International Journal of Advances in Engineering Research http://www.ijaer.com
(IJAER) 2012, Vol. No. 4, Issue No. IV, October **ISSN: 2231-5152**

SOFTWARE TESTING LEVELS APPROACH AND OBJECTIVE OF TESTING

Dr. Umesh Sehgal[1], Arshdeep[2]
[1,2]Arni School of Computer Science, Arni University, HP

ABSTRACT

Software development has long aspired to merit the status of a professional engineering discipline like those of the established engineering branches. This paper discusses this aspiration with particular reference to software-intensive or computer-based systems. Some opportunities are pointed out for learning important lessons from the established branches. These lessons stem above all from the highly specialized nature of traditional engineering practice. They centre on the crucial distinction between radical and normal design, the content of normal design practice, and the social and cultural infrastructures that make effective specialization possible.
KEYWORD:* - *Software Testing

INTRODUCTION

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, but are not limited to, the process of executing a program or application with the intent of finding software bugs (errors or otherdefects).

Software testing can be stated as the process of validating and verifying that a computer program/application/product:

- meets the requirements that guided its design anddevelopment,
- works asexpected,
- can be implemented with the samecharacteristics,

Software testing, depending on the testing method employed, can be implemented at any time in the development process. Traditionally most of the test effort occurs after the requirements have been defined and the coding process has been completed, but in the Agile approaches most of the test effort is on-going. As such, the methodology of the test is governed by the chosen software development methodology.
Different software development models will focus the test effort at different points in the development process. Newer development models, such as Agile, often employ test-driven and place an increased

portion of the testing in the hands of the developer, before it reaches a formal team of testers. In a more traditional model, most of the test execution occurs after therequirements

Software development process

- Requirements
- Specification
- Architecture
- Design
- Implementation
- Testing
- Debugging
- Maintenance

- TestingLevels
    - Unit testing
    - Integrationtesting
    - Systemtesting
    - Acceptancetesting

1. Testinglevels

Tests are frequently grouped by where they are added in the software development process, or by the level of specificity of the test. The main levels during the development process as defined by the SWEBOK guide are unit-, integration-, and system testing that are distinguished by the test target without implying a specific process model. Other test levels are classified by the testing objective.

- **Unittesting**

  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 assure that the building blocks the software uses work independently of each other.

- **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 localised 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 asystem.

OBJECTIVES OF TESTING

1 Installationtesting

An installation test assures that the system is installed correctly and working at actual customer's hardware

2 Compatibilitytesting

A common cause of software failure (real or perceived) is a lack of its compatibility with other application software, operating system (or operating system versions, old or new), or target environments that differ greatly from the original (such as a terminal or GUI application intended to be run on the desktop now being required to become a web application, which must render in a web browser). For example, in the case of a lack of backward compatibility, this can occur because the programmers develop and test software only on the latest version of the target environment, which not all users may be running. This result in the unintended consequence that the latest work may not function on earlier versions of the target environment or on older hardware those earlier versions of the target environment was capable of using. Sometimes such issues can be fixed by proactively abstracting operating system functionality into a separate program module orlibrary..

3 Smoke and sanitytesting

Sanity testing determines whether it is reasonable to proceed with further testing.
Smoke testing is used to determine whether there are serious problems with a piece of software, for example as a build verification test.

4 Regressiontesting

Regression testing focuses on finding defects after a major code change has occurred. Specifically, it seeks to uncover software regressions, or old bugs that have come back. Such regressions occur whenever software functionality that was previously working correctly stops working as intended. Typically, regressions occur as an unintended consequence of program changes, when the newly developed part of the software collides with the previously existing code. Common methods of regression testing include re-running previously run tests and checking whether previously fixed faults have re-emerged. The depth of testing depends on the phase in the release process and the risk of the added features. They can either be complete, for changes added late in the release or deemed to be risky, to very shallow, consisting of positive tests on each feature, if the changes are early in the release or deemed to be of lowrisk.

5 Acceptancetesting

Acceptance testing can mean one of two things:

1. A smoke test is used as an acceptance test prior to introducing a new build to the main testing process, i.e. before integration or regression.
2. Acceptance testing performed by the customer, often in their lab environment on their own hardware, is known as user acceptance testing (UAT). Acceptance testing may be performed aspart of the hand-off process between any two phases ofdevelopment

CONCLUSION

The Software testing is the process to make the software or project error free or having the all features or implementation of the software easy, user friendly or system compatible so that user will not phase any problem in the time of access the software. At the last we can say that Software have the Quality Assurence.

REFERENCES

1. Exploratory Testing, CemKaner, Florida Institute of Technology, *Quality Assurance Institute Worldwide Annual Software Testing Conference*, Orlando, FL, November2006.
2. Software Testing by Jiantao Pan, Carnegie MellonUniversit.
3. Kolawa, Adam; Huizinga, Dorota (2007). Automated Defect Prevention: Best Practices inSoftware Management. Wiley-IEEE Computer Society Press. pp.41–43.
4. see D. Gelperin and W.C.Hetzel.
5. Myers, Glenford J. (1979). *The Art of Software Testing*. John Wiley and Sons. ISBN 0-471-04328-1.