Software Testing Fundamentals: A Study

Jyoti Goyat
GJUS&T, Hisar

Sakshi Dhingra
GJUS&T, Hisar

Vinod Goyal
GJUS&T, Hisar

Vinay Malik
GJUS&T, Hisar

Abstract- Software provides a complete set of application development tools for building stand-alone, client-server, and Internet-enabled applications. Software testing has three main purposes: verification, validation, and defect finding. Testing software is operating the software under controlled conditions, 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. This paper discusses about the different issues related to software testing fundamentals.

I. INTRODUCTION

Software testing is more than just error detection. Testing software is operating the software under controlled conditions, to verify, to detect errors, and to validate that what has been specified is what the user actually wanted. Verification is the checking or testing of items, including software, for conformance and consistency by evaluating the results against pre specified requirements. 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. Validation looks at the system correctness-i.e. is the process of checking that what has been specified is what the user actually wanted. In other words, validation checks to see if we are building what the customer wants/needs, and verification checks to see if we are building that system correctly. Both verification and validation are necessary, but different components of any testing activity. 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.

Testing helps in Verifying and Validating if the Software is working as it is intended to be working. This involves using Static and Dynamic methodologies to Test the application. Software testing should not be confused with debugging. Debugging is the process of analyzing and locating bugs when software does not behave as expected. Although the identification of some bugs will be obvious from playing with the software, a methodical approach to software testing is a much more thorough means of identifying bugs. Debugging is therefore an activity which supports testing, but cannot replace testing.

However, no amount of testing can be guaranteed to discover all bugs. Other activities which are often associated with software testing are static analysis and dynamic analysis. Static analysis investigates the source code of software, looking for problems and gathering metrics without actually executing the code. Dynamic analysis looks at the behavior of software while it is executing, to provide information such as execution traces, timing profiles, and test coverage information.

II. SOFTWARE TESTING FUNDAMENTAL

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as yet undiscovered error. A successful test is one that uncovers an as yet undiscovered error. Minimize the risk of product failure. 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.

Software Testing Life Cycle (STLC). Software Testing is not a just a single activity. It consists of series of activities carried out methodologically to help certify software product. These activities (stages) constitute the Software Testing Life Cycle (STLC) as shown in figure 1.

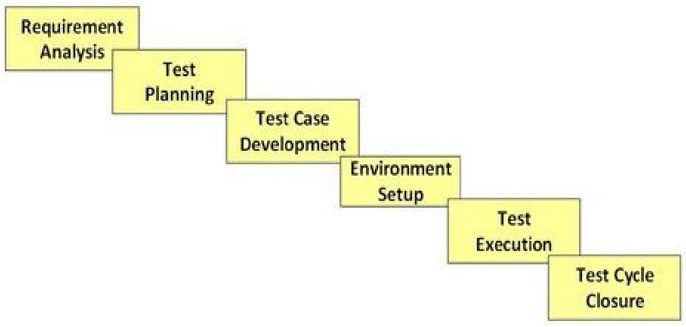

Fig. 1: STLC

Each of these stages has a definite Entry and Exit criteria, Activities & Deliverables associated with it.
Requirement Analysis
During this phase, test team studies the requirements from a testing point of view to identify the testable requirements. The QA team may interact with various stakeholders (Client, Business Analyst, Technical Leads, System Architects etc) to understand the requirements in detail. Requirements analysis, also called requirements engineering, is the process of determining user expectations for a new or modified product. These features, called requirements, must be quantifiable, relevant and detailed. In software engineering, such requirements are often called functional specifications. Requirements could be either Functional (defining what the software must do) or Non Functional (defining system performance /security availability) .Automation feasibility for the given testing project is also done in this stage.
Activities: 1. Identify types of tests to be performed. Gather details about testing priorities and focus. 2. Prepare Requirement Traceability Matrix (RTM). 3. Identify test environment details where testing is supposed to be carried out. 3. Automation feasibility analysis (if required).
Deliverables: RTM Automation feasibility report. (if applicable)

III. TEST PLANNING

This phase is also called Test Strategy phase. Typically, in this stage, a Senior QA manager will determine effort and cost estimates for the project and would prepare and finalize the Test Plan. A Software Test Plan is a document describing the testing scope and activities. It is the basis for formally testing any software/product in a project. A document describing the scope, approach, resources and schedule of intended test activities. It identifies amongst others test items, the features to be tested, the testing tasks, who will do each task, degree of tester independence, the test environment, the test design techniques and entry and exit criteria to be used, and the rationale for their choice, and any risks requiring contingency planning. It is a record of the test planning process.

