Different Software Testing Levels for Detecting Errors

Mohd. Ehmer Khan *ehmerkhan@gmail.com*
Lecturer
Department of Information Technology
Al Musanna College of Technology
P.O. Box-191, PC-314, Sultanate of Oman

Abstract

Software testing is the process to uncover requirement, design and coding errors in the program. But software testing is not a “miracle” that can guarantee the production of high quality software system, so to enhance the quality of a software and to do testing in a more unified way, the testing process could be abstracted to different levels and each level of testing aims to test different aspects of the system. In my paper, I have described different level of testing and these different levels attempt to detect different types of defects. The goal here is to test the system against requirement, and to test requirement themselves.

Keywords: Acceptance Testing, Integration Testing, Regression Testing, System Testing, Unit Testing

1. INTRODUCTION

Software testing is the process of accessing the functionality and correctness of a software through analysis. It also identifies most important defects, flaws, or errors in the application code that must be fixed. The system must be tested in steps with the planned build and release strategies. The key to successful testing strategies is selecting the right level of test at each stage in a project.

The level of testing have a hierarchical structure which build up from the bottom-up where higher level assume successful and satisfactory completion of lower level test. Each level of test is characterized by an environment i.e. type of people, hardware, data etc. and these environmental variables vary from project to project. [1] Each completed level represent a milestone on the project plan and each stage represents a known level of physical integration and quality. These integrated stages are known as level of testing.

The various levels of testing are:

