Original Article

BRIEF REVIEW ON DIFFERENT MANUAL SOFTWARE TESTING APPROACHES & PROCEDURE

Prasad Dhore[1], Lalit Wadhwa[2], Pankaj Shinde[3], Deepti Chaudhri[4], Priyanka Vyas[5]

[1, 3, 4] Assistant Professor, Department of Computer Science and Engineering, Nutan College of Engineering and Research, Talegaon Dabhade, Pune, MH, India.
[2] Professor, Department of Electronics & Communication, Dr. D.Y Patil Institute of Technology, Pimpri, Pune, MH, India.
[5] Assistant Professor, Department of Computer Science and Engineering- Artificial Intelligence, Nutan College of Engineering and Research, Talegaon Dabhade, Pune, MH, India.
Email: dhoreprasad89@gmail.com[1]

DOI: 10.47750/pnr.2023.14.S02.56

Abstract

Manual testing is a machine testing method in which testing cases are carried out manually rather than utilizing an automatic process. Both test cases were handled manually by the tester from the perspective of the end consumer. It guarantees that the program performs, regardless of whether or not it is specified in the prerequisite text. In order to complete approximately 100% of the application, test cases are planned and implemented. Manually produced test case reports are common. Since it can spot both visible and hidden program bugs, manual testing is one of the most effective testing techniques. A defect is described by the software as the difference between expected and actual performance. The bugs were corrected and turned over to the tester by the creator. Manual training is expected before automatic testing for any newly created software. This research requires a lot of time and resources, but it ensures the entire program is bug-free. Manual research entails knowledge of manual testing methods but does not require the use of an automated evaluation method. Manual checking is necessary since one of the fundamental software testing concepts is that "100% automation is not feasible."

Keywords: Image processing, testing, Manual.

Introduction

Manual testing is the process of manually testing applications to find bugs, problems, and defects. The aim of a software test is to split the machine and figure out how it reacts to various scenarios. Manual testing is the practice of manually testing software for product bugs, faults and defects. The role of a computer tester is to break the device and find out how it responds to various scenarios [17]. The system's observed behavior is sometimes opposed to the system's expected or desired behavior. If the two differ, the tester raises the issue and files a bug report. A tester may use a range of manual machine evaluation methods to assess some aspect of the application, either functional or non-functional. The following steps of manual control on a software product were done. The engineers first assess each test functionality unit, then merge individual units and eventually assemble the entire device. Device inspection is performed at this stage to ensure that the whole device is performed as anticipated. [18].

Stages of manual testing

After completing the business application verification, the software product is transferred to consumers. The users will perform Customer Acceptance Testing before the app is approved or rejected.

A. Unit Testing

The unit checking of the developer is the testing of individual devices and components. A computer is the smallest computing machine to evaluate. Unit research is a kind of white box evaluation and is usually done using a programming language. It is important that the developers test the app before the release is passed on to QA. There are a host of benefits for unit testing. It leaves the code flexible and unit control needs a modular coding approach. It even simplifies debugging.

B. Integration Testing

Integration monitoring is performed during device control. When different software units, components and modules are combined, integration tests are performed. Integration checking ensures usability, stability and reliability of the units. The main subjects of integration research are the functional areas that are directly or indirectly affected by integration, such as features that take input from one module and provide output from the other.

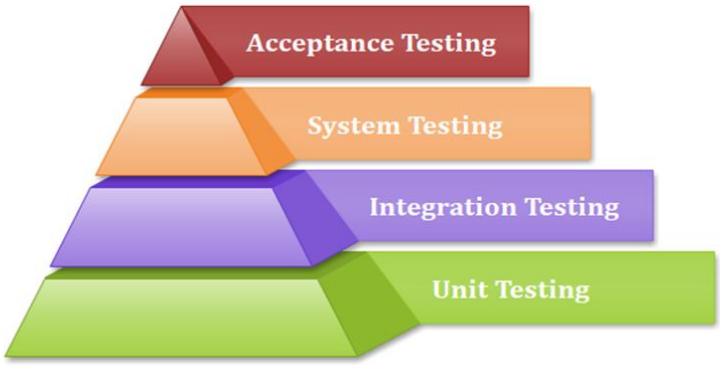
C. System Testing

After integration testing, the next conceptual step is device testing. The computer device test assesses the behaviour of a machine and its compliance in a fully integrated consumer product with the software requirements (SRS). Because you have no chance to study computing devices, it can be difficult. In order to ensure that the whole system meets the requirements of the user, it is advisable to build a research scenario close to the manufacturing environment to generate outcomes in the real world. Both information device evaluation approaches include functionality, accuracy, scalability, tension and regression testing. We also recently explored in great detail the many aspects of system testing, which every new tester should be acquainted with. [18]

