IJCSE International Journal of Computer Sciences and Engineering Open Access
Survey Paper Vol.-7, Issue-2, Feb 2019 E-ISSN: 2347-2693

# Boundary Analysis for Equivalent Class Partitioning by using Binary Search

**Sandeep Chopra[1*], Lata Nautiyal [2], M.K. Sharma [3]**

[1]MCA, Uttrakhand Technical University, Dehradun, India, 248001
[2]MCA , Graphic Era University, Dehradun, India, 248002
[3]MCA , Amarapali Institute, Haldwani, India, 263139

**Corresponding Author: tosandeepchopra2016@gmail.com, 9897884345*

DOI: https://doi.org/10.26438/ijcse/v7i2.601605 | Available online at: www.ijcseonline.org

Accepted: 13/Feb/2018, Published: 28/Feb/2019

***Abstract***—Testing of Software is an indispensible phase of software development. It helps us to improve functional and non-functional characteristics. To implement functional test scenario black box testing process is used, and the test bases are the functional requirement. Nonfunctional requirement does not describe the function, but the attribute of the function i.e function quality, usability, efficiency and reliability. To implement testing, the most difficult part is to design test cases. There are numerous processes available which can help us to design test cases. This paper will present the novel algorithm of Equivalence class partitioning. Here the input is partitioned by using a strategy that is inspired by binary search. Based on the input data, the complete range is divided into two sub ranges, and this partition continues until a threshold is reached. The proposed novel algorithm of testing will increase the reliability of the software product.

***Keywords:*** Software testing, functional testing, black box testing, binary search, class partitioning.

## I. INTRODUCTION

Software testing is a process of ensuring acceptable degree of quality attributes of software. The main objective is to achieve correctness, robustness , reliability etc. Testing is the process of finding errors at any level or stage of development that modifies non functional attributes like reliability andquality of a software product[1,2].
One can know about the functional testing by verifying all the functionality present in the software.

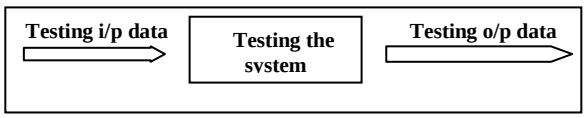



**Fig1: Black box testing**

The testing process includes two types of inputs - (i) software configuration (ii) test configuration.

Test are performed with the test configuration on the software configuration and all possible outcomes are stored. These stored results are compared with expected results .When the outcome result is not matched with the expected result it means there is an error in the software, then this error is reported for debugging[3,4,5]. The main objective of this paper is to design a new algorithm for the tester, so that one can divide input cases according to one's requirement. Boundary value analysis tests values only at the boundary, but for the larger value only test at boundary level is not sufficient.

## II. TESTING OF DIFFERENT SOFTWARE COMPONENTS

There is another aspect of testing i.e component based testing[6]. This is typical because the tester has to integrate two or more than two component before the testing can be performed. This involves coupling i.e( joining more than two module) and the good softwareshould possess maximum cohesion and minimum coupling. This process increase the reliability of a software[7].
Some popular definations of componenet testing are.

- TheIEEE defines software testing as " the process of analyzing a software item to detect differences between existing and required condition and to evaluate the feature s of the software item".[8]
- Component testing is the activity in which individual components are tested to ensure that they operate correctly. Each component is tested independently and correctly, without other system components gap and errors." Sommerville[9]

Broadly testing is divided into three category.

- Equivalence class partitioning testing

- Boundary Value testing ( BVA )
- Decision table based testing

**2.1 Equivalence class partitioning testing**:
Equivalence class analysis is black box software testing technique that minimizes the number of test cases to a necessary minimum and selects a right test case that could be represented to cover the possible similar scenario[10,11 ].

