e-ISJN: A4372-3114
p-ISJN: A4372-3115

ISSN: 2321-7782 (Online)
ISSN: 2347-1778 (Print)
Impact Factor: 7.529

Volume 9, Issue 7, July 2021

International Journal of Advance Research in Computer Science and Management Studies

Research Article / Survey Paper / Case Study

Available online at: www.ijarcsms.com

A Review on Software Testing Tools and Techniques

Priyanka Yadu[1]
(Author)
Computer Science Department,
School of Information Technology,
MATS University,
Raipur, C.G., India.

Dr. Vaibhav Sharma[2]
(Guide)
Computer Science Department,
School of Information Technology,
MATS University,
Raipur, C.G., India.

Abstract: ***The Process of developing error free application known as code testing. The bug free code production is that the aims of testing team that carrier in each level of testing. To succeed in the standard of product, code testing may be a helpful method. Testing eliminates all defect by providing various options and useful to scale back the price of code. Testers aim to produce code with low maintenance value. However the main issue in checking is to seek out appropriate test cases to check the code. We've got many advances in testing. However still the code has got to be completely tested before it's delivered to the client. during this paper, numerous forms of testing, its objective outlined and conjointly describe the code checking life cycle and test cases and phases are describe very well with their importance and significance.***

Keywords: Software Testing; Software Quality; Verification; Validation; Manual Testing; Automation Testing; Test Cases.

I. INTRODUCTION

Before development user wants and constraints should be determined and expressly explicit so as to develop a software package. The code should be designed to accommodate implementers, users, maintainers; fastidiously enforced and completely tested supply code; and should ready supporting documents. Several testing techniques involve activities performed by engineers in order that ought to study the utilization of such techniques by those Engineers. Testing is a very important engineering activity liable for a big portion of the prices of developing and maintaining code B. Bezier [1], H. Leung [2]. The processes of code testing and defect detection still challenge the -software community. Although the code testing and defect detection activities are Inexact and Inadequately understood, they're crucial to the success of a code project.

In this investigation, common testing techniques were applied to differing kinds of code by subjects that had a good vary of skilled expertise. This work is meant to characterize however testing effectiveness relates to many factors: testing technique, code kind, fault type, tester expertise, and any interactions among these factors Victor R et al. [3]. In straightforward words, Testing is looking for however well one thing works. Relating it to the code, code Testing may be an essential part of code quality assurance and represents the last word review of specification, style and code generation.

Here we are going to discuss code testing fundamentals and techniques for code action style. The preponderating objectives for code testing case style in code testing fundamentals. Check cases style focuses on a group of techniques for the creation of check cases that meet overall testing objectives. In technical terms, we can say, the mixture of Verification and Validation is termed testing. A definition of code testing would be "A method of proving that a program performs its meant functions in code testing".

II. Objectives of Software Testing

The major role of code Testing is that there ought to be no discrepancy in code development method. There are various goals and objectives of testing, that build a defect less and satisfactory code once achieved helps developers. Some of the Objectives of code Testing are summarized below-

A. *To Find and Prevent Defects*

To find defects within the code and report them to developer is that the foremost task of a tester, in order that they'll be corrected. Maximum defects will be arising for that a checker should type best set of test cases in order that. After all, a decent check encompasses a high chance of finding a blunder and bugs.

B. *Satisfies the SRS*

The Software Requirement Specification is that the another objective of testing is to evaluate whether or not the developed code satisfies the need and alternative Specification or not as a result of till or unless the code is satisfying user necessities, it's of no use to the client inspite of exploitation best programming skills, coming up with and tools.

C. *Writing High Quality Test Cases*

A test case may be a set of conditions beneath that a checker can confirm whether or not associate degree application beneath test satisfies necessities or works properly. A scientific approach to testing these strategies provided to the developer. A lot of vital strategies give a mechanism which will facilitate to make sure the completeness of check and supply the best probability for uncovering errors in code. The more accurate are the test cases; the higher are testing method.

D. *Software Reliability Estimation*

Testing conjointly helps in estimating the dependability of the code. Code dependability is that the chance of failure-free code operation for a such that amount of your time in an exceedingly such that atmosphere. Dependability estimation helps to find the common lifetime of the code and also the variety of failures occurring in an exceedingly such that quantity of your time to seek out the most reason behind failure etc.

