Journal of Basic and Applied Engineering Research
Print ISSN: 2350-0077; Online ISSN: 2350-0255; Volume 1, Number 8; October, 2014 pp. 62-66
© Krishi Sanskriti Publications
http://www.krishisanskriti.org/jbaer.html

Gray Box Testing: Proactive Methodology for the Future Design of Test Cases to Reduce Overall System Cost

Ruhi Saxena[1], **Monika Singh**[2]

[1, 2]*Faculty of Engineering, Mody University of Science and Technology (MUST)*

Abstract: **Software testing is a highly complex and time consuming activity & it is exhaustive. Grey box is a technique to test the application with limited knowledge of the internal working of an application and also has the knowledge of fundamental aspects of the system. Gray box testing is a powerful idea if one knows something about how the product works on the inside; one can test it better, even from the outside. This paper is focused on gray box testing & its different techniques plus applications.**

1. INTRODUCTION

Software testing is a process of *verifying* and *validating* that a software application or program meets the business and technical requirements that guided its design and development, and works as expected. Software testing also identifies important *defects*, flaws, or errors in the application code that must be fixed.

Software testing has three main purposes: verification, validation, and defect finding [4].

- The *verification* process confirms that the software meets its technical specifications. A "specification" is a description of a function in terms of a measurable output value given a specific input value under specific preconditions. A simple specification may be along the line of "a SQL query retrieving data for a single account against the multi-month account-summary table must return these eight fields <list> ordered by month within 3 seconds of submission."
- The *validation* process confirms that the software meets the business requirements. A simple example of a business requirement is "After choosing a branch office name, information about the branch's customer account managers will appear in a new window. The window will present manager identification and summary information about each manager's customer base: <list of data elements>." Other requirements provide details on how the data will be summarized, formatted and displayed.
- A *defect* is a variance between the expected and actual result. The defect's ultimate source may be traced to a fault introduced in the specification, design, or development (coding) phases.[2]

The Attributes according to testing strategies & testing levels are shown below:

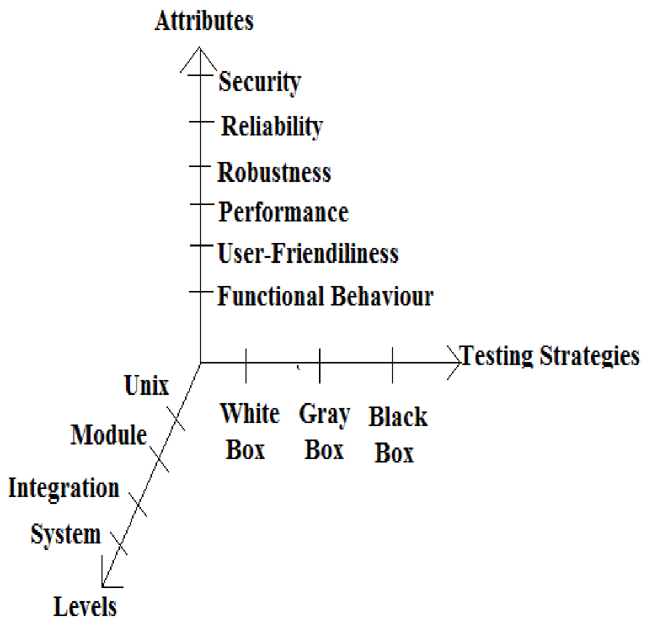

Fig. 1. 1: Software Testing Dimensions

1.1 Testing strategies:

1.1.1 White box testing: White-box testing is also known as structural testing, clear box testing, and glass box testing. White box testing involves looking at the structure of the code. When the internal structure of a product is known, tests can be conducted to ensure that the internal operations performed according to the specification and all internal components have been adequately exercised.

Types of White-Box Testing:

- Code coverage
- Segment coverage
- Branch Coverage or Node Testing
- Compound Condition Coverage
- Basis Path Testing
- Data Flow Testing (DFT)
- Path Testing
- Loop Testing

1.1.2 Black Box Testing: Black-box testing is also known as functional testing. It treats the system as a “black-box”, so it doesn’t explicitly use knowledge of the internal structure or code & test engineer need not know the internal working of the “Black box” or application; the main focus is on the functionality of the system as a whole [1][3].

