International Journal of Research (IJR)
P-ISSN: 2348-6848
E-ISSN: 2348-795X
Vol. 8 Issue 9
September 2021

Software Testing Methods, Tools and Techniques

Dr. Mithelesh Parihar

Associate Professor

Department of Computer Science

Sun Rise University Alwar (Rajasthan)

Email: manju14parihar@gmail.com

Abstract:

In this progress report main testing methods and techniques are shortly described. The purpose of this paper is to offer visibility in to the software testing process using measurements. The focus of this thesis is at the possible measurements in the Software Test Plans, Methods and Techniques of the System for testing level using the Software Development Process. In this paper we conducted a literature study on all testing techniques together that are related to both Black and White box testing techniques. The overall aim of this literature study is to clearly explain different testing techniques along with a case situation and their advantages. The aims and objectives of the research work is to investigate testing methods and techniques for support provided to the test planning and test design activities for Concurrent Development Validation Testing Model, thereby, enabling the management to have better insight in to the software testing techniques. The scope of the research is restricted to the investigation for the role of software engineering in testing methods is formed software testing processes and techniques.

Keywords: Software Testing Methods, Testing Spectrum, Black Box Testing, White Box Testing, Testing Techniques.

1. Introduction

Software testing is an important means of assessing the software to determine its quality. Since testing typically consumes 40~50% of development efforts, and consumes more effort for systems that require higher levels of reliability, it is a significant part of the software engineering**: Pressman (2001, 2019).** With the development of fourth generation languages (4GL), which speeds up the implementation process, the proportion of time devoted to testing increased. A Survey of Software Testing Techniques and Analysis have been studied by **Parihar** and **Anu Bharti (2019-a)**. As the amount of maintenance and upgrade of existing systems grow, significant amount of testing will also be needed to verify systems after changes are made. Despite advances in formal methods and verification techniques, a system still needs to be tested before it is used. Testing remains the truly effective means to assure the quality of a software system of non-trivial complexity as well as one of the most intricate

and least understood areas in software engineering. Software Testing Fundamentals: Levels, Types, and Methods have investigated by **Parihar** and **Anu Bharti (2019-b).**

2. The Taxonomy of Testing Techniques

Software testing is a very broad area, which involves many other technical and non-technical areas, such as specification, design and implementation, maintenance, process and management issues in software engineering. Our study focuses on the state of the art in testing techniques, as well as the latest techniques which representing the future direction of this area**: Bezier (1990, 2002).** Before stepping into any detail of the maturation study of these techniques, let us have a brief look at some technical concepts that are relative to our research.

2.1. The Goal of Testing

In different publications, the definition of testing varies according to the purpose, process, and level of testing described. Miller gives a good description of testing: **Bertolino (2004).** The general aim of testing is to affirm the quality of software systems by systematically exercising the software in carefully controlled circumstances.

2.2. The Testing Spectrum

Testing is involved in every stage of software life cycle, but the testing done at each level of software development is different in nature and has different objectives.

- ***Unit Testing*** is done at the lowest level. It tests the basic unit of software, which is the smallest testable piece of software, and is often called unit, module, or component interchangeably. A code based testing which is performed by developers, this testing is mainly done to test each and individual units separately. This unit testing can be done for small units of code or generally no larger than a class.

- ***Functional Testing:*** Done for a finished application, this testing is to verify that it provides all of the behaviours required of it.

- ***Integration Testing*** is performed when two or more tested units are combined into a larger structure. The test is often done on both the interfaces between the components and the larger structure being constructed, if its quality property cannot be assessed from its components.

- ***System Testing*** tends to affirm the end-to-end quality of the entire system. System test is often based on the functional/requirement specification of the system. Non-functional quality attributes, such as reliability, security, and maintainability, are also

checked. Reveals that the system works end-to-end in a production and like location to provide the business functions specified in the high-level design.

- ***Acceptance Testing*** is done when the completed system is handed over from the developers to the customers or users. The purpose of acceptance testing is rather to give confidence that the system is working than to find errors. It is conducted by business owners, the purpose of acceptance testing is to test whether the system does in fact, meet their business requirements.

- ***Regression Testing:*** This testing is done to make sure that the reliability of each software release, testing after changes has been made to ensure that changes did not introduce any new errors into the system.

