Comparative analysis of Domain Specific Testing Techniques

Neha Kumawat, Yashika Sharma
Computer Science Department, MRCE, Faridabad ,India

***Abstract*- Software Testing is the process of evaluating the quality of the system with the intent to find whether it satisfies the specified requirements or not. Quality *has* to be improved for the sake of customer satisfaction as *well* as growth [1]. There are many methods for an effective testing. Black Box Testing is one of them. Black box testing is a software testing technique that focuses on the analysis of software functionality, versus internal system mechanisms. Domain Testing Method is considered as a type of Black Box Testing. Equivalence class testing(ECT) methods and boundary value testing(BVT) methods are well known as domain testing methods. Basic algorithm is used to generate test case. The main focus is on size of test suite and the complexity of domain testing methods. Comparative study of BVA and ECP includes size of test suite and the complexity of domain testing methods of Equivalence Partitioning and Boundary Value Analysis. Complexity of Domain Testing Method is calculated using Time Estimation Method. Fault coverage matrix is generated for the test suite using both techniques , that helps determine whether the behaviors specified by the requirements have been adequately tested during software validation.**

***Index Terms*- Domain Testing, Black Box Testing, Equivalence class testing(ECT), Boundary value testing(BVT)**

I. INTRODUCTION

Software quality is measured with verification and validation. Verification is a method used to ensure that a product is built correctly. Validation is a method to ensure that the correct product is built [2]. While both methods are important, this thesis will focus on verification. Well-known software verification techniques are testing, review, program proving, and model checking. This thesis focuses on testing. The number and importance of test cases that are performed in testing and the coverage of faults during executing those test cases gives a measure of the confidence of software quality. Which software Testing technique will be more suitable for the software can be determined by comparison of different techniques. Domain testing is one of the technique commonly used now-a-days for testing.

1.2 Domain Testing

It is a software testing technique in which a small number of test cases are selected from a nearly infinite group of test cases. The strategy goes under several names, such as equivalence partitioning, boundary analysis, and category partitioning.[3]

1.2.1 Equivalence partitioning

Equivalence partitioning is a **software testing** technique that divides the input data of a software unit into partitions of equivalent data from which test cases can be derived. In principle, test cases are designed to cover each partition at least once.[4]

1.2.2 Boundary value analysis

Boundary value analysis is a **software testing** technique in which tests are designed to include representatives of boundary values in a range. The idea comes from the **boundary**.[5]

1.3 Methodology

The methodologies adopted to complete the research. Comparative analysis of Boundary Value Analysis and Equivalence Class Partitioning Methods is done by using the following steps:

- For test case generation basic algorithms are used by Domain testing methods.
- Fault coverage matrix is generated for the test suite using both techniques.
- Complexity of Domain Testing Method is calculated using Total Estimation time

1.4 Organisation of Paper

Section2 describes the Case study taken for Test Case Generation. Section3 give the detail of Test Case Generation for BVA and ECP. Section4

discuss the result of the experiment1. Section5 explains the formula for Total Estimation Time.
Section6 summarizes the result of experiment II. Section7 gives the overall comparison of BVA and ECP on the basis of Experiment I and Experiment II. Finally, section 8 concludes this survey with a summary of most important results and some future directions of research and references are presented in sections 9.

II. CASE STUDY

Consider a software called GENERATE-GRADING-LINGUISTIC-PERFORMANCE,
which is used by the system to calculate student's grade and linguistic performance based on an examination mark and sessional mark. The component is passed an exam mark (out of 100) and a sessional mark (out of 50) from which it generates a grade and linguistic performance for the course in the range' A' to 'D'. The grade is calculated from the overall mark, which is calculated as the sum of the exam mark and sessional marks, as follows.

OVERALL MARK GRADE

Total Marks	**Obtained Marks**	**Grade**	**Linguistic Terms**
Out of 150	Greater than or equal to 105	A	Excellent
Out of 150	Greater than or equal to 75 but less than 105	B	Very Good
Out of 150	Greater than or equal to 75 but less than 105	C	Good
Out of 150	Less than 45	D	UnSatisfactory

