRunner".
* He made JavaScript Runner open-source which was later re-named as Selenium Core.
* Birth of Selenium Remote Control
* Testers using Selenium Core had to install the whole application under test and the web server on their own local computers because of the restrictions imposed by the same origin policy.
* Paul Hammant, decided to create a server that will act as an HTTP proxy to "trick" the browser into believing that Selenium Core and the web application being tested come from the same domain. This system became known as the Selenium Remote Control.

## *11.2) Birth of Selenium Grid*

* Selenium Grid was developed by Patrick Lightbody to address the need of minimizing test execution times as much as possible. He initially called the system "Hosted QA."
* It was capable of capturing browser screenshots during significant stages, and also of sending out Selenium commands to different machines simultaneously.

## *11.3) Birth of Selenium IDE*

* Shinya Kasatani of Japan created Selenium IDE, a Firefox extension that can automate the browser through a record-and-playback feature.
* He came up with this idea to further increase the speed in creating test cases.

## *11.4) Birth of Web Driver*

* Simon Stewart created Web Driver circa 2006 when browsers and web applications were becoming more powerful and more restrictive with JavaScript programs like Selenium Core.
* It was the first cross-platform testing framework that could control the browser from the OS level.

\*[Reference: 11]

# Internet Links Referred (\*)

1. <https://www.tutorialspoint.com/software_testing_dictionary/test_case.htm>
2. <http://www.zyxware.com/articles/3403/levels-of-priority-in-software-testing-explanation-with-a-simple-example>
3. <http://www.softwaretestinghelp.com/test-case-template-examples/>
4. <https://en.wikipedia.org/wiki/Software_testing>
5. <http://istqbexamcertification.com/why-is-testing-necessary/>
6. https://www.tutorialspoint.com/software\_testing/software\_testing\_overview.htm
7. https://www.tutorialspoint.com/software\_testing\_dictionary/beta\_testing.htm
8. http://www.drdobbs.com/testing/the-death-of-beta-testing/240151159
9. http://softwaretestingfundamentals.com/black-box-testing/
10. <http://softwaretestingfundamentals.com/white-box-testing/>
11. <http://www.guru99.com/introduction-to-selenium.html>)