Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 10, October 2014, pg.988 – 993

RESEARCH ARTICLE

Software Testing Techniques

Sukhdev Singh Ghuman

SBDSM Khalsa College Domeli (Kapurthala) Punjab, India

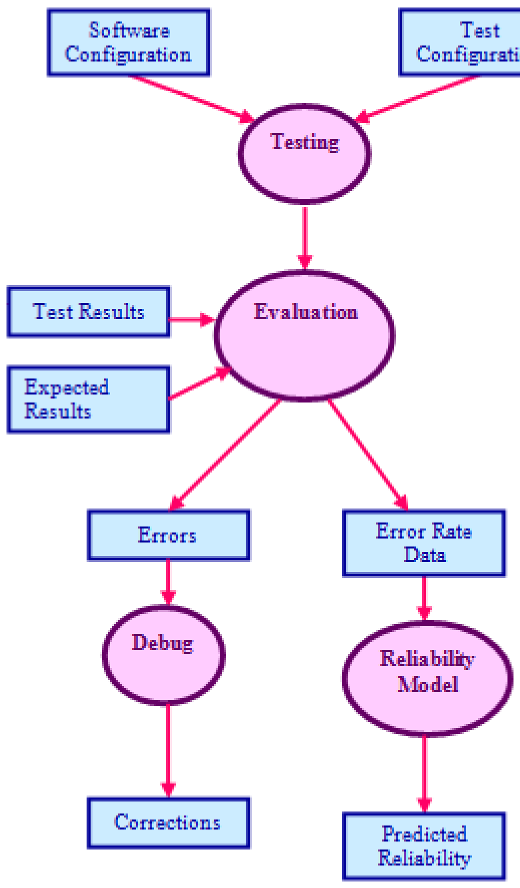
Abstract: Software testing in very important activity in software development. It is one of the important activity which require lot of time and labour. Different testing techniques are used to find bugs in the software. Testing is involved at different stages of software development like unit testing, integration testing, system testing, acceptance testing etc. Different testing techniques namely dynamic testing, functional testing and structural testing are used to test software.

I. INTRODUCTION

Software testing is conducted to ascertain 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. Testing techniques include the process of executing a program or application with the intent of finding software bugs [1]. The testing of software is an important means of assessing the software to determine its quality. Since testing typically consumes 40 to 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. Testing is very important area of computer science.

II. TEST INFORMATION FLOW

Testing is an activity to evaluate the quality of software and improving it by removing errors in it. Hence, the goal of testing is systematical detection of different classes of errors in a minimum amount of time and with a minimum amount of effort [6]. The information flow diagram is as given below:-



III. DIFFERENT LELVELS OF TESTING

Tests are frequently grouped by where they are added in the software development process, or by the level of specificity of the test. Testing is involved in every stage of software life cycle, but the testing done at each level of software development is different in nature and has different objectives.

Unit Testing

It is done at the lowest level. It tests the basic unit of software, which is the smallest testable piece of software. It is also called module or component testing. It 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.

Integration Testing

It is performed when two or more tested units are combined into a larger structure. The test is often done on both the interfaces between the components and the larger structure being constructed, if its quality property cannot be assessed from its components. 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 one at a time or all components together. Integration testing works to expose defects in the interfaces and interaction between integrated components. 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 [1] [2].

System Testing

It tends to affirm the end-to-end quality of the entire system. System test is often based on the functional or requirement specification of the system. Non-functional quality attributes, such as reliability, security, and maintainability, are also checked [1].

Acceptance Testing

Acceptance is used to conduct operational readiness of a product, service or system as part of a quality management system. It 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 [1]. It is done when the completed system is handed over from the developers to the customers or users. The purpose of acceptance testing is rather to give confidence that the system is working than to find errors [2].

Alpha testing

Alpha testing is simulated or actual operational testing by potential users/customers or an independent test team at the developers' site. Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing, before the software goes to beta testing. [1]

Beta testing

Beta testing comes after alpha testing and can be considered a form of external user acceptance testing. Versions of the software, known as beta versions, are released to a limited audience outside of the programming team. The software is released to groups of people so that further testing can ensure the product has few faults or bugs. Sometimes, beta versions are made available to the open public to increase the feedback field to a maximal number of future users. [1]

Regression testing

It 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 low risk.

TABLE 1: LEVELS OF TESTING [5]

Testing Type	Specification	General Scope	Opacity	Tester
Unit	Low level design; Actual code	Classes	White box	Programmer
Integration	Low level design; High level design	Multiple classes	White box; Black box	Programmer
Functional	High level design	Whole product	Black box	Independent tester
System	Requirement Analysis	Whole product in environment	Black box	Independent tester
Acceptance	Requirement	Whole product in	Black box	Customer

	Analysis	environment		
Beta	Ad hoc	Whole product in environment	Black box	Customer
Regression	Changed Documentation; High level design	Any of the above	Black box; White box	Programmer or Independent tester

IV. TESTING GOALS [4]

A goal is a projected state of affairs that a person or system plans or intends to achieve. A goal has to be accomplishable and measurable. It is good if all goals are interrelated. In testing we can describe goals as intended outputs of the software testing process. Software testing has following goals:

2.1 Verification and Validation

It would not be right to say that testing is done only to find faults. Faults will be found by everybody using the software. Testing is a quality control measure used to verify that a product works as desired [10]. Software testing provides a status report of the actual product in comparison to product requirements (written and implicit). Testing process has to verify and validate whether the software fulfills conditions laid down for its release/use [8]. Testing should reveal as many errors as possible in the software under test, check whether it meets its requirements and also bring it to an acceptable level

of quality.

