Software Testing basics

Murali Chemuturi

Introduction

Testing is carried out primarily for unearthing any and all defects present in the system and to prevent a defective product reaching the customers. Secondarily, testing is also carried out to convince the customer that the product conforms to and fulfills the specifications and the functionality specified and agreed to.

Software Testing is recognized as a very important activity in software development. Of late, the importance of independent testing – that is testing by persons not involved in the development of software – is increasingly being recognized, so much so, software companies specialized only in software testing are established and are doing good business. Also significant is the fact that - as the complexity and size of the software increased multi-fold – so has the complexity of testing as well as the types of testing increased.

Testing Basics

There are basically two techniques of testing

1. White Box
2. Box Testing

White Box testing involves stepping thru every line of code, and every branch in the code. To use this technique, the tester should be knowledgeable about the programming language and should know the structure of the program.

In Black Box testing, a set of inputs is given to the software and the outputs delivered by the software are compared with the expected outputs. To use this techniques, the tester should have knowledgeable about the functionality of the system and should be able to use the computer.

Testing Scenarios

Software testing as stated above is carried in two independent scenarios –

1. Project Testing or Embedded Testing – that is testing, which is carried out as part of a software development project – this is carried out to ensure that the development work is defect-free.
2. Product Testing – testing that is carried out for a COTS (Commercial Off-The-Shelf) software product. This is to ensure that the products work without any defects in a variety of customer scenarios.

These scenarios are described below.

Project Testing / Embedded Testing

When software is developed as product that is delivered to a single client or intended to be used at a single location, the following testing takes place, in addition to software inspections (peer reviews) –

1. **Unit Testing** – this is certainly carried out by the person who wrote the code and by an independent peer using white box testing technique.
2. **Integration Testing** – carried out either as one-off (that is, when all integration is completed) of incrementally (that is, whenever one unit of software integrated and continued till all units are integrated). Black Box testing is used in one-off Integration Testing and white box testing can be, perhaps, used in incremental integration testing
3. **System Testing** to ensure that the software works in all intended target systems.
4. **User Acceptance Testing** to obtain customer sign-off so that software can be delivered and payments received

Optionally, many other tests can be conducted at the behest of the customer.

Product Testing

Product would be developed as a project first and would undergo all the tests that a project normally undergoes, namely, unit, integration, and system testing. System testing is carried out more rigorously and on multiple systems. In addition, it needs some more rigorous tests. These are –

1. **Load Testing** – in web applications and multi-user applications, large numbers of users are logged in and try to use the software in a random manner. The objective is to see if the software is managing multiple requests and serving up accurate results or mixing them up. . This unearths the issues connected with the bandwidth, database, sufficiency of RAM, hard disk etc
2. **Volume Testing** – subject the software to a high volume of data and see the performance, whether it degrades.
3. **Functional Testing** – test that all functions expected of the software are functioning correctly.
4. **End-to-End Testing** – in this type of testing, one entity is tracked from birth to death in the application. For example, in a payroll application, an employee joins the system; then is promoted; then is demoted; salary increases are effected, salary decreases are effected; kept in abeyance; transferred, then retired, dismissed, terminated and so on to ensure that the state transitions designed in the applications happen as desired
5. **Parallel Testing** – a number of users using the same function and are either inputting or requesting same data. This brings out the system's ability to handle requests at the same time and preserving the data integrity.
6. **Concurrent Testing** – Concurrent testing is carried out to unearth issues when two or more users use the same functionality and update or modify same data with different values at the same time – normally using a testing tool. For example, take ticket reservation scenario, there is only one seat and it is shown as available to two people. When both confirm purchase, the system should give to only one and reject the other request. It should not happen that money is collected from both credit cards and reserve for only one – the credit card transaction must be reversed for the rejected party. Scenarios like this will be tested.
7. **Stress Testing** – cause stress to the software by making expected resources unavailable or causing deadlock like scenarios or not releasing resources and so on to ensure that the software has routines built in to handle such stress. This will bring out software responses for events like machine-rest, Internet disconnection, server timeouts etc.
8. **Positive Testing** – test the software as specified and not trying any negative acts – to ensure that all defined functions are performing. Used mostly for customer / end user acceptance testing.

9. **Negative Testing** – using the software in a manner that is not expected to be used – this will bring out all hidden defects in the software. This is to ensure even malicious usage would not affect the software or data integrity.
10. **User Manual Testing** – use the software conforming to the user manual to ensure that they both are in synch with each other
11. **Deployment Testing** – Simulate the target environment and deploy the software and ensure that deployment specified is appropriate.
12. **Sanity Testing** – this cursory testing to ensure that the components, of software package, are complete and are of appropriate versions, carried out before delivery or before making a software-build.
13. **Regression Testing** – testing carried out after unearthed defects are fixed
14. Security Testing – testing to ensure vulnerability against the threat of viruses and spy-ware
15. **Performance Testing** – testing to ensure that the response times are in acceptable range
16. **Usability Testing** – testing the software for different types of usage to ensure that it satisfactorily fulfills the requirements of specified functional areas
17. **Install** / **uninstall Testing** – test the software on all target platforms to ensure that install and uninstall operations are satisfactorily performed
18. **Comparison Testing** – testing the product with competing products to contrast the differences for determining the relative position of the product
19. **Intuitive Testing** – testing without reference to user manuals to see if the product can be used without much reference to user guides

It is rare that all the above types of testing are carried out for every project that is executed in the organization. But it is common for product testing to include many of the above tests.

