Overview On Test Case Design Technique For Black Box Testing

Kanika Agarwal, Mr. Ashendra Saxena

Student, Teerthanker Mahaveer University, CCSIT, Moradabad

Assistant Professor,Teerthanker Mahaveer University, CCSIT, Moradabad

agarwalskani@gmail.com

ashendrasaxena@gmail.com

***Abstract*—** Software testing is done for analyzing software to find the difference between required and existing condition. In software development life cycle,Software testing is the most important and time consuming part. It's main purpose is to find software failures so that defects may be recovered and corrected in early phase.In this review paper I have explained one of the software testing technique i.e. Black Box Testing. It is a method of generating test cases that are independent of software internal structure, I have also briefly described various different techniques for finding errors in black box testing . Black box testing strategies play pivotal role in detecting possible defects in software and can help in successful completion of software according to functionality. Black box testing techniques are important to test the functionality of the software without knowing its inner detail which makes sure correct, consistent, complete and accurate behavior or function of a system.

***Keywords*—– Software Testing, Black Box Testing, Test Cases Techniques, Approaches Techniques.**

I. Introduction

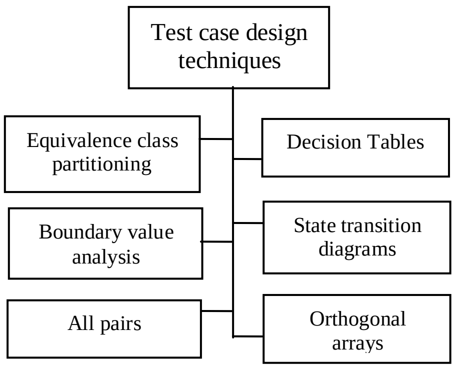
Black box testing is also called as behavioral testing, in which the interior structure, logic of software that is being tested is unknown to analyzer. This testing is based on requirement specification and it's not necessary to analyze code. It is basically performed under the end user point of view. It helps to identify incomplete and unpredictable specifications, so that they can be rectified later. Black box testing is done from beginning of software project development cycle. Testers need to gather end user requirements and based on that test scenarios have to be prepared. Black box testing strategies are used to test logical, data or behavioral dependencies, to generate test data and quality of test cases which have potential to guess more defects. Black box testing strategies play pivotal role to detect possible defects in system and can help in successful completion of system according to functionality. The studies of five companies regarding important black box testing strategies are

presented in this thesis. This study explores the black box testing techniques which are present in literature and practiced in industry as well. Black box testing have little or no knowledge to the internal logical structure of the system. Thus, it only examines the fundamental aspect of the system. It makes sure that all inputs are properly accepted and outputs are correctly produced.

1.1. Test Case Design-Need:

Just as code is designed and developed, test cases too must be designed and then written. Exhaustive testing of any non-trivial system is impractical. The input date domain is extremely large. Test case design is required to derive an optimal test suite which is of reasonable size and uncovers as many errors as possible. Randomly selecting or writing test cases does not indicate effectiveness of the testing. Writing a large number of test cases does not mean that many errors in the system would be uncovered.

I.2.Test case design techniques for black box testing:


Test case design techniques
Equivalence class partitioning
Decision Tables
Boundary value analysis
State transition diagrams
All pairs
Orthogonal arrays

Figure(I.2.a)

Figure above shows the different types of Black box testing techniques.

I.3. Advantages of Black box testing:

i. Tests are effective on large units of code then glass box testing
ii. Tester needs no knowledge of implementation, including specific programming languages.
iii. Tester and programmer are independent of each other.
iv. Testing is done from a user's point of view.
v. Will help to expose any ambiguities or inconsistencies in the specifications.
vi. Test cases can be designed as soon as the specifications are complete.

I.4. Disadvantages of Black box testing:-

i. Test cases are hard to design without clear specifications.
ii. Only small numbers of possible input can actually be tested.
iii. Some parts of the back end are not tested at all.
iv. Chances of having unidentified paths during the testing.
v. Chances of having repetition of tests that are already done by programmer.

II. VARIOUS BLACK BOX TESTING TECHNIQUES

II.1. <u>Equivalence Class Partitioning</u>**:** In this procedure information area of system is separated into equivalent classes from where experiments can be done. This reduces the quality of experiments. This procedure can be applied at any testing level and tests one condition from each class.Basically it is a method that divides the input data of a software unit into partitions of data from which test cases can be derived. It reduces no. of test cases. In equivalence class partitioning an equivalence class is formed of the inputs for which the behavior of the system is specified or expected to be similar. An equivalence class represents a set of valid or invalid states for input conditions. Typically, an input condition is either a specific numeric values, array of values, a set of related values or Boolean condition. Once we select equivalence classes for each of the input, the next issue is to select the test cases suitably.