1. Unit Testing
2. Integration Testing
3. System Testing
4. Acceptance Testing
5. Regression Testing

2. LEVELS OF TESTING

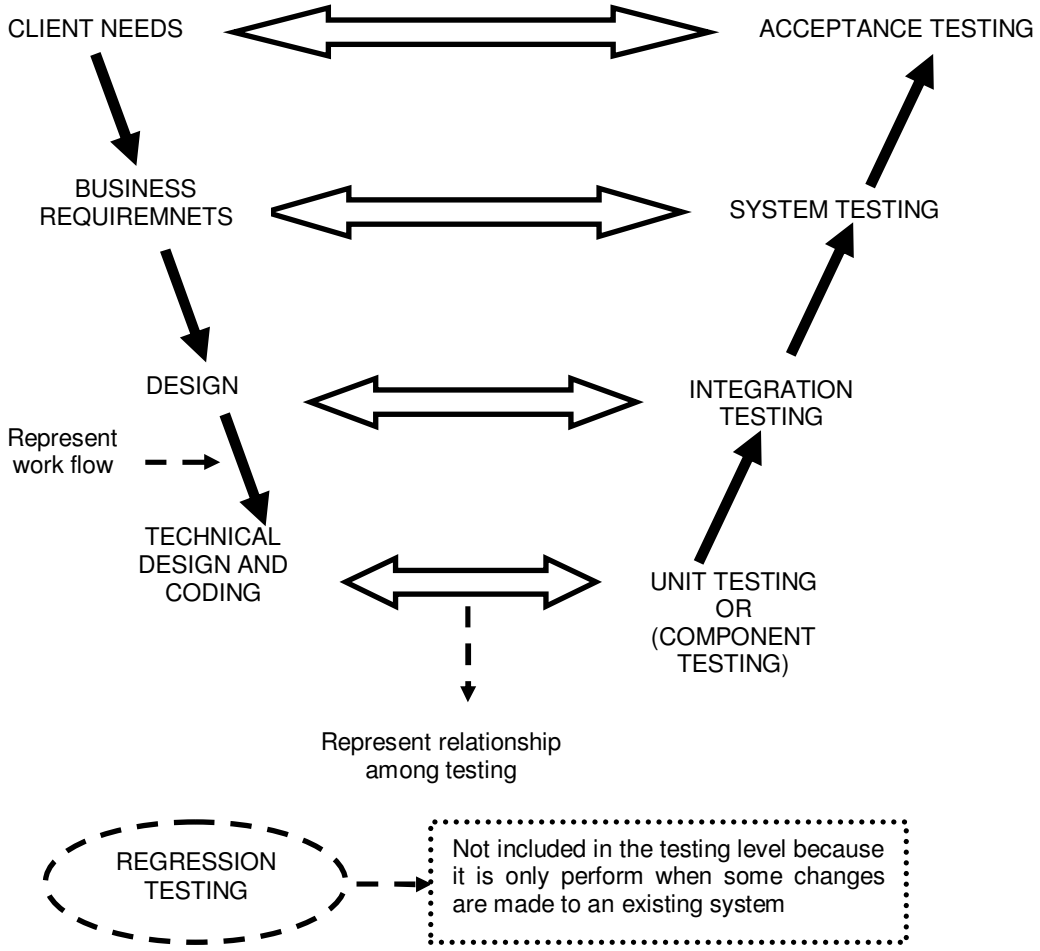


FIGURE 1: Represent various levels of testing

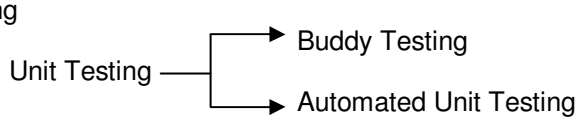
2.1 Unit Testing

Unit testing is also known as component testing, is the first and the lowest level of testing. In this level, individual units/components of software are tested and it is typically done by a programmer of the unit or module (Unit is the smallest piece of software that can be tested). Unit testing help to expose bugs that might appear to be hidden. Unit testing focuses on implementation and also require thorough understanding of the systems functional specification. [1]

Approximate level of documentation is needed for unit testing and there are certain minimum requirements for that documentation. They are as follows:

1. It must be reviewable
2. All the record must be archivable
3. Test can be repeatable

Two Types of unit testing



2.1.1 Buddy Testing

A better approach that has been encouraged is the use of a process called "Buddy Testing". It takes two-person team approach for code implementation and unit testing. Developer A writes the test cases for developer B, while B performs the unit test and vice versa. There are certain advantages of this team approach (buddy testing) – [1]

1. Models of the program specification requirements are served properly as the test cases are created prior to coding
2. Buddy testing provide cross-training on application.
3. The testing process is more objective and productive.

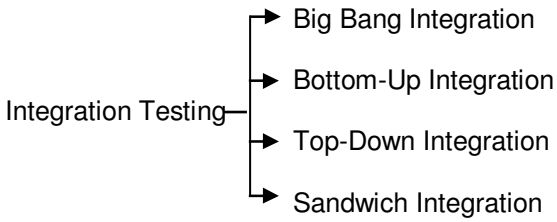
The one disadvantage with buddy testing is the extra time required to write the test cases up front and for the developer to familiarize themselves with each other specification and code.

2.1.2 Automating Unit Testing

Unit testing is usually automated and it is performed within the IDE of programmers. JUNIT (for JAVA) is an example of automated unit testing. RUTE-J a randomized unit testing example of JAVA is an effective and efficient method of testing.

2.2 Integration Testing

The level after unit testing is integration testing either the developer or an independent tester performs integration testing. It involves combining and testing different units of the program. The purpose of integration testing is to verify functional, performance and reliability requirements placed on major design items. [2] Approximately 40% of software errors are revealed during integration testing, so the need of integration testing must not be overlooked. The key purpose of integration testing is to leverage the overall integration structure to allow rigorous testing at each phase while minimizing duplication of efforts. Some different types of integration testing are



2.2.1 Big Bang Integration Testing

Big bang is an approach to integration testing where almost all the units are combined together and tested at one go. It is very effective for saving time in the integration testing process. Usage model testing is a type of big-bang testing and can be used in both software and hardware integration testing. [2]

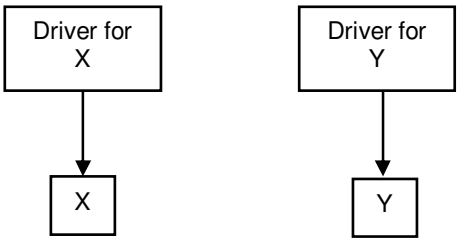
There are certain disadvantages with big bang [3]

1. High cost of repair
2. Minimum observability, diagnosabilty, efficacy and feedback
3. Defects are identified at a very late stage.
4. High probability of missing critical defects.
5. Difficult to isolate the defect found.