Organizations carry out some combination of the types of testing described above. Normally every organization conducts the following types of testing –

1. **Functional Testing** to ensure that all the functionalities allocated to the software are working and there are no inaccuracies, when used properly
2. **Integration testing** – to ensure that the coupling between various software modules is in order
3. **Positive Testing** / **Acceptance Testing** to get the software accepted by the client
4. **Load Testing** to ensure that the system does not crash when heavy loads are placed on it

Organizations carry out the remaining types of testing, sometimes, on a “if time and budget are available” –– basis” or “if mandated” basis.

The “How” of testing

When we come to methodology of testing, we find –

1. **Test Cases Based Testing** – there is a set of test cases and testing is carried out only against the test cases
2. **Intuitive Testing** – there may be a general description of the functionality and suggestions/guidelines for intuitive testing, as to how to go about unearthing defects. Testing is carried out using the experience and intuition of the tester. Some amount of creativity or common sense is expected from the tester.

For any software project there would always be a high-level test plan. For every intended testing, there ought to be set of test cases against which testing is carried out. But consider the implications –

1. For every numeric (including date type data) data input, we need to have five test cases – using the Partitioning and Boundary Value Analysis techniques –
    a. One value in the acceptable range – to be accepted by the system
    b. One value above acceptable range – to be rejected by the system
    c. One value below acceptable range – to be rejected by the system
    d. One value at the upper boundary of the acceptable range – to be accepted by the system
    e. One value at the lower boundary of the acceptable range– to be accepted by the system
2. Size checks for all non-numeric data, one per every data item
3. Logical testing for presence of invalid data – like two decimal points in numeric data, numeric and special characters in name data fields etc.

Thus, the test case set for even a moderately complex unit will be voluminous. Modern projects are large in size and the effort required to prepare exhaustive test case set would be significantly high. Therefore, it is common (not always, perhaps) to prepare test cases where it is expected that the tester cannot intuitively figure out the test cases all by him/her self. It is common to have guidelines for the following tests –

1. GUI Testing
2. Navigation Testing
3. Negative Testing
4. Load Testing
5. Stress Testing
6. Parallel and Concurrent Testing
7. Unit Testing

Organizations make use of these guidelines and avoid, not in all organizations perhaps, preparing test cases, exhaustively.

It is not uncommon that unit testing is carried out without any test cases. Integration testing, system testing and acceptance testing are normally carried out against test cases.

Test Strategy

Before we can start our discussion on Test Effort Estimation, we need to understand test strategy. ***Test Strategy is concerned with unearthing as many defects as possible within the allocated budget of time and cost and maximizing the impact of such testing***.

Fist step in finalizing the test strategy is to set testing objectives. These could be –

1. **Quality Objectives** – these are concerned with the level of unearthing of defects ranging from
    a. All defects irrespective of time or cost
    b. Almost all possible defects within the time available – cost and time are a criterion
    c. All possible defects within the time available – time is the main criteria
2. **Customer Acceptance Objectives** – the main objective of testing to obtain customer sign-off so that customer pays our money
3. **Product Certification Objectives** – carry out the tests specified by the customer and certify as requested by the client. These certifications could be –
    a. Virus and Spy-ware free
    b. Functionality

c. Usability
d. Comparison and relative position
e. Product Rating
f. Etc.

In addition to objectives, the following are also part of Test Strategy

1. **Types of tests to be included in testing** – what are the tests that have to be conducted to achieve the project objectives
2. **How of testing** – the methodology of testing
   a. Plan and Test Case based testing or intuitive testing
   b. White Box or Black Box
   c. Manual Testing or Tool based testing
3. **Regression Testing** – number of iterations for Regression Testing – only once or iterated till all defects are closed
4. **Criteria for successful completion of testing** – is it time and cost based or closure of defects or until all defects are unearthed
5. **Mechanisms** for defect closure and escalation when necessary
6. **Progress Reporting** during the project execution
7. **Defect Analysis** – such as ABC analysis, Category analysis etc – whether required or not

Defect Analysis

The following types of Defect Analysis are carried out as a means for reducing defects in the coming projects.

1. Severity Analysis
2. ABC Analysis
3. Defect Origin Analysis

Severity Analysis

Normally defects are categorized into Critical, Major and Minor defects.

1. A Critical Defect is one that delivers wrong results or does not allow some functionality or such serious defect. It is called a “Showstopper” in testing parlance.
2. Major Defect – is one that may not be a showstopper but is detrimental to the functionality and usage of software.
3. Minor Defect is one that does not cause any major issues but it would be desirable if it is fixed. It is called “Cosmetic” in testing parlance.

This analysis looks at the relative frequency of each of these categories. It is preferable to have critical and major defects hovering around zero and minor defects being majority.

ABC Analysis

ABC stands for “Always better Control. In this, defects are divided into three classes, namely,

1. A class defects – these form about 10% of all defects occurring about 90% of the time
2. B Class Defects– these form about 20% of all defects occurring about 20% of the time
3. C Class Defects– these form about 90% of all defects occurring about 10% of the time

This classification facilitates focus on those 10% of the defects occurring 90% time so that corrective action can be implemented to prevent future defects.

Defect Origin Analysis

Defects are categorized according their origin. Some of the origins are –

1. Misunderstood requirements
2. Wrong Design
3. Forgotten steps
4. Loops not properly terminated
5. Control statements without exit criteria
6. So on – there are nearly 70 categories that are normally used.

When defects are so categorized, it will give clarity on where to improve to reduce defects. This analysis coupled with the above two analyses, would help bring about significant reduction in defects and improve quality

Final words

Testing is a very important activity in software development. It should be carried out in accordance with a well-defined process. Defect analysis ought to be carried out for effecting reduction in defects and improvement in quality.

>>.
Your feedback is gratefully accepted at murali@chemuturi.com .
>>