ISSN:0975-9646

Testing Approaches to Generate Automated Unit Test Cases

Parul Sharma, Neha Bajpai

Centre for Development of Advance Computing,
Guru Gobind Singh Indraprastha

Abstract: In Software Development Life Cycle software testing is very essential activity. It is used to find out the errors present within the application code. The manual testing is difficult and time consuming and it may be impossible for large system to test. There have been several methods proposed for the automatic generation of test cases. This paper explores various approaches to generate test cases.

Keywords: Software testing, Black Box Testing, White Box Testing, Test case Automation.

INTRODUCTION

To satisfy the specific requirement software testing is the process of exercising a system by manual or automated means. Software testing is an activity that should be done throughout the whole development process. As testing is a very important and expensive step in software development and maintenance, the 50 percent of the total time and 80 percent of the total cost is consumed only for testing. A challenging part of this phase involves the generation of test cases. A test case is a set of conditions performed in a sequence and related to a test objective, will produce a number of tests comprising specific input values, observed output, expected output, and any other information needed for the test to run, such as environment prerequisites [1]. A good test case should have the quality to cover more features of test objective. The techniques for the automatic generation of test cases try to efficiently find a small set of cases that allow a given criterion to be fulfilled, thus resulting in the decreased cost of software testing. Using automated unit test tools instead of manual testing can help with all three problems. Automated testing can reduce the time and effort needed to design and implement unit tests [2].Test cases can be derived from use cases, system requirements and directly mapped to. One of the great advantage of generating test cases from requirements and design is that they can be used before the construction of program. Detection and removal of errors in the early phase of life cycle will definitely bring down the cost of building the software systems. Software testing is classified mainly into two types i.e. Black box & White box. Black box testing focuses mainly on the outputs generated in response to selected inputs and execution conditions. In this testing technique knowledge of the internal functionality and structure of the system is not available. White box testing focuses on the internal structure of the software code. The white box tester knows what the code looks like and writes test cases by executing methods with certain parameters. Using white box testing methods, the software tester generates test cases that confirm all independent paths within a module have been exercised atleast once. All logical decisions are checked on their true and false conditions and exercises all loops at their boundaries and within their boundary range. On the other side White Box Testing has some disadvantages like it brings complexity to testing while testing every important aspect of the program. Further one must have complete understanding of the program because in several test cases few test conditions may be untested.

LITERATURE REVIEW

1. White on Black: A White-Box-Oriented Approach for Selecting Black-Box-Generated Test Cases

This paper [3] describes that many useful test case construction methods that are based on important aspects of the specification have been proposed in the literature. A test suite hence obtained is often very large and is non-redundant with respect to the aspects identified from the specification. This paper points out on the selection of relevant test cases from the existing test suite and the use of white box criteria to select test cases from the initial black-box-generated test suite. The approach involves the following important steps:

(1) Obtain an initial test suite which is complete and contains no redundant test case according to a black box criterion.
(2) Choose a white box criterion to be used in step (3) .
(3) Select a classification (called candidate classification) X and two of its classes (called candidate classes) x1 and x2 that are expected to be CW-equivalent (Test cases that are considered to be processed similarly by a white box criterion are said to be CW-equivalent).
(4) Identify all matching pairs with individualize classes x1 and x2 from the initial test suite.
(5) Check the results from the implementation if all the identified matching pairs are CW-equivalent or not. If so, select only one test case from every such matching pair. Otherwise, all of the test cases are reserved
(6) Repeat steps (3) to (5) if appropriate.

2. A Graphical Class Representation for Integrated Black- and White-Box Testing

This paper [4] talks about that when both black- and white-box testing have the same aim, namely detecting faults in a program, they are often handled separately. Reason is the lack of techniques and tools integrating both approaches, although integration of both can considerably decrease testing costs. An integrated technique can generate a reduced test suite, as single test case can cover both requirement specification and implementation at the same time. A new graphical representation of classes has been proposed, which can be used for integrated black and white-box testing. This approach is different from others in the scenario that each method of a class is shown from two perspectives, namely the requirement specification and implementation view. Both the perspectives are represented in the form of control flow graphs. On the other hand for the adjustment of white box test cases to black box test cases a reduction technique has been proposed. An integrated technique supported by a single tool can have several benefits like less maintenance effort is required and tester has to be friendly with concepts of one tool.

