International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Software Testing - Principles, Lifecycle, Limitations and Methods

Anupriya[1], Ajeta[2]

[1]Department of Computer Science, MKJK College, Rohtak, Haryana , India

[2]Department of Computer Science and Engineering, Chhotu Ram Polytechnic, Rohtak , Haryana, India

Abstract: ***Software testing provides a means to reduce errors, cut maintenance and overall software costs. Various software development and testing methodologies, tools, and techniques have emerged over the last few decades promising to enhance software quality. While it can be argued that there has been some improvement it is apparent that many of the techniques and tools are isolated to a specific lifecycle phase or functional area. One of the major problems within software testing area is how to get a suitable set of cases to test a software system. This set should assure maximum effectiveness with the least possible number of test cases. There are now numerous testing techniques available for generating test cases.***

Keywords: Software testing, Unit testing, System testing, Acceptance testing**.**

1. Introduction

Software testing is as old as the hills in the history of digital computers. The testing of software is an important means of assessing the software to determine its quality. Since testing typically consumes 40 - 50% of development efforts, and consumes more effort for systems that require higher levels of reliability, it is a significant part of the software engineering. Software testing is more than just error detection; Testing software is operating the software under controlled conditions[1], 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 wanted.

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. [*Validation: Are we building the right system?*]

The definition of testing according to the ANSI/IEEE 1059 standard is that testing is the process of analyzing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item. The purpose of testing is verification, validation and error detection in order to find problems – and the purpose of finding those problems is to get them fixed.

1.1 Objectives of testing

- Executing a program with the intent of finding an *error*.
- To check if the system meets the requirements and be executed successfully in the Intended environment.
- To check if the system is "Fit for purpose".
- To check if the system does what it is expected to do.

1.2 Features of Good Test case

- A good test case is one that has a probability of finding an as yet undiscovered error.
- A successful test is one that uncovers a yet undiscovered error.
- A good test is not redundant.
- A good test should be "best of breed".
- A good test should neither be too simple nor too complex.

1.3 Objective of a Software Tester

- Find bugs as early as possible and make sure they get fixed.
- To understand the application well.
- Study the functionality in detail to find where the bugs are likely to occur.
- Study the code to ensure that each and every line of code is tested.
- Create test cases in such a way that testing is done to uncover the hidden bugs and also ensure that the software is usable and reliable

1.4 Testing Principles

Principle is the rule or method in action that has to be followed. Different testing principles are as follows: [2]

1) Test a program to try to make it fail: Testing is the process of executing a program with the intent of finding errors. We should expose failures to make testing process more effective.
2) Start testing early: This helps in fixing enormous errors in early stages of development, reduces the rework of finding the errors in the initial stages.
3) Testing is context dependant: Testing should be appropriate and different for different points of time.
4) Define Test Plan: Test Plan usually describes test scope, test objectives, test strategy, test environment, deliverables of the test, risks and mitigation, schedule,

levels of testing to be applied, methods, techniques and tools to be used. Test plan should efficiently meet the needs of an organization and clients as well.

5) Design Effective Test cases: Test case must be specified in a way that is measurable so that testing results are unambiguous.
6) Test for valid as well as invalid Conditions: In addition to valid inputs, we should also test system for invalid and unexpected inputs/conditions
7) Testing must be done by different persons at different levels: Different purpose addressed at different level of testing so different person should perform testing differently using different testing techniques at different level.
8) End of Testing: Testing has to be stopped somewhere. The testing can be stopped when risk is under some limit or if there is limitation.

2. Software Testing Lifecycle Phases

1. Requirements study
- Testing Cycle starts with the study of client's requirements.
- Understanding of the requirements is very essential for testing the product.

2. Test Case Design and Development
- Component Identification
- Test Specification Design
- Test Specification Review

3. Test Execution
- Code Review
- Test execution and evaluation
- Performance and simulation

4. Test Closure
- Test summary report
- Project De-brief
- Project Documentation

5. Test Process Analysis
Analysis done on the reports and improving the application‟s performance by implementing new technology and additional features.

3. Level of Testing

3.1 Unit Testing

It is the lowest level of testing as particular code modules have to be tested. It is done after modules are being coded. Internal Application Design, Master Test Plan, Unit Test Plan act as input in this testing[3]. As an output, unit test output report is generated. It is done by white box testing techniques and test coverage techniques. Debug, Restructure, Code Analyzers, Path/statement coverage tools can also be used to complete the testing. It is done by programmers as shown in figure 1 (not by testers).