D. Acceptance Testing

Acceptance testing is the study that users conduct before adopting a Product functionality, which is often known as User Acceptance Testing. The cases from real-world examples of the end consumer are usually covered in this exam. For this purpose, UAT is carried out by users with different roles and privileges [18]. we must run intelligent consumer acceptance checks as the findings would decide if a software application is accepted or refused by management. You can feel obliged to create a UAT testing strategy to help you handle testing for user acceptance.

Figure 1: Stages of Manual Testing



METHODS & PROCESS OF MANUAL SOFTWARE TESTING

Software testing can be broadly classified into the following methods:

A. Black Box Testing

As the name suggests, the Black Box approach believes the tester has no knowledge of the code or structure of the Program. The tester works with the software and tests its usable as well as non-functional features. There are a number of testing methods in black boxes, which help the tester to find bugs and defects. We can run over a few key manual control techniques in a second.

B. White Box Testing

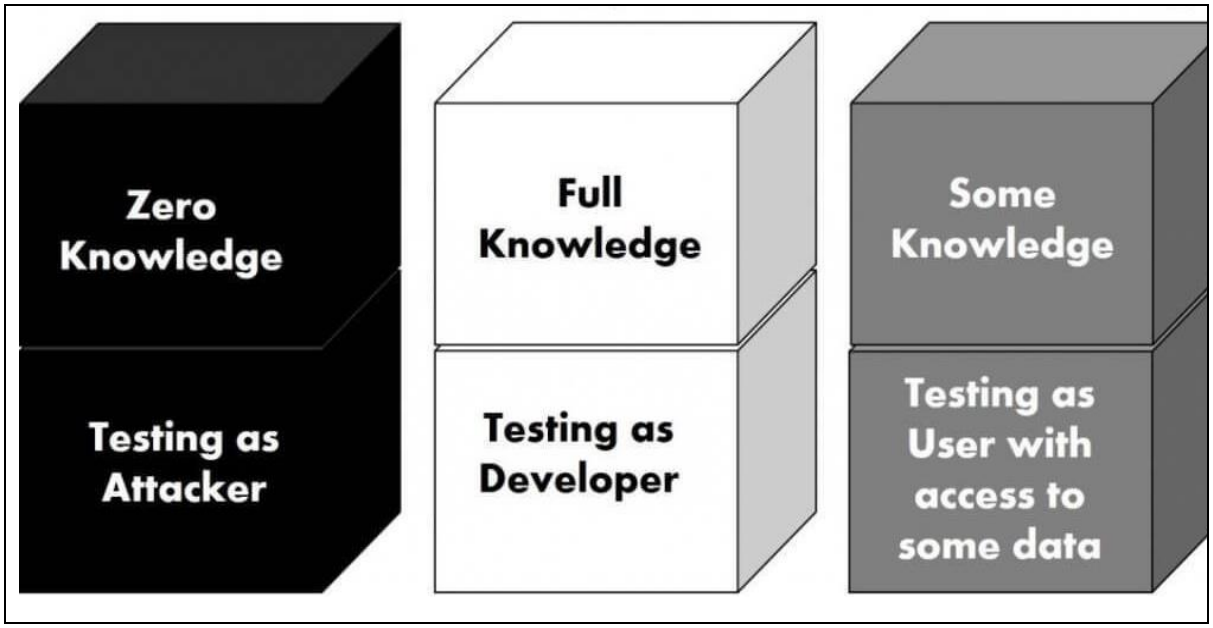
Checking the white box is a validation procedure in which the tester knows the code and configuration of the Program. It is often known as transparent study box or checking glass box. Developers are using this to inspect units. Whitebox testing techniques also include control flow control verification, data flow review, branch testing, claim coverage, judgement coverage, and path testing.

