© March 2016 | IJIRT | Volume 2 Issue 10 | ISSN: 2349-6002

Comparative Study between Equivalence Class Partitioning and Boundary Value Analysis Testing Methods

Neha Kumawat, Yashika Sharma, Urmila Pilania
Computer Science Department, MRCE

***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. One of them is Domain Testing Method. 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 ,an Empirical Analysis of Equivalence Partitioning and Boundary Value Analysis that helps in comparing effectiveness of both. For comparing effectiveness, an experimental methodology is used in which all possible input values are considered that satisfy a test technique and also all possible input values that would cause a module to fail and determine absolute values for the effectiveness for each test technique.**

I. INTRODUCTION

One of the mostly used software testing technique is Domain testing. It is a method in which there are infinite group of candidate test cases from which a small number of test cases are selected. Domain knowledge plays a very critical role while testing domain-specific work[2][3]. Boundary value analysis and equivalence partitioning are test case design strategies in black box testing.

1.1 Equivalence Partitioning:

In this method the input domain data is divided into different equivalence data classes. This method is typically used to reduce the total number of test cases to a finite set of testable test cases, still covering maximum requirements. In short it is the process of taking all possible test cases and placing them into classes. One test value is picked from each class while testing. Equivalence partitioning uses fewest test cases to cover maximum requirements[4].

1.2 Boundary value analysis:

It is widely recognized that input values at the extreme ends of input domain cause more errors in system. More application errors occur at the boundaries of input domain. 'Boundary value analysis' testing technique is used to identify errors at boundaries rather than finding those exist in centre of input domain[4].

The effectiveness of the two most popular black box testing techniques used in the module testing phase of software development, equivalence partitioning (EP) and boundary
value analysis (BVA), are compared with each other in this paper. There are two parameters for comparing the testing methods – relative effectiveness and cost [5]. The complexity of the method is closely related with the cost of the method. If there is a large count of test cases in the test suite it means that complexity is high. When the size of the test suite grows, more resources are necessary for testing. A method is considered effective if the software tested thoroughly according to that method is almost correct [5]. But method should be efficient too – if the test case fails, it is better if there is a small count of candidates (input values of the test case) to blame. Hence, complexity and effectiveness both are important during testing.

1.3 Methodology

The methodology used in this study is considering every possible value in an equivalence
partition, so that an absolute measure of test
effectiveness for given modules and testing techniques was generated. The aim of this paper is to review some ECT and BVT methods and assess their complexity.

Section2 describes the first experiment conducted to compare the effectiveness of BVA and ECP.
section3 discuss the result of the experiment1.
section4 explains the testing criteria of domain testing methods. There is complexity of equivalence class testing methods in section 5 and complexity of boundary value testing methods in section 6.In sections 5 and 6, the algorithm of test case generation is also described which is considered as experiment2 and
the complexity of each identified domain testing method is assessed. Section 7 summarizes the result of experiment2 and provides the hierarchy of
domain testing methods. 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. EXPERIMENT I

This experiment compares the effectiveness of Equivalence Class Partitioning and Boundary Value Analysis. It has following five stages: Failure Analysis, Creation of Fault Finding Sets, Creation of Test Case Sets, Derivation of Probabilities, and Analysis of Results. In this research Fault Finding Sets is dependent on Failure Analysis of the system.
The sub-domains were identified to satisfy test coverage criteria which are then defined as the test case sets i.e. the third stage. Techniques were used to define both the fault-finding sets and test case sets. Then, the concepts of domain testing is used to calculate a definitive probability of detection in terms of 'overlapping' N-space convex polyhedra, which correspond to the test case sets and the fault-finding sets respectively.