Types of Black box Testing:

- Graph Based Testing Methods
- Error Guessing
- Boundary Value Analysis
- Equivalence Partitioning
- Behavioral Testing
- Random Testing/Stochastic Testing
- Syntax Testing
- Stress Testing

1.1.3 Gray Box Testing: Gray testing is also known as translucent testing. Gray Box Testing is a software testing method which is a combination of black box testing method and white box testing method. It is a technique to test the application with limited knowledge of the internal workings of an application [7]. The aim of this testing is to search for the defects if any due to improper structure or improper usage of applications.

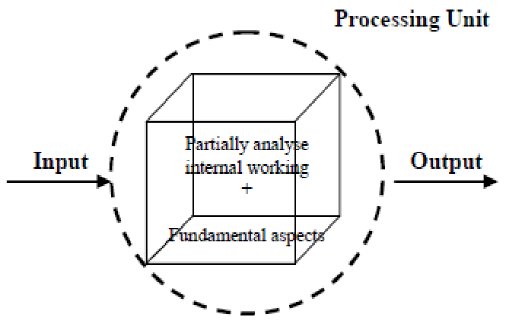

Fig. 1. 1. 3. 1: Gray Box testing

2. WHY WE NEED GRAY-BOX TESTING

There are certain problems with WBT & BBT listed below which are overcome by GBT:

2.1 Problems with White-Box Testing (WBT)

Following are the problems with white-box testing:

- Potentially most exhaustive of the three, because It is not possible to test each and every path of the loops in program.
- Since test cases are written on the code, specifications missed out in coding would not be revealed.
- Due to the fact that a skilled tester is needed to perform white box testing, the costs are increased.
- Sometimes it is impossible to look into every nook and corner to find out hidden errors that may create problems as many paths will go untested.
- It is difficult to maintain white box testing as the use of specialized tools like code analyzers and debugging tools are required.
- Internals fully known.

2.2 Problems with Black-Box Testing (BBT)

Following are the problems with white-box testing:

- Low granularity
- It can test only by trial and error
- The test inputs needs to be from large sample space.
- Blind Coverage- It is difficult to identify all possible inputs in limited testing time. So writing test cases is slow and difficult
- Chances of having unidentified paths during this testing
- The test can be redundant if the software designer has already run a test case.
- The test cases are difficult to design.
- Internals NOT known

2.3 Choosing Gray box testing over WBT and BBT based upon various parameters

Gray box testing (GBT) scores over WBT & BBT because:

- It would not suffer from deficiency as describe for WBT. It can test data domains, internal boundaries and over flow, if known.
- It factors in high level design environment and the inter operability conditions.

- It addresses problems that are not as easily considered by a black box or white box analysis, especially problems of end-to-end information flow and distributed hardware / software system configuration and compatibility.
- Context-specific errors that are germane to web systems are commonly uncovered in this process.
- It is platform & language independent.
- It will increase the testing coverage by focusing on all of the layers of any complex system through the combination of all existing white and black box techniques.

3. GRAY BOX TESTING

Gray Box testing is a technique to test the application with limited knowledge of the internal workings of an application. In software testing, the term *the more you know the better* carries a lot of weight when testing an application [3][7].

Unlike black box testing, where the tester only tests the application's user interface, in grey box testing, the tester has access to design documents and the database. Examples of grey box testing technique are:

- Architectural Model
- Unified Modeling language (UML)
- State Model (Finite State Machine)

Gray box testing encountered two types of issues:

- Test execution is in constant flow but content of the output is not correct. Somewhere in system, data is processed incorrectly and system thus generates error in showing results.
- Any failure due to uncertain reasons and thus process is aborted

Gray box Software Testing from Industry expert's point of views:

Industry experts have provided some definitions of gray box testing; few of them are given below.