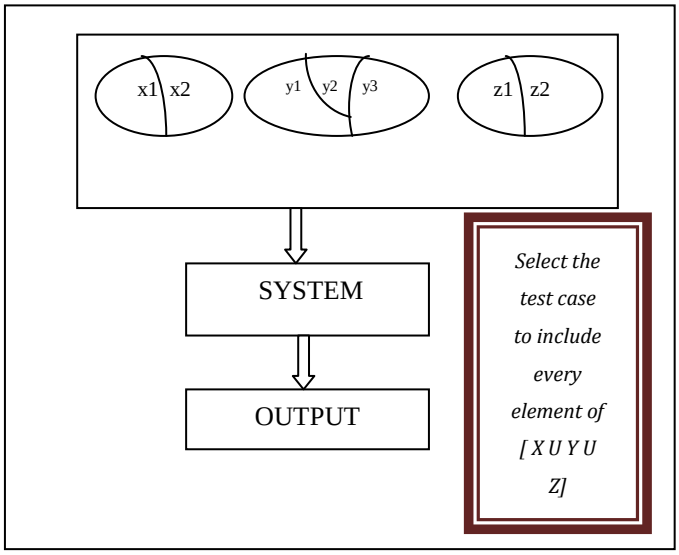



**Fig 2. Equivalence class testing**

The following major steps are involved in equivalence class analysis are

- Find and list all the input variables
- Find the equivalence classes for each variable and its boundaries
- Write test cases for these classes

**2.2 Boundary Value Analysis**:
Boundary Value analysis concentrates on the behavior of the system on its boundary condition or the boundary of its input variables because system changes its nature very frequently at the boundary[ 12 ].It means it is not stable at the boundary. The boundary of a variable includes the maximum and the minimum valid value allowed to attain the system . It pin points on the data at the "edges" of an equivalence class [13,14 ].

**2.3 Decision Table based Testing**
A decision table is a compact way to model system behavior for different input conditions and integrates the functionality of the system with input predicates ( if – else, switch case). It tells the actions to be implemented for a given closure of the system input. The decision table consists four quadrants[ 15,16 ]

i. Condition stub: filled with all possible condition
ii. Condition entry stub: filled with all unique combination of existence of the input condition
iii. Action stub: filled with all functionalities the system is expected to perform during its execution.
iv. Action entry stub: a cross is marked in front of the action to be taken in a particular combination of the input condition.

## III. DIFFERENT CHALLENGES IN SOFTWARE TESTING

After going through all the criteria of software testing , one identifies the problems and challenges while implementing the testing. Some of them are given below

- The method of verification and validation are different in conventional approach and component approach.[17]
- By adopting CBSE approach one can higher quality product by reducing development cost and time . There is some problem when we integrate diffrenr component and composition of third party.[18,19]
- In Component based development , component vendor implements testing criteria during early phases of component development on the other hand user performs testing activities during application engineering.[20,21]

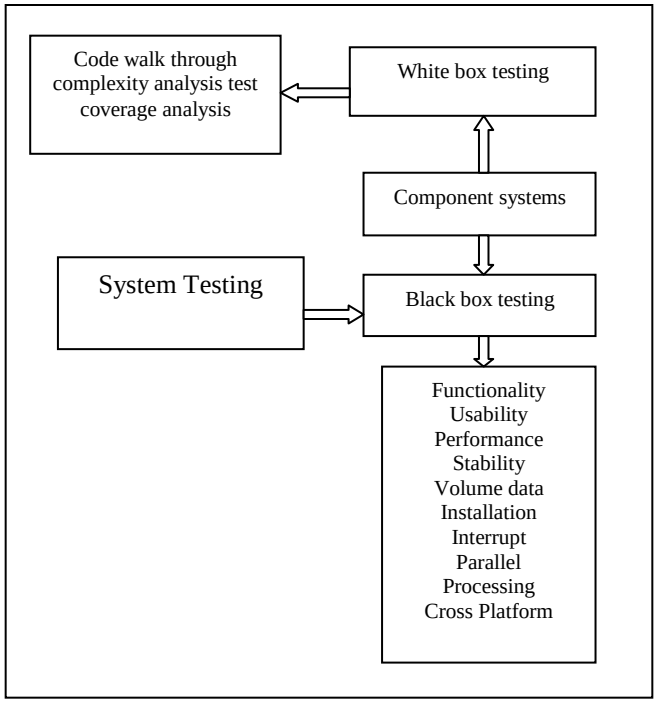



**Fig 3. Testing types at a glance**

## IV. ANALYSIS OF THE RELATED WORK DONE

This section highlights the previous work done in the field of software testing. In [24] researcher focuses on sequence

