International Journal of Advances in Software Engineering & Research Methodology– IJSERM
Volume 1 : Issue 3 *[ISSN 2374– 1619]*
Publication Date : 30 September, 2014

Demystifying Black Box over White Box Testing

Ruhi Oberoi, Harpreet Oberai, Aniket Ramgiri

Abstract— **Within the automated testing world there are two predominating testing methodologies: black-box and white-box. This paper seeks to explore the pros and cons of both approaches and to identify when each approach should be used to ensure quality applications are delivered to market. In the end, this paper concludes that while black-box testing has its drawbacks in the past, innovative approaches to black-box testing makes it the likely choice to deal with the ever increasing complexity of applications and deliver lower Total Cost of Ownership (TCO) and a better Return on Investment (ROI) to organizations although it can be used with a degree of white box testing involved in it.**

Keywords— Black Box, White Box, Total Cost of Ownership, Return of Ownership.

I. Introduction

A. *Black-box*

This testing methodology looks at what are the available inputs for an application and what is the expected output. It is not concerned with the inner workings of the application, the process that the application undertakes to achieve a particular output or any other internal aspect of the application that may be involved in the transformation of an input into an output. Most black-box testing tools employ either coordinate based interaction with the applications graphical user interface (GUI) or image recognition. An example of a black- box system would be a search engine. You enter text that you want to search for in the search bar, press "Search" and results are returned to you. In such a case, you do not know or see the specific process that is being employed to obtain the search result. One simply provides an input – a search term – and receives an output – for the search results. [1]

B. *White- box*

This testing methodology looks under the cover and into the subsystem of an application. Whereas black-box testing concerns itself exclusively with the inputs and outputs of an application, white-box testing enables you to see what is happening inside the application. White-box testing provides a degree of sophistication that is not available with black -box testing as the tester is able to refer to and interact with the objects that comprise an application rather than only having access to the user interface. An example of a white-box system would be in-circuit testing where someone is looking at the interconnections between each component and verifying that each internal connection is working properly. Another example from a different field might be an auto-mechanic who looks at the inner-working of a car to ensure that all of the individual parts are working correctly to ensure the car drives properly [2].In white box testing one can be sure that all parts through the test objects are properly executed. Some synonyms for white box testing are [5]:

- Design Based Testing
- Open Box Testing
- Transparent Box Testing
- Clear Box Testing
- Glass Box Testing
- Structural Testing

Some important types of white box testing techniques are [5]:

- Control Flow Testing
- Branch Testing
- Path Testing
- Data flow Testing
- Loop Testing

There are some pros & cons of white box testing-

1) Pros

- Side effects are beneficial.
- Errors in hidden codes are revealed.
- Approximate the partitioning done by execution equivalence.
- Developer carefully gives reason about implementation.

2) Cons-

- It is very expensive.
- Missed out the cases omitted in the code.

The main difference between black-box and white-box testing is the areas on which they choose to focus. In simple terms,

Ruhi Oberoi
Assistant Professor, Jawaharlal Nehru Engineering College
Aurangabad, India

Harpreet Oberai
BFSI Practice, Zensar
Pune, India

Aniket Ramgiri
SME, Amdocs
Pune, India

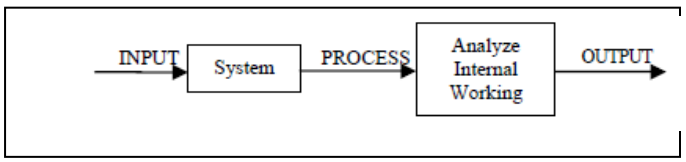

Figure 1. Working Process for White Box Testing

black-box testing is focused on results. If an action is taken and it produces the desired result then the process that was actually used to achieve that outcome is irrelevant. White- box testing, on the other hand, is concerned with the details. It focuses on the internal workings of a system and only when all avenues have been tested and the sum of an application's parts can be shown to be contributing to the whole is testing complete [2].

II. Different Types Of Testing

There are several advantages to black-box testing. A few of the most commonly cited are listed below:

A. *Pros*

1) **Unbiased:** Since the tester and developer are independent of each other, the testing conducted is balanced and unprejudiced. This makes it easier to identify vagueness and contradictions in the functional specifications since each specification is analyzed objectively
2) **Tester can be non-technical**: This is because Black box testing has no concern with the programming language or the details of implementation of the software and is saved from having a detailed knowledge of the functioning of the system
3) **Tests are reproducible**: The invested efforts to test the functional specifications can be used multiple times. This is of great help especially when testing large scale systems and cuts down on a plethora of repetitive tasks.

