A Review of Testing Techniques and Principles in Software Quality Assurance Testing

B. Vasundhara Devi

ABSTRACT: To judge any software product it should completely free from errors, faults, bugs, and vulnerabilities that is, product should completely correct, complete, fit for use Nothing but it should satisfy all the internal, external requirements and should comply with the given functionalities then only anyone can judge the quality of the software product for these we have got plenty of tools which are entrepreneur and open source This paper describes Software testing techniques and importance of in engineering the software product and also tells about Software testing goals and principles. And also describes Software testing techniques and strategies. Finally it describes the difference between software testing and debugging.

Keywords—Debugging, verification, validation, Software Testing Goals, Software Testing principles, Software Testing Techniques, Software Testing strategies

IJSER

I. INTRODUCTION

Software testing is a process used to identify the correctness, completeness, and quality of developed computer software**.** It includes a set of activities conducted with the intent of finding errors in software so that it could be corrected before the product is released to the end users. In simple words, software testing is an activity to check whether the actual results match the expected results and to ensure that the software system is defect free**.**

1.1 Why Testing is Important?

- China Airlines Airbus A300 crashing due to a software bug on April 26, 1994 killing 264 innocent lives.
- Software bugs can potentially cause monetary and human loss, history is full of such examples. In 1985, Canada's Therac-25 radiation therapy machine malfunctioned due to software bug and delivered lethal radiation doses to patients, leaving 3 people dead and critically injuring 3 others.
- In April of 1999 ,a software bug caused the failure of a $1.2 billion military satellite launch, the costliest accident in history
- In may of 1996, a software bug caused the bank accounts of 823 customers of a major U.S. bank to be credited with 920 million US dollars As you see, testing is important because software bugs could be expensive or even dangerous.
- As Paul Elrich puts it - "To err is human, but to really foul things up you need a computer."

II. SOFTWARE TESTING GOALS

You simply say that software testing is nothing but validation and verification. Main goal of software Testing is to ensure that software should always be defect free and easily maintained. Software Testing has different goals and objectives. The major objectives of Software testing are as follows:

- Finding defects which may get created by the programmer while developing the software.
- Gaining confidence in and providing information about the level of quality.
- To prevent defects.
- To make sure that the end result meets the business and user requirements.
- To ensure that it satisfies the BRS that is Business Requirement Specification and SRS that is System Requirement Specifications.
- To gain the confidence of the customers by providing them a quality product.

Testing should systematically uncover different classes of errors in a minimum amount of time and with a minimum amount of effort. A secondary benefit of testing is that it demonstrates that the software appears to be working as stated in the specifications. The data collected through testing can also provide an indication of the software's reliability and quality. But, testing cannot show the absence of defect -- it can only show that software defects are present.

III. SOFTWARE TESTING PRINCIPLES

There are seven principles of testing. They are as follows:

1) Testing shows presence of defects: Testing can show the defects are present, but cannot prove that there are no defects. Even after testing the application or product thoroughly we cannot say that the product is 100% defect free. Testing always reduces the number of undiscovered defects remaining in the software but even if no defects are found, it is not a proof of correctness.

2) Exhaustive testing is impossible: Testing everything including all combinations of inputs and preconditions is not possible. So, instead of doing the exhaustive testing we can use risks and priorities to focus testing efforts. For example: In an application in one screen there are 15 input fields, each having 5 possible values, then to test all the valid combinations you would need 30 517 578 125 (5^{15}) tests. This is very unlikely that the project timescales would allow for this number of tests. So, accessing and managing risk is one of the most important activities and reason for testing in any project.

3) Early testing: In the software development life cycle testing activities should start as early as possible and should be focused on defined objectives.

4) Defect clustering: A small number of modules contains most of the defects discovered during pre-release testing or shows the most operational failures.

5) Pesticide paradox: If the same kinds of tests are repeated again and again, eventually the same set of test cases will no longer be able to find any new bugs. To overcome this “Pesticide Paradox”, it is really very important to review the test cases regularly and new and different tests need to be written to exercise different parts of the software or system to potentially find more defects.

6) Testing is context depending: Testing is basically context dependent. Different kinds of sites are tested differently. For example, safety – critical software is tested differently from an e-commerce site.

7) Absence – of – errors fallacy: If the system built is unusable and does not fulfill the user’s needs and expectations then finding and fixing defects does not help.