➢ ***Approach*****:**
i. Divide the input domain into classes of data for which test cases can be generated.
ii. Attempting to uncover classes of error.s
iii. Divides the input domain of a program into classes of data.
iv. Derives test case based on these partitions.
v. An equivalence class is a set of valid or invalid states of input.
vi. Test case design is based on equivalence classes for an input domain.

➢ ***Advantages:***
i. It eliminates the need for exhaustive testing, which is not feasible.
ii. It enables a tester to cover a large domain of inputs or outputs with a smaller subset selected from an equivalence class.
iii. It enables a tester to select a subset of test inputs with a high probability of detecting a defect.

II.2. <u>Boundary Value Analysis:</u> This procedure mainly concentrates on boundaries or extreme boundaries values that are created in software that being tested. It also incorporates inside and outside limits Boundaries and conditions are two major sources of defect in software products. In boundary value analysis, typical programming errors occur at the boundaries of equivalence classes This may be purely due to psychological factors. Programmers often fall to see special processing required at the boundaries of equivalence classes or Programmers may improperly use < instead of <=.

➢ ***Approach*****:**
i. For a range of value bounded by a and b, test(a-precision value),a,(a+ precision value),(b- precision value),b,(b+ precision value)
ii. If input conditions specify a number of values n, test with(n-1),n and (n+1) input values.
iii. Apply 1 and 2 to output conditions(e.g., generate table of minimum and maximum size).
iv. If internal program data structures have boundaries(e.g.,buffer size, table limits), use input data to exercise structures on boundaries.

➢ ***Advantages:***
Boundaries and conditions are two major sources of defect in software products. This technique aims to identify defects in these areas.

II.3. <u>Decision tables:</u> Decision tables can be used when the outcome or the logic involved in the program is based on a set of decisions and rules which need to be followed. A decision table lists the various decision variables, the conditions(or values)assumed by each of the decision variables and the actions taken in each combination or conditions. Variables that contribute to the decision table are listed as the columns of the table. Last column of the table is the action to be taken for combination of values of the decision variables.

➢ ***Approach:***
The steps for using decision table testing are as given below:
Step1:
Analyze the given test inputs or requirements and list out the various conditions in the decision table.
Step2:
Calculate the number of possible combination(Rules)
Step3:

Fill columns of the decision table with all possible combinations(Rules)
Step4:
Find out cases where the values assumed by a variable are immaterial for a given combination. Fill the same by *don't care* symbol.
Step5:
For each of the combination of values, find out the action or expected result.
Step6:
Create at least one test case for each rule. If the rules are binary, a single test for each combination is probably sufficient. Else if a condition is a range of values, consider testing at both the low and high end of range.

Advantages:

i. Enables us to get a "Complete" view with no consideration of dependence
ii. Enables us to look at and consider "dependence," "impossible," and "not relevant" situations and eliminate some test cases.

II.4. <u>State transition based testing</u>: State is represented by a circle. Transition is represented by a label on the transition.Thus from the starting state to the end state the various transitions and routes are represented in the form of a transition diagram as mentioned. Create test cases in such a way that all states are visited at least once, all events are triggered at least once and all paths are executed at least once (I.e. all transitions in the system are tested at least once).

- ***<u>Approach</u>***:

The steps for using state transition testing are:
Step1:

a) Understand the various states that the system, user, or object can be in, including the initial and final states.
b) Examples of states can be:'User raising a purchase order' or 'leave request is accepted'. These states will be represented as:

Step 2:
Identify transitions, events, conditions, and actions that can- and can't- apply in each state.
Step 3:
Use a graph or table to model the system. This graph or table also serves as an oracle to predict correct system behavior along with a requirments specification.
Step 4:
For each event and condition- that is, each transition- verify that the correct action and next state occurs.
Step 5:
Create test cases in such a way that all states are visited at least once, all events are triggered at least oce and all paths are executed at least once(I.e. all transitions in the system are tested at least once)

- ***Advantages:***

i. All possible states and transitions in a system would be covered(including valid and invalid).
ii. Critical when testing high risk systems like Avionics or medical devices where testing of all possible states and transitions is required(not just valid ones).

II.5. <u>Orthogonal Array Testing:</u> This procedure used when number of inputs to software is comparatively small but too complex for carrying complete testing of every possible input to software. A device for selecting a "good" subset of all possible combinations. Too many combinations to consider. Risky to skip testing large parts of the functionality or combinations. So what is there as a compromise solution. All PAIRS(each option with every other option ONCE, but not all combinations across all options). Exercise multiple pairs simultaneously. Requires knowledge of all legitimate combinations.

- ***Approach:***

The steps for using orthogonal Array technique are:
Step1:
Analyze the given test inputs or requirements and list out the variables that needs to be tested for interaction.
Step2:
Determine the number of choices or values for each variable.
Step3: Locate an orthogonal array which has a column for each variable and values within the columns that correspond to the values for each variable.
Step4:
Map the variables with their values on to the orthogonal array.
Step5:
Each row in the table corresponds to a test condition or a unique test case.

