Software Testing Techniques for Faults/Errors Detection

Mamta Mor
Department of CSE, OITM, GJUS&T, Hisar
mamtamor12121990@gmail.com

Abstract

Software testing is one of the important phases in Software Development Life Cycle. It provides several means/methods to reduce errors, cut maintenance and overall software costs. It is a process which focuses on evaluating an attribute or capability of a program and ensures that it meets the required result. One of the major issues within software testing phase is how to get a suitable set of test cases with maximum effectiveness to test a software system. This paper presents several testing techniques for error detection.

1. Introduction

A Software testing consists of set of steps aimed at finding errors in software. It is an important means of assessing the software to determine its quality. An effective software testing technique always contributes in delivering higher quality software, more satisfied user, lower maintenance costs, and more accurate and reliable results. It is one the significant phases in the software development life cycle, it typically consumes one-third to one-half of the development efforts, and consumes more effort for systems that require higher levels of reliability. Modern software programs must be extremely reliable and correct. Software testing is not a “silver bullet” that can guarantee the production of high quality software systems. It is almost impossible to test a software application thoroughly because [1]:

1) Verification is the checking or testing of items, including software, for conformance and consistency by evaluating the results against pre-specified requirements. [Verification: Are we building the system right?]

2) 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.

3) Validation looks at the system correctness – i.e. is the process of checking that what has been specified is what the user actually wanted.

2. SOFTWARE TESTING OBJECTIVES

All The overall aim of testing is to find errors and fix them to improve quality, reliability of software. Software testing typically represents 40% of a software development budget [4].

There are four main objectives of software testing:

1) Demonstration: It demonstrates functions under special conditions and shows that products are ready for integration or use.

2) Error Detection: It discovers defects, errors and deficiencies. It determines system capabilities and limitations, quality of components, work products and the system.

3) Error Prevention: It provides information to prevent or reduce the number of errors clarify system specifications and performance. Identify ways to avoid risk and problems in the future.

4) Improving Quality: By doing effective testing, we can minimize errors and hence improve the quality of software.

Testing involves the configuration of proper inputs, execution of the software over the input, and the analysis of the output. “Test Configuration” includes a) test cases, b) test plan and procedures, and c) testing tools. The information flow of testing is shown in Figure 1.
From above points it can be concluded that the main purpose of testing can be quality assurance, reliability estimation, validation or verification. The other objectives software testing includes.[2][3].

• The better it works the more efficiently it can be tested.
• Better the software can be controlled more the testing can be automated and optimized.
• The fewer the changes, the fewer the disruption to testing.
• A successful test is the one that uncovers an undiscovered error.
• Testing is a process to identify the correctness and completeness of the software.
• The general objective of software testing is to affirm the quality of software system by systematically exercising the software in carefully controlled circumstances.

3. Software testing life-cycle phases
Software Testing is not a just a single task. It consists of series of tasks carried out to make the software product more reliable. All these tasks/activities collectively form the STLC. The various activities (stages) constituting the Software Testing Life Cycle (STLC) which are shown in figure 2. The various phases of STLC are discussed in brief below [5][6]:

3.1 Requirements study
a. Testing Cycle starts with the study of client’s requirements.
b. Understanding of the requirements is very essential for testing the product.

3.2 Test Case Design and Development
a. Component Identification
b. Test Specification Design
c. Test Specification Review

3.3 Test Execution
a. Code Review
b. Test execution and evaluation
c. Performance and simulation

3.4 Test Closure
a. Test summary report
b. Project De-brief
c. Project Documentation

3.5 Test Process Analysis
a. Analysis done on the reports and improving the application’s performance by implementing new technology and additional features.

4. Software Testing Techniques
4.1 White Box testing
This testing technique is based on knowledge of the internal logic of an application’s code (Figure 3). Tests are based on coverage of code statements, branches, paths, conditions [7]. It is the process of forwarding input to the system and analyzing how the system processes that input to generate the required output. It is necessary for a tester to have the full knowledge of the source code. White box testing is applicable at integration, unit and system levels of the software testing process.
It is further divided into two types:

A. **Static white box testing**: It has 3 types[8]
   a) *Desk Checking*: It is the primary testing done on code. Testers with great knowledge of the programming language will be involved in the testing.
   b) *Code Walk Through*: In this testing process a group of technical people go through the code. This is one type of semi-formal review technique.
   c) *Formal Inspections*: Inspection is a formal, efficient and economical method of finding errors in design and code. It includes a formal review and aimed at detecting all faults, violations and other side effects.

B. **Structural white box testing**: It has 4 types
   a) *Control flow testing*: It is a structural testing strategy that uses the program control flow as a model control flow and favors more but simpler paths over fewer but complicated path.
   b) *Basis path testing*: It allows the test case designer to produce a logical complexity measure of procedural design and then uses this measure as an approach for outlining a basic set of execution paths.
   c) *Data Flow testing*: In this type of testing the control flow graph is annotated with the information about how the program variables are defined and used.
   d) *Loop Testing*: It exclusively focuses on the validity of loop construct.
      . It makes sure that input is properly accepted and output is correctly produced.