The most common consumers of the White Box analysis approach are the developers, not observers. But don't get sidetracked by checking the white box. Recognize the gaps between analysis methodologies in black box and white box and focus on improving your evaluation abilities in black boxing [18]. However, as your career advances, you realise that you must learn white box testing to expand your skills and bring value to your business.

C. Grey Box Testing

The technology of Grey Box blends the benefits of black and white box inspection. The objective of grey box testing is to identify problems and defects in an application induced by its structure or improper usage. Where a critical flaw has to be corrected in a web Program, grey box scanning is used. Many tools are accessible to help you with grey testing apps.

Figure 2: Methods of Manual Testing Process



Manual Software Testing Process

Given that this is a guide to manual testing for beginners, we believe that you need to provide a brief description of the software test process. This enables you to consider the subjects that we would discuss appropriately in the subsequent pages. A software testing process typically involves many steps to ensure that the product has been thoroughly checked. The first step is to gain a detailed industrial interpretation of the application. The next step is to create a study Program that will act as a guideline for the testing process. This test schedule is created by the test manager or the QA manager.

After the test design has been developed, test situations and test cases are manually created. If the commodity has been completed and tested ready, test cases are executed and the outcomes recorded as 'Pass/Fail.' When a test case fails, a problem or error is presented. This mistake is assigned to a developer who addresses it and the tester retests it. The loop is repeated until all bugs are fixed.

Human perspective: only people may look at and judge the basic usability and appearance of the application. As the software is for users only, validation from a customer point of view can only be a good work. A broader range of system workflows and views: Manual verification offers a wider perspective of the overall software. In contrast to a coding system which repeats the same steps every time, the human mind remains in a state of exploration? The validation of the system will then have a broader spectrum of coverage [5].

Automation cost is the extensive automation trials are always unfair due to project schedules or scale and we also prefer fast manual validation for automation research. Unautomated situations: a multitude of cases are either not automated or lack high customer confidence when experimenting with automation alone. For example, a number of scenarios have occurred on mobile devices like user intervention, including "Tap & Pay," with various behaviors, where tools are automatic, then when manually validated.

Advantages of Manual Testing

- There are no automated mechanisms used, resulting in a cost-effective operation
- Manual control captures most of the bugs
- People monitor and evaluate rather than digital resources

Manual checking relates to the method of reviewing an application manually. The research cases/scenarios are manually conducted without the use of pre-made facilities and the results are checked by Testers (software testing professionals). Manual checking is, thus, a process whereby we compare the activities of a piece of software (which may be a vector, feature, entity or something else) with the predicted behavior we developed at the early stages of the SDLC.

Manual verification is the most common form of software testing. Someone who has never used a tool before will do this. A student with a good understanding of the curriculum or testing of a system may conduct manual inspection. However, it is an important step in the course of software testing. Any new system or technology must be manually tested before automating tests.

How to perform Manual Testing

1. Read and understand the application documentation/guides. Even, if one is available, check the test application (AUT).
2. 2. Have test cases with all specifications of the documents.
3. Analysis and baseline test cases with the help of the team leader and the client (as applicable)
4. Run the AUT test cases.
5. Make a note of fault.
6. Re-run failing test cases after the bugs have been corrected to guarantee they are passed

TYPE OF TESTING

A. Smoke Testing

Smoke inspection is a high-level manual test to see how the Program achieves its main goals and is free of critical flaws. Smoke checking is a non-exhaustive method since it just tests the program's key features. After adding a new function to the software, smoke checks are often used to verify the build. Until doing multiple smoke checks in a suite, the QA team normally determines which device components should be tested. Smoke checks are a form of preliminary examination that occurs before more severe, in-depth examinations.

Smoke Checking is a software testing technique that determines whether or not a piece of software is stable. For the QA squad, smoke checking is proof of ongoing device testing. In order to verify Program functionality, it consists of a minimum number of tests per build. "Build Verification Testing" is another name for smoke inspection. Simply stated, we inspect the construction and see whether critical aspects work and if there are any show-stoppers. It's a quick-return mini-test for big features. It's a simple test that verifies that the product is fit for testing. This assists in determining if the build is flawed in order to avoid wasting time and resources on additional checks.

