International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048
Volume 6, Issue 10, pp 142-144, October-2022
https://doi.org/10.47001/IRJIET/2022.610028

Software Testing Using White Box

[1]Prof. S. B. Bele, [2]Arjun A. Harinkhede, [3]Vaibhav M. Bhatti

[1]Assistant Professor, Department of MCA, Vidya Bharati Mahavidyalaya, Amaravati, India
[2,3]Student, Department of MCA, Vidya Bharati Mahavidyalaya, Amaravati, India

***Abstract* - In Software Development Life Cycle (SDLC) one of the most time-consuming parts is Software Testing. Software Testing is simply used to detect the bugs, failures, and faults in software and recover and free from the error in the early phases of the Software Developing Life Cycle. Software testing is the process of evaluating and performing the software with the aim of finding out the errors and bugs. There are different types of techniques and methods used to perform testing there are mainly two types namely White Box Testing and Black Box Testing, and there is one type of testing known as Grey Box Testing.**

Keywords: Software testing, white box testing, Testing techniques, Unit Testing, Integration Testing, System Testing.

I. INTRODUCTION

One of the most crucial, important, high-rank steps of the software development life-cycle is software testing to perform. It can also be called testing is executing, exercising a program with the aim of detecting the errors before delivery to the end-user or customers. It is one of the box testing approaches to software testing. White box testing has a counterpart black box testing involves testing from an external or end-user way of perspective. White box testing in software engineering is based on the inner working of an application or software used in the internal testing types. White box testing term white box is known as the see-through box contain.

II. TYPE OF TESTING/LEVEL OF TESTING

In the Testing Field of a Software Development Life Cycle testing perform in many different ways used to improve the performance of software and help to prevent unnecessary crashes, delivering futile products and irrelevant bugs from the product/software. There are some types of software testing are:

A) Unit Testing

Unit testing is the very basic testing term used for checking the smallest part of a code is performing correctly or not. Unit testing performs by a developer by himself before the setup or assembling of the units' part of the software. This testing is also called the first level of testing before any type of testing.

B) Integration Testing

Integration Testing is the testing process where two or more parts of testing are integrated into each other and check whether they work together and cooperate after combining them. Integration testing is mainly focusing on the interfaces specified in the low-level design of the software. This testing is also known as the second level of the software testing process, following unit testing.

In this testing, the integration tester uses different types of testing methods first the Bottom-up Integration Testing method where the their approach is that bottom-level units are tested first then they go to the higher-level unit to perform testing step by step.

And another is the Top-down Integration Testing method where the modules are tested from the high level followed up by the low-level modules, And the last approach is the Big-Bang testing method, where all or most of the units are combined and tested simultaneously. This approach is taken when the testing team receives the entire software in a bundle.

Note: In a comprehensive software development environment, bottom-up testing is usually done first, followed by top-down testing.

C) System Testing

In the system testing process, the integrated software is tested. After all the components are integrated, the application as a whole is tested to see that it meets the specified quality standards to full fill the customers'/users demand. System testing reveals that the system works end-to-end production-like uses to provide the business function specified in the high-level design. In this, we just focus on the required input and output without focusing on the internal work also we have to do security testing, stress testing, and performance testing to ensure that the software is ready or not to deliver to the end user.

III. WHITE BOX TESTING

The testing approach of software testing consists of black-box testing and white-box testing. We are discussing here white box testing which is also known as glass box or structural testing, clear box testing, open box testing, and

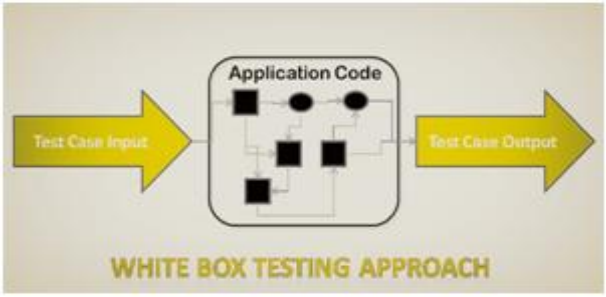
International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 6, Issue 10, pp 142-144, October-2022

https://doi.org/10.47001/IRJIET/2022.610028

transparent box testing. The testing done by the developer himself checks whether the code is working properly.



What Is White Box Testing?

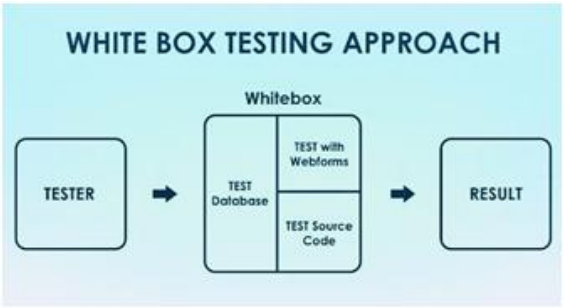
White box testing or clear box testing is the one where the internal structure and components of the application are exposed to the tester to see through the code. The software developer does a bit of glass testing at his end too while performing unit testing. A software tester who wants to perform this testing needs to have good knowledge of the coding language and logic to be efficient.