- ***Alpha Testing:*** Usually in the existence of the developer at the developer's site will be done.

- ***Beta Testing:*** Done at the customer's site with no developer in site.

2.3. Static Analysis and Dynamic Analysis

Based on whether the actual execution of software under evaluation is needed or not, there are two major categories of quality assurance activities:

- ***Static Analysis*** focuses on the range of methods that are used to determine or estimate software quality without reference to actual executions. Techniques in this area include code inspection, program analysis, symbolic analysis, and model checking.

- ***Dynamic Analysis*** deals with specific methods for ascertaining and/or approximating software quality through actual executions, i.e., with real data and under real (or simulated) circumstances. Techniques in this area include synthesis of inputs, the use of structurally dictated testing procedures, and the automation of testing environment generation. Generally the static and dynamic methods are sometimes inseparable, but can almost always discuss separately. In this paper, we mean dynamic analysis when we say testing, since most of the testing activities (thus all the techniques studied in this paper) require the execution of the software.

2.4. Functional Technique and Structural Technique

The information of testing involves the configuration of proper inputs, execution of the software over the input, and the analysis of the output. The software configuration includes requirements specification, design specification, source code, and so on: **Beizer (1990, 2002).** The test configuration includes test cases, test plan and procedures, and testing tools. Based on the testing information flow, a testing technique specifies the strategy used in testing to select input test cases and analyze test results. Different techniques reveal different quality aspects of a software system, and there are two major categories of testing techniques, functional and structural.

- ***Functional Testing***: the software program or system under test is viewed as a **"black box".** The selection of test cases for functional testing is based on the requirement or design specification of the software entity under test. Functional testing emphasizes on the external behaviour of the software entity.

- ***Structural Testing***: the software entity is viewed as a "**white box**". The selection of test cases is based on the implementation of the software entity. The goal of selecting such test cases is to cause the execution of specific spots in the software entity, such as specific statements, program branches or paths. The expected results are evaluated on a set of coverage criteria.

3. Testing Methods

Testing methods are usually conducted in order and include Unit testing, Integration testing, System testing, Acceptance testing**: Jorgensen (2002).**

- **Unit testing:** This type of testing is performed by developers before the setup is handed over to the testing team to formally execute the test cases. Unit testing is performed by the respective developers on the individual units of source code assigned areas**: Jorgensen (2002).** The developers use test data that is different from the test data of the quality assurance team.

- **Integration testing:** Integration testing is defined as the testing of combined parts of an application to determine if they function correctly. Integration testing can be done in two ways: Bottom-up integration testing and Top-down integration testing**: Jorgensen (2002).**

- **System testing:** System testing tests the system as a whole. Once all the components are integrated, the application as a whole is tested rigorously to see that it meets the specified Quality Standards. This type of testing is performed by a specialized testing team**: Jorgensen (2002).**

- **Acceptance testing:** This is arguably the most important type of testing, as it is conducted by the Quality Assurance Team who will gauge whether the application

meets the intended specifications and satisfies the client's requirement. The QA team will have a set of pre-written scenarios and test cases that will be used to test the application**: Jorgensen (2002).**

4. Types of Software Testing Techniques

There are two main categories of software testing techniques:

- **Static Testing Techniques** are testing techniques which are used to find defects in Application under test without executing the code. Static Testing is done to avoid errors at an early stage of the development cycle and thus reducing the cost of fixing them.
- **Dynamic Testing Techniques** are testing techniques that are used to test the dynamic behavior of the application under test that is by the execution of the code base.

4.1. Static vs. Dynamic Testing

There are many approaches available in software testing are referred to as static testing whereas actually executing programmed code with a given set of test cases referred to as dynamic testing. **Static testing** is often implicit, as proofreading, plus when programming tools/text editors check source code structure or compilers (pre-compilers) check syntax and data flow as static program analysis. **Dynamic testing** takes place when the program itself is run. Dynamic testing may begin before the program is 100% complete in order to test particular sections of code and are applied to discrete functions or modules**: Rakitin (2001).**

Static testing involves verification, whereas dynamic testing involves validation**.** Together they help improve software quality. Among the techniques for static analysis, mutation testing can be used to ensure the test-cases will detect errors which are introduced by mutating the source code**: Miller (2001).**

