Effective Test Case Design: A Review

Ved Parkash
Research Scholar
Department of Computer Science Engineering
Kurukshetra University, Kurukshetra (Haryana)

Amit Kumar
Research Scholar
Department of Computer Science Engineering
Kurukshetra University, Kurukshetra (Haryana)

Abstract

This paper describes the different techniques of testing the software. This paper explicitly addresses the idea for testability and the important thing is that the testing itself—not just by saying that testability is a desirable goal, but by showing how to do it. Software testing is the process we used to measure the quality of developed software. Software Testing is not just about error-finding and their solution but also about checking the client requirements and testing that those requirements are met by the software solution. It is the most important functional phase in the Software Development Life Cycle (SDLC) as it exhibits all mistakes, flaws and errors in the developed software. Without finding these errors, technically termed as ‘bugs,’ software development is not considered to be complete. Hence, software testing becomes an important parameter for assuring quality of the software product. We discuss here about when to start and when to stop the testing of software. How errors or Bugs are formed and rectified. How software testing is done i.e. with the help of Team Work.

Keywords: SDLC, Concepts for Application Test Management, ISO 9126

I. INTRODUCTION

Testing is a process used to identify the correctness, completeness and quality of developed computer software. With in mind that, testing can never completely establish the correctness of Computer software. There are many approaches for software testing, but effective testing on complex products is essentially a process of investigation, not merely a matter of creating and following rote procedure. The definition of testing is “the process of questioning a product in order to evaluate it”, where the “questions” are things the checker tries to do with the product, and the product answers with its behaviour in reaction to the probing of the tester. Although most of the intellectual processes of testing are nearly identical to that of review or inspection, the word testing is not to mean the dynamic analysis of the product—putting the product through its paces. The quality of the application can and normally does vary widely from system to system but some of the common quality attributes include reliability, stability, portability, maintainability and usability. Testing helps in verifying and validating if the Software is working as it is intended to be working. Things involve using Static and Dynamic methodologies to Test the application. Because of the fallibility of its human designers and its own abstract, complex nature, software development must be accompanied by quality assurance activities. It is not unusual for developers to spend 40% of the total project time on testing. For life-critical software (e.g. flight control, reactor monitoring), testing can cost 3 to 5 times as much as all other activities combined. The destructive nature of testing requires that the developer discard preconceived notions of the correctness of his/her developed software. The importance of software testing and its impact on software cannot be underestimated. Software testing is a fundamental component of software quality assurance and represents a review of specification, design and coding.

A. Software Testing Fundamentals:

Testing is the one step in the software process that can be seen by the developer as destructive instead of constructive. Software engineers are typically constructive people and testing requires them to overcome the concepts of correctness and deal with conflicts when errors are identified.

B. Testing Objectives Include:

1) It is a process to finding the intent of Error.
2) A test case is called to be a good when it discovered the undiscovered error.
3) A test is successful when all errors are uncovered as yet undiscovered error.

Testing should systematically uncover different classes of errors in a minimum amount of time and with a minimum amount of effort. One more 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.

C. When Testing should be started?

Early Testing in life cycle can reduce the error. Test cases are associated with every phase of development. The goal of Software Tester is to find bugs, find them as early as possible, and make them sure they are fixed. The number one cause of Software bugs is the Specification. There are several reasons specifications are the largest bug producer.
In many point or instance the specification isn’t written. Other reasons may be that the spec isn’t thorough enough, it’s constantly changing, or it’s not communicated well to the entire team. Planning software is vitally important. If it’s not done correctly bugs will be created.

The next largest source of bugs is the Design, That’s where the programmers lay the plan for their Software. Compare it to an architect creating the blue print for the building. Bugs occur here for the same reason they occur in the specification. It’s rushed, changed, or not well communicated.

Coding errors may be more familiar to you if you are a programmer. Typically these can be traced to the Software complexity, poor documentation, schedule pressure or just plain dump mistakes. It’s important to note that many bugs that appear on the surface to be programming errors can really be traced to specification. The other category is the catch-all for what is left. Some bugs can blamed for false positives, conditions that were thought to be bugs but really weren’t. There may be duplicate bugs, multiple ones that resulted from the square root cause. Some bugs can be traced to Testing errors.