IV. SOFTWARE TESTING TECHNIQUES

In this Section the focus is mainly on the different software testing Techniques. Software Testing Techniques can be divided into two types:-

4.1. MANUAL TESTING (Stress Testing)

It is a slow process and laborious where testing is done statically .It is done in early phase of life cycle. It is also called static testing. It is done by analyst, developer and testing team. Different Manual testing Techniques are as follows:- A) walk through B) Informal Review C) Technical Review D) Inspection

4.2. AUTOMATED TESTING (Dynamic Testing)

In this tester runs the script on the testing tool and testing is done. Automated testing is also called dynamic testing. Automated testing is further classified into four types

1. Correctness testing
2. Performance testing
3. Reliability testing
4. Security testing

IJSER

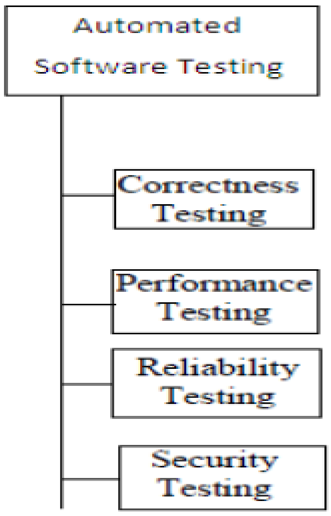

Fig 1:-Further classification of Automated software Testing.

4.2.1. CORRECTNESS TESTING

Correctness is the minimum requirement of software. Correctness testing will need some type of oracle, to tell the right behavior from the wrong one. The tester may or may not know the inside details of the software module under test. [3] Therefore either white box testing or black box testing can be used. Correctness testing has following three forms:-

A. White box testing
B. Black box testing
C. Grey box testing

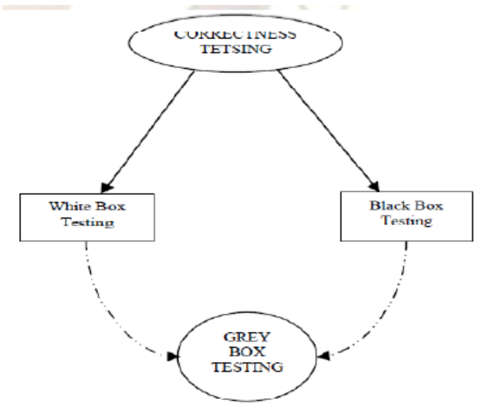

Fig 2:- Different form of Correctness testing [3].

A. WHITE BOX TESTING

White box testing is highly effective in detecting and resolving problems, because bugs can often be found before they cause trouble.[5] White box testing is the process of giving the input to the system and checking how the system processes that input to generate the required output. White box testing is also called white box analysis, clear box testing or clear box analysis.[5] White box testing is applicable at integration, unit and system levels of the software testing process.[3] White box testing is considered as a security testing method that can be used to validate whether code implementation follows intended design, to validate implemented security functionality, and to uncover exploitable vulnerabilities. Some Different types of white box testing techniques are as follows:-

1) Basis Path Testing
2) Loop Testing
3) Control Structure Testing

Advantages of white box testing:-

1) All independent paths in a module will be exercised at least once.
2) All logical decisions will be exercised.
3) All loops at their boundaries will be executed.
4) Internal data structures will be exercised to maintain their validity.
5) Errors in hidden codes are revealed.
6) Approximate the partitioning done by execution equivalence.
7) Developer carefully gives reason about implementation.

Disadvantages of white box testing:-

1) Missed out the cases omitted in the code.
2) As knowledge of code and internal structure is a prerequisite, a skilled tester is needed to carry out this type of testing, which increases the cost.
3) And it is nearly impossible to look into every bit of code to find out hidden errors, which may create problems, resulting in failure of the application.

B. BLACK BOX TESTING

Black box testing is testing software based on output requirements and without any knowledge of the internal structure or coding in the program.[5]

Basically Black box testing is an integral part of „Correctness testing" but its ideas are not limited to correctness testing only. The goal is to test how well the component conforms to the published requirement for the component. Black box testing have little or no regard to the internal logical structure of the system, it only examines the fundamental aspect of the system. It makes sure that input is properly accepted and output is correctly produced. [3] Some Different types of Black box testing techniques are as follows:-

1) Equivalent Partitioning
2) Boundary value Analysis
3) Cause-Effect Graphing Techniques
4) Comparison Testing
5) Fuzz Testing
6) Model-based testing

Advantages of Black box testing:-

1) The number of test cases are reduced to achieve reasonable testing
2) The test cases can show presence or absence of classes of errors.
3) Black box tester has no "bond" with the code.
4) Programmer and tester both are independent of each other.
5) More effective on larger units of code than clear box testing.