B. *Cons*

Even though Black box testing has several advantages, several drawbacks came to light when attempts were made to create black box test systems, thus resulting in the feasibility of black box testing approach to be questioned. Some of the most commonly cited issues were:

1) **Script maintenance**: Black Box tools rely on the method where the input provided is consistent. Though using an image-based approach to testing is advantageous, there's a possibility that the user interface may undergo changes constantly. In the wake of such an event, Script maintenance becomes very tedious and a difficult task.
2) **Fragility**: The test scripts are left fragile when interacted with GUI. The reason for the occurrence is that the GUI may not be rendered consistently at regular intervals when deployed on different hardware platforms or machines. The frequency of failure of test scripts is high, unless the tool in use possesses the ability of dealing with differences in rendering of the GUI.
3) **Lack of introspection**: It's a great irony that one of the greatest criticisms of black-box testing it isn't like white-box testing; that it only analyses the behavior and ignores the structural aspects which are dealt with in white-box testing. Due to this reason, we can never quite test a system completely, for many program paths may remain untested.
4) **Necessary vs. Luxury**: When employing automated testing methods, it is important to understand what is necessary vs. what is a Luxury. Both testing methodologies have some merit to their credit.

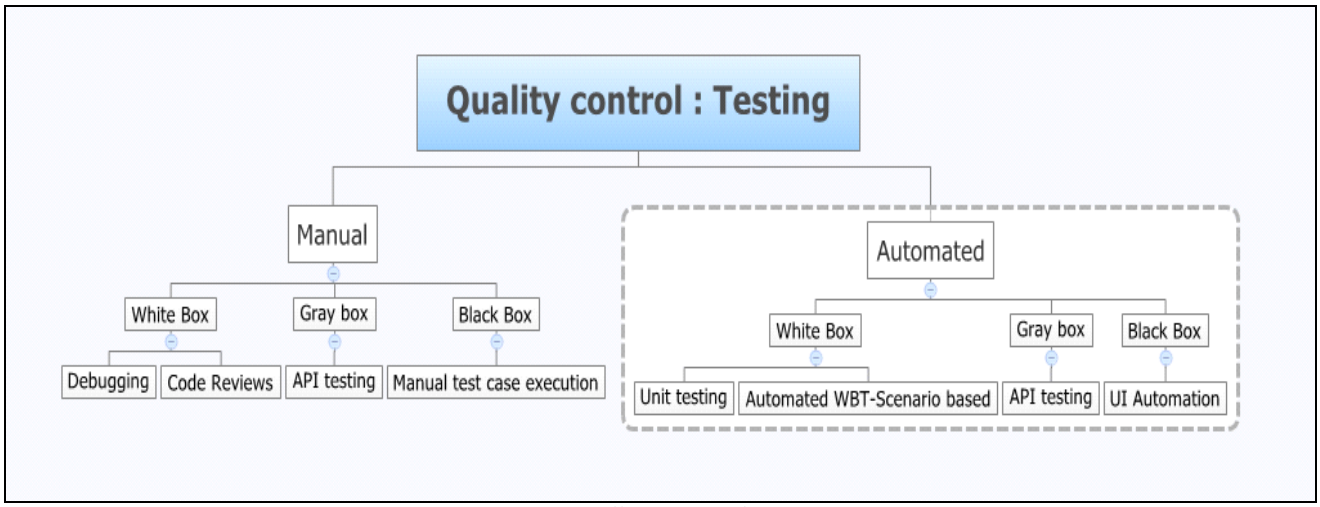

Figure 2. Different Types of Testing

To help one understand what approach should likely be used there are a few questions that every company should try to find the answers to:

- Who will be the primary user of the application?
- What parts of the application must be tested before deployment, and why?
- What programming language will my application be coded in?
- When can one expect noticeable changes to the GUI?
- Will the changes affect the underlying code?
- In what areas is the application likely to be used?
- How will the application be put to use?
- Which platforms does the application need to run on?