D. When should we Stop Testing?
This can be difficult to determine. Many modern software applications are so complex, and run in such as interdependent environment, that complete testing can never be done. "When to stop testing” is one of the most difficult questions to a test engineer. Common factors in deciding when to stop are:

1) Deadlines (Set the deadline for testing)
2) Test cases completed with certain percentages passed
3) Test budget declared
4) Coverage of code/functionality/requirements reaches a specified point
5) The rate at which Bugs can be found is too small
6) Beta or Alpha Testing period ends
7) The risk in the project is under acceptable limit.

Practically, we feel that the decision of stopping testing is based on the level of the risk acceptable to the management. As testing is a never ending process we can never assume that 100 % testing has been done, we can only minimize the risk of shipping the product to client with X testing done. The risk can be measured by Risk analysis but for small duration / low budget / low resources project, risk can be deduced by simply: -

1) Measuring Test Coverage.
2) Number of test cycles.
3) Number of high priority bugs.

E. Concepts for Application Test Management:
Testing should be pro-active following the V-model. Test execution can be a manual process, Test execution can be an automated process. It is possible to plan the start date for testing. It is not possible to accurately plan the end date of testing. Ending testing is through risk assessment. A fool with a tool is still a fool Testing is not a diagnosis process. Testing is a triage process. Testing is expensive but Not testing, can be more expensive.

F. How Software Defects Arise?
The International Software Testing Qualifications Board says that software faults occur through the following process:
A human being can make an error (mistake), which produces a defect (fault, bug) in the code, in software or a system, or in a document. If a defect in code is executed, the system will fail to do what it should do (or do something it shouldn’t), causing a failure.

Defects in software, systems or documents may result in failures, but not all defects do so.
A fault can also turn into a failure when the environment is changed. Examples of these changes in environment include the software being run on a new hardware platform, alterations in source data or interacting with different software.

G. Inability to find all faults:
A problem with software testing is that testing all combinations of inputs and preconditions is not feasible when testing anything other than a simple product. This means that the number of defects in a software product can be very large and defects that occur infrequently are difficult to find in testing.

H. When Testing is Carried Out?
A common practice of software testing is that it is performed by an independent group of testers after the functionality is developed but before it is shipped to the customer. This practice often results in the testing phase being used as project buffer to compensate for project delays, thereby compromising the time devoted to testing. Another practice is to start software testing at the same moment the project starts and it is a continuous process until the project finishes.

Another common practice is for test suites to be developed during technical support escalation procedures,[documentation ] Such tests are then maintained in regression testing suites to ensure that future updates to the software don't repeat any of the known mistakes.
II. MEASURING SOFTWARE TESTING

Usually, quality is constrained to such topics as correctness, completeness, security, but can also include more technical requirements as described under the ISO standard ISO 9126, such as capability, reliability, efficiency, portability, maintainability, compatibility, and usability.

Testing is a process of technical investigation, performed on behalf of stakeholders, that is intended to reveal quality-related information about the product with respect to the context in which it is intended to operate. This includes, but is not limited to, the process of executing a program or application with the intent of finding errors. Quality is not an absolute; it is value to some person. With that in mind, testing can never completely establish the correctness of arbitrary computer software; testing furnishes a criticism or comparison that compares the state and behaviour of the product against a specification. An important point is that software testing should be different from the separate discipline of Software Quality Assurance (SQA), which encompasses all business process areas, not just testing. Today, software has grown in complexity and size. The software product developed by a developer is according to the System Requirement Specification (SRS). Every software product has a target audience. For example, video game software has its audience completely different from banking software. Therefore, when an organization invests large sums in making a software product, it must ensure that the software product must be acceptable to the end users or its target audience. This is where Software Testing comes into play. Software testing is not merely finding defects or bugs in the software; it is the completely dedicated discipline of evaluating the quality of the software.

- There are 4 Testing Steps:

A. Select What Has To Be Measured:
Code tested for correctness with respect to:
   1) Requirements
   2) Architecture
   3) Detailed Design

B. Decide How The Testing Is Done For Each Level Of Testing:
   1) Code inspection
   2) Black-box, white box, grey box.
   3) Select integration testing strategy (big bang, bottom up, top down, sandwich)

C. Develop Test Cases:
A test case is a set of test data or situations that will be used to exercise the unit (code, module, system) being tested or about the attribute being measured.

D. Create The Test Oracle:
An oracle contains of the predicted results for a set of test cases i.e., expected output for each test. The test oracle has to be written down before the actual testing takes place. This is the difficult step White box, black box, and grey box testing:

White box and black box testing are terms used to describe the point of view that a test engineer takes when designing test cases.

