SOFTWARE & MANUAL TESTING

Abhishek Jain

Abstract- **Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding software bugs . Software testing can be conducted as soon as executable software (even if partially complete) exists. The overall approach to software development often determines when and how testing is conducted. For example, in a phased process, most testing occurs after system requirements have been defined and then implemented in testable programs. In contrast, under an agile approach, requirements, programming, and testing are often done concurrently.**

Introduction

A primary purpose of testing is to detect software failures so that defects may be discovered and corrected. Testing cannot establish that a product functions properly under all conditions but can only establish that it does not function properly under specific conditions. The scope of software testing often includes examination of code as well as execution of that code in various environments and conditions as well as examining the aspects of code: does it do what it is supposed to do and do what it needs to do. In the current culture of software development, a testing organization may be separate from the development team. There are various roles for testing team members. Information derived from software testing may be used to correct the process by which software is developed.

Every software product has a target audience. For example, the audience for video game software is completely different from banking software. Therefore, when an organization develops or otherwise invests in a software product, it can assess whether the software product will be acceptable to its end users, its target audience, its purchasers and other stakeholders. Software testing is the process of attempting to make this assessment.

What is Software Testing?

Software testing is more than just error detection; Testing software is operating the software under controlled conditions, to (1) verify that it behaves "as specified"; (2) to detect errors, and (3) to validate that what has been specified is what the user actually want

1. **Verification** is the checking or testing of items, including software, for conformance and consistency by evaluating the results against pre-specified requirements. [Verification: Are we building the system right?]
2. **Error Detection**: Testing should intentionally attempt to make things go wrong to determine if things happen when they shouldn‟t or things don‟t happen when they should.
3. **Validation** looks at the system correctness – i.e. is the process of checking that what has been specified is what the user actually wanted

Basic Terminologies

1.failure: Actual deviation of the component or system from its expected delivery, service or result.
2.Error-A human action that produces an incorrect result. [After IEEE 610]
3.Defect: A flaw in a component or system that can cause the component or system to fail to perform its required function, e.g. an incorrect statement or data definition. A defect, if encountered during execution, may cause a failure of the component or system.
4.**software quality:** The totality of functionality and features of a software product that bear on its ability to satisfy stated or implied needs. [After ISO 9126]
5.specification: A DOCUMENT that specifies, ideally in a complete, precise and verifiable manner, the requirements, design, behavior, or other characteristics of a component or system, and, often, the procedures for determining whether these provisions have been satisfied. [After IEEE 610]

OBJECTIVES

1 .A good test case is one that has a probability of finding an as yet undiscovered error.
2.A good test is not redundant.

3.A successful test is one that uncovers a yet undiscovered error.
4.A good test should be "best of breed".
5.A good test should neither be too simple nor too complex.
6.To check if the system does what it is expected to do.
7.To check if the system is "Fit for purpose".
8.To check if the system meets the requirements and be executed successfully in the Intended environment.
9.Executing a program with the intent of finding an error.

Testing Levels

There are generally four recognized levels of tests: unit testing, integration testing, component interface testing, and system testing. Tests are frequently grouped by where they are added in the software development process, or by the level of specificity of the test.

Unit testing

Unit testing, also known as component testing, refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors

These types of tests are usually written by developers as they work on code (white-box style), to ensure that the specific function is working as expected. One function might have multiple tests, to catch corner cases or other branches in the code. Unit testing alone cannot verify the functionality of a piece of software, but rather is used to ensure that the building blocks of the software work independently from each other.

Unit testing is a software development process that involves synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce software development risks, time, and costs. It is performed by the software developer or engineer during the construction phase of the software development lifecycle. Rather than replace traditional QA focuses, it augments it. Unit testing aims to eliminate construction errors before code is promoted to QA; this strategy is intended to increase the quality of the resulting software as well as the efficiency of the overall development and QA process.

Depending on the organization's expectations for software development, unit testing might include static code analysis, data flow analysis, metrics analysis, peer code reviews, code coverage analysis and other software verification practices.

Integration testing

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or all together ("big bang"). Normally the former is considered a better practice since it allows interface issues to be located more quickly and fixed.

Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.[33]

System testing

System testing, or end-to-end testing, tests a completely integrated system to verify that it meets its requirements.[36] For example, a system test might involve testing a logon interface, then creating and editing an entry, plus sending or printing results, followed by summary processing or deletion (or archiving) of entries, then logoff.

Acceptance testing

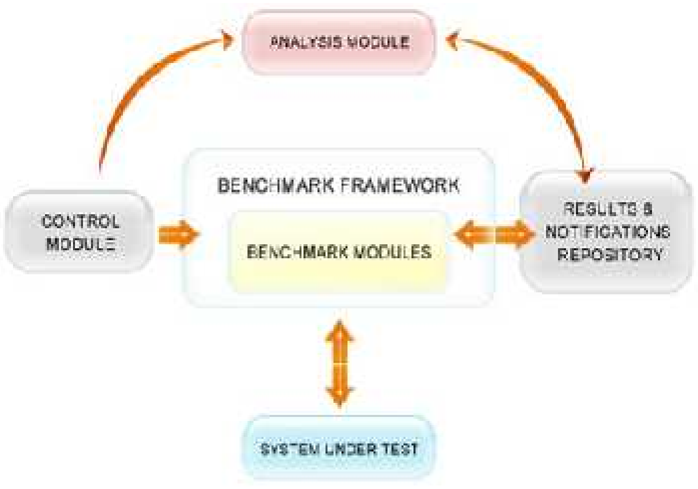
Acceptance is used to conduct operational readiness (pre-release) of a product, service or system as part of a quality management system. OAT is a common type of non-functional software testing, used mainly in software development and software maintenance projects. This type of testing focuses on the operational readiness of the system to be supported, and/or to become part of the production environment. Hence, it is also known as operational readiness testing (ORT) or Operations Readiness and Assurance (OR&A) testing. Functional testing within OAT is limited to those tests which are required to verify the *non-functional* aspects of the system.