E. *Minimum Cost And Effort*

Testing is so high - it's a meth. There’s perpetually a motto that we should always pay less for testing and a lot of for maintenance. However in actual reality, if there's no correct testing, it may result in improper style of code which is able to be high-priced to handle and can lead to huge loss.

F. *Gain Client Confidence*

Software testing helps customers gain their trust by providing a high quality product.

III. Testing Phrase

The testing phrase is formed of numerous stages of testing, reflective a level-up correspondence within which code is intended and engineered.

A. *Unit Testing*

Unit testing focuses verification effort on the tiniest unit of code style the code element or module. Making certain that the code works properly as such that by the careful style; do check a code module in isolation. Smart unit tests facilitate to reconstruct future code, as they assure that the revised code still works evidently and may thus be incorporated into the project.

B. *Integration Testing*

Testing of communication and interaction between various codes modules that are to be integrated. Integration tests are defined based on the architectural design of the system, and assure that all modules can work together to achieve the functionality specified in the design. The level of coverage for previous tests determine by code coverage analysis. If the level does not meet the predefined limit, the test cases should be extended until a satisfactory coverage level is attained. Since the coverage of test cases depends on the actual code implementation, coverage is evaluated to change the code to maintain coverage.

C. *System Testing*

Testing of system level requirements as stipulated by the software requirements specification. This might include tests for performance, interoperability, portability, usability, install ability, etc.

D. *Acceptance Test*

Testing of final product against specification of user requirements. The 'user' can refer to the real end-users using the program, or in the case of prototypes or novel applications, that the developers define what they are trying to achieve.

IV. EXISTING SOFTWARE TESTING TECHNIQUES

Software testing is a process that is used to measure the quality of software developed and also the process of exposing errors in a program and makes it a possible task. This is a useful process of executing the program with the intention of finding the bug. Fig. 1 represents the most prevalent techniques of software testing that are purposefully classified [4]. The first step is to generate test cases while commencement of the testing process. The various test cases are used during development, for the effective and accurate testing. The internal working of a product, tests can be conducted to ensure that is, internal operations are performed according to specifications and all internal components have been adequately exercised. The first test approach is Black box testing and the second is White Box testing and the one is Grey Box testing J. Irena [7].

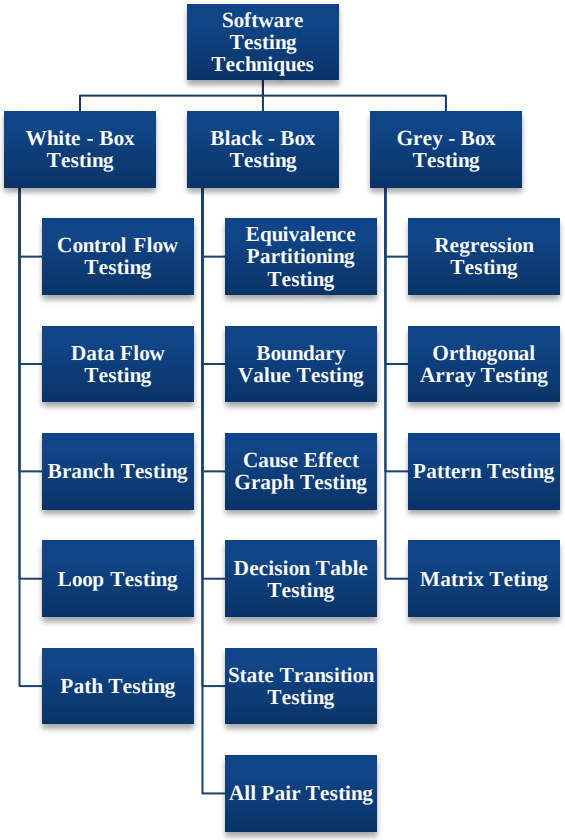

Fig. 1 Software Testing Techniques

A. WHITE BOX TESTING

White-box testing, sometimes called glass-box testing is a test case design method that uses the control structure of procedural design to derail test cases. Using white-box testing methods, the software engineer can obtain test cases:

- Guarantee that all independent paths are used at least once in a module.
- Exercise all the ends on their true and false sides.
- Perform all operations at their limits and within their operational limits, and
- Use internal data structures to ensure their validity Roger Pressman Book [11].

White box testing is also known as clear box testing, open box testing, structural testing, transparent box testing, code-based testing and glass testing. This is usually done by developers. Such testing can be applied at all levels, including unit, integration or system testing.