5. The Box Approach

Software testing methods are traditionally divided into white- and black-box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

5.1. White-Box Testing

White-box testing (also known as **clear box testing**, **glass box testing**, **and transparent box testing** and **structural testing**) tests internal structures or workings of a program, as opposed to the functionality exposed to the end-user. In white-box testing an internal

perspective of the system, as well as programming skills, are used to design test cases**: Shao et al. (2007).** The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g. In-Circuit Testing (ICT): **Hutcheson (2003).** While white-box testing can be applied at the unit, integration and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

5.2. Techniques used in White-Box Testing are as follows:

- Application Programming Interface (API): testing of the application using public and private API.
- Code coverage: creating tests to satisfy some criteria of code coverage (e.g., the test designer can create tests to cause all statements in the program to be executed at least once)
- Fault injection method intentionally introducing faults to gauge the efficacy of testing strategies
- Mutation testing methods
- Functional Technique and Structural Technique.
- Static and dynamic testing methods.

5.3. Black-Box Testing

Black-box testing treats the software as a "black box", examining functionality without any knowledge of internal implementation. The testers are only aware of what the software does during the testing Black-Box testing methods include equivalence partitioning , boundary value analysis, all-pairs testing, static transition tables, decision table testing, fuzz testing, model based testing, use case testing, exploratory testing and specification-based testing**: Beizer (1995).**

6. Software Testing and Techniques

P-ISSN: 2348-6848
E-ISSN: 2348-795X
Vol. 8 Issue 9
September 2021

Software testing techniques are the ways employed to test the application under test against the functional or non-functional requirements gathered from business: **Limaye (2009)**. Each testing technique helps to find a specific type of defect. For example, Techniques which may find structural defects might not be able to find the defects against the end-to-end business flow. Hence, multiple testing techniques are applied in a testing project to conclude it with acceptable quality:

6.1. Principles of Testing

Below are the principles of software testing:

a. All the tests should meet the customer requirements.
b. To make our software testing should be performed by a third party
c. Exhaustive testing is not possible. As we need the optimal amount of testing based on the risk assessment of the application.
d. All the test to be conducted should be planned before implementing it
e. Start testing with small parts and extend it to large parts.

6.2. Objectives of Testing

Below are the objectives and goals software testing is as follows:

- Analysing the key phases in the Software Testing Life Cycle
- Understanding of the software testing methods and generated during the software test planning and test design activities.
- Understanding of the role of measurements in improving Software Testing process
- Investigation into the attributes of Test Planning and Test Design processes those are measurable
- Understanding of the current metric support available to measure the identified attributes of the Test Planning and Test Design processes
- Analysis of when to collect those metrics and what decisions are supported by using those techniques for software testing.

7. Benefits of Software Testing

The benefit of software testing is that it allows removing errors and bugs before the products get shipped to the market. That will prevent unsatisfied clients and unnecessary expenses that bring customer support**: Myers (2004, 2011)**.

- Software Development Perspective
- To discover defects.
- To avoid the end user from defect problems.
- Number of defects detected will tell about the reliability of the software.
- To ensure that product works as user expected.

8. Impact on Software Development without Testing

a. Lot of defects at UAT.

b. We have to pay penalty for slippage of delivery.

c. High Cost of Defect fixing.

d. Dissatisfied Customer.

e. Loss of business.

f. Low moral to developer.

9. Conclusions

Software testing is an important and critical part of the software development life cycle and software development process. Testing has been widely used as a way to help engineers develop high-quality systems, and the techniques for testing have evolved from an ad-hoc activities means of small group of programmers to an organized discipline in software engineering. However, the maturation of testing techniques has been fruitful, but not adequate. Software testing is an activity to ensure the correctness; completeness and quality of the software product with respect all the requirements for testing. It is also the process of executing a program or application with the intend of finding error. The testing methods and

techniques will be more elaborately and would be covered all testing techniques together for software testing.

Software testing is a broad term encompassing a wide spectrum of different activities, from the testing of a small piece of code by the developer (unit testing), to the customer validation of a large information system (acceptance testing), to the monitoring at run-time of a network-centric service-oriented application. Software testing is a most often used technique for verifying and validating the quality of software. Fundamental research that addresses the challenging problems, development of methods and tools, and empirical studies should be carried out so that we can expect significant improvement in the way we test software. The successful use of these techniques in industrial software development will validate the results of the research and drive future research. The pervasive use of software and the increased cost of validating it will motivate the creation of partnerships between industry and researchers to develop new techniques and facilitate their transfer to practice. Development of efficient testing techniques and tools that will assist in the creation of high-quality software will become one of the most important research areas in the near future.