In addition, the software testing should ensure that the portability of the system, as well as working as expected, does not also damage or partially corrupt its operating environment or cause other processes within that environment to become inoperativ[e]

A sample testing cycle

Although variations exist between organizations, there is a typical cycle for testing.[47] The sample below is common among organizations employing the Waterfall development model. The same practices are commonly found in other development models, but might not be as clear or explicit.

- **Requirements analysis**: Testing should begin in the requirements phase of the software development life cycle. During the design phase, testers work to determine what aspects of a design are testable and with what parameters those tests work.
- **Test planning**: Test strategy, test plan, testbed creation. Since many activities will be carried out during testing, a plan is needed.
- **Test development**: Test procedures, test scenarios, test cases, test datasets, test scripts to use in testing software.
- **Test execution**: Testers execute the software based on the plans and test documents then report any errors found to the development team.
- **Test reporting**: Once testing is completed, testers generate metrics and make final reports on their test effort and whether or not the software tested is ready for release.
- **Test result analysis**: Or Defect Analysis, is done by the development team usually along with the client, in order to decide what defects should be assigned, fixed, rejected (i.e. found software working properly) or deferred to be dealt with later.
- **Defect Retesting**: Once a defect has been dealt with by the development team, it is retested by the testing team. AKA Resolution testing.
- **Regression testing**: It is common to have a small test program built of a subset of tests, for each integration of new, modified, or fixed software, in order to ensure that the latest delivery has not ruined anything, and that the software product as a whole is still working correctly.
- **Test Closure**: Once the test meets the exit criteria, the activities such as capturing the key outputs, lessons learned, results, logs, documents related to the project are archived and used as a reference for future projects.



Regression Testing

Test Plan

A] Purpose of preparing a Test Plan

- Validate the acceptability of a software product.
- Help the people outside the test group to understand „why‟ and „how‟ of product validation.
- A Test Plan should be

-Thorough enough (Overall coverage of test to be conducted)

-Useful and understandable by the people inside and outside the test group.

B]. Scope

- The areas to be tested by the QA team.
- Specify the areas which are out of scope (screens, database, mainframe processes etc)

C]. Test Approach

- Details on how the testing is to be performed.
- Any specific strategy is to be followed for testing (including configuration management).

Testing methodologies and testing

1. Black box testing
2. White box testing

1. BLACK BOX TESTING No knowledge of internal design or code required Tests are based on requirements and functionality

1.1 Black Box - Testing Technique Incorrect or missing functions , Interface errors,Errors in data structures or external database access, Performance errors,Initialization and termination errors

1.2. Black box / Functional Testing- Based on requirements and functionality Not based on any knowledge of internal design or code Covers all combined parts of a system Tests are data driven

2. WHITE BOX TESTING Knowledge of the internal program design and code required.Tests are based on coverage of code statements, branches, paths, conditions.

2.1 White Box - Testing Technique All independent paths within a module have been exercised at least once Exercise all logical decisions on their true and false sides Execute all loops at their boundaries and within their operational bounds Exercise internal data structures to ensure their validity

2.2 White box Testing / Structural Testing Based on knowledge of internal logic of an application's code Based on coverage of code statements, branches, paths, conditions. Tests are logic driven

2.3 Other White Box Techniques Statement Coverage – execute all statements at least once

Example :

```
 A + B If (A = 3)
Then B = X + Y
 End-If
While (A > 0) Do
Read (X)
A = A - 1
End-While-Do
```

Decision Coverage – execute each decision direction at least once

Example:

```
If A< 10 or A > 20 Then
B = X + Y
End -if
```

Condition Coverage – execute each decision with all possible outcomes at least once

Example: A = X

```
 If (A > 3) or (A < B)
Then B = X + Y
End-If-Then
 While (A > 0) and (Not EOF) Do
 Read (X)
 A = A – 1
End-While-Do
```

Conclusion

Testing can show the presence of faults in a system; it cannot prove there are no remaining faults. Component developers are responsible for component testing; system testing is the responsibility of a separate team Integration testing is testing increments of the system; release testing involves testing a system to be released to a customer. Use experience and guidelines to design test cases in defect testing. Interface testing is designed to discover defects in the interfaces of composite components. Equivalence partitioning is a way of discovering test cases - all cases in a partition should behave in the same way. Structural analysis relies on analysing a program and deriving tests from this analysis. Test automation reduces testing costs by supporting the test process with a range of software tools.

.**Refrences**

[1] Lessons Learned in Software Testing, by C. Kaner, J. Bach, and B. Pettichord

[2]. Testing Computer Software, by C. Kaner, J. Falk, and H. Nguyen

[3]. Effective Software Testing, by E. Dustin

[4]. Software testing, by Ron Patton

[5]. Software engineering, by Roger Pressman

[6]. http://people.engr.ncsu.edu/txie/testingresearchsurvey.htm

[7]. http://www.engpaper.com

[8]. http://www.people.engr.ncsu.edu/txie/testingresearchsurvey.htm

[9]. www.cs.cmu.edu/luluo/Courses/17939Report.pdf

[10]. www.findwhitepapers.com

[11]. www.scribd.com