This type of test is also called a security test that satisfies the need to determine whether information systems protect data or maintain intended functionality. Since such a test process uses the software's internal logical system, it is able to test all independent paths of a module, every logical decision is used, all ends are tested at each boundary level, And internal data structures are also exercised. However, due to the inclusion of programming skills in the testing process [8] [9] a white box test serves a purpose for being a complex testing process.

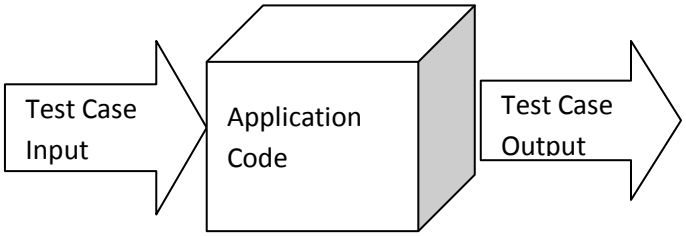

Fig 2. White Box Testing

B. BLACK- BOX TESTING

Black box testing is also called behavioral testing which focuses on the functional requirement of software. That is, black box testing enables the software engineer to obtain sets of input conditions that will fully utilize all functional requirements for a program. Black box testing is not an alternative to white-box techniques. To uncover a different error than white-box methods it is complementary approach **Roger Pressman Book** [12]. Black box testing is defined as a testing technique in which the functionality of the application under test (AUT) is tested without looking at the internal code structure, implementation details, and knowledge of the software's internal paths. This type of testing is entirely based on software requirements and specifications. In Fig 3 black box testing we focus on the input and output of the software system without bothering about the internal knowledge of the software program **M. Shaw** [6] **E. F. Miller** [9].

Fig 3. Black Box Testing

C. GRAY BOX TESTING

Gray Box testing is a technique for testing with partial knowledge of the internal workings of a software product or application. The purpose of this test is to discover defects due to improper code structure or improper working usage of an application. In this process, context-specific errors that are related to web systems are commonly recognized. It increases test coverage by focusing on all layers of any complex system. Gray box testing is a software testing technique which combines the feature of white box testing and black box testing.

Fig 4. Gray Box Testing

V. Evolution of Testing

The concept of testing was of no importance in the early days. Developers develop the product and finalized without testing. Glenford Mayon was the one who introduced the concept of testing. He began the trial with all possible combinations.

There are 5 stages in the development of the test.

In step 0, the test was based on debugging. This was considered a positive test.

In Phase 1, test cases were written and tested. But Phase 1 failed due to the possibility that with increasing test cases, the software could fail.

The test in Phase 2 was performed to test whether the software worked under normal conditions and did not fail to work under abnormal conditions.

Testing in Phase 3 was based on the risk of working with a generally acceptable value.

Phase 4 is the final stage of testing and is the test performed in recent times. The worm found in the test is called Lunank and the testing team will work on finding the defects **Glenford J. Myers et al.** [5].

VI. Software Testing Tools

There are two types of software testing - manual and automation. By using various tools Automation testing can be done. There are many software testing tools available, so it becomes quite difficult to choose the best tool for the project and these tools are also categorized on the basis of certain parameters like test management tools, load testing tools, mobile testing tools, defect tracking tools, is done. Some of the common testing equipment's like safety testing equipment are summarized below-

1. *SELENIUM*

For testing web applications Selenium is the best open source portable framework. It is written in Java. It has many components like Selenium IDE, Selenium Client API, Selenium Web Driver, Selenium Remote Control and more.

2. *TESTPAD*

Testpad is a simple test management tool. Testpad has many features such as mobile and tablet friendly, easy, easy, natural way of testing, drag and drop organization etc.

3. *JEFFIRE*

Zephyr, test management software provides a suite of tools to optimize speed and test quality. When the test load increases, the agile team expands, ensuring Zephyr reliability and testability.

Apart from the above tools, there are many other tools like qTest, QMetry, QAComplete, Ranorex etc.

VII. Software Testing Life Cycle (STLC)

In Fig 5 discusses the STCL stages, stages and stages that software undergoes during the testing process. However, there is no fixed standard of software or application undergoing STLC, and it varies by region worldwide M. Shaw [6].