Definition – 1: "Gray box testing consists of methods and tools derived from the knowledge of the application internals and the environment with which it interacts, that can be applied in black box testing to enhance testing productivity, bug finding and bug analyzing efficiency – by Nguyen H.G

Definition – 2: "Gray box testing is using inferred or incomplete structural or design information to expand or focus black box testing". - Dick Bende

Definition – 3: "Gray box testing is designing of the test cases based on the knowledge of algorithms interval states, architectures or other high level descriptions of program behavior". - Dong Hoffmar

Definition – 4: "Gray box testing involves inputs and outputs, but test design is educated by information about the code or the program operation of a kind that would normally be out of scope of view of the tester". - Cem Kanei

3.1 Gray Box Methodology

The Gray box methodology is a ten step process for testing computer software. The methodology starts by identifying all the input and output requirements to a computer system. This information is captured in the software requirements documentation.

The Gray box methodology utilizes automated software testing tools to facilitate the generation of test unique software. Module drivers and stubs are created by the toolset to relieve the software test engineer from having to manually generate this code.

Table 1: 10 Steps Graybox Methodology

Steps	Description
1	Identify Inputs
2	Identify Outputs
3	Identify Major Paths
4	Identify Subfunction (SF)X
5	Develop Inputs for SF X
6	Develop Outputs for SF X
7	Execute Test Case for SF X
8	Verify Correct Result for SF X
9	Repeat Steps 4:8 for other SF
10	Repeat Steps 7&8 for Regression

The toolset also verifies code coverage by instrumenting the test code. "Instrumentation tools help with the insertion of instrumentation code without incurring the bugs that would occur from manual instrumentation". By operating in a debugger or target emulator, the Graybox toolset controlled the operation of the test software. The Graybox methodology has provide us the way to let a debugger into the real world and into real-time. The methodology can be used in real-time by changing the basic premise that inputs can be sent to the test software via normal system messages and outputs are then tested using the system output messages [3].

Gray-box testing uses assertion methods to preset all the conditions required, prior to a program being tested. Testing using Formal specification languages is one of the commonly used techniques for ensuring that a core program is correct to a very large extent. If the requirement specification language is being used to specify the requirement, it would be easy to execute the requirement stated and verify its correctness. Gray-box testing will use the predicate and verifications defined in requirement specification language as inputs to the requirements based test case generation phase.

4. GRAY BOX TESTING TECHNIQUES

Different forms of grey box testing techniques are briefly described below:

- Matrix Testing
- Strategy for Gray box Regression testing
- Pattern Testing
- Orthogonal Array Testing

4.1 Matrix Testing

In matrix testing the status report of the project is stated. The idea of beginning your testing activities with a list of variables used in the software is not new. You may have heard the term CRUD (created, read, update and deleted) method. Basically it starts with developers defining all the variables that exist in their programs. Each variable will have an inherent technical risk, business risk and can be used with different frequency during its' life cycle.

4.2 Software Regression Testing

Regression testing -testing that is performed after making a functional improvement or repair to the program [6]. Its purpose is to determine if the change has regressed other aspects of the program. This can be accomplished by executing the following testing strategies:

- **Retest all:** Rerun all existing test cases
- **Retest Risky Use Cases:** Choose baseline tests to rerun by risk
- **Retest By Profile:** Choose baseline tests to rerun by allocating time in proportion to operational profile
- **Retest Changed Segment:** Choose baseline tests to rerun by comparing code changes. (White box strategy)
- **Retest within Firewall:** Choose baseline tests to rerun by analyzing dependencies. (White box strategy)

4.3 Pattern Testing

Pattern testing is best accomplished when historical data of previous system defects are analyzed. The analysis template will include specific reasons for the defect, which will require system analysis. Unlike black box testing where the types of failures are tracked, gray box testing digs within the code and determines why the failure happened. This information is extremely valuable, as future design of test cases will be proactive in finding the other failures before they hit production. The coding structure in place influences gray box test case design[9][10].