Smoke checking is critical in the construction of software because it ensures that the architecture is correct from the start. This encourages them to and the amount of time we spend on testing. As a result, smoke scans ensure that the system is in proper working order. We only begin functional testing after we have completed smoke testing.

Smoke inspection can identify all show stoppers in the construction, and smoke checks will be done after the QA has been published. We automate the detection and repair of significant flaws through smoke testing. QA team can find glitches in the application's functionality that might have shown up in the most recent code by using smoke screening.

B. Cross Browser Testing

Cross Browser Testing is a method of test that tests if an application works properly through several browsers. It's a method of ensuring the application's compatibility through various browsers. There are no promises that a website can appear the same in any browser since each browser will behave differently and build the website based on its own interpretation. Because of these factors, conducting cross-browser tests prior to the creation of a website is critical. To ensure that all browsers have a consistent experience, cross-browser testing is performed.

Browser testing examines the consistency, design, accessibility, and responsiveness of an application. Cross-browser testing can begin at the end of the development period, so that any or all of the key functionality can be tested on how they render in various web browsers. Cross-browser checking is usually done by the QA team and/or the Programrs. Since the design team is comfortable with each pixel, using them may be beneficial. Much of this cross-browser testing may sound daunting, but you just need to plan ahead of time to ensure that you evaluate it in the right ways and that you don't run into any unexpected issues. When operating on a big project, you should monitor it often to ensure the new improvements are usable for the target group and that new programming revisions should not break existing features.

The purpose of the Cross Browser Testing can be summarized as

- Knowing and fixing what is wrong.
- Improve performance, user experience and company.
- Any pitfalls to be told

C. Acceptance Testing

Acceptance assessments are distinct from most kinds of manual tests in that they focus on detecting glitches. Acceptance testing, also known as User Acceptance Testing or UAT, has the aim of demonstrating how closely the application matches the wishes and expectations of the consumer. Acceptance checking is performed until all bugs have been addressed. During acceptance checks, the system should be ready for the market, since this type of research gives the user a good view of how the software Program looks and works in actual life. Acceptance checking can be performed by the product's client or end user. Since it is performed after implementation and bug fixing, it is one of the most important types of tests.

The most important assessment method is acceptance testing, which decides whether or not the user approves the application/software. It could involve the app's flexibility, usability, performance, and user interface (U.I.). User approval assessment is also regarded as OAT testing or end-user testing. It's one of the last steps of the program's testing period, and it will happen before a customer accepts a new proposal. Acceptance checks are black-box interface tests. In real-time situations, device owners perform tests to see whether the software/application meets all specifications.

Uses of Acceptance Testing

- Analysis of Business Requirements
- Creation of UAT test plan
- Identify Test Scenarios
- Create UAT Test Cases
- Preparation of Test Data(Production like Data)

- Run the Test cases
- Record the Results
- Confirm business objectives

D. Beta Testing

Beta testing is a common way of gathering feedback from real consumers after a soft launch before releasing a product to the general public. It allows tech teams to get useful input from a diverse range of customers via real-world usage apps. The product will be sent for beta testing after internal team testing is finished. At this point, it's important to believe that the app can manage a high amount of traffic, especially if the beta testing audience is public.

Closed and open beta training would necessitate rigorous practise. Closed beta testing grants access to the platform to a small group of users who have been identified by a request and acceptance phase. Open beta testing means that those involved in utilising the product in its unreleased version can learn from feedback from a diverse group of test users. Beta testing, also known as user testing, is done at the end user's site to ensure that the web is accessible, usable, compatible, and reliable. Beta testing adds value to the software creation life cycle by supplying "actual" customers with feedback on a product's design, functionality, and usability. These inputs are useful not only for product production, but also as an investment in potential products, whether the data collected is efficiently handled.