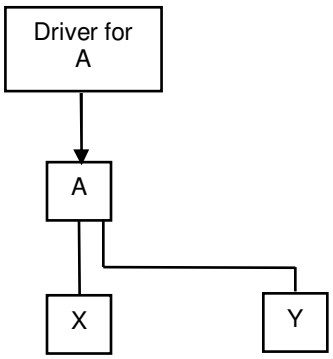
2.2.2 Bottom-Up Integration Testing

In this approach, testing starts at the bottom of the tree. Bottom-Up integration uses test drivers to drive and pass appropriate data to the lower level module. At each stage of bottom-up integration, the units at the higher levels are replaced by drivers (drivers are throw away pieces of code that are used to simulate procedure calls to child). [1]

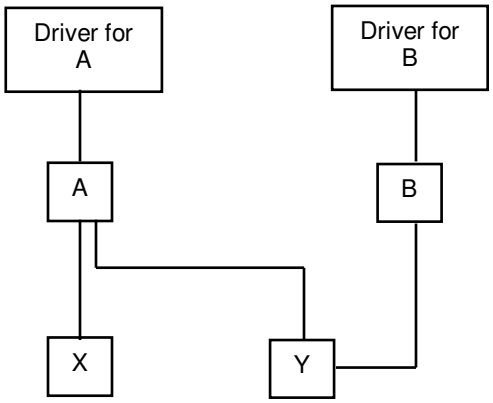
In this approach behaviour of interaction point are crystal clear. On the other hand, writing and maintaining test drivers is more difficult than entering stub.

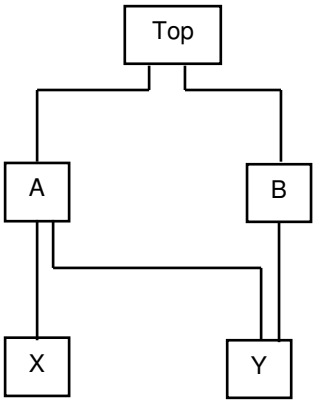


Separate drivers for each module, No. of tests are V(X) + V(Y)



Test a basis set of paths A in integration units X and Y using a driver, No. of tests V(A)



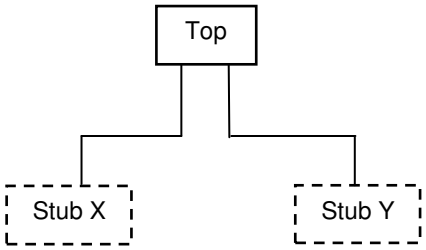


Integrated into single subsystem (Complete Bottom UP)

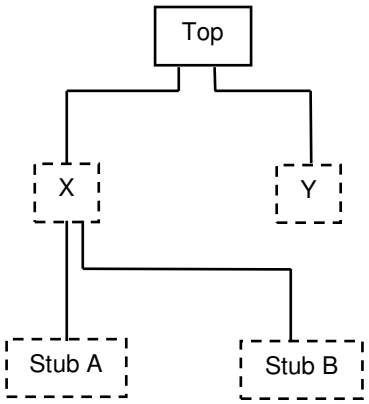
FIGURE 2: Represent complete bottom-up integration testing

2.2.3 Top-Down Integration Testing

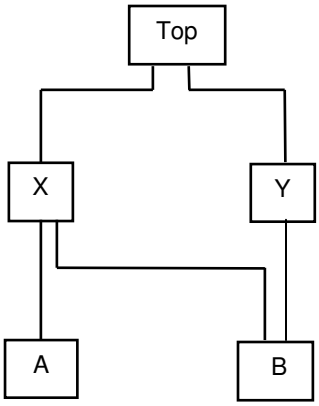
Top-down integration testing starts from the top and then proceeds to its child units. Any other lower level nodes that may be connected should be create as a stub. [4] As we add lower level code, we will replace stubs with actual components. This testing can be performed either breadth first or depth first. It is up to the tester to decide how many stubs should be replaced before the next test is performed. As the system prototype can be developed early on in the project process, this will make work easier and design defect can be found and corrected early. But one disadvantage with top-down approach is that extra work is needed to be done to produce large number of stubs.