3. Analysis of Black Box Software Testing Techniques: A Case Study

This paper [5] compares the two technique of black box testing i.e. boundary value analysis and robustness technique. For this they considered the **line equation problem** and generated different test cases using both the techniques and finally concluded that robustness technique is better than boundary value analysis.

Consider a **straight line problem** with (m1,c1) and (m2,c2) defining the lines of the form y=mx+c with following conditions

i. Parallel lines (m1=m2, c1=/c2)
ii. Intersecting lines (m1=/m2)
iii. Coincidental lines (m1=m2, c1=c2)

They have summarized their results in the following table and boundary range assigned was from 0 to 100:

S.No	**Type Of Test**	**No. Of Test Cases**
1	BVA	17
2	Robustness Testing	25
TYPE OF LINE	BVA	Robustness Testing
Intersecting Lines	8	12
Parallel Lines	8	12
Coincidental Lines	1	1

4. Boundary Value Analysis using Divide-and-Rule Approach

This paper [6] talks about a new boundary value analysis algorithm using a divide and rule approach. This new algorithm was proposed because of the functional dependency among the input parameters . This new algorithm generates some necessary test cases that couldn't be generated using the traditional methods.

Characteristics of divide and rule approach:

- breaks the dependencies among the variables
- produces multiple independent sets of variables
- assures that the variables in the new sets do not influence each other's boundary values.

Comparison with traditional boundary value analysis:

The limitation with the traditional approach is that it does not cover cases that the parameters have dependency relationship [8]. BVA fails to generate sufficient test cases when there are dependencies among the variables. In the NextDate function the variables Month and Day have the boundaries 1-12 and 1-31 respectively. Therefore, the dates 28th February and 29th February are never generated as test cases. There is also no stress on the leap years. New BVA algorithm using divide and rule approach explained with NEXTDATE EXAMPLE as follows:

The function has three variables : month, day, year where boundary values for day depends on values assumed by month and year.

D= set of dependent variable, B = set of boundary determining variable, I = set of independent variable

Step 1 D = {Day};B = {Month; Year}; I = {null}.

Step 2 $V_{B,Day}$ = {Month; Year}.

Step 3 Day_1 = {1 : : : 31}, Day_2 = {1 : : : 30}, Day_3 = {1 : : : 28}, Day_4 = {1 : : : 29}.

Step 4 $Month_1$ = {1; 3; 5; 7; 8; 10; 12} (31 days),
$Month_2$ = {2} (28 or 29 days),
$Month_3$ = {4; 6; 9; 11} (30 days).
$Year_1$ = {1904; 1908; : : : ; 2000}(Leap Years),
$Year_2$ = {1900; 1901; 1902; 1903; 1905; :; 1999} (Non-Leap Years)

Step 5 Prepare all possible combinations of boundary determining sets for every dependent variable. NextDate function has only one dependent variable Day. It has the following Determining Value Sets:
$Month_1Year_1$; $Month_1Year_2$; $Month_2Year_1$; $Month_2Year_2$; $Month_3Year_1$; $Month_3Year_2$;

Step 6 Assign a boundary value set to every determining set combination:

Determining Set	Boundary Value Set
$Month_1$ $Year_1$	Day_1
$Month_1$ $Year_2$	Day_1
$Month_2$ $Year_1$	Day_4
$Month_2$ $Year_2$	Day_3
$Month_3$ $Year_1$	Day_2
$Month_3$ $Year_2$	Day_2

Step 7 Possible combinations for all Determining Sets are same as listed in Step 5.

Step 8 To every combination from Step 7, assign it boundary-value sets. Following sets are formed:
$Month_1Year_1Day_1$; $Month_1Year_2Day_1$;
$Month_2Year_1Day_4$; $Month_2Year_2Day_3$;
$Month_3Year_1Day_2$; $Month_3Year_2Day_2$;

Step 9 This step is skipped since there are no independent variables.

Step 10 Generate test cases.