Looking at each of these questions, some of the possible answers that one may encounter are listed below:

- End users
- Any part of the application that the end user is likely to be exposed to
- Python using Django framework
- Bi-Annually because our customers are always expecting a refreshing experience
- Unless there are drastic changes, the underlying code will be unaffected
- On the cloud
- Customers will login via the web and will enter information through a series of screens
- Web browsers like Firefox, IE, Opera and Safari

When we analyze the answers to the questions listed above, it is possible for us to make an educated guess and pick the tool which would be ideal for use. Based on the answers given one can make an educated decision around the type of tool that should be used. If we consider the answers to the question listed above, then for the application, a black-box tool would be ideal since it has a customer-centered approach and focuses on testing the GUI rather than the code of the application. Moreover, a black box tool will possibly support multiple platforms which are required in the application described above.

On reviewing the questions suggested above does expose a more fundamental question related to the nature of testing itself. Why is it necessary to have an application tested? Even though the question looks silly, it makes the long-term viability of the two approaches being explored herein clairvoyant.

All companies test their applications before deploying because their users do not tolerate bugs. So, the main reason to carry out testing is to satisfy the customer. Given this, It is obvious that testing should be conducted in a manner which is fixated on analyzing how a customer will use an application. If we agree with this statement, unless and until black-box testing is applied on an application, testing cannot be said to be completed.

This argument is highly unlikely to be taken in good spirits by the white-box tool vendors and distributors and is likely to be criticized. It does bring up an interesting point nevertheless. Let us introspect on what we know. Customers interact with the GUI and do not deal with code generally. They interact with the application by entering details as input in some fashion and wait for the outcome to be displayed as output. If the application processes the details successfully, they get an acknowledgement which informs them of a favorable result, making them happy. If not, they find it problematic. This is similar to the approach followed by black-box testing tools. Hence, it would be logical that black-box tool be employed on the application. This doesn't mean that white-box testing tools don't have a place in the testing life cycle at all, but it touches on the point that they are ill-equipped to provide sufficient coverage and an organization cannot declare an application to be fully tested simply by employing white-box tools.

The previous statement is likely to draw flak from White-box tool vendors who would object strongly. They would present a case explaining that their introspective capabilities are superior when compared to a black-box testing approach. However, a white-box vendor cannot guarantee a consistent UI which properly reflects the code of the application, whereas the converse can be said to be true by a black-box vendor. A reason why this may happen is that one of the properties of an object might be set to "visible" but due to an error in a different part of the application or even due to a problem outside it, the object might not be visible on the GUI when viewed by the user. A white-box tool would log a change in state and the record would state the change as having passed. This is because the property of visibility was successfully updated. It is not equipped enough to find out whether the user is able to see the object or not, and the possibility of such an occurrence happening is largely overlooked. Even after the introduction of image-recognizing elements which tackle the problem of visibility of an object highlighted above, white-box tools are unable to deal with frequent differences encountered while rendering an image.
Even though some shortcomings of white-box testing are mentioned above, it shouldn't lead to a decisive conclusion which states that black-box testing is a comprehensive coverage solution to test an application. On the contrary, what is clairvoyant at this stage is that only after combining white-box and black-box testing tools do we achieve an optimum level of comprehensive test coverage. The reason in favour of this statement is that both the methodologies are related to

varying aspects involved in testing an application. For black-box testing, the focus is on the appearance of the GUI, which should be consistent enough to satisfy the user, providing him a refreshing experience. Compared to this, the primary focus of white-box testing is to deal with the internal structure of the system and ensure that the application works as smoothly and efficiently as possible based on the design. Due to this reason, these two methodologies can be treated as complimentary to each other. Organizations that have funds and resources available are strongly encouraged to take advantage of both.

III. Black-box Testing: A Necessary Step

So far this position piece has compared and contrasted black-box approaches related to testing: On analyzing the points, it has been observed that in black-box testing, the emphasis is on the end user; and that undertaking black- box testing is the best way of ensuring that the parts related to the interface will work as planned. This point is stressed upon because the user will spend a majority of his time interacting with the GUI. On combining with the simplicity in terms of using it with large scale systems, quicker test case development and simplicity, black-box testing represents a lower cost related to the initial stages than white-box testing and delivers Return of investment in a shorter period of time.

The budgetary and time constraints faced by organizations are well known and we can state that black-box testing is necessary whereas white-box testing is a luxury in the quality assurance process.