Working from top level towards bottom



Write stubs of used module at each step in construction

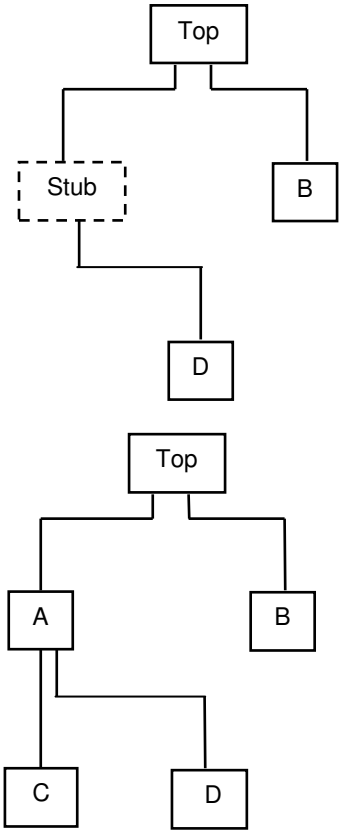


Top down is complete, and all functionality can be tested

FIGURE 3: Represent complete top-down integration testing

2.2.4 Sandwich Integration Testing

This approach combines the functionality of both bottom-up and top-down approach. The lower section unit are tested using bottom-up integration and higher section unit are tested by using top-up integration.[1] Less throw away codes are used by sandwich testing as compare to top down approach. [5]



Working from extremes toward center

It is flexible but complex to plan

FIGURE 4: Represent complete sandwich integration testing

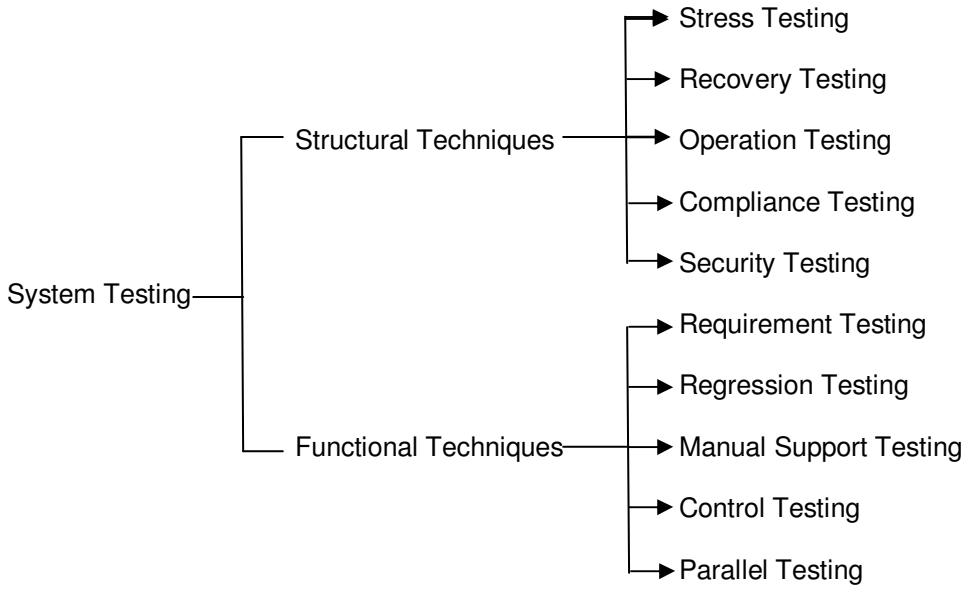
2.3 System Testing

Major level of testing or we can say the core of testing comes in this level, it is called system testing. This phase demands additional skills from a tester because various structural and functional techniques are carried out in this phase. System testing occurs when the system has been deployed onto a standard environment and all necessary components has been released internally.

Besides functional testing, system testing may include testing configuration, security, optimal utilization of resources and performance of the system. System testing is needed as:

1. It reduces cost
2. It increase productivity
3. It reduces commercial risk

The main goal of system testing is to evaluate the system as a whole and not its part. Various forms of testing under system testing are [6]



2.3.1 Structural Techniques

2.3.1.1 Stress Testing

It puts the program under heavy load or stress or we can also define stress testing as type of performance testing conducted to evaluate a system or components at all beyond limits of its specified work load. Stress testing may have a more specified meaning in certain industries, such as fatigue testing for material. [7]

2.3.1.2 Recovery Testing

It is a process of testing to determine the recoverability of the software. Recovery testing is executed to show that whether the recovery function of a system work in a correct manner or not. It also handles how the system recovers from the failure and it handles corrupted data such as data in DBMS and operating system.

