International Journal of Scientific and Technical Advancements
IJSTA
ISSN: 2454-1532

Software Testing Goals, Methods and their In-depth Analysis

Anil Gupta, Ayush Malhotra, Supriya Gupta
Department of Information Technology, MIET, Kot Bhalwal, Jammu, India

***Abstract*—**Software Testing is an important area of research in the field of software engineering because it is the deciding factor for successful project release and directly proportion to software quality and reliability. A lot of research has been already done in this field, likely to be leveraged in future as software has become integral part of each and every domain to make it faster, easier and smarter. Software testing is regarded as the most important phase of software development process, as without testing it is impossible to diagnose errors. This paper explains the phases, principles and limitations of software testing. We tried to accumulate principles after detailed analysis of phases of testing, based on available methodologies, users involved & importance of software testing. This paper describes Software testing, need for software testing and Software testing goals. Further it describes about different Software testing techniques and different software testing strategies. Finally it describes the difference between software testing and debugging.

***Keywords*—** Debugging, software testing goals, software testing principles, software testing techniques, software testing strategies

I. INTRODUCTION

Software testing refers to process of evaluating the software with intention to find out error in it. Software testing is a technique aimed at evaluating an attribute or capability of a program or product and determining that it meets its quality. Software testing is also used to test the software for other software quality factors like reliability, usability, integrity, security, capability, efficiency, portability, maintainability, compatibility etc.

For many years now we are still using the same testing techniques .some of which is crafted method rather than good engineering methods. Testing can be costly but not testing software can be even more costly. Software testing aims at achieving certain a goals and principles which are to be followed.

1.1. Need for Software testing

Software Testing is necessary because we all make mistakes. Some of those mistakes are unimportant, but some of them are expensive or dangerous. We need to check everything and anything we produce because things can always go wrong – humans make mistakes all the time.

Since we assume that our work may have mistakes, hence we all need to check our own work. However some mistakes come from bad assumptions and blind spots, so we might make the same mistakes when we check our own work as we made when we did it. So we may not notice the flaws in what we have done.

Ideally, we should get someone else to check our work because another person is more likely to spot the flaws.

There are several reasons which clearly tells us as why Software Testing is important and what are the major things that we should consider while testing of any product or application.

Software testing is very important because of the following reasons:

Software testing is really required to point out the defects and errors that were made during the development phases.

It's essential since it makes sure of the Customer's reliability and their satisfaction in the application.

It is very important to ensure the Quality of the product. Quality product delivered to the customers helps in gaining their confidence.

Testing is necessary in order to provide the facilities to the customers like the delivery of high quality product or software application which requires lower maintenance cost and hence results into more accurate, consistent and reliable results.

Testing is required for an effective performance of software application or product.

It's important to ensure that the application should not result into any failures because it can be very expensive in the future or in the later stages of the development.

It's required to stay in the business.

1.2. Goals for software testing

1. Short-term or immediate goals of software testing: - These goals are the immediate results after performing testing. These goals even may be set in the individual phases of SDLC. Some of them are completely discussed below: a) Bug discovery: The immediate goal about software testing is to find errors at any stage of software development. More the bugs discovered at early stage, better will be the success rate about software testing. b) Bug prevention: It is the consequent action of bug discovery. From the behavior and analysis of bugs discovered, everyone in the software development team gets to learn how to code so that bugs discovered should not be repeated in later stages or future projects. Though errors always cannot be prevented to zero, they can be minimized. In this sense prevention of a bug is a superior goal of testing. 2. Long-term goals of software testing: - These goals affect the product quality in the deep run, when one cycle of the SDLC is over. Some of them are completely discussed below: a) Quality: Since software is also a product, so its quality is primary from the user's point of view. Thorough testing ensures superior quality.

Anil Gupta, Ayush Malhotra, and Supriya Gupta, "Software testing goals, methods and their in-depth analysis," *International Journal of Scientific and Technical Advancements*, Volume 2, Issue 4, pp. 235-240, 2016.