Once you've compiled this program, you'll use it to search the following 3x3 matrix for some number. You are to carry out black-box testing of this program, beginning with the matrix [4]:

| 45 77 93 |

| 78 79 85 |

| 72 96 77 |

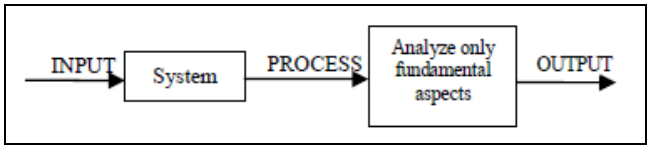

Figure 3. Explaining the working process for Black Box Testing

- Search for the value 77. What output is produced?
- Search for the value 99. What output is produced?

Try searching for other values and/or with another matrix. Try enough to show that your testing has turned up an error. Record what input you used and the resulting output. TABLE I. shows a sample of inputs used and resulting output.

Approach: Black box testing must be implemented wherever the software has a tangible / testable output.
A degree of white box should be implemented depending on the project complexity and Risk.
Project Complexity deals with interdependent components (modules) and integration points (interfaces) in the project.

A. *Example for project complexity:*

1) **Low**: Informative website with a majority of its content being static and involving mechanism involving feedback or comments.
2) **Medium**: System which provides a wide range of services and involving processing on the cloud, such as an online compiler for multiple programming languages.
3) **Complex**: Logistical Analytics related application which deals with a huge amount of data, An Enterprise application which consisting of several sub products which are tightly integrated.

B. *Example for integration point impact:*

If there are two components- output of 1 is input to the other. First has 5 conditions and 5 outcomes second one is internal and takes 2 inputs from DB – Black Box may need 5*2 = 10 cases to provide complete test coverage however as White Box has knowledge of internal systems- can test component 1 with 5 cases and component 2 with only 2 cases – also making sure the internal logic developed is correct as it will be reviewed.

TABLE I. Sample Testing Scenario with inputs used and resulting outputs.

TEST_ ID	INP UT	EXPECTED _OUTPUT	ACTUAL_ OUTPUT	STA TUS
MAT_01	77	TRUE	TRUE	PASS
MAT_02	99	FALSE	FALSE	PASS
MAT_03	85	TRUE	TRUE	PASS

IV. Conclusion:

While black-box testing has had its ups and downs in the past, recent innovative approaches to black-box testing try to solve its shortcomings, making it the likely choice to deal with the ever increasing complexity of applications and deliver a lower Total Cost of Ownership (TCO) and a better Return on Investment (ROI) to organizations. It can be used whenever there is a tangible/testable output involved along with white-box testing involved to a certain extent, depending on the risk involved as well as the complexity of the project.

Acknowledgment

We sincerely thank everyone associated with this work.

References

[1] Pressman R.,"Software Engineering, A Practitioners Approach",6th Edition ,Tata McGraw Hill Publication,2004,ISBN 007-1240837.
[2] Pankaj Jalote,"Software Engineering",Narosa Publishing House.``
[3] Furquan Naseer, Shafiq ur Rehman and Khalid Hussain.-Using Meta-data technique for component based Black Box Testing\\6th Internationalconference on emerging Technologies[ICET]2010,IEEE
[4] Ying jiang,Yin-a-Li and Xiao-Dong Fu.-The Support of InterfaceSpecifications in Black-box Component Testing.Fifth International Conference on Frontier of Computer Science and Tecnology 2010.IEEE,2010,p.305-311.
[5] Mohd.Ehmer Khan,"Different Forms of Software Testing Techniques for Finding Errors,"IJCSI,Vol.7,Issue 3,No 1,opp 11-16,May 2010.
[6] Mohd.Ehmer Khan,A ComparativeStudy og White Box,Black Box and Grey Box Testing Techniques."International Journal of Advanced Computer Science and Applications,Vol.3,No.6,2012.

About Authors:

Ruhi Oberoi has been working with Jawaharlal Nehru Engineering College as Assistant Professor since last 10 Years. Her profile includes testing, graphical passwords, data warehouse modeling and data warehouse testing. She has completed her M.E. from Government college of Engineering College, Aurangabad.

Aniket Ramgiri has his interest in testing, graphical passwords, data warehouse modeling and data warehouse testing. He has completed his B.E. from Jawaharlal Nehru Engineering College, Aurangabad.

Harpreet Oberai has been working with Zensar Technologies Ltd and has total of 11 years of experience in BFSI domain. His profile includes Project Management, requirement gathering, data modelling, testing and estimation. He has completed his B.E. from Bharati Vidyapeeth COE, Navi Mumbai and MMS from IES Management College,Mumbai.