2.3.1.3 Operation Testing

Testing conducted to evaluate a component or system in its operational environment. [8] Operation testing also test how the system is fits in with existing operations and procedure in the user organization.

2.3.1.4 Compliance Testing

It is usually done to determine the compliance of a system. It tests adherence to standards.

2.3.1.5 Security Testing

It helps to protect data and maintains the functionality of the system. The main concepts covered by security testing are [7]

1. Confidentiality
2. Integrity
3. Authentication
4. Authorization
5. Availability
6. Non Duplication

Internet based application are good candidate for security testing due to the continuous growth in the number of e-commerce applications.

2.3.2 Functional Techniques

2.3.2.1 Requirement Testing

It is the most fundamental form of testing and it checks and make sure that the system does what it is required to do.

2.3.2.2 Regression Testing

It is performed to uncover new errors, in existing functionality after changes have been made to the software. It assures that a change, such as a bugfix, did not introduce new bugs. [7] Regression testing ensures that the unchanged functionality remains unchanged.

2.3.2.3 Manual Support Testing

It includes user documentation and tests whether the system can be used properly or not. [6]

2.3.2.4 Control Testing

It is the process of testing various required control mechanism for system.

2.3.2.5 Parallel Testing

In parallel testing the same input is feed into two different versions of the system to make sure that both the versions of the system produces the same result. [6]

2.4 Acceptance Testing

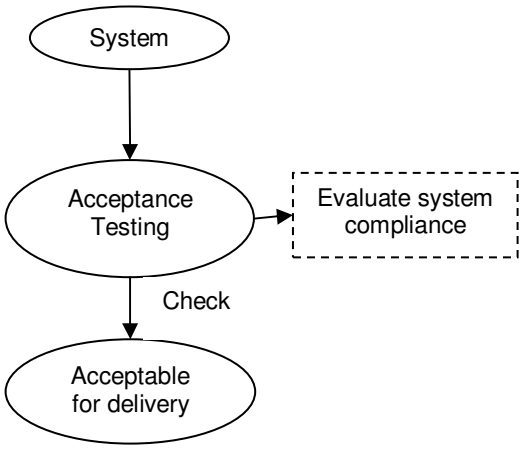
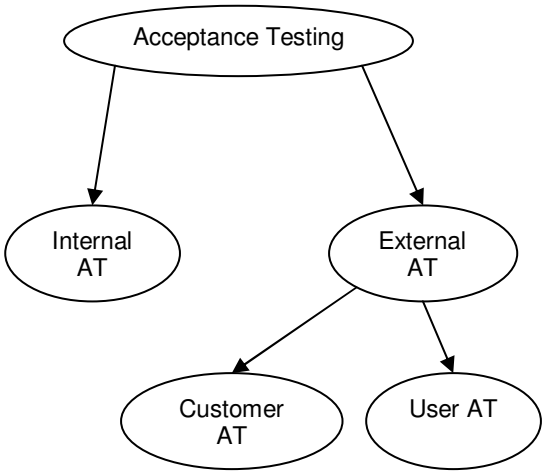


FIGURE 5: Represent acceptance testing

In software engineering acceptance testing is a level of software testing where the system is tested for user acceptability. Acceptance testing checks the system against requirement.

Acceptance testing is performed after system testing and before making the system available for actual use. [9] Sometimes acceptance testing also involves compatibility testing, it happens when a new system is developed to replace the old one.

Types of system testing



2.4.1 Internal Acceptance Testing
It is also known as alpha testing, which is simulated or actual operational testing and it is carried out by the test team who are not directly involved in the project.

2.4.2 External Acceptance Testing
It is performed by the external (not employed in an organization that developed the system)

2.4.2.1 Customer Acceptance Testing
Testing is performed by the customers who asked the organization to develop the software for them (software not being owned by the organization that developed it) [9]

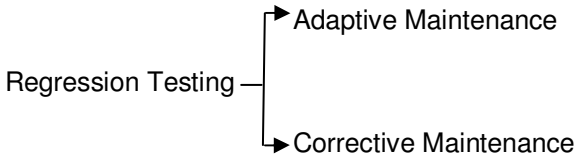
2.4.2.2 User Acceptance Testing
It is also known as beta testing which is operational testing by potential and existing customer at an external site not otherwise involved with the developer, to determine whether or not system satisfies customer needs.