Quality depends on various factors, such as correctness, integrity, efficiency, and reliability. So to achieve quality you have to achieve all the above mentioned factors of Quality. b) Customer satisfaction: From the user's perspective, the prime goal of software testing is customer satisfaction only. If we want the client and customer to be satisfied with the software product, then testing should be complete and thorough. A complete testing process achieves reliability, reliability enhances the quality, and quality in turn, increases the customer satisfaction. 3. Post-implementation goals of software testing: - These goals are become essential after the product is released. Some of them are completely discussed below: a) a) Reduced maintenance cost: The maintenance cost about any software product is not its physical cost, as effective software does not wear out. The only maintenance cost in a software product is its failure due to errors. Post- release errors are always costlier to fix, as they are difficult to detect. Thus, if testing has been done rigorously and effectively, then the chances about failure are minimized and as a result of this maintenance cost is reduced. b) Improved software testing process: A testing process for one project may not be blooming successful and there may be area for improvement. Therefore, the bug history and post-implementation results can be analyzed to find out snags in the present testing process, which can be determine in future projects. Thus, the long-term post-implementation goal is to improve the testing process for future projects.

1.*3. 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 limitations.

II. METHODS OF SOFTWARE TESTING

1) 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. White-box testing is the detailed investigation of internal logic and structure of the code. White-box testing is also called glass testing or open-box testing. In order to perform white-box testing on an application, a tester needs to know the internal workings of the code.

The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

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) As the tester has knowledge of the source code, it becomes very easy to find out which type of data can help in testing the application effectively.
2) It helps in optimizing the code.
3) Extra lines of code can be removed which can bring in hidden defects.
4) Due to the tester's knowledge about the code, maximum coverage is attained during test scenario writing.

Disadvantages of white box testing:-

1) Due to the fact that a skilled tester is needed to perform white-box testing, the costs are increased.
2) Sometimes it is impossible to look into every nook and corner to find out hidden errors that may create problems, as many paths will go untested.
3) It is difficult to maintain white-box testing, as it requires specialized tools like code analyzers and debugging tools.

2) Black box testing

Black box testing is a software testing techniques in which functionality of the software under test (SUT) is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software.

This type of testing is based entirely on the software requirements and specifications.

Basically Black box testing is an integral part of „Correctness testing" but its ideas are not limited to

Anil Gupta, Ayush Malhotra, and Supriya Gupta, "Software testing goals, methods and their in-depth analysis," *International Journal of Scientific and Technical Advancements*, Volume 2, Issue 4, pp. 235-240, 2016.

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

3) Grey box testing

The Graybox Testing Methodology is a software testing method used to test software applications. The methodology is platform and language independent. The current implementation of the Graybox 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 Graybox 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]

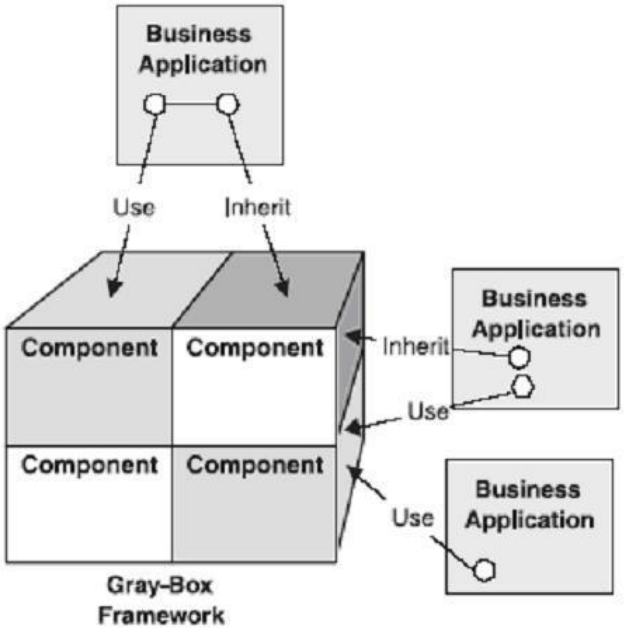
The Graybox methodology is a ten step process for testing computer software.