Disadvantages of Black box testing:-

1) Test cases are hard to design without clear specifications.
2) Only small numbers of possible input can actually be tested.
3) Some parts of the back end are not tested at all.
4) Chances of having unidentified paths during this testing
5) Chances of having repetition of tests that are already done by programmer

C. GREY BOX TESTING

The Gray box Testing Methodology is a software testing method used to test software applications. The methodology is platform and language independent. The current implementation of the Gray box methodology is heavily dependent on the use of a host platform debugger to execute and validate the software under test. Recent studies have confirmed that the Gray box method can be applied in real time using software executing on the target platform. Grey box testing techniques combined the testing methodology of white box and black box. Grey box testing technique is used for testing a piece of software against its specifications but using some knowledge of its internal working as well. The understanding of internals of the program in grey box testing is more than black box testing, but less than clear box testing. [3] The Gray box methodology is a ten step process for testing computer software. Ten Step Gray box Methodology.

1) Identify Inputs
2) Identify Outputs
3) Identify Major Paths
4) Identify Sub function SF X
5) Develop Inputs for SF X
6) Develop Outputs for SF X
7) Execute Test Case for SF X
8) Verify Correct Result for SF X
9) Repeat Steps 4:8 for other SF
10) Repeat Steps 7&8 for Regression

The Gray box methodology utilizes automated software testing tools to facilitate the generation of test unique software. Module drivers and stubs are created by the toolset to relieve the software test engineer from having to manually generate this code. The toolset also verifies code coverage by instrumenting the test code. "Instrumentation tools help with the insertion of instrumentation code without incurring the bugs that would occur from manual instrumentation". By operating in a debugger or target emulator, the Gray box toolset controlled the operation of the test software. The Gray box methodology has moved out of a debugger into the real world and into real-time. The methodology can be applied in real-time by modifying the basic premise that inputs can be sent to the test software via normal system messages and outputs are then verified using the system output messages

4.2.2. PERFORMANCE TESTING

Performance Testing involve all the phases as the mainstream testing life cycle as an independent discipline which involve strategy such as plan, design, execution, analysis and reporting. [3] Not all software has specification on performance explicitly. But every system will have implicit performance requirements. Performance has always been a great

IJSER

concern and driving force of computer evolution. The goals of performance testing can be performance bottleneck identification, performance comparison and evaluation. By performance testing we can measure the characteristics of performance of any applications. One of the most important objectives of performance testing is to maintain a low latency of a website, high throughput and low utilization. [3]

PERFORMENCE TESTING HAS TWO FORMS:-

1. LOAD TESTING:

Load testing is the process of subjecting a computer, peripheral, server, network or application to a work level approaching the limits of its specifications. Load testing can be done under controlled lab conditions to compare the capabilities of different systems or to accurately measure the capabilities of a single system. In this we can check whether the software can handle the load of many users or not.

2. STRESS TESTING:

Stress testing is a testing, which is conducted to evaluate a system or component at or beyond the limits of its specified requirements to determine the load under which it fails and how. [3]

4.2.3. RELIABILITY TESTING

The purpose of reliability testing is to discover potential problems with the design as early as possible and, ultimately, provide confidence that the system meets its reliability requirements. Reliability testing is related to many aspects of software in which testing process is included; this testing process is an effective sampling method to measure software reliability. In system after software is developed reliability testing techniques like analyze or fix techniques can be carried out to check whether to use the software.

4.2.4. SECURITY TESTING

Software quality, reliability and security are tightly coupled. Flaws in software can be exploited by intruders to opens security holes. Security testing makes sure that only the authorized personnel can access the program and only the authorized personnel can access the functions available to their security level. The security testing is performed to check whether there is any information leakage in the sense by encrypting the application or using wide range of software's and hardware's and firewall etc.