and state diagram, thereby new test cases will be generated automatically. In [25] researcher tried to identified the problem arises when one of the component has changed or modified. In [26] researcher proposed a state machine based method to detect robustness problem and test the invalid inputs. In [27] researcher proposed path-oriented random testing, which minimize or rejects the number of inputs. In [28] researcher partitioned the input in two classes i.e even and odd and worked for integer inputs only. In [29] researcher proposed the test cases of a boundary value analysis based on strings ,which is a non-numeric variable.

## V. TEST CASES FOR THE PROPOSED WORK

The most important thing while testing a software is to design a test cases.

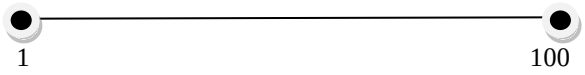
The main disadvantage of equivalence class partitioning is that it gives values at the boundary .for example suppose we have an element from [ 1 to 100 ].

so our test cases will be

**Test Case 1:**

a) i. x<1 ii. x=1 iii. x>1

b) i. x<100 ii. x=100 iii. x>100




The equivalence class gives only few values but if the database is very large and if we test only at the boundaries , the reliability of the software will be very low. So this paper proposes a new algorithm where the tester is free to create more boundary in between the equivalent class which helps him to test as many caseshe want which will increase the reliability . Suppose in the above example if the tester wants to create two more points then the number of partitions will be two and the position of partitioning will be entered by the tester .For example position in the first case is 50, so in the above array the partition will be [ 1 to 50] and [ 51 to 100].

so our test cases will be

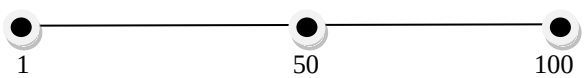
**Test Case 2:**

a) i. x<1 ii. x=1 iii. x>1

b) i. x<50 ii. x=50 iii. x>50

c) i. x<51 ii. x=51 iii. x>51

d) i. x<100 ii. x=100 iii. x>100




For example position in the second case is 25, so in the above array the partition will be [ 1 to 25] , [26 to 50] and [ 51 to 100].

so our test cases will be

**Test Case 3:**

a) i. x<1 ii. x=1 iii. x>1

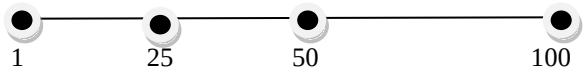
b) i. x<25 ii. x=25 iii. x>25

c) i. x<26 ii. x=26 iii. x>26

d) i. x<50 ii. x=50 iii. x>50

e) i. x<51 ii. x=51 iii. x>51

f) i. x<100 ii. x=100 iii. x>100




## VI. PROPOSED ALGORITHM

**I.** Begin

**II.** Intialize variables

int *boundary,n // used for dynamic array and n is the sizre of array

int i ,outerloop, innerloop , pos , found // global variables

int split_pos[25]; // it store all indexes from where array splits,

**III**. Enter the size of an array from the tester and input values dynamically.

**IV**.Enter the position (pos) where user wants to split the boundary into two parts

i Check ( pos> n-1 OR || pos<=0) then print Invalid position

ii Check or the duplicacy position whether boundary is already splitted at this position.