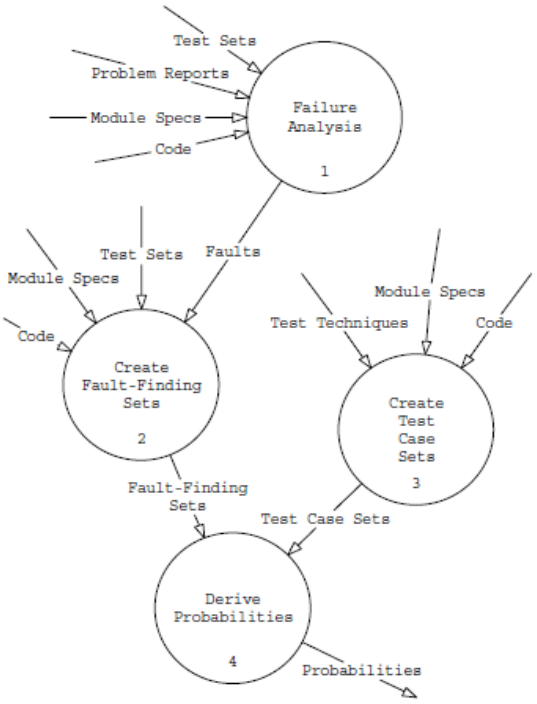

Figure 1: Experimental methodology[6][7].

2.1. Hypotheses

The experiment considered three pair wise comparisons.
In each case the first technique in each pair was more effective than the second.
- BVA (one-to-one) with EP (one-to-one)
- EP (one-to-one) with Minimised EP
- BVA (one-to-one) with Minimised BVA

2.2. Failure analysis

Failure analysis was done on one of the system already in use to identify the faults in the system by using module testing. Then analysis of these faults were done using the original and modified source code, design specifications and test specifications. All the faults were identified by analysing the complete system. Hetzel [8] states that the "defects of greatest interest are those still in the product after release, not those already discovered." Module testing detects some faults which are considered as a small subset of all faults found with the system (less than 10%) which are then used to create the fault-finding sets.

2.3. Creation of fault finding sets

The fault-finding set is created by using the set of inputs to the module that would cause the fault to be detected. An example is shown in Figure 2[6-7], with two fault-finding sets shown.
These fault-finding sets take the form of equalities and inequalities simultaneously on the module

inputs which represents the input conditions for detecting the fault.

In the example the fault-finding sets (FF1 and FF2) could be described by:

-4 □□XIN □□0 AND 1 □□YIN □□4 FF1

2 □□XIN □□4 AND 1 □□YIN □□5 FF2

where the two input parameters are XIN and YIN, as shown in figure 2.

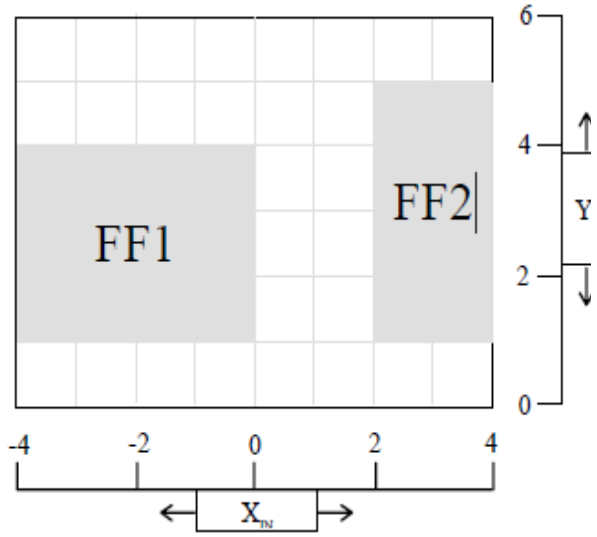

Figure 2: Fault-finding sets[6][7]

2.4. Creation of test case sets

On the basis of definitions of testing techniques the test case sets were all derived to decrease the work of testers. The test case design techniques is used in such a way that test coverage criteria is covered totally.

Like the fault-finding sets, the test case sets also take the form of equalities and inequalities simultaneously on the module inputs. For a given test coverage criterion there will be a number of test case sets, each set belongs to one of the sub-domains. If we use an EP 'one-to-one' approach then four test cases would be created:

-4 □□XIN □□4 AND 4 < YIN □□6 TC1

-4 □□XIN □□4 AND 3 □□YIN □□4 TC2

-4 □□XIN □□4 AND 0 □□YIN < 3 TC3

-1 < XIN < 1 AND 0 □□YIN □□6 TC4

2.5. Derivation of probabilities

The degree of intersection between the fault-finding sets and the test case sets derives fault-finding capability of a technique for a particular module.