Objectives of Unit testing: Following are the main objectives of Unit testing**:**

1. To test the function of a program or unit of code such as a program or module
2. To test internal logic
3. To verify internal logic.
4. To test path and condition coverage.
5. To test exception conditions & error handling

3.2 Incremental Integration Testing

Continuous testing of an application as and when a new functionality is added is done. This testing is done after unit testing. Applications functionality aspects are required to be independent enough to work separately before completion of development[4]. This testing is done by programmers or testers as shown in figure 1. Internal and external Application Design, Master Test Plan, Integration Test Plan act as input in this testing. As an output, Integration test output report is generated. It is done by white and black box testing techniques and Configuration Management techniques. Debug, Restructure, Code Analyzers can also be used to complete the testing.

Integration Testing
It involves building a system from its components and testing it for problems that arise from component interactions.

(i) **Top-down integration:** Develop the skeleton of the system and populate it with components.
(ii) **Bottom-up integration:** Integrate infrastructure components then add functional components.

Objective of Integration Testing: To technically verify proper interfacing between modules, and within sub-systems.

3.3 System Testing

This testing is done after integration testing. This testing is done by development teams as shown in figure 1. Detailed Requirements & External Application Design, Master Test act as input in this testing. System Test Report is generated. Configuration management Recommended set of tools can also be used to complete the testing.

Objectives of System testing: Following are the main objectives of system testing**:**

1) To verify that the system components perform control functions
2) To perform inter-system test
3) To demonstrate that the system performs both functionally and operationally as specified
4) To perform appropriate types of tests relating to Transaction Flow, Installation, Reliability, Regression

Figure 1

3.4 Acceptance Testing

This testing is done after system testing. This testing is done by User / End User as shown in figure 1. Business Needs & Detailed Requirements, Master Test Plan, User Acceptance Test Plan act as input in this testing. User Acceptance Test report is generated. It is done by Black Box techniques; Configuration Management tools can also be used. Compare, Keystroke capture & Playback, Regression testing can be used as tools to complete the testing.

Objective of Acceptance Testing:- To verify that the system meets the user requirements

4. Software Testing Limitations

Limitation is a principle that restricts the extent of any application. Software testing has also few limitations that should be considered to set realistic expectations about its benefits. Inspite of being most widely used verification technique, software testing as various following limitations:

1) Testing can be used to show the presence of errors, but never to show their absence [5]. It can only identify the known issues or errors. It gives no idea about defects still uncovered. Testing cannot guarantee that the system under test is error free.
2) Testing provides no help when we have to make a decision to either "release the product with errors for meeting the deadline" or to "release the product late compromising the deadline"[6].
3) Testing cannot establish that a product functions properly under all conditions but can only establish that it does not function properly under specific conditions [7].
4) Software testing does not help in finding root causes which resulted in injection of defects in the first place. Locating root causes of failures can help us in preventing injection of such faults in future.

5. Conclusion

This paper on Software testing discusses about software testing, objectives of software testing, principles. It further discusses about features of a good tester and a test case. Software testing is often less formal and rigorous than it should be, and a main reason for that is because we have struggled to define best practices, methodologies, principles, standards for optimal software testing. To perform testing effectively and efficiently, everyone involved with testing should be familiar with basic software testing goals, principles, limitations and concepts. Further different Software testing methods i.e. unit testing, integration testing, system testing and acceptance testing are described. Already a lot of work has been done in this field, and even continues today. Implementing testing principles in real world software development, to achieve the testing objectives to maximum extent keeping in consideration the testing limitations will validate the research and will become a way for future research.

References

[1] Ian Somerville, Software Engineering, Addison-Wesley, 2001.
[2] Myers, Glenford J., ―The art of software testing, New York: Wiley, c1979. ISBN: 0471043281
[3] Miller, William E. Howden, "Tutorial, software testing &validation techniques", IEEE Computer Society Press, 1981.
[4] S.M.K Quadri and Sheikh Umar Farooq,"Software Testing-Goals,Principles and Limitations," *International Journal of Computer Applications, Volume 6-No.9,September 2010*.
[5] Mohd. Ehmer Khan, "Different Forms of Software Testing Techniques for Finding Errors",*IJCSI International Journal of Computer Science Issues,Vol. 7, Issue 3, No 1, May 2010*.
[6] Roger S. Pressman, "Software engineering: A practitioner‟s Approach," fifth edition, 2001.
[7] Salim Istaq et al., ―Debugging, Advanced Debugging and Runtime Analysis―, (IJCSE) International Journal on Computer Science and Engineering, Vol. 02, No. 02, 2010, 246-249.