V. SOFTWARE TESTING STRATEGIES

A strategy for software Testing integrates software test case design methods into a well planned Series of steps that result in successful Construction of software that result in successful construction of software. Software testing Strategies gives the road map for testing. A software testing Strategy should be flexible enough to promote a customized testing approach at same time it must be right enough. Strategy is generally developed by project managers, software engineer and testing specialist. There are four different software testing strategies.

5.1. UNIT TESTING:

Unit is the smallest module i.e. smallest collection of lines of code which can be tested. Unit testing is just one of the levels of testing which go together to make the big picture of testing a system. IT complements integration and system level testing. It should also complement code reviews and walkthroughs. Unit testing is generally seen as a white box test class. That is it is biased to looking at and evaluating the code as implemented. Rather than evaluating conformance to some set of requirements.

Benefits of Unit Testing:

1) Unit level testing is very cost effective.
2) It provides a much greater reliability improvement for resources expanded than system level testing. In particular, it tends to reveal bugs which are otherwise insidious and are often catastrophic like the strange system crashes that occur in the field when something unusual happens.
3) Be able to test parts of a project without waiting for the other parts to be available,
4) Achieve parallelism in testing by being able to test and fix problems simultaneously by many engineers,
5) Be able to detect and remove defects at a much less cost compared to other later stages of testing,
6) Be able to take advantage of a number of formal testing techniques available for unit testing, 7)

IJSER

Simplify debugging by limiting to a small unit the possible code areas in which to search for bugs,
8) Be able to test internal conditions that are not easily reached by external inputs in the larger integrated systems
9) Be able to achieve a high level of structural coverage of the code,
10) Avoid lengthy compile-build-debug cycles when debugging difficult problems.

Unit Testing Techniques:

A number of effective testing techniques are usable in unit testing stage. The testing techniques may be broadly divided into three types:

1. Functional Testing
2. Structural Testing
3. Heuristic or Intuitive Testing

5.2. INTEGRATION TESTING:

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested components and build a program structure that has been dictated by design.

IJSER

Different Integration testing Strategies are discussed below:-

1. Top down Integration testing
2. Bottom up Integration testing

5.2.1. Top down Testing:

Top-down integration testing is an incremental approach to construct program structure. Modules are integrated by moving downward through the structure, beginning with the main control module. Modules subordinate to the main control module are incorporated into the structure in either a depth-first or breadth-first manner. [4]

The integration process is performed in a series of five steps:

1. The main control module is used as a test driver and stubs are substituted for all components directly subordinate to the main control module.
2. Depending on the integration approach selected subordinate stubs are replaced one at a time with actual components.
3. Tests are conducted as each component is integrated.
4. On completion of each set of tests, another stub is replaced with the real component.
5. Regression testing may be conducted to ensure that new errors have not been introduced.

It is not as relatively simple as it looks. In this logistic problem can arise. Problem arises when testing low level module which requires testing upper level. Stub replace low level module at the beginning of top down testing. So no data can flow in upward direction.

5.2.2. Bottom up Testing:

Bottom-up integration testing, as its name implies, begins construction and testing with atomic modules. Because components are integrated from the bottom up, processing required for components subordinate to a given level is always available and the need for stubs is eliminated. [4]

A bottom-up integration strategy may be implemented with the following steps:

1. Low-level components are combined into clusters that perform a specific software sub function.
2. A driver is written to coordinate test case input and output.
3. The cluster is tested.
4. Drivers are removed and clusters are combined moving upward in the program structure.

5.3. ACCEPTENCE TESTING:

Acceptance testing (also known as user acceptance testing) is a type of testing carried out in order to verify if the product is developed as per the standards and specified criteria and meets all the requirements specified by customer. [4] This type of testing is generally carried out by a user/customer where the product is developed externally by another party.

Acceptance testing falls under black box testing methodology where the user is not very much interested in internal working/coding of the system, but evaluates the overall functioning of the system

and compares it with the requirements specified by them. User acceptance testing is considered to be one of the most important testing by user before the system is finally delivered or handed over to the end user.
Acceptance testing is also known as validation testing, final testing, QA testing, factory acceptance testing and application testing etc. And in software engineering, acceptance testing may be carried out at two different levels; one at the system provider level and another at the end user level.

Types of Acceptance Testing:

5.3.1. User Acceptance Testing:

User acceptance testing in software engineering is considered to be an essential step before the system is finally accepted by the end user. In general terms, user acceptance testing is a process of testing the system before it is finally accepted by user.