A. Test Plan Types

1. Master Test
Plan: A single high-level test plan for a project/product that unifies all other test plans.
2. Testing Level Specific Test Plans: Plans for each level of testing. Unit Test Plan, Integration Test Plan, System Test Plan, Acceptance Test Plan. Testing Type Specific Test Plans**:** Plans for major types of testing like Performance Test Plan and Security Test Plan.
Activities**:** 1**.**Preparation of test plan/strategy document for various types of testing Test tool selection**.** 2. Test effort estimation. 3. Resource planning and determining roles and responsibilities. 4. Training requirement
Deliverables: Test plan /strategy document, Effort estimation document.

B. Test Case Development

This phase involves creation, verification and rework of test cases & test scripts. Test data, is identified/created and is reviewed and then reworked as well.
A set of inputs, execution preconditions, and expected outcomes developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement.

It is a detailed procedure that fully tests a feature or an aspect of a feature. Whereas the test plan describes what to test, a test case describes how to perform a particular test. We need to develop a test case for each test listed in the test plan.
Activities: 1. Create test cases, automation scripts (if applicable) 2. Review and baseline test cases and scripts 3. Create test data (If Test Environment is available)
Deliverables: Test cases/scripts, Test data

C. Test Environment Setup

Test environment decides the software and hardware conditions under which a work product is tested. Test environment set-up is one of the critical aspects of testing process and can be done in parallel with Test Case Development Stage**.** Test team may not be involved in this activity if the customer/development team provides the test environment in which case the test team is required to do a readiness check (smoke testing) of the given environment.
Activities: 1.Understand the required architecture, 2.environment set-up and prepare hardware and software requirement list for the Test Environment 3. Setup test Environment and test data 4. Perform smoke test on the build
Deliverables: Environment ready with test data set up, Smoke Test Results

D. Test Execution

During this phase test team will carry out the testing based on the test plans and the test cases prepared. Bugs will be reported back to the development team for correction and retesting will be performed.
Activities: 1. Execute tests as per plan 2. Document test results, and log defects for failed cases 4. Map defects to test cases in RTM 4. Retest the defect fixes 5. Track the defects to closure.
Deliverables: Completed RTM with execution status, Test cases updated with results, Defect reports

E. Test Cycle Closure

Testing team will meet, discuss and analyze testing artifacts to identify strategies that have to be implemented in future, taking lessons from the current test cycle. The idea is to remove the process bottlenecks for future test cycles and share best practices for any similar projects in future.
Activities: 1.Evaluate cycle completion criteria based on Time, Test coverage, Cost, Software, Critical Business Objectives, Quality 2. Prepare test metrics based on the above parameters. 3. Document the learning out of the project 4. Prepare Test closure report 5.Qualitative and quantitative reporting of quality of the work product to the customer. 6. Test result analysis to find out the defect distribution by type and severity.
Deliverables
• Test Closure report
• Test metrics

IV. TYPES OF TESTING

Black box testing-Internal system design is not considered in this type of testing as shown in figure 2.

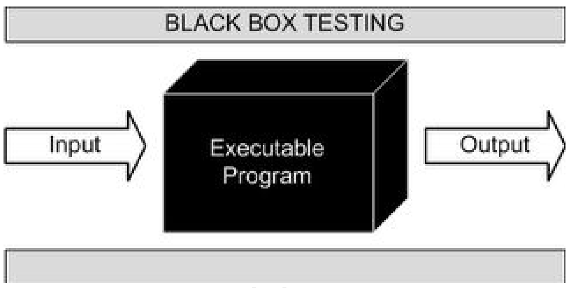

Figure2: Black Box Testing

Tests are based on requirements and functionality. Black box testing takes an external perspective of the test object to derive test cases. These tests can be functional or non-functional, though usually functional. The test designer selects valid and invalid input and determines the correct output. There is no knowledge of the test object's internal structure.
White box testing- This testing is based on knowledge of the internal logic of an application's code. Also known as Glass box Testing. Internal software and code working should be known for this type of testing. Tests are based on coverage of code statements, branches, paths, conditions. White Box Testing also known as Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing or Structural Testing.

A. Unit Testing

Testing of software components or modules typically done by the programmer not by testers, as it requires detailed knowledge of the internal program design and code which may require developing test driver modules. Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. Unit testing is often automated but it can also be done manually.

B. Integration Testing

Testing of integrated modules to verify combined functionality after integration. Modules are typically code modules, individual applications, client and server applications on a network, etc. This type of testing is especially relevant to client/server and distributed systems. Testing is that in which modules are combined and tested as a group. Modules are typically code modules, individual applications, client and server applications on a network, etc. Integration Testing follows unit testing and precedes system testing.

C. System Testing

Entire system is tested as per the requirements. Black-box type testing that is based on overall requirements specifications, covers all combined parts of a system. The purpose of this test is to evaluate the system's compliance with the specified requirements.