Convex polyhedra in N-space is considered for the domains of both the fault-finding sets and the test case sets, where N is the number of input parameters to the module. The 'overlap' between the fault-finding polyhedra and the test case polyhedra helps in measuring the probability of a fault being detected .

$P_{DET} = m/d$

where D denote the module's input domain, of size d,

(thus d=|D|)

FF represents the union of all fault finding sets, then m=|FF|

$P_{DET}(TC,FF) = m_{TC/}\ d_{TC}$ **where** dTC=|TC| and mTC=|FF□TC|.

Thus, the probability of missing the fault is

$P_{MISS}(TC,FF) = 1 - P_{DET}(TC,FF) = 1 - m_{TC/}\ d_{TC}$

Then, given that n test case sets (TC1,TC2,..,TCn) are required to satisfy the test coverage criterion, C, the probability of the fault (represented by the fault-finding sets FF) being missed is the product of the miss probabilities for each of the n test case sets, thus:

$P_{MISS}(C,FF) = \prod_{i=1}^{n} P_{MISS}(TC_i,FF)$

$P_{DET}(C,FF) = 1 - P_{MISS}(C,FF) = 1 - \prod_{i=1}^{n} (1 - P_{DET}(TC_i,FF))$

For example, calculation can be shown by considering the fault-finding sets and test case set TC_1 to TC_4

$P_{DET(TC1)} = 1/8 \Rightarrow P_{MISS(TC1)} = 7/8$

$P_{DET(TC1)} = 3/4 \Rightarrow P_{MISS(TC1)} = 1/4$

$P_{DET(TC1)} = 1/2 \Rightarrow P_{MISS(TC1)} = 1/2$

$P_{DET(TC1)} = 1/4 \Rightarrow P_{MISS(TC1)} = 3/4$

$\Rightarrow P_{MISS} = 7/8*1/4*1/2*3/4 = 21/256 = 0.082 = 8.2\%$

and $P_{DET} = (1-0.082) = 0.918 = 91.8\%$

Module ID	EP (1 to 1)	Minimised EP	BVA (1 to 1)	Minimised BVA
1	0.000153	0.000055	0.000421	0.72
2	0.000379	0.00017413	0.00106045	0.72
3	0.00021111	0.0001085	0.00044791	0
4	0.0000705	0.0000215	0.000204	0
5	0.00003	0.00001529	1	1
6	0.1787	0.09375	0.3843	0
7	0.578125	0.578125	1	1
8	0.9076	0.85086	1	1
9	0.00052074	0.00052074	1	1
10	1	1	1	1
11	1	1	1	1
12	0.000095497	0.00006499895	1	1
13	0.00004826	0.00003322	1	1
14	0.00052074	0.00052074	1	1
15	0.06261	0.0276	0.98686	0.9961
16	0.95551	0.75	0.999997	0.9961
17	1	1	1	1
Mean	**0.33**	**0.31**	**0.73**	**0.79**
Sum	**5.7**	**5.3**	**12.4**	**13.4**

TABLE I : Probabilities of detection

Module ID	EP (1 to 1)	Minimised EP	BVA (1 to 1)	Minimised BVA
1	12	6	44	14
2	12	6	44	14
3	6	3	24	10
4	6	3	24	10
5	6	3	20	12
6	4	3	13	5
7	4	4	8	5
8	6	5	19	14
9	4	4	24	5
10	8	4	24	13
11	9	4	26	13
12	10	9	26	23
13	10	9	26	23
14	4	4	8	5
15	10	6	35	26
16	10	6	35	26
17	9	4	26	13
Mean	**7.6**	**4.9**	**25.1**	**13.6**

TABLE II : Number of test cases

III. EXPERIMENT II

This experiment is based on Adequacy criteria.
Adequacy criterion explicitly specifies test case selection, determines whether a test set is adequate, and determines observations that should be done during the testing process[9]. Adequacy criteria of domain testing methods can be characterized by three aspects:
1) which kind of values to choose for testing, for example, only boundary values or only representants of equivalence classes of valid values;

2) data coverage principle;
3) strategy how the chosen values are combined in test cases according to the data coverage principle.

3.1 Complexity of Equivalence Class Testing and Boundary Value Testing

For calculation of complexity test case is derived for each of the selected values and the other input parameters of the test case assume some
valid nominal value.