10. Future Scopes

Working in this area of research a scope for future work has been observed by focus the testing techniques using testing tools, testing methods, testing types for software testing of the product. There need of further testing technique and analysis for good quality of product for testing purpose. In this paper, moreover main focus on all Functional and Structural Testing Techniques are discussed. Software testing is an important means of accessing quality of software. The work is focused on the technology maturation of testing techniques, including these functional and structural techniques that have been influential in the academic world and widely used in practice. In future work will be done for Software testing activities of the software development life cycle (SDLC). It helps in developing the confidence of a developer that a program does what it is intended to do so. In other words, we can say it's a process of executing a program with intends to find errors. Verification and Validation (V&V), black box testing is often used for validation and white box testing is often used for verification. This study emphasizes the need to investigate various testing techniques in software testing field and conducted a literature review to obtain the reviews from state-of-art of software testing techniques. Testing, an important research area within computer science is likely to become even more important in the future.

References

1. Beizer, B. (1990), "Software Testing Techniques," 2nd Edition, International Thomson Computer Press, New York, Vol.18, Issue 3, pp.82-88.

2. Beizer, B. (1995), "Black-Box Testing: Techniques for Functional Testing of Software and Systems", New York: John Wiley & Sons, Inc., 2nd Edition, Vol.2, Issue 1, pp.16-20.

3. Beizer. B. (2002), "Software testing techniques", 2nd edition, Van Nostround Reinhold, New York, Vol. No. 6, Issue 1, pp.21-32

4. Bertolino, A. (2004),"Software Testing In: *Guide to the Software Engineering Body of Knowledge* (*SWEBOK)",*2nd Edition, IEEE Computer Society, Chapter 5, Prentice Hall, Inc., New Jersey,Vol.9, Issue 2, pp. 5-16.

5. Hutcheson, M. L. (2003), "Software Testing Fundamentals: Methods and Metrics". John Willey & Sons.

6. Jorgensen, P. (2002), "Software testing: A Craftsman's Approach", 2nd Edition, Auer Bach Publications Boston, MA, USA, CRC Press, Volume 32, No.1, pp. 167-171.
7. Miller, E. F. (2001), "Introduction to Software Testing Technology Tutorial", Software Testing &Validation Techniques, Second Edition, IEEE Catalog No. EHO 180-0, pp. 4-16

8. Myers, G. J. (2004), "The Art of Software Testing", 2nd Edition" Published by John Wiley & Sons, Inc., Hoboken, New Jersey.

9. Myers, G. J. (2011), "The Art of Software Testing", 3rd Edition" Published by John Wiley & Sons, Inc., Hoboken, New Jersey, pp.240-256

10. Parihar, Mithelesh and Anu Bharti. (2019-a), "A Survey of Software Testing Techniques and Analysis", International Journal of Research, **Vol**. 06, Issue: 03, March 2019, Pages:1038-1048.

11. Parihar, Mithelesh and Anu Bharti. (2019-b), **"**Software Testing Fundamentals: Levels, Types, and Methods", International Journal of Research, **Vol**. 06, Issue: 06, May 2019, Pages: 649-667.

12. Pressman, Roger S. (2001), "Software Engineering: A Practitioner's Approach", 5th Edition, Boston: McGraw-Hill.

13. Pressman, Roger S. (2019), "Software Engineering: A Practitioner's Approach", 8th Edition, Boston: McGraw-Hill.

14. Shao. D, Khurshid, S. and Perry, D. E. (2007), "A Case for White-box Testing Using Declarative Specifications Poster Abstract", in Testing: Academic and Industrial Conference Practice and Research Techniques - MUTATION, TAICPART-MUTATION, and pp.137.

15. Rakitin, S. R. (2001), "Software Verification and Validation for Practitioners and Managers", 2nd Edition, Artech House Publisher, pp. 368-370.

16. Limaye M.G. (2009),"Software Testing: Principles, Techniques and Tools", 1st Edition, McGraw-Hill.