In this software testing method, internal details, and the structure of the system are made visible. This type of testing is highly efficient in detecting and resolving problems because bugs can often be found before they cause trouble and bugs. We can thus define this method as testing software with the knowledge of its internal structure and coding. White box testing is also called clear box testing, white box analysis, or clear box analysis. It is a strategy for finding errors in which thetester has complete knowledge of how the program components interact. In this method of testing, the test cases are calculated based on an analysis of the internal structure of the system based on code coverage, branch coverage, paths coverage, condition coverage, etc. It involves several features such as

1) Tester has full knowledge of the internal working of the software
2) Data domains and internal boundaries can be easily tested
3) Best suited for Algorithm testing
4) Mostly done by testers and developers
5) Granularity is high

Various Names of white Box Testing

1) Clear box Testing
2) Open box Testing
3) Glass box Testing
4) Transparent Box testing
5) Code-based Testing
6) Structural Testing
7) Logic-driven Testing



Advantages of White Box Testing

1) All the possible conditions are considered and test cases are generated. Hence all the functionalities are being tested.
2) Testing can commence even before the GUI is ready.
3) It is executed at different levels such as system, integration, and unit level of software development.
4) It involves testing a series of predefined inputs against expected or desired output so that specified input does not result in the expected output. The bug was obtained, and works on those bugs.
5) It helps to optimize the code well and helps to remove extra or unnecessary code it may contain bugs or errors that may contain hidden defects.
6) Due to the tester's knowledge of the code, the maximum coverage is attained during test scenario writing.

Disadvantages of White Box Testing

1) In this type of testing requires intimate knowledge of coding, targeted system testing tools, and coding languages.
2) It requires specialized tools such as source code analyzers, debuggers, and injector type's tools to check the system/software.

IV. CONCLUSION

White box testing in software testing should be done on a software application as it is being developed after it is written and again after each. White box testing can be quite complex. The complexity involved has a lot to do with the application being tested. A small application that performs a single simple operation could be written box tested in a few minutes, while larger programming applications take days, weeks, and even longer to fully test.

White box testing helps the tester find bugs or errors in the application at the unit, integration, and system levels. It helps to detect optimum loops, and the code is perfect with fewer lines without any extra looping works under the specific area of a function. And helps to verify the logical conditions and their values with all dependent and independent paths

ISSN (online): 2581-3048
Volume 6, Issue 10, pp 142-144, October-2022
https://doi.org/10.47001/IRJIET/2022.610028

within the module and functions using the different techniques available under those testing techniques. Test cases are a waste if changes in the implementation code are done frequently.

REFERENCES

[1] [Eckel] B. Eckel, Thinking in Java, Third Edition, ISBN 0-13-100287-2, Prentice Hall, 2003.

[2] [Frankel] D. Frankel, Model Driven Architecture, ISBN 0-471-31920-1, Wiley, 2003.

[3] "A path-oriented automatic random testing based on double constraint propagation" Ruilian Zhao, Yuandong Huang International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.2, March 2012.

[4] black box and white box testing techniques –a literature review, Srinivas Nidhra, and JagruthiDondeti, International Journal of Embedded Systems and Applications (IJESA) Vol.2, No.2, June 2012.

[5] White Box Coverage and Control Flow Graphs Venezia Elodie, 2011.

[6] M. Shaw, "What makes good research in software engineering?" International Journal on Software Tools for Technology Transfer (STTT), vol. 4, no. 1, pp. 1–7, 2002.

[7] M. R. Keyvanpour, H. Homayouni, and H. Shirazee, "Automatic Software Test Case Generation," Journal of Software Engineering, vol. 5, no. 3, pp. 91–101, Mar. 2011.

[8] M. Sharma and B. S. Chandra, "Automatic Generation of Test Suites from Decision Table - Theory and Implementation," in Software Engineering Advances (ICSEA), 2010 Fifth International Conference on, 2010, pp. 459 –464.

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

[10] H. Liu and H. B. Kuan Tan, "Covering code behavior on input validation in functional testing," Information and Software Technology, vol. 51, no. 2, pp. 546–553, Feb. 2009.

[11] P. Mitra, S. Chatterjee, and N. Ali, "Graphical analysis of MC/DC using automated software testing," in Electronics Computer Technology (ICECT), 2011 3rd International Conference on, 2011, vol. 3, pp. 145 –149.

[12] G. J. Myers, T. Badgett, T. M. Thomas, and C. Sandler, The Art of Software Testing. John Wiley & Sons, 2004.

[13] J. H. Hayes and A. J. Offutt, "Increased software reliability through input validation analysis and testing," in Software Reliability Engineering, 1999. Proceedings. 10th International Symposium on, 1999, pp. 199 –209.

[14] P. Jorgensen, Software testing: a craftsman's approach, CRC Press, 2002. p. 359.

[15] B. Beizer, Software Testing Techniques. Dreamtech Press, 2002.

[16] S. R. Rakitin, Software Verification, and Validation for Practitioners and Managers. Artech House, 2001.

[17] T. Murnane, K. Reed, and R. Hall, "On the Learnability of Two Representations of Equivalence Partitioning and Boundary Value Analysis," in Software Engineering Conference, 2007. ASWEC 2007. 18th Australian, 2007, pp. 274 –283.

[18] G. Scollo and S. Zecchini, "Architectural Unit Testing," Electronic Notes in Theoretical Computer Science, vol. 111, no. 0, pp. 27–52, Jan. 2005.

[19] Mohd. Ehmer Khan, F. K., 2012. "A Comparative Study of White Box, Black Box and Grey Box Testing Techniques". International Journal of Advanced Computer Science and Applications, 3(6).

[20] Harsh Bhasin, E. K., 2014." Black Box Testing based on Requirement Analysis and Design Specifications." International Journal of Computer Applications, 87(18), pp. 0975-88.

Citation of this Article:

Prof. S. B. Bele, Arjun A. Harinkhede, Vaibhav M. Bhatti, "Software Testing Using White Box" Published in *International Research Journal of Innovations in Engineering and Technology - IRJIET*, Volume 6, Issue 10, pp 142-144, October 2022. Article DOI https://doi.org/10.47001/IRJIET/2022.610028

****