Some synonyms of white box testing are:
   a) Logic Driven Testing, b) Design Based Testing,
   c) Open Box Testing, d) Transparent Box Testing,
   e) Clear Box Testing f) Glass Box, g) Testing Structural Testing

**Advantages of white box testing:**
   a) Errors in hidden codes are revealed.
   b) Approximate the partitioning done by execution equivalence.
   c) Developer carefully gives reason about implementation.

**Disadvantages of white box testing:**
   a) It is very expensive.
   b) Missed out the cases omitted in the code.
   c) A skilled tester is needed to carry out this testing because knowledge of internal structure is required.

4.2 **Black Box testing**

In black box testing the code is tested without any reference to its internal working. So it is based on the requirements specifications and there is no need of examining the code in black box testing. 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 [9]. It makes sure that input is properly accepted and output is correctly produced.
It is further divided into six types: [9]

a) **Equivalence Partitioning:** It divides the input domain of a software program into equivalence classes from which test cases can be derived, so reducing the number of test cases.

b) **Boundary Value Analysis:** It focuses on testing at the boundaries or periphery. It includes minimum, maximum, just inside/outside boundaries, error values and typical values.

c) **Fuzzing:** It feeds random input to software application. It is used for finding implementation bugs, using malformed/semi-malformed data injection in an automated or semi-automated session.

d) **Cause-Effect Graph:** A graph is created in this technique and then relations are established between effect and its causes.

e) **Orthogonal Array Testing:** It is used where the input domain is very small, but too large to accommodate exhaustive testing.

f) **All Pair Testing:** Test cases are designed to execute all possible discrete combinations of each pair of input parameters. Its main objective is to have a set of test cases that covers all the pairs.

**Advantages of Black box testing:**

a) Black box tester has no “bond” with the code which means he need not to have great knowledge of programming language.

b) Tester perception is very simple.

c) Programmer and tester both are independent of each other.

d) More effective on larger units of code than clear box testing.

e) Testing is done from user’s point of view.

f) It helps to expose any ambiguities or inconsistencies in the requirement specifications.

**Disadvantages of Black box testing:**

a) Test cases are hard to design without clear specifications.

b) Only small numbers of possible input can actually be tested.

c) Some parts of the back end are not tested at all.

**4.3 Gray Box testing**

Grey box testing techniques combines the techniques of white box and black box. Grey box testing technique is used for testing a code against its fundamental specifications but using some knowledge of its internal working as well [9] [10]. So the understanding of internals of the program in grey box testing is more than black box testing, but less than clear box testing. Gray-box testing is well suited for web applications and functional or business domain testing. It has four types:

I. **Orthogonal Array Testing:** It is used as subset of all possible combinations.

II. **Matrix Testing:** In matrix testing the status report of the project is stated.

III. **Regression Testing:** If new changes are made in software, regression testing implies running of test cases.

IV. **Pattern Testing:** Pattern testing verifies the good application for its architecture and design.

**Advantages of Gray box testing**

a) It provides combined benefits of both white-box and black-box testing

b) It is based on functional specification, UML Diagrams, Database Diagrams or architectural view.

c) Intelligent Test Authoring: Grey-box tester can design complex test scenario more intelligently

d) Unbiased Testing: It maintains a boundary for testing between tester and developer.

**Disadvantages of Gray box testing**

a) Partial code coverage: In grey-box testing, complete white box testing cannot be done due to inaccessible source code/binaries.

b) Defect Identification: It is difficult to associate defects when we perform Grey-box testing for a distributed system applications.
5. Comparison among the techniques

<table>
<thead>
<tr>
<th>ATTRIBUTES</th>
<th>WHITE BOX TESTING</th>
<th>BLACK BOX TESTING</th>
<th>GRAY BOX TESTING</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition</td>
<td>The internal structure/ design/ implementation of the software being tested are known to the tester.</td>
<td>The internal structure/ design/ implementation of the software being tested are completely unknown to the tester.</td>
<td>The internal structure/ design/ implementation of the software being tested are partially known to the tester.</td>
</tr>
<tr>
<td>Effort &amp; Time</td>
<td>Potentially most exhaustive and time consuming.</td>
<td>Least exhaustive and time consuming.</td>
<td>Somewhere lies between the two.</td>
</tr>
<tr>
<td>Levels Applicable To</td>
<td>Mainly applicable at higher levels: Acceptance testing, System testing</td>
<td>Mainly applicable at lower levels: Unit testing, Integration testing</td>
<td>Applicable on Integration Testing</td>
</tr>
<tr>
<td>Basis for Test Cases</td>
<td>Detailed Design</td>
<td>Requirements Specification</td>
<td>Both are partially required</td>
</tr>
<tr>
<td>Granularity</td>
<td>High</td>
<td>Low</td>
<td>Medium</td>
</tr>
</tbody>
</table>

Table 1 Comparison among techniques

5. Conclusion

Software testing is an important technique for the improvement and measurement of a software system quality. It can detect faults in a software system but it cannot prove that there are no remaining faults. It is not possible to find out all the errors in a code using any of the techniques above, but the error detection can be maximized by using appropriate technique. This paper has presented an analysis on the testing techniques to carry out software testing in a more effective way.

6. References