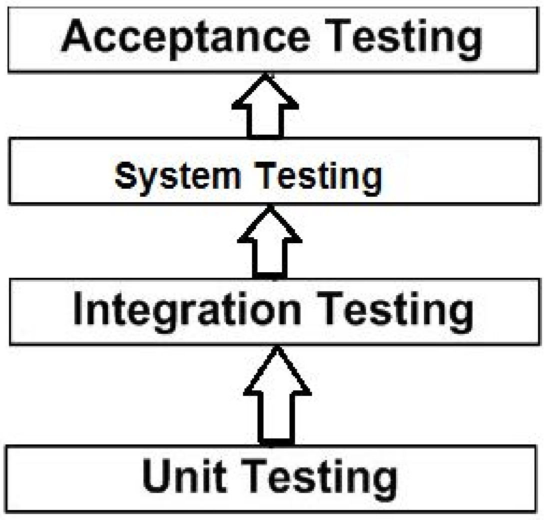

Fig. 3: Hierarchy of Testing

1) Regression testing

Which is selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still complies with its specified requirements (IEEE, 1990). As with integration testing, regression testing can be done via black-box test cases, white-box test cases, or a combination of the two. White-box unit and integration test cases can be saved and rerun as part of regression testing.

2) Acceptance testing

Testing to verify a product meets customer specified requirements. A customer usually does this type of testing on a product that is developed externally. Acceptance testing is a test conducted to determine if the requirements of a specification or contract are met.

3) Beta testing

Testing when development and testing are essentially completed and final bugs, problems need to be found before the final release. Beta Testing is typically done by end-users or others, not by programmers or testers.

4) Functional testing

Validating an application or Web site conforms to its specifications and correctly performs all its required functions. This entails a series of tests which perform a feature by feature validation of behavior, using a wide range of normal and erroneous input data. This can involve testing of the product's user interface, APIs, database management, security, installation, networking, etc. testing can be performed on an automated or manual basis using black box or white box methodologies.

5) Stress testing

System is stressed beyond its specifications to check how and when it fails. Doing perform under heavy load like putting large number beyond storage capacity, complex
database queries, continuous input to system or database load. Testing 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. A graceful degradation under load leading to non-catastrophic failure is the desired result. Often Stress Testing is performed using the same process as Performance Testing but employing a very high level of simulated load. Mutation Testing: Mutation testing is a method for determining if a set of test data or test cases is useful, by deliberately introducing various code changes ('bugs') and retesting with the original test data/cases to determine if the 'bugs' are detected. Proper implementation requires large computational resources Sanity testing: Typically an initial testing effort to determine if a new software version is performing well enough to accept it for a major testing effort. For example, if the new software is crashing systems every 5 minutes, bogging down systems to a crawl, or destroying databases, the software may not be in a 'sane' enough condition to warrant further testing in its current state.

V. CONCLUSION

Software testing is a critical element in the software development life cycle and has the potential to save time and money by identifying problems early and to improve customer satisfaction by delivering a more defect-free product. Unfortunately, it is often less formal and rigorous than it should, and a primary reason for that is because the project staff is unfamiliar with software testing methodologies, approach has, and tools.
Presently we are unaware about the relative ordering of software testing techniques and if we are to make software testing more effective by selecting effective testing techniques then we need to place existing software testing techniques at least on an ordinal scale. To do so we need to carry out experimentation on large scale but that needs to in a way that can be compared and will have no contradictions. For that we also need to establish common and standard parameters so that there are little variations in experimentation goals. However the actual research settings of creating reasonable comparative models have not been totally explored.

VI.REFERENCES

[1] B. Beizer, *Software Testing Techniques*. London: International Thompson Computer Press, 1990.
[2] B. Beizer, *Black Box Testing*. New York: John Wiley & Sons, Inc., 1995.
[3] A. Bertolino, "Chapter 5: Software Testing," in *IEEE SWEBOK Trial*
[4] *Version 1.00* May 2001.
[5] B. W. Boehm, *Software Engineering Economics*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1981.
[6] L. Copeland, *A Practitioner's Guide to Software Test Design*. Boston: Artech ,House Publishers, 2004.
[7] R. D. Craig and S. P. Jaskiel, *Systematic Software Testing*. Norwood, MA: Artech, House Publishers, 2002.
[8] E. W. Dijkstra, "Notes on Structured Programming," Technological
[9] University, Eindhoven T.H. Report 70-WSK-03, Second edition, April, 1970.
[10] D. Galin, *Software Quality Assurance*. Harlow, England: Pearson, Addison,Wesley, 2004.
[11] Talby, D., Keren, A., Hazzan, O., and Dubinsky, Y. "Agile software
[12] testing in a large-scale project,"*IEEE Software* (23:4), 2006,
[13] Hutcheson, M. L. *Software testing fundamentals: Methods and metrics*, Wiley Publishing Inc., Indianapolis, Indiana, 2003.
[14] Issac, G., Rajendran, C., and Anantharaman, R. N."Determinants of software quality: Customer's perspective," *TQM & Business Excellence* (14:9),2003.