**V.** Assign the position of splitting in array split_pos and increment the index

**VI**. Now Call the **sort function** and pass the address split_pos ,indexposition

**VII**.Now create boundary & insert symbol "[" at starting and insert symbol "]" at the end

```
for (outerloop=0 to outerloop<split_pos_max_index)
        end_pos = split_pos[outerloop];
        print " ["      // at the starting boundary
for ( innerloop=start_pos to innerloop<=end_pos) then
// initially start_pos =0  print the boundary elements of an array
print  "] "      // at the end of the boundary
  //       for the remaining array
print  "\n [";    // at the starting of the next boundary
for innerloop=start_pos to innerloop<n) //  n is the number of elements in an arry
print the boundary elements of an array;
print  "] "      // at the end of the boundary
```

**VIII.** This process continues for the remaining array.

**IX**. End

```
void sort(int *split_pos,intsplit_pos_max_index)  // definition of sort function
        Begin
```

**i.** for i=0 to split_pos_max_index
**ii.** for j=i+1 to split_pos_max_index
**iii.** if split_pos[i] >split_pos[j] then
swap ( split_pos[i] and split_pos[j])
**iv.** End of step ii //End for loop ii.
**v .** End of step i //End for loop i.
**End**

**Out put: For simplicity , the size of an array is in between [ 0 – 9 ]. The code is written in C language.**

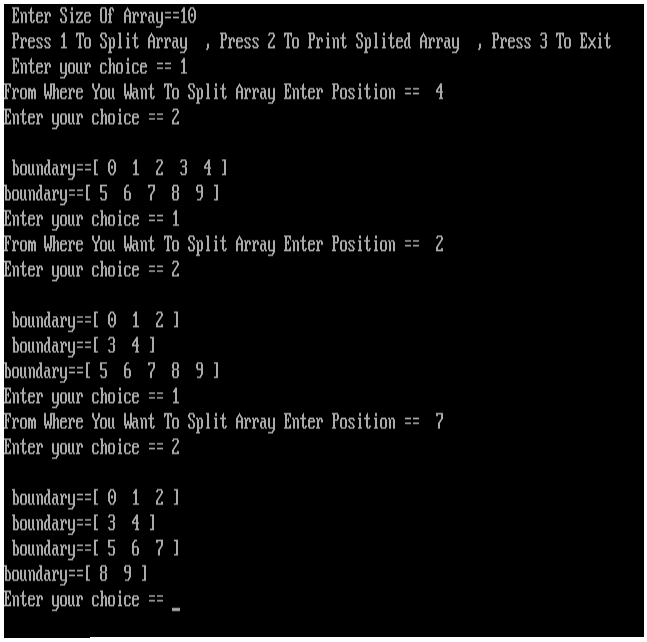



**Figure 3**

## VII. COMPLEXITY OF THE ABOVE ALGORITHM

Suppose we have n elements in an array taking n=10 and the element of an array is {0,1, 2, 3, 4, 5, 6, 7, 8, 9} . Now suppose the tester wants to split at position 4.
Now the result of splitted array will be [0, 1, 2, 3, 4]and, [ 5,6,7,8,9].
Now suppose again the tester wants to split at position 2 .
Now the result of splitted array will be [0, 1, 2] [ 3, 4] and, [ 5, 6,7,8,9 ].
This process continues until we split the last element. So the process can define as

Step1: n elements in search space
Step 2: n/2 elements in search space
Step 3: n/4 elements in search space
Step 4: 1 element in search space.

The problem is , how many times can we divide M by 2 until we have 1 i.e
the last splited location in an array, Mathematically it can be expressed as

$1 = M / 2^x$
$2^x = M$
Taking log both side
$\log 2^x = \log M$
$x \log 2 = \log M$
$x = \log M / \log 2$
$x = \log_2 M$----------(1)

This means tester can divide $\log_2 M$ times until a threshold is reached. But every time when we split the boundary , the sort function is called. This sort function arrange the index of split_pos []array. Now the complexity of sort function