5.3.2. Alpha Testing & Beta Testing:

Alpha testing is a type of acceptance testing carried out at developer’s site by users.[4] In this type of testing, the user goes on testing the system and the outcome is noted and observed by the developer simultaneously.

Beta testing is a type of testing done at user’s site. The users provide their feedback to the developer for the outcome of testing. This type of testing is also known as field testing. Feedback from users is used to improve the system/product before it is released to other users/customers.

5.3.3 Operational Acceptence Testing:

This type of testing is also known as operational readiness/preparedness testing. It is a process of ensuring all the required components (processes and procedures) of the system are in place in order to allow user/tester to use it.

5.3.4. Contact and Regular Acceptance Testing:

In contract and regulation acceptance testing, the system is tested against the specified criteria as mentioned in the contract document and also tested to check if it meets/obeys all the government and local authority regulations and laws and also all the basic standards.

5.4. STRESS TESTING:

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all work to verify that system elements have been properly integrated and perform allocated functions.

Some of Different types of system testing are as follows:-

1. Recovery testing
2. Security testing
3. Graphical user interface testing
4. Compatibility testing

5.4.1. Recovery testing:

Recovery Testing Recovery testing is a system test that forces the software to fail in a variety of ways and verifies that recovery is properly performed. If recovery is automatic, re-initialization, check pointing mechanisms, data recovery, and restart are evaluated for correctness. If recovery requires human intervention, the mean-time-to-repair is evaluated to determine whether it is within acceptable limits.

5.4.2. Security Testing:

Security testing attempts to verify that protection mechanisms built into a system will, in fact, protect it from improper penetration.

During security testing, the tester plays the role(s) of the individual who desires to penetrate the system. Anything goes! The tester may attempt to acquire passwords through external clerical means; may attack the system with custom software designed to breakdown any defenses that have been constructed; may overwhelm the system, thereby denying service to others; may purposely cause system errors, hoping to penetrate during recovery; may browse through insecure data, hoping to find the key to system entry.

IJSER

5.4.3. Graphical User Interface Testing:

Graphical user interface testing is the process of testing a product's graphical user interface to ensure it meets its written specifications. This is normally done through the use of a variety of test cases.

5.4.4. Compatibility Testing:

Compatibility testing, part of software non-functional tests, is testing conducted on the application to evaluate the application's compatibility with the computing environment.

VI. DISCUSSION

In this section difference between testing and debugging is shown. Software testing is a process that can be systematically planned and specified. Test case design can be conducted, a strategy can be defined, and results can be evaluated against prescribed expectations.

Debugging occurs as a consequence of successful testing. That is, when a test case uncovers an error, debugging is the process that results in the removal of the error. The purpose of debugging is to locate and fix the offending code responsible for a symptom violating a known specification. Debugging typically happens during three activities in software development, and the level of granularity of the analysis required for locating the defect differs in these three. [1]

The first is during the coding process, when the programmer translates the design into an executable code. During this process the errors made by the programmer in writing the code can lead to defects that need to be quickly detected and fixed before the code goes to the next stages of development. Most often, the developer also performs unit testing to expose any defects at the module or component level. [1]

The second place for debugging is during the later stages of testing, involving multiple components or a complete system, when unexpected behavior such as wrong return codes or abnormal program termination may be found. A certain amount of debugging of the test execution is necessary to conclude that the program under test is the cause of the unexpected behavior. [1]

VII. CONCLUSIONS

Software testing is the activity that executes software with an intention of finding errors in it. Software testing can provide an independent view of the software to allow the business to appreciate and understand the risk of software implementation. To carry out software testing in a more effective manner, this paper provides a comparative study of techniques of software testing

REFERENCES:

1. Software testing for wikipedia available at http://en.wikipedia.org/wiki/grey_box_testing#grey_box_tetsing
2. White box testing from wikipedia, the free encyclopedia.
3. Security testing-wikipedia the free encyclopedia available at http://en.wikipedia.org/wiki/security-tetsing.
4. Software testing glossary available at http://www.aptest.com/glossary.html#performance testing
5. Software testing by Jiantao Pan available at http://www.ece.cmu.edu/~roopman/des-899/sw_testing/

IJSER

Vasundhara Devi received a degree in M.Tech Computer Science and Engineering from university of JNTUH and currently working as Assistant Professor in Sreenidhi Engineering College. Her research interest includes Software Testing.