3.1.1 Robust Weak Equivalence Class Testing

Robust weak equivalence class testing examines one representant from each equivalence class of valid values of each parameter [2, 5, 28–32, 42].In addition, it also considers equivalence classes of invalid values [1, 2, 5, 28–31, 34–38,42] according to the following algorithm:
1) for valid values choose only one value from each equivalence class; furthermore, all parameters have valid values in each test case ;
2) for invalid values choose one value from each equivalence class and in each test case combine one invalid value with all other valid values. Invalid values for two or more input parameters of the program in the same test case are not allowed
In N parameters' case, the count of generated test cases is

$$\max_{i=1}^{N}(M_i) + \sum_{i=1}^{N} Q_i$$

where Qi is the size of the set of equivalence classes of invalid values for parameter Xi.

3.1.2 Robust Weak Boundary Value Testing

The robust weak boundary value testing method [2, 5, 34–36, 38, 42] examines
boundary values of equivalence classes, inner and outer OFF points, nominal case.
The size of the generated test suite can be obtained using following method –
If the program has two parameters, test cases are generated according to the
following algorithm.
1) Hold nominal value of the first equivalence class for the first parameter and obtain 4 test cases by changing the min, min+, max-, max points for the first
class of the second parameter.
2) repeat step 1 for each equivalence class of X2.
There are 4M2 test cases obtained so far.
3) Optimize the test suite – exclude redundant test cases raised by overlapped
boundary values of adjacent equivalence classes.
Now we have 4M2 – L2 test cases in our test suite.
4) Repeat steps 1–3 for each equivalence class of parameter X1. There are (4M2 – L2)M1 test cases obtained so far.
5) Repeat steps 1–4 with parameters in exchanged roles. There are (4M1 – L1)M2 + (4M2 – L2)M1 test cases obtained during steps 1–5.
6) Now we have to add point test cases when both parameters have nominal values for all elements of Cartesian product of valid equivalence classes of both parameters.
So, we obtain M1M2 test cases in this step.
The size of the resulting test suite is:
there will be no less than

$$(6N+1)\prod_{i=1}^{N} M_i - \sum_{i=1}^{N}(L_i \prod_{\substack{j=1 \\ j\neq i}}^{N} M_j)$$

test cases and no more than

$$(6N+1)\prod_{i=1}^{N} M_i$$

test cases .

IV. CONCLUSION

Equivalence partitioning and boundary value analysis techniques are frequently mentioned in the books about software testing. In most cases, they describe common principles how to derive equivalence classes, boundary values, and present some strategies how to make test cases from them. This paper compares effectiveness and complexity of domain testing methods. The complexity of testing methods is treated from the aspect of the size of the test suite generated by the method.

The results of experiment I is that -

- BVA was the most effective technique studied, achieving a highest mean probability of detection of 0.79, compared with 0.33 for EP. However, to achieve this, nearly twice as many test cases were required (13.6 for BVA , 7.6 for EP).
- Minimised BVA focuses testing more on the extremes of the input domain than one-to-one BVA; the study found little to choose between the two approaches.
- Minimised EP was consistently slightly less effective than one-to-one EP, although it required marginally fewer test cases. Again, there was little to choose between the two approaches.

The result of experiment II is that -

No.	Method	Complexity	
		Lower bound	Upper bound
1.	Robust weak equivalence class testing	$\max_{i=1}^{N}(M_i)+\sum_{i=1}^{N}Q_i$	The same as lower bound
2.	Robust weak boundary value testing	$(6N+1)\prod_{i=1}^{N}M_i-2\sum_{i=1}^{N}(L_i\prod_{\substack{j=1\\j\neq i}}^{N}M_j)$	$(6N+1)\prod_{i=1}^{N}M_i$

REFERENCES

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

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

[3] Weyuker E. J., Jeng B. ,'Analyzing Partition Testing Strategies', IEEE Trans. Softw. Eng., July 1991,vol. 17,No. 7,pp 703–711.

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

[5] Weyuker E. J., Weiss S. N., Hamlet D. (1991) Comparison of program testing strategies. In: Proceedingsof the Symposium on Testing, Analysis, and Verification, TAV4. ACM, New York, NY, 1–10.

[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.