- ***Advantages:***

i. Provide uniformly distributed coverage of the test domain.
ii. Concise test set with fewer test cases are created.
iii. All pair-wise combinations of test set are created.
iv. Simpler to generate and less error prone than test sets created manually
v. Reduces testing cycle time.

II.6. <u>All Pair Testing:</u> In this procedure all possible combinations of input parameters are designed and executed. Its main aim is to cover all possible inputs. Testing deals with validating the different values for all variables in the system. We generate test cases by pairing values of different variables.

- ***Approach:***

Step1:
List out the variables in the application to be tested and the various possible values each of the variables can hold.
Step2:
Combine or group the values where ever possible.

Step3:
Create the all pairs table by putting the variables in the Top row and start by filling in the values in each column.
Step4:
If a combination does not exist, then swap around with the values to see if the combination can be got.
Step5:
Else add a new row
Step6:
Each row in the table corresponds to a test case.

Advantages:

i. Significantly reduces the number of test cases.
ii. Pairwise testing protects against pairwise bugs which represent the majority of combinatorial bugs and such bugs are a lot more likely to happen than ones that only happen with more variables.
iii. Tools are available which can create the All pairs table automatically(and are no longer crated by hand).
iv. Efficiency(I.e., amount time and resources required to conduct testing) is improved because the much smaller pairwise test suite achieves the same level of coverage as larger combinatorial test suites.

III. Conclusion

Hence, Software testing is important, and testing techniques are too, because they have the main aim to improve and make easier this process. There is considerable controversy between Software testing writers and consultants about what is important in software testing and what constitutes responsible in software testing. First quality is main focus of any software engineering project. Without measuring, we cannot be sure of the level of quality in a software. So the methods of measuring the quality of software testing techniques. This paper relates various types of testing techniques that we apply in measuring various quality attributes. Software testing research is the driving element of development and application. In this era of new and higher demand of software testing, it is important to constantly summarize new achievements and propose different ideas about software testing.

References

[1] 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.
[2] F. Saglietti, N. Oster amd F. Pinte, ”White and grey-box verification and validation approaches for safety-and security- critical software systems,” Information Security Techincal Report, Vol. 13, no. 1, pp 10-16, 2008.
[3] 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
[4] Irena Jovanovic,”Software Testing Methods and Techniques”, IPSI BGD Internet Research Society, Vol. 5, No. 1, pp 30-41, January 2009.
[5] 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.
[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] Mohd. Ehmer Khan, “Difference Approaches to Black box testing technique for finding Errors”, IJSEA, Vol. 2, No. 4, pp 31-40, October 2011.
[8] P. Jorgensen, Software testing: a craftman’s approach, CRC Press, 2002. p. 359.
[9] S.Liu and Y.Chen ,“A Relation-based method combining functional and structiural testing for test case generation,” Journal of Systems and Software, Vol.81,No. 2,pp, 234-248, Feb.2008.
[10] S.M.K Quadri and shiek umar Farooq, “Software Testing- Goals and Techniques,” International Journal of Computer Applications (0975-8887) Vol. 6, No.9,September 2011.
[11] S.M.K Quadri and shiek umar Farooq,”Testing Techniques Selection: A Systematic Approach”, Proceedings of the 5th National Conference: INIACom-2011.pp-279-281, March 10-11, 2011.
[12] Shivkumar Hamukhrai Trivedi,”Software Testing Techniques”, International journal of Computer Science and Software Engineering. Vol. 2. Issue 10, October2012, ISSN:2277 128X
[13] Srinivas Nidhra and Jagruthi Dondeti, “Black box and White box techniques-A Literature review.”, International Journal of Embedded Systems and Applications, Vol 2, No. 2, June 2012.
[14] Surendra Singh Rathod, “A Scenario of Different types of Testing Techniques in Software Engineering”, International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 4, Issue 6, June 2014, ISSN:2277 128X.
[15] T. Murnane and K. Read “On the effectiveness of mutation analysis as a black box testing technique,” in software Engineering Conference. 2001. Proceedings. 2001 Australian. 2001, pp. 12-20.
[16] Taraq Hussain and Dr. Satyaveer Singh, “A Comparative Study of Software Testing Techniques Viz. White box testing Black box testing and Grey box testing”, International Journal of Allied Practice, Research and Review, Vol. 2, Issue 4, 2015, pp 26-33, ISSN:2350-1294.
[17] User Acceptance Testing available at “http://guide.agilealliance.org/guide/acceptance.html”
[18] User Acceptance Testing available at “https://www.technopedia.com/definition/3887/user-acceptance-testing-uat”