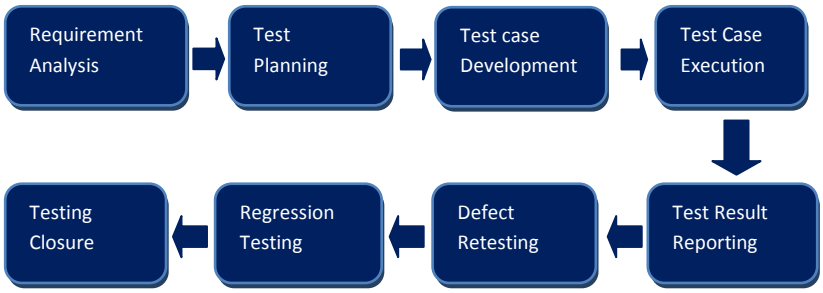

Fig 5. Software Testing Life Cycle (STLC)

This phase is related to the preparation of the test plan, which will be the final deliverables of this phase. The test plan is a mandatory document for the functional testing of the application, without which the testing process is not possible M. Shaw [6]. The test designing phase is the stage where the test case develops, and the test planning activity stops. Suitable test cases are written manually or in some cases by the QA team, automated test cases are generated.

The test case specifies a set of test inputs or data, execution conditions, and expected results. The specified set of test data should be chosen in such a way that it produces the expected result as well as intentionally incorrect data that will cause an error during testing. This is usually done to check what the conditions are for applying M. Shaw [6].

The test execution phase consists of execution based on the test plan produced during the execution phase of the test plan. If the functionality passes the execution phase without a bug report, the test is cleared or passed, and each failed test case will be associated with the bug or error found. The delivery of such activity is a defect or bug report.

Test reporting is the reporting of results generated after the execution of test cases including bug reporting which is then passed on to the development team so that it can be fixed M. Shaw [6].

VIII. LITERATURE SURVEY

Pavithra L et al. [13] done several studies within the field of code testing are helpful in future as a result of cluster of author will forestall defects earlier. So this paper provides details regarding code testing techniques and regarding the various testing tools. Code testing is typically less formal attributable to the robust apply and methodologies of testing. Code testing could be a team primarily based project work and each individual got to perform their role for manufacturing 100% bug free code. Although testers aim at manufacturing 100% bug free code there'll be defects found throughout the upkeep.

Maria Siniaalto and Pekka Abrahamsson [14] group of author target Test-driven development is one in every of the foremost elementary practices of agile development. It's aforementioned to yield many edges, nevertheless a number of these claims haven't been by trial and error studied or supported.

Mika V. Mantyla [15] author gift gamification proposals got to each ends of testing – automatic unit-testing that's technical, and end-user connected testing, i.e. beta-testing and searching testing. Second, the multitude of various roles and crowd sourcing in testing were recognized. Third, varied chemical action parts were gift.

Shunkun principle et al. [16] researchers projected a brand new intelligent search-based technique that generates take a look at cases mechanically for coverage-oriented soft- ware testing. The projected technique, referred to as the regenerate genetic algorithmic rule (RGA), adds a brand new regeneration strategy to ancient GA. during this projected RGA, the population aging method is directed by process a brand new population aging issue to see the cur- rent degree of population aging.

André van Hoorn, J. Waller, and W. Hasselbring [17] during this framework presentation the picturing of observance records, observance go online Online/offline performance analysis and feedback, e.g., Continuous observance of application behaviour and usage. Performance anomaly detection and identification (Self-) adaptation management, Extraction of code

subject field (performance) models and visualizations, Simulation (replaying antecedently monitored stimuli; activity, logging, and analysis).

Boby Georgea, Laurie Williams [18] a vital thought during this analysis is that the management pairs were asked to put in writing take a look at cases when they developed code. However, just one cluster wrote any worthy take a look at cases. This resulted in Associate in nursing uneven comparison of the time taken and thence a limitation to the present study. There are edges ensuing from the take a look at cases created by the TDD programmers. First, the TDD pairs made take a look at assets together with the implementation code. Second, the code developed is testable.

Chayanika Sharma et. al [19] in this paper, a survey of various code testing techniques wherever GA is expeditiously used is given. This paper is split into four sections. Section a pair of describes in brief the operating of a GA. In section three, applications of GA in numerous varieties of code testing is delineate. Section four concludes the paper and offers an outline of our future work.