Black box testing treats the software as a black-box without any understanding as to how the internals behave. It aims to test the functionality according to the requirements. Thus, the tester inputs data and only sees the output from the test object. This level of testing usually requires thorough test cases to be provided to the tester who then can simply verify that for a given input, the output value (or behavior), is the same as the expected value specified in the test case.

White box testing, however, is when the tester has access to the internal data structures, code, and algorithms. For this reason, unit testing and debugging can be classified as white-box testing and it usually requires writing code, or at a minimum, stepping through it, and thus requires more knowledge of the product than the black-box tester. If the software in test is an interface or API of any sort, white-box testing is almost always required. In recent years the term grey box testing has come into common usage. This involves having access to internal data structures and algorithms for purposes of designing the test cases, but testing at the user, or black-box level. Manipulating input data and formatting output do not qualify as grey-box because the input and output are clearly outside of the black-box we are calling the software under test. This is particularly important when conducting integration testing between two modules of code written by two different developers, where only the interfaces are exposed for test.

Grey box testing could be used in the context of testing a client-server environment when the tester has control over the input, inspects the value in a SQL database, and the output value, and then compares all three (the input, SQL value, and output), to determine if the data got corrupt on the database insertion or retrieval.
III. SOFTWARE TESTING LIFE CYCLE

A. Requirements Analysis:
Testing should begin in the requirements phase of the software development life cycle. During the design phase, testers work with developers in determining what aspects of a design are testable and with what parameters those tests work.

B. Test Planning:
Test Strategy, Test Plans, Test Bed creation.
A lot of activities will be carried out during testing, so that a plan is needed.

C. Test Development:
Test Procedures, Test Scenarios, Test Cases, and Test Scripts to use in testing software.

D. Test Execution:
Testers execute the software based on the plans and tests and report any errors found to the development team.

E. Test Reporting:
Once testing is completed, testers generate metrics and make final reports on their test effort and whether or not the software tested is ready for release.
Retesting the Defects. Not all errors or defects reported must be fixed by a software development team. Some may be caused by errors in configuring the test software to match the development or production environment. Some defects can be handled by a workaround in the production environment. Others might be deferred to future releases of the software, or the deficiency might be accepted by the business user. There are yet other defects that may be rejected by the development team (of course, with due reason) if they deem it.

F. What should the Test Team do?
1) Programmer Management
2) Strong Change Management
3) Strict Configuration Control
4) Pro Active Scope Creep Management
5) Inclusion in the decision making process

G. What are the Test Team Deliverables?
1) Test Plans
2) Test Script Planner
3) Test Scripts
4) Test Execution Results
5) Defect Reports

IV. CONCLUSION

Software testing accounts for a large percentage of effort in the software development process, but we have only recently begun to understand the subtleties of systematic planning, execution and control. For an IT organization, developing a software system that meets the business needs of clients is always a challenge. The company needs to ensure that the software system that gets delivered to their clients is free from bugs or defects and achieves the demands as per client requirements. But this can only be ensured by following rigorous software testing and quality assurance procedures.

Software testing is a process without which the Software Development Life Cycle (SDLC) stands incomplete. It is the process that identifies the correctness, completeness and quality of the software developed during the SDLC process. Software bugs and improperly tested codes cost millions in damages and millions more in time and money to fix the defect. Organizations try to develop software applications that should act in a way that cause the least amount of surprises to the user. In short they should be bug free. Due to this, the software testing field has emerged from the shadows in the world IT space and has claimed its rightful place in the IT market. Gone are the days when software testing was considered a poor cousin of software development. In this article, we talk about software testing techniques, trends that are coming up in this arena and also new software development paradigms.

The ways in which testing can be done are broadly classified as Manual Testing and Automated Testing.

- The manual testing of the software happens in several phases. Self-testing, which is done by developers themselves or by small development teams, should be restricted to build cycle itself and should be done while the software development is in the production stage. The errors of the Bugs can be Corrected or Verified easily by the Performance
of the Team. Hence there should be a good understanding between the Team Members by which we get the Software Tested Successfully. It encourages team and organisational learning.

- Team Work focuses team efforts towards the respective goals. (i.e., intent of finding the errors)
- Team Work increases motivation and accountability of individual employees by which knowledge will be shared among themselves.
- Team Work encourages continuous improvement.
- Team Work provides adequate feedback thus, allowing situational awareness, capability assessment, problem diagnosis, intervention and remediation.

REFERENCES

[2] Nancy Bordelon, A comparison of automated software testing tools
[4] Software testing, by Ron Patton