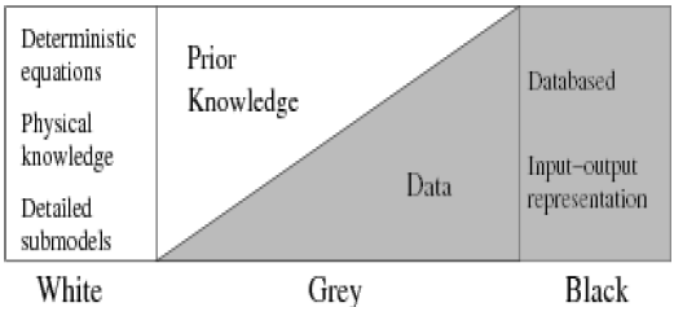
Ten Step Graybox Methodology

1) Identify Inputs
2) Identify Outputs
3) Identify Major Paths

4) Identify Subfunction (SF)X

The Graybox methodology is a ten step process for testing computer software.





5) Develop Inputs for SF X
6) Develop Outputs for SF X

2.2.2. 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 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]

Performance testing has two forms:-

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

Anil Gupta, Ayush Malhotra, and Supriya Gupta, "Software testing goals, methods and their in-depth analysis," *International Journal of Scientific and Technical Advancements*, Volume 2, Issue 4, pp. 235-240, 2016.

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 user or not.

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]

2.2.3. 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.

2.*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.

III. 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.

1) Unit testing
2) Integration testing
3) Acceptance/Validation testing
4) System testing

3.1. 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) 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

3.2. 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.

Different Integration testing Strategies are discussed below:-

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

Top down Integration

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.

Anil Gupta, Ayush Malhotra, and Supriya Gupta, “Software testing goals, methods and their in-depth analysis,” *International Journal of Scientific and Technical Advancements*, Volume 2, Issue 4, pp. 235-240, 2016.

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.

Bottom up Integration

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 subfunction.
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.

3.3. Acceptance 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

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.

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.

Operational Acceptance 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.

Contact and Regulation 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.

3.4. System 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

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.

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.

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.

Compatibility testing

Anil Gupta, Ayush Malhotra, and Supriya Gupta, “Software testing goals, methods and their in-depth analysis,” *International Journal of Scientific and Technical Advancements*, Volume 2, Issue 4, pp. 235-240, 2016.

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

IV. 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 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. 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 behaviour.

V. CONCLUSION

This paper on Software testing describes in detail about software testing, need of software testing, Software testing goals and principles. . Software testing is often less formal and rigorous than it should, 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.

We further explains different Software testing techniques such as Correctness testing, Performance testing, Reliability testing, Security testing. Further we have discussed the basic principles of black box testing, white box testing and gray box testing. We have surveyed some of the strategies supporting these paradigms, and have discussed their pros and cons. We also describes about different software testing strategies such as unit testing, Integration testing, acceptance testing and system testing.

Finally there is comparison between debugging and testing. Testing is more than just debugging .Testing is not only used to locate defects and correct them it is also used in validation, verification process and measurement.

REFERENCES

[1] Sahil Batra and Dr. Rahul Rishi,"IMPROVING QUALITY USING TESTING STRATEGIES," Journal of Gobal Research in Computer Science, Volume 2,No.6,June 2011.
[2] 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.
[3] 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.
[4] Ajay Jangra, Gurbaj Singh, Jasbir Singh and Rajesh Verma,"EXPLORING TESTING STRATEGIES," International Journal of Information Technology and Knowledge Management, Volume 4, NO.1,January-June 2011.
[5] Jovanovic and Irena,"Software Testing Methods and Techniques," May 26, 2008.
[6] S. Pressman, "Software engineering".

Anil Gupta, Ayush Malhotra, and Supriya Gupta, "Software testing goals, methods and their in-depth analysis," *International Journal of Scientific and Technical Advancements*, Volume 2, Issue 4, pp. 235-240, 2016.