E. Exploratory Testing

There isn't anything in the way of a structure or guidelines for exploratory science. When conducting fly checks, instead of implementing a preset script for each test scenario, the tester can use their effort and curiosity to "Explore" and think about the sample. Exploratory research is a form of ad hoc test that may be used at some point during the production and testing phase if the team believes it is essential. It is often performed by non-testers, such as artists, product managers, or engineers, due to the lack of formalities involved.

This kind of software testing involves testers reviewing the system on the run rather than pre-creating test cases. Before you start the exam, you may get some suggestions about what to calculate. The majority of exploratory science focuses on "thinking" testing. Exploratory testing is a form of testing that is often used in Agile models that is concerned with innovation, discovery, and learning. It emphasises the tester's personal independence and responsibility. The scanning test has the following advantages: it needs fewer preparation, it is easy to find critical glitches, and it normally activates the mind rather than running Programd experiments at runtime.

Another significant benefit is that testers will steer their future fly experiments using deductive inference focused on the results of previous observations. They are not required to complete a current collection of script tests before concentrating on or moving to a more target-rich environment. If done correctly, this will also help in bug identification.

Features Exploratory testing -

- Rigorous
- Is cognitively (thinking) structured in contrast to the predetermined research methodological framework.
- Chartering, time boxing, and other factors contribute to this arrangement.
- instructional and practical.

- It's not a tactic nor an approach

F. Negative Testing

Negative evaluations examine how an implementation responds to intentionally invalid inputs. Negative checking is possible at all levels of growth and testing, but not before error handling and exceptions are introduced. This sort of test is usually performed by the QA team or engineers, who often work with copywriters to ensure that all relevant messages are used. Negative checks are a method of evaluating an app or system that ensures the app plot satisfies the standards and can tolerate unfavourable input and consumer behaviour. Invalid data is applied to equate the output to the required information.

Negative testing ensures that the software can handle inappropriate or intrusive consumer behavior. If a user types a letter into a numeric field, for example, the appropriate response is to display the message "Incorrect data form, insert a number." Negative checks are intended to detect certain situations and prevent Programs from crashing. Negative evaluations often aid in the improvement of the application's consistency and the identification of its flaws. The main difference between positive and bad tests is that throwing an exception is not a common phenomenon. When running negative checks, assume exceptions; this means that the software handles improper user behaviour. Negative testing is a form of software test that looks for unusual input data and conditions in a software Program.

Unexpected details or scenarios may vary from erroneous information to serious cyber attacks. Negative checks are used to deter software applications from failing as a result of negative criticism, to increase accuracy and usability, and to guarantee that our technology only runs under normal circumstances. To guarantee that our device is fully defect-free, we must ensure that it can survive unexpected situations.

Purpose of Negative Testing:

- Negative checks are used to prevent an application from crashing and to improve the reliability of an application by identifying defects.
- Negative checking allows you to maximise the evaluation coverage on the submission.
- Negative checks improve the program's stability and reliability.
- Negative checks, in addition to positive tests, enable users to verify an application using either correct (or invalid) input results.

G. Usability Testing

Since it is associated about how a buyer thinks when faced with the device, usability testing is the most mentally active in manual testing. This type of evaluation analyses the user's behaviour and emotional responses to determine the usefulness of the Program. Usability testing (UX) is a research tool for determining how user-friendly and basic a software application is. A Program interface is used to expose usability vulnerabilities in a small number of target end users. The user's ease of usage of the Program, the application's flexibility in managing commands, and the application's potential to accomplish its goals are the primary considerations of usability testing.

Under defined circumstances, this evaluation decides the software product's comprehension, ease of learning, ease of use, and appeal to consumers. & When doing usability research exercises, participants talk aloud to the interviewer and observers to express their thoughts. It is beneficial for the client to think aloud in order to learn. Usability checks can be conducted at some point during the production phase to analyse and compare specific functionality or an entire Program, based on its size. Usability checking involves involving genuine users of the

software that were not interested in its creation in order to provide real-world feedback that can be used to improve the product.

Uses Usability Testing

- It aids in the discovery of accessibility issues prior to the product's sale.
- Increases consumer satisfaction • Increases the device's capacity
- During a usability evaluation, it allows you to gather real feedback from your audience who are actually using your product. You don't have to depend on the "opinions" of strangers.