2.2 Priority Coverage

Exhaustive testing is impossible [9]. We should perform tests efficiently and effectively, within budgetary and scheduling limitations. Therefore testing needs to assign effort reasonably and prioritize thoroughly. Generally every feature should be tested at least with one valid input case. We can also test input permutations, invalid input, and non-functional requirements depending upon the operational profile of software. Highly present and frequent use scenarios should have more coverage than infrequently encountered and insignificant scenarios. A study by [3] on 25 million lines of code also revealed that 70-80% of problems were due to 10-15% of modules , 90% of all defects were in modules containing 13% of the code, 95% of serious defects were from just 2.5% of the code. Pareto principle also states that 80 percent of all software defects uncovered during testing will likely be traceable to 20 percent of all program components [2]. The problem, of course, is to isolate these suspect components and to thoroughly test them. Overall we target a wide breadth of coverage with depth in high use areas and as time and budget permits.

2.3 Balanced

Testing process must balance the written requirements, real-world technical limitations, and user expectations. The testing process and its results must be repeatable and independent of the tester, i.e., consistent and unbiased [4]. Apart from the process being employed in development there will be a lot unwritten or implicit requirements. While testing, the software testing team should keep all such requirements in mind. They must also realize that we are part of development team, not the users of the software. Testers personal views are but one of many considerations. Bias in a tester invariably leads to a bias in coverage. The end user's viewpoint is obviously vital to the success of the software, but it is not all that matters as all needs cannot be fulfilled because of technical, budgetary or scheduling limitations. Every defect/shortcoming has to be prioritized with respect to their time and technical constraints.

2.4 Traceable

Documenting both the successes and failures helps in easing the process of testing. What was tested, and how it was tested, are needed as part of an ongoing testing process. Such things serve as a means to eliminate duplicate testing effort [10]. Test plans should be clear enough to be re-read and comprehended. We should agree on the common established documentation methods to avoid the chaos and to make documentation more useful in error prevention.

2.5 Deterministic

Problem detection should not be random in testing. We should know what are we doing, what are we targeting, what will be the possible outcome. Coverage criteria should expose all defects of a decided nature and priority. Also, afterward surfacing errors should be categorized as to which section in the coverage it would have occurred, and can thus present a definite cost in detecting such defects in future testing. Having clean insight into the process allows us to better estimate costs and to better direct the overall development.

V. TESTING TECHNIQUES

The different testing techniques are available for testing software. We can use as per our requirements.

Static Testing

Static program analysis is the analysis of computer software that is performed without actually executing programs. [1] In most cases the analysis is performed on some version of the source code, and in the other cases, some form of the object code. Different static or Manual testing Techniques are as listed below [3]

- Walk through
- Informal Review
- Technical Review
- Inspection

Dynamic Testing

Dynamic testing is also known as dynamic analysis. It is a term used in software testing to describe the testing of the dynamic behavior of the program or code. It is concerned with the examination of the physical response from the system to variables that are not constant and change with time. In dynamic testing the software must actually be compiled and run. It involves working with the software, giving input values and checking if the output is as expected by executing specific test cases which can be done manually or with the use of an automated process. This is in contrast to static testing. Unit tests, integration tests, system tests and acceptance tests utilize dynamic testing.

Functional Testing

Functional testing is a quality assurance process and a type of black box testing that bases its test cases on the specifications of the software component under test. Functions are tested by feeding them input and examining the output, and internal program structure is rarely considered (not like in white-box testing). [2] Functional testing usually describes what the system does. [1] The selection of test cases for functional testing is based on the requirement or design specification of the software entity under test. Examples of expected results sometimes are called test oracles, include requirement/design specifications, hand calculated values, and simulated results. Functional testing emphasizes on the external behavior of the software entity. [2]

Functional testing typically involves six steps [1]

- The identification of functions that the software is expected to perform
- The creation of input data based on the function's specifications
- The determination of output based on the function's specifications
- The execution of the test case
- The comparison of actual and expected outputs
- To check whether the application works as per the customer need.

Structural Testing:

In structural testing the software entity is viewed as a white box. The selection of test cases is based on the implementation of the software entity. The goal of selecting such test cases is to cause the execution of specific spots in the software entity, such as specific statements, program branches or paths. The expected results are evaluated on a set of coverage criteria. Examples of coverage criteria include path coverage, branch coverage, and data-flow coverage. Structural testing emphasizes on the internal structure of the software entity. [2]

VI. CONCLUSION

Testing is a process to evaluate the quality of software. It is labour intensive activity. Different types of testing are used to test software. There is scope for automation in the activities of testing but testers experience is very much important for successful testing. Software testing is component of software quality control. Different types of tests are used to testing like unit testing, integration testing, acceptance testing, system testing are used to test a system. Techniques of testing like static testing, functional testing, dynamic testing and structural testing have been used to test the system.

REFERENCES

[1] https://en.wikipedia.org/wiki/Software_testing

[2] Lu Luo ,“ Software Testing Techniques Technology Maturation and Research Strategy”, Class Report for 17-939A.

[3] Abhijit A. Sawant, Pranit H. Bari, P. M. Chawan , “Software Testing Techniques and Strategies “ , International Journal of Engineering Research and Applications ,Vol. 2 Issue 3, May-Jun 2012, pp.980-986

[4] SMK Qadri et. al, “Software Testing – Goals, Principles, and Limitations”, International Journal of Computer Applications, Volume 6– No.9, September 2010

[5] http://www.cs.umd.edu/~mvz/cmsc435-s09/pdf/slides10.pdf

[6] Jovanović Irena, “Software Testing Methods and Techniques”