Analysis Template will include:

- The problem addressed
- Applicable situation
- The problem that can be discovered
- Related problems
- Solution for developers that can be implemented

New test cases will include:

- Security related test cases
- Business related test cases
- GUI related test cases
- Language related test cases
- Architecture related test cases
- Database related test cases
- Browser related test cases
- Operating System related test cases

4.4 Orthogonal Array Testing

Orthogonal Array Testing is a statistical testing technique implemented by Taguchi. This method is extremely valuable for testing complex applications and e-comm. products. The e-comm. world presents interesting challenges for test case design and testing coverage. The black box testing technique will not adequately provide sufficient testing coverage. The underlining infrastructure connections between servers and legacy systems will not be understood by the black box testing team. A gray box testing team will have the necessary knowledge and combined with the power of statistical testing, an elaborate testing net can be set-up and implemented.

The theory -Orthogonal Array Testing (OAT) can be used to reduce the number of combinations and provide maximum coverage with a minimum number of test cases. OAT is an array of values in which each column represents a variable - factor that can take a certain set of values called levels. Each row represents a test case. In OAT, the factors are combined pair-wise rather than representing all possible combinations of factors and levels.

5. ADVANTAGES & DISADVANTAGES

Some of the advantages & disadvantages of gray box testing are listed below:

5.1 Advantages:

- **Offers combined benefits**: It serves advantages from both black-box & white-box testing
- **Non-Intrusive:** Based on functional specification, architectural view whereas not on source code or binaries which makes it invasive too.
- **Intelligent test authoring**: Tester handles intelligent test scenario. Ex: data type handling, communication protocol, exception handling.
- **Unbiased Testing:** Gray-box testing maintains boundary for testing between and developer**.**

5.2 Disadvantages

- **Partial code coverage:** Source code or binaries are missing because of limited access to internal or structure of the applications results in limited access for code path traversal.
- **Defect Identification:** In distributed application, it is difficult to associate defect identification.

6. APPLICATIONS OF GRAY BOX TESTING

Gray box testing is well suited for Web Applications. Web applications have distributed network or systems; due to absence of source code or binaries it is not possible to use white box testing. Black box testing is also not used due to just contract between customer and developer, so it is more efficient to use gray box testing as significant information is available in Web Services Definition Language(WSDL)

Gray box testing is suited for functional or business domain testing. Functional testing is done which is basically a test of user interactions may be with external systems. As gray box testing can efficiently suits for functional testing due to its characteristics; it also helps to confirm that software meets the requirement. Functional testing due to its characteristics; it also helps to confirm that software meets the requirement [3][8].

7. CONCLUSION

By implementing gray box testing, the overall cost of system defects can be reduced and prevent more from passing the testing stage. If the development process tries to accumulate the benefits of black box and gray box then gray box testing is preferable due to moderate granularity. Gray box testing is well suited for Web applications, Web services, functional or business domain testing, security assessment, GUI, distributed environments, etc.

REFERENCES

[1] Boris Beizer, “*Software Testing Techniques”*, Van Nostrand Reinhold CompanyInc, New York, 1983.

[2] Cem Kaner, James Bach, Bret Pettichord, “*Lessons Learned in Software Testing”*, 2002.

[3] Boris Beizer, “*Black-Box Testing: Techniques for Functional Testing of Software and Systems”, 2003.*

[4] Jerry Gao, Raquel Espinoza, Jingsha He, "*Testing Coverage Analysis for Software Component Validation*", In Proceedings *29th Annual International Computer Software and Applications Conference (COMPSAC’05)*, 2005.

[5] J. Gao, D. Gopinathan, Quan Mai, Jingsha He, “*A Systematic Regression Testing Method and Tool For Software Components*”. In *Proceedings 30th Annual International on Computer Software and Applications Conference*, IEEE Computer Society, September 2006, Volume 1.

[6] André Coulter, “*Graybox Software Testing Methodology – Embedded Software Testing Technique*”, In *proceedings 18th IEEE Digital Avionics Systems Conference*, 1999.

[7] Boris Beizer, “*Software System Testing and Quality Assurance”,* Van Nostrand Reinhold Company Inc, New York, 1984.

[8] P. C. Jorgensen, “*Software Testing: A Craftsman’s Approach”*, CRC Press, 2nd ed., 2002.

[9] M. Buchi and W. Weck., “*The Graybox approach: when blackbox specifications hide too much”*, Technical Report TUCS TR No. 297, Turku Centre for Computer Science, 1999.