Conclusion

In this review paper we present Manual testing technique for finding the bug, error or fault by using the different process. Similarly we apply the different method like testing as attacker used the black box , testing as developer for white box &testing as user with access same data used the gray box method for the finding the defect.

References

1. R.S. Pressman & teal, “Software Engineering A Practitioner’s Approach” 6/e; Chapter 14: Software Testing Techniques, 2005
2. Mohd. Ehmer Khan, “Different Approaches to Black Box Testing Technique for Finding Errors,” IJSEA, Vol. 2, No. 4, pp 31-40, October 2011
3. M. A. Jan, P. Nanda, X. He and R. P. Liu. 2013. “Enhancing lifetime and quality of data in cluster-based hierarchical routing protocol for wireless sensor network”, 2013 IEEE International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC & EUC), pp. 1400-1407.
4. M. A. Jan, P. Nanda, and X. He. 2013. “Energy Evaluation Model for an Improved Centralized Clustering Hierarchical Algorithm in WSN”, in Wired/Wireless Internet Communication, Lecture Notes in Computer Science, pp. 154–167, Springer, Berlin, Germany.
5. F. Khan, K. Nakagawa. 2012. “Performance Improvement in Cognitive Radio Sensor Networks” in the IEICE Japan.
6. Zhang Hongchun, Research on New Techniques and Development Trend of Software Testing. Myers, Glenford J.(1979), IBM Systems Research Institute, Lecturer in Computer Science, Polytechnic Institute of New York, The Art of Software Testing, by John Wiley & Sons, Inc.
7. Fu Bo (2007), Automatic Generation Method of Test Data Based on Ant Colony Algorithm, Computer Engineering and Applications.
8. R. Chauhan and I. Singh, "Latest Research and Development on Software Testing Techniques and Tools", INPRESSCO, International Journal of Current Engineering and Technology, 2014.
9. Prasad Dhore, Lalit Wadhwa, Pankaj Shinde, Deepak Naik , Sanjeevkumar Angadi “proficient exploration of malnourishment with machine learning by cnn procedure”, “journal of northeastern university” , Volume 25 Issue 04, 2022 ISSN: 1005-3026, pp-1916-1932
10. Prasad Dhore, Aparna Pande, Shital Mehta, Saili Sable, “Human pose estimation and classification: a review”,Neuroquantology , November 2022, volume 20, issue 15, page 3199-3213.
11. S. P and D. N, "Automation of Software Testing in Agile Development – An Approach and Challenges with Distributed Database Systems", GJRA - GLOBAL JOURNAL FOR RESEARCH ANALYSIS, vol. 3, no. 7, 2014.
12. N. Bhateja, "A Study on Various Software Automation Testing Tools", International Journal of Advanced Research in Computer Science and Software Engineering, vol. 5, no. 6, 2015.
13. S. Sharma, "Study And Analysis Of Automation Testing Techniques", Journal of Global Research in Computer Science, vol. 3, no. 12, 2012.
14. "Analysis of Automation and Manual Testing Using Software Testing Tool", IJIACS, vol. 4, ISSN 2347 – 8616, 2017.

15. S. Thummalapenta, S. Sinha and N. Singhania, "Automating Test Automation", IBM T. J. Watson Research Center, IBM Research - India, 2017.

16. "An Approach of Software Design Testing Based on UML Diagrams", International Journal of Advanced Research in Computer Science and Software Engineering, vol. 4, no. 2, 2014.

17. Prasad Dhore, Yogesh Kumar Sharma, Prashant Kumbharkar, “Malnutrition Detection and Administration Organization”, International Journal of Computer Techniques, Volume 8 Issue 1, January 2021, pp. 1-4.

18. Prasad dhore, lalit wadhwa, deepak naik, pankaj shinde, “study on various machine learning techniques for plant disease detections in agricultural sector”, journal of pharmaceutical negative results, volume 13, special issue 7, 2022, page 3336-3348.

19. Prasad dhore, lalit wadhwa, pankaj shinde, deepak naik, sanjeevkumar angadi, “proficient exploration of malnourishment with machine learning by cnn procedure”, journal of northeastern university , volume 25 issue 04, 2022 issn: 1005-3026 page 1916-1932.