5. An Improved Algorithm for Basis Path Testing

Basis path testing [7] is an important and wildly applied white box testing technology. It is based on the cyclomatic complexity and always generates the basis path set through the baseline method. This testing fully depends on the CFG of the program, some basis paths we get are feasible in mathematics but inexecutable in the actual test. To overcome this condition, they proposed an improved method that when generating the CFG, connect the causal paths of the two series judgment parts and omit the intermediate nodes. For this they constructed a correspondence between the premise and the following branch, to make sure that every basis path is feasible. Under the condition that the results of the previous judge fragment are the premise of the next judgment parts, this method makes sense.

ANALYSIS

The table [1] provides the comparative analysis of the various approaches. From this we can easily understand the importance of each approach.

S.No	Approach	Advantage	Disadvantage	Conclusion
1	White on Black [3]	Selecting a Subset of test cases from a test suite Substantial amount of testing effort can be saved by using this approach	Difficult to select a proper subset of test cases from given test suite	Uses white box approach to select test cases from the black box generated initial test suite because if any other black box approach is considered it could have already been taken into account and secondly white box is complementary to black box testing.
2	Integration of Black Box & White Box Testing at Class Level [4]	Class specification & implementation graph generation algorithm has been enhanced with a test suite reduction algorithm to allow the generation of test cases.	Requires a deep knowledge of both white box and black box testing techniques for their integration	Generates a reduced test suite, as single test case can cover both specification and implementation at the same time.
3	Comparison of Boundary value analysis & Robustness testing [5]	Test cases are derived on the basis of values that lie on boundary/edges.	Boundary value doesn't check for values that lie outside the range. It only considers valid values and doesn't check for invalid values.	Robustness testing gives better results, we considered a line equation and generated test cases for both. BVA generated 17 test cases while Robustness testing generated 25 test case covering both valid and invalid values.
4	Divide and Rule Approach [6]	Breaks dependency among variables and produces multiple independent sets of variables.	Boundary Value Analysis has limitation that it doesn't support dependency among variables	New enhanced divide and rule approach overcome this limitation by dividing the variables according to their type into different independent sets.
5	Basis Path testing [7]	Provides test coverage metric and avoids infeasible paths	Finding out feasible path is difficult task as there are many infeasible path exists within a code	Executes each feasible path and how it can be used to detect errors within a piece of code

Table 1. Comparative Analysis

CONCLUSION

This study exhibits a clear picture about various test case generation approaches. The working principal of each approach is explained and their advantages and disadvantages are tabulated. From this we can easily identify which approach is suitable for which particular application. To sum up, there are many techniques available for generating test cases. An integrated approach adjusts White Box test cases to Black Box test cases [4]. Divide & Rule Approach breaks dependency among variables [6]. A path oriented approach identifies path for which test case has to be generated [7].

REFERENCES

1. M.Prasanna, S.N. Sivanandam, R.Venkatesan, R.Sundarrajan, "A SURVEY ON AUTOMATIC TEST CASE GENERATION", International Journal of Software Engineering and Its Applications Vol. 6, No. 4, October, 2012.
2. Shuang Wang and Jeff Offutt," Comparison of Unit-Level Automated Test Generation Tools",IEEE International Conference on Software Testing Verification and Validation Workshops,2009.
3. T. Y. Chen, P. L. Poon "White on Black: A White-Box-Oriented Approach for Selecting Black-Box-Generated Test Cases". In Proceedings of the eighth conference on Quality Software, pages 140-154.
4. Sami Beydeda, Volker Gruhn, Michael Stachorski "A Graphical Class Representation for Integrated Black- and White-Box Testing".
5. Mumtaz Ahmad Khan, Mohd. Sadiq, Analysis of Black Box Software Testing Techniques: A Case Study, 2011 IEEE, pg 1-5.
6. Karan Vij and Wenying Feng ,"Boundary Value Analysis using Divide- and – rule Approach",Fifth International Conference on Information and Technology, 2008, IEEE, pg 70-75.
7. Du Qingfeng, Dong Xiao, "An Improved Algorithm for Basis Path Testing",2011, IEEE, pg 175-178.
8. WENYING FENG," A Generalization of Boundary Value Analysis for Input Parameters with Functional Dependency",9th IEEE/ACIS International Conference on Computer and Information Science.