- To sort the first location of spli_pos array compiler will have to (m-1) times.
- To sort the second location of spli_pos array compiler will have to (m-2) times.
- This process continues until a the last location i.e a thresh hold is reached,
  So this will make a series i.e (m-1) + (m-2) + (m-3)+…….+ 3 + 2 + 1.
  This is an Arithmetic progression and the sum of arithmetic progression will be
  S={ n / 2*( 2*a + (n-1)*d)}
  Where n is the total number of elements , a is the first term and d is the common difference,
  In the above series a = m-1 , d= -1, no. of elements = m
  S = m / 2*( 2*(m-1) + (m-1)* -1)
  = m*(2m-2-m +1)/ 2
  = m*(m-1)/2
  = $m^2/2$ - m / 2
  S= $O(m^2)$ i.e higher order of m----------(2)

Combining (1) and (2) we will get $\mathbf{m^2 \log_2 M}$. This is the complexity of this algorithm.

## VIII. CONCLUSION

In this paper author has proposed a novel approach of equivalence class partioning. The proposed algorithm is implemented in C language and the output of the same is shown in the figure3. The complexity of the algorithm is also calculated.

This proposed algorithm has given a free hand to the tester to do the partitions the number of times he wants and also increases the number of inputs. As the number of inputs have increased which increases the number of test cases and there by increases the reliability of a software product.

## REFERENCES

[1]. B. Meyer. “The grand challenge of trusted components,” In Proc. ICSE
[2]. 2003, pages 660–667. IEEE Computer Society Press, 2003,

[3]. Bertrand Meyer et al. "Providing Trusted Component to the Industry ,"
[4]. vol 31,no 5,pp104-105. IEEE, May 1998
[5]. C. Szyperski, Component Oriented Programming: Beyond Object
[6]. Oriented. Reading, MA: Pearson Education, 1999.
[7]. G. J. Myers, The art of software testing, 2nd ed. United States: John
[8]. Wiley & Sons, 2004.
[9]. Sandeep Chopra, M.K.Sharma, LataNautiyali, "Analogous Study of
[10]. Component-Based Software Engineering Models",IJARCSSE, Vol.7
[11]. ,Issue 6, pp. 597-603, 2017
[12]. M. Sitariman and B. Weide, „Component-based software using
[13]. RESOLVE", ACM SIGSOFT Software Engineering Notes, vol. 19, no.
[14]. 4, pp. 21–22, Oct. 1994
[15]. Sandeep Chopra etal"Elena – A Novel Component Based Life Cycle
[16]. Model", European Journal of Advances in Engineering and
[17]. Technology 2017, 4(12): 932-940
[18]. IEEE "Standard Glossary of Software Engineering Terminology", IEEE
[19]. Standards Board, Sep. 1990.
[20]. IanSommerville . Software Engineering . 9thedition.Addison Wesley
[21]. March 2010
[22]. LataNautiyal,Preeti, "A Novel Approach of Equivalent Class
[23]. Partitioning for Numerical Input",ACM SIGSOFT ,Vol41, Number 1,
[24]. Jan 2016
[25]. LataNautiyal,etal, "A Novel Approach to Component Based Software
[26]. Testing",ACM SIGSOFT ,Vol 39, Number 6, Nov 2014
[27]. J.MVoas ," A dynamic testing complexity metric" Software Quality
[28]. Journal, vol 1 issue 2, pp 101-114 , june 1992
[29]. C.Ramamoorthyetal," On the automated generation of program test
[30]. data", IEEE Transaction on Software Engineering ,2(4): 293-300, april
[31]. 1976
[32]. J. Voas and J. Payne, „Dependability certification of software
[33]. components", Journal of Systems and Software, vol. 52, no. 2–3, pp.
[34]. 165–172, Jun. 2000.
[35]. J. Voas, J. Payne, R. Mills, and J. McManus, "Software testability",
[36]. Proceedings of the 1995 Symposium on Software reusability - SSR '95,
[37]. April. 1995.
[38]. E.J. Weyukar. More experience with data flow testing ,IEEE
[39]. Transaction on Software Engineering ,19(9), : 912- 919 , 1993
[40]. Beatriz Pérez Lamancha, Pedro Reales Mateo, Ignacio Rodríguez de
[41]. Guzmán, Macario Polo Usaola, and Mario Piattini Velthius, "Automated
[42]. model-based testing using the UML testing profile and QVT,"
[43]. Proceedings of the 6th International Workshop on Model-Driven
[44]. Engineering, Verification and Validation (MoDeVVa 2009),Denver,
[45]. Colorado, USA, 05 Oct 2009. ACM International Conference
[46]. Proceedings Series, vol. 413, ACM Press, 2009.
[47]. Brown, Alan W., Wallnau, Kurt C. (1998): The Current State of CBSE.
[48]. IEEE Software Journal, September/October 1998, pp. 37-46.
[49]. Han, Jun (1998): Characterization of Components. In proceedings of
[50]. International Workshop on Component- Based Software Engineering,
[51]. 1998.
[52]. Hong Zhu, Patrick A. V. Hall and John H. R. May, "Software Unit Test
[53]. Coverage and Adequacy," ACM Computing Survey, vol. 29, no. 4, pp.
[54]. 366–427, Dec 1997.
[55]. Paul Baker, Zhen Ru Dai, Jens Grabowski, Øystein Haugen, Ina
[56]. Schieferdecker, and Clay Williams, Model-Driven Testing Using the
[57]. UML Testing Profile. Springer, 08 Nov 2007.
[58]. 22 . A. Bertolino and E. Marchetti. 2005. Introducing a Reasonably
[59]. Complete and Coherent Approach for Model-based Testing. Electronic
[60]. Notes in Theoretical Computer Science, Elsevier, 116, 85–97
[61]. Leonardo Mariani and Mauro Pezz`e. 2005. A Technique for Verifying
[62]. Component-Based Software, Electronic Notes in Theoretical Computer
[63]. Science, Elsevier, 116, 17–30.
[64]. Bin Lei, Zhiming Liu, Charles Morisset and Xuandong Li, 2010. State
[65]. Based Robustness Testing for Components. Electronic Notes in
[66]. Theoretical Computer Science, Elsevier. 260, 173– 188.
[67]. Arnaud Gotlieb and Matthieu Petit. 2006. Path Oriented random testing.
[68]. Proceedings of the First International Workshop on Random Testing,
[69]. July 20, 2006, Portland, ME, USA Copyright ACM.
[70]. A. Jain, S. Sharma, S. Sharma and D. Juneja. 2010. Boundary value
[71]. analysis for non-numerical variables: Strings, Oriental Journal of
[72]. Computer Science & Technology, Vol. 3(2), 323-330.