Vahid Garousi [20] during this article group of author shows that current maturity models and techniques in TMA/TPI provide cheap recommendation for business and also the analysis community. Group of authors counsel directions for follow-up work, e.g., exploitation the findings of this MLR in industry-academia cooperative comes and empirical analysis of models and techniques within the space of TMA/TPI as rumoured during this article.

IX. CONCLUSION

Software testing is an activity done to evaluate software quality and improve it. This is one of the broad topics that seek immediate attention in this era of new and high demand for quality software. To test in a more effective way, this paper presented a comprehensive description of all the terminology that relates to software testing. Usually, it happens that people know the best techniques, methods, tools involved in testing. But they do not know the basic test goals and this is reflected in their poor test report. Therefore, in order to test efficiently and properly, everyone involved in testing must be familiar with the basic software testing goals, objectives, principles, and concepts. Only then, robust, reliable, accurate, flexible and efficient software can be delivered to the public. Focusing more on WHY TO TEST rather than HOW TO TEST is the need of the hour because HOW is easier when WHY is clear.

References

1. B. Bezier. “Software Testing Techniques”, Van Nostrand Reinhold, New York, NY, 1990.
2. H. Leung and L. White, “Insights into regression testing”, In Conf. Softw. Maint. Pages 60–69, Oct. 1989.
3. Victor R. Basili Richard W. Selby, Jr. “Comparing the Effectiveness of Software Testing Strategies” May 1985.
4. Software testing by Jiantao Pan available at http://www.ece.cmu.edu/~roopman/des-899/sw_testing.
5. Glenford J. Myers, Corey Sandler and Tom Badgett, (2011), “The Art of Software Testing”, (3rd edition). Wiley Publishing.
6. M. Shaw, “Prospects for an engineering discipline of software”, IEEE Software, November 1990, pp.15-24.
7. J. Irena. “Software Testing Methods and Techniques”, 2008, pp. 30-35.
8. Guide to the Software Engineering Body of Knowledge, Swebok, A project of the IEEE Computer Society Professional Practices Committee, 2004.
9. F. Miller, “Introduction to Software Testing Technology”, Software Testing & Validation Techniques, IEEE, 1981, pp. 4-16.
10. L.S. Chin, D.J. Worth, and C. Greenough (2007) “A Survey of Software Testing Tools for Computational Science”.
11. Roger Pressman, (2010), Testing Conventional Applications. In McGraw.Hill International Edition (7th Eds.), Software Engineering (pp. 485).
12. Roger Pressman, (2010), Testing Conventional Applications. In McGraw.Hill International Edition (7th Eds.), Software Engineering (pp. 495).
13. Pavithra L et. Al. (2019), “Survey on Software Testing”, Journal of Network Communications and Emerging Technologies (JNCET), Volume 9, Issue 3, March (2019).
14. Siniaalto, M. and Abrahamsson, P. (2007). A Comparative Case Study on the Impact of Test-Driven Development on Program Design and Test Coverage”, First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007).
15. Mantyla, M., & Smolander, K. (2011), “Gamification of Software Testing -an MLR”, (p. 1).
16. Shunkun Yang et al., “RGA: A Lightweight and Effective Regeneration Genetic Algorithm for Coverage-Oriented Software Test Data Generation” Information and Software Technology, vol. 76, 1 Aug. 2016, pp. 19–30.
17. André van Hoorn, J. Waller, and W. Hasselbring (2012) “A Framework for Application Performance Monitoring and Dynamic Software Analysis”

18. Boby Georgea, Laurie Williams (2004) "A structured experiment of test-driven development".
19. Chayanika Sharma et. al. (2013), "A Survey on Software Testing Techniques using Genetic Algorithm".
20. Vahid Garousi, (2017), "A systematic literature review of literature reviews in software testing", Software Engineering Research Group Department of Computer Engineering Hacettepe University, Ankara, Turkey, vahid.garousi@hacettepe.edu.tr.

AUTHOR(S) PROFILE

Priyanka Yadu, received the PG diploma in Computer Application from Makan Lal University, Bhopal in 2005, M.Sc. degree in Information Technology from Guru Ghasidas University in 2008 and M.C.A degrees in Computer Application from Sikkim Manipal University in 2010. Since 2008, She is a Assistant Professor in Central College Of I.T. in Computer Science Department. And doing Ph.D. in Computer Science from MATS University, Raipur (C.G.)