Hence the goal of acceptance testing should verify the overall quality, correct operation, scalability, completeness, usability, portability and robustness of the functional component which is supplied by software system [6]

2.5 Regression Testing
Regression testing is performed when the software or its environment is changed. It is testing of a previously tested program following modification to ensure that defects have not been introduce or uncovered in unchanged areas of software, as a result of the changes made. [8]

Another important reason for regression testing is that it is often extremely difficult for a programmer to find out how the changes in one part of the software effects the other part. [10] Regression testing is a very important aspect of the system maintenance.

Two strategies of regression testing are [1]



2.5.1 Adaptive Maintenance
In adaptive maintenance the system specification are modified. Adaptive regression requires the generation of new test cases to suit the new specification.

2.5.2 Corrective Maintenance
In corrective maintenance the system specification are not modified and its support the reuses of test cases.

3. COMPARING DIFFERENT SOFTWARE TESTING LEVELS

Comparing different levels of testing that are done throughout the software development process are outlined in the table below

Level	Description	Importance
Unit Testing	Verify the functionality of a specific section of code at functional level.	White Box
Integration Testing	Tests the interfaces between component against a software design.	White Box / Black Box
System testing	Test a completely integrated software system and verifies that it satisfies the requirement.	Black Box
Acceptance Testing	It is performed as a part of hand-off process between any two phases of software development process.	Black Box
Regression Testing	Tests the defects that are occurred after a major code change i.e. tests new functionality in a program.	White Box

TABLE 1: Comparison between different software testing levels.

4. CONCLUSION

Software testing has the potential to save time and money by identifying errors early, and to improve customer satisfaction by delivering a more error free product. Software testing normally involves different levels of test case specification, test case generation, test execution, test evaluation and regression testing. Each of these levels plays an important role in the production of the program and meets their desired specification.

We have seen different level of testing so far. Starting from unit testing which is at the lowest level and ensures that the implementation fits the functional specification. Integration testing is next to unit testing and it tests the communication between different components of the system. After integration testing, system testing comes which tests the functionality of software as a complete system. The last level is acceptance testing and it verifies whether the end user is satisfy with the

system or not. Lastly, the regression testing which ensures that the modification applied to a system has not adversely change system behavior.

Irrespective of different levels of testing the testing should encompass the following

1. Cost of failure
2. Identify defect before customer finds them
3. Reduce the risk of releasing
4. Evaluation of product with an independent perspective

5. REFRENCES

[1] Levels of Testing written by Patrick Oladimeji supervised by Prof. Dr. Holger Schlingloff & Dr. Markus Roggenbach published on 1/12/2007 available at http://www.cs.swan.ac.uk/~csmarkus/CS339/dissertations/OladimejiP.pdf

[2] Integration testing available at http://en.wikipedia.org/wiki/Integration_testing

[3] Big bang integration available at http://www.testinggeek.com/big-bang-integration-testing

[4] Integration testing by Thomas Bradley (368100) published on February 8, 2008 available at http://sucs.org/~tobeon/testing.pdf

[5] Integration testing & Component based software testing chapter # 21 by Mauro & Michal Young, 2007 (c) available at http://ix.cs.uoregon.edu/~michal/book/slides/pdf/PezzeYoung-Ch21-integration.pdf

[6] Cognizant Technology Solution available at pp 44, 50, 52 & 61 available at http://www.scribd.com/doc/6749799/Software-Testing-COGNIZANT-Notes

[7] System testing available at http://en.wikipedia.org/wiki/System_testing

[8] Standard glossary of terms used in Software Testing (ISTQB) version 2.1 (dd. April 1st, 2010) by Erik van Veenendaal (The Netherlands) available at http://istqb.org/download/attachments/2326555/ISTQB+Glossary+of+Testing+Terms+2+1.pdf

[9] Levels of software testing available at http://softwaretestingfundamentals.com/software-testing-levels/

[10] Regression testing available at http://en.wikipedia.org/wiki/Regression_testing