III. TEST CASE GENERATION

3.1 Test Case for Equivalence Partitioning

Equivalence partitioning is predicated on the premise that the inputs and outputs of a element will be divided into categories that , in step with the component's specification, are going to be treated equally by the element. so the results of testing one worth from associate equivalence partition is taken into account representative of the entire partition.
The following nineteen equivalence partitions have been identified for the component-
$0 \leq$ exam mark ≤ 100
exam mark > 100
exam mark < 0
$0 \leq$ sessional mark ≤ 50
sessional mark > 50
sessional mark < 0
sessional mark = real number
sessional mark = alphabetic
$105 \leq$ total mark ≤ 150
$75 \leq$ total mark < 105
$45 \leq$ total mark < 75
$0 \leq$ total mark < 45
total mark > 150
total mark < 0
output = 'Z'
output = 'B+'
output = 'null'
Nineteen partitions were identified leading to nineteen test cases.
It can be seen that several of the test cases are similar, where the main difference between them is the partition targetted. As the component has two inputs and one output, each test case actually 'hits' three partitions; two input partitions and one output partition.. Thus it is possible to generate a smaller 'minimal' test set that still 'hits' all the identified partitions by deriving test cases that are designed to exercise more than one partition.
Thus the test case suite of **eleven** test cases corresponds to the minimised test case suite approach where each test case is designed to hit as many new partitions as possible rather than just one.

3.2 Test Case for Boundary Value Analysis

Boundary Value Analysis is based on the following premise. Firstly, that the inputs and outputs of a component can be partitioned into classes that, according to the component's specification, will be treated similarly by the component (in the same way as in equivalence partitioning) and, secondly, that developers are prone to making errors in their treatment of the boundaries of these classes. Thus test cases are generated to exercise these boundaries.
On the basis of partitions total **27** test cases are generated . In fact these partitions are bounded on their other side by implementation-dependent

maximum and minimum values. For integers held in sixteen bits these would be 32767 and -32768 respectively. This leads to total **18** test cases.
So, total **45** test cases are generated for BVA.

IV. EXPERIMENT I

This experiment compares the fault finding capability of Equivalence Class Partitioning and Boundary Value Analysis. Total Faults found in ECP are 13.
This implies that total fault finding coverage is 13/19 * 100 = 68.42%
After minimising the test case suite by hitting as many new partitions as possible rather than just one it is observed that 7 Faults are found.
This implies that total fault finding coverage is 7/11 * 100 = 63.63%
Total Faults found in BVA are 24.
This implies that total fault finding coverage is 24/45 * 100 = 53.33%

V. TOTAL ESTIMATION TIME

This will help in Estimating Testing effort -
Step 1 : Count no. of Test Cases (NTC)
Step 2 : Set Avg Execution Time (AET) per a test case (ideally 10 min depends on your system)
Step 3 : Calculate Total Execution Time (TET)
TET = Total number of test cases * AET
Step 4 : Calculate Test Case Creation Time (TCCT)
usually we will take 1.5 times of TET as TCCT
TCCT = 1.5 * TET
Step 5 : Time for Re-Test Case Execution (RTCE)
this is for retesting
usually we take 0.5 times of TET
RTCE = 0.5 * TET
Step 6 : Set Report generation Time (RGT)
usually we take 0.2 times of TET
RGT = 0.2 * TET
Step 7 : Set Test Environment Setup Time (TEST)
it also depends on test plan
Step 8 : Total Estimation time = TET + TCCT+ RTCE + RGT + TEST

VI. EXPERIMENT II

This experiment is based on calculating time complexity of BVA and ECP using Total Estimation Time Method.

6.1 Total Estimation Time for ECP

No. of Test cases(NTC) : 19
Average Execution Time (AET) : 10
Total Execution Time (TET) = 19 * 10 / 60 = 3.166 hr
Time for creating test cases (TCCT) : 1.5 * 3.166 = 4.749 hr
Time for retesting (RTCE) : 0.5 * 3.166 = 1.583 hrs
Report Generation(RGT) = 1.5 hrs
Test Environment Setup Time(TEST) = 1 hrs
Total Estimation time = 3.166 + 4.749 + 1.583 + 1.5 + 1 = 11.998 hrs

6.2 Total Estimation Time for Minimised ECP

No. of Test cases(NTC) : 11
Average Execution Time (AET) : 10
Total Execution Time (TET) = 11 * 10 / 60 = 1.833 hr
Time for creating test cases (TCCT) : 1.5 * 1.833 = 2.749 hr
Time for retesting (RTCE) : 0.5 * 1.833 = 0.9165 hrs
Report Generation(RGT) = 1.5 hrs
Test Environment Setup Time(TEST) = 1 hrs
Total Estimation time = 1.833 + 2.749 + 0.9165 + 1.5 + 1 = 7.998 hrs