## Authors Profile

*Mr. Sandeep Chopra* pursed Master of Computer Application(MCA) from G.B.Pant Engineering college (Pauri)in 2004. He has also done two more Master courses i.e MSc(Maths) and MTech(IT). He has cleared UGC-NET(thrice), U-SET, GATE exam in the field of computer science. He is currently pursuing Ph.D. from Uttrakahand Technical University(Dehradun) and currently working as Assistant Professor in Department of Computer and Information and technology, University of SGRR since 2006. His main research work focuses on Component based software engineering.He has published 6 papers in international and 2 papers in national journals. He has 14 years of teaching experience and 2 years of research experience.

D*r. Lata Nautiyal* pursed Doctor of Philosophy PhD from Gurukul Kangri University in 2016.. She has also done three Master courses i.e MSc(CS),MCA and MTech(CS). She is currently working as Associate Professor in Department of Computer Application in Graphic Era University. Her main research work focuses on Component based software engineering.She has published 15 papers in international and 12 papers in national journals. She has 15 years of teaching experience and 6 years of research experience.

*Dr. M.K.Sharma* pursed Doctor of Philosophy PhD. He is currently Professor and Head inMCA Department of Amrapali Institute of technology,Haldwani. He has published more than 25 papers in national and international journals. He has 18 years of teaching experience and 7 years of research experience.