6.3 Total Estimation Time for BVA

No. of Test cases(NTC) : 45
Average Execution Time (AET) : 10
Total Execution Time (TET) = 45 * 10 / 60 = 7.5 hr
Time for creating test cases (TCCT) : 1.5 * 7.5 = 11.25 hr
Time for retesting (RTCE) : 0.5 * 7.5 = 3.75 hrs
Report Generation(RGT) = 1.5 hrs
Test Environment Setup Time(TEST) = 1 hrs
Total Estimation time = 7.5 + 11.25 + 3.75 + 1.5 + 1 = 25 hrs

VII. COMPARISON

Comparison on the basis of Experiment I and Experiment II -

7.1 Fault Coverage Matrix

On the basis of Experiment I the following fault coverage matrix can be created -

	TC 1	TC 2	TC 3	TC 4	TC 5	TC 6	TC 7	TC 8	TC 9	TC 10	TC 11	TC 12
ECP	×	✓	✓	×	✓	✓	✓	✓	✓	✓	✓	×
Min. ECP	×	×	×	×	✓	✓	✓	✓	✓	✓	✓	-
BVA	✓	×	×	×	×	✓	✓	×	×	×	×	✓

	TC 13	TC 14	TC 15	TC 16	TC 17	TC 18	TC 19	TC 20	TC 21	TC 22	TC 23	TC 24
ECP	×	×	×	✓	✓	✓	✓	-	-	-	-	-
Min. ECP	-	-	-	-	-	-	-	-	-	-	-	-
BVA	✓	×	×	×	×	×	×	×	×	×	×	×

	TC 25	TC 26	TC 27	TC 28	TC 29	TC 30	TC 31	TC 32	TC 33	TC 34	TC 35	TC 36
ECP	-	-	-	-	-	-	-	-	-	-	-	-
Min. ECP	-	-	-	-	-	-	-	-	-	-	-	-
BVA	×	×	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

	TC 37	TC 38	TC 39	TC 40	TC 41	TC 42	TC 43	TC 44	TC 45
ECP	-	-	-	-	-	-	-	-	-
Min. ECP	-	-	-	-	-	-	-	-	-
BVA	✓	✓	✓	✓	✓	✓	✓	✓	✓

7.2 Comparison Table

	ECP	**Min. ECP**	**BVA**
No. of partitions	**19**	**19**	**19**
No. of test cases	**19**	**11**	**45**
(Time Complexity) Time required for Executing test cases	**11.998 hrs**	**7.998 hrs**	**25 hrs**
Cost required for test case generation	**Moderate**	**Low**	**High**
Total Fault Covered	**13**	**7**	**24**
Fault Coverage Percentage	**68.42 %**	**63.63 %**	**53.33 %**

VIII. CONCLUSION AND FUTURE WORK

From the above table in Comparison it is clear that ECP is more better than BVA as it works well with less number of test cases due to which it is time efficient . We can also see that the test case suite can be reduced in ECP without any loss of any partition left uncovered which is not possible in BVA. Also we can see that fault coverage percentage of ECP (68.42 %) and with minimized ECP (63.63 %) is greater than BVA (53.33%) .

Total time required for test case execution for ECP is 11.998 hrs, minimised ECP is 7.998 hrs and BVA is 25 hrs .

All this leads to the result that ECP is much better and efficient technique in time and cost as compared to BVA.

In future this work can be exceeded with implementation on the same instead of manual testing. Comparison can also be done between other techniques.

REFERENCES

[1] http://www.softwaretestinghelp.com/software-testing-career-and-secrets-of-a-richest-tester

[2] Barry W. Boehm. Software Engineering Economics. IEEE Transactions on Software Engineering, SE-10(1):4–21, January 1984.

[3] Kaner, C. The impossibility of complete testing, Software QA, 1997, vol. 4, pp. 28

[4] http://www.softwaretestinghelp.com/what-is-boundary-value-analysis-and-equivalence-partitioning/

[5] https://en.wikipedia.org/wiki/Equivalence_ partitioning

[6]http://www.researchgate.net/publication/ 220902769_An_Empirical_Analysis_of_Equivalence_Partitioning_Boundary_Value_Analysis_and_Random_Testing

[7] Stuart C. Reid, 'An Empirical Analysis of Equivalence Partitioning, Boundary Value Analysis and Random Testing', Cranfield University Royal Military College of Science, Shrivenham, Swindon, Wilts SN6 8LA, UK

[8] Hetzel, W.C., 'Making Software Measurement Work',*QED Publishing Group*, 1993

[9] Zhu H., Hall P. A., 'Software unit test coverage and adequacy', *ACM Comput. Surv.* 29, May J. H. (1997),pp 366–427.