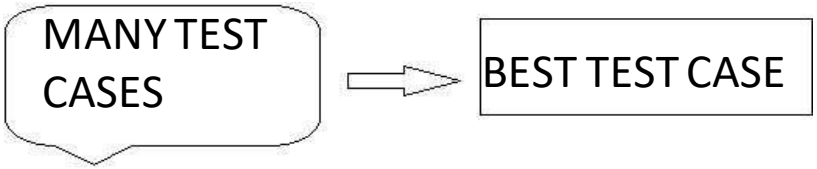
Unit II

Chapter 3: Boundary Value Testing

The practice of testing software has become one of the most important aspects of the process of software creation. When we are testing software the first and potentially most crucial step is to design test cases.



Boundary Value Testing

- Boundary value analysis is a type of black box or specification based testing technique in which tests are performed using the boundary values.
- Practically, due to time and budget considerations, it is not possible to perform exhausting testing for each set of test data, especially when there is a large pool of input combinations.
- We use two techniques - **Equivalence Partitioning** & **Boundary Value Analysis testing techniques** to achieve this.

Definitions

“ Bugs lurk in corners and congregate at boundaries”

Boris Beizer

A boundary value:

- an input value or output value which is on the edge of an equivalence partition or at the smallest incremental distance on either side of an edge, for example the minimum or maximum value of a range.

What is **Boundary value analyze (BVA)?**

- Is a technique to refine equivalence partitioning
- based on testing at the boundaries between partitions

Goals of BVA Testing

Goals:

- Find errors associated with the boundary values

Test levels:

- Unit
- Integration
- System
- System-integration

This technique is based on the risks:

- A large number of problems occur at the boundaries of the input variables
- Even if found equivalent classes correctly, boundary values may be incorrectly assigned to another class.

Introduction

- Boundary testing is the process of testing between extreme ends or boundaries between partitions of the input values.
- So these extreme ends like Start- End, Lower- Upper, Maximum-Minimum, Just Inside-Just Outside values are called boundary values and the testing is called "boundary testing".
- The basic idea in boundary value testing is to select input variable values at their:
- Minimum
- Just above the minimum
- A nominal value
- Just below the maximum
- Maximum

Boundary Value Analysis

- In software Testing we can identify two major domains to test. Those are INPUTS & OUTPUTS.

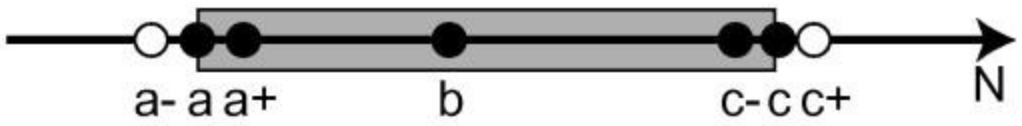
- Here for each of the domain how to use knowledge of the functional nature of a program to identify test cases for the program.

- Input domain testing also called “boundary value testing” is the best-known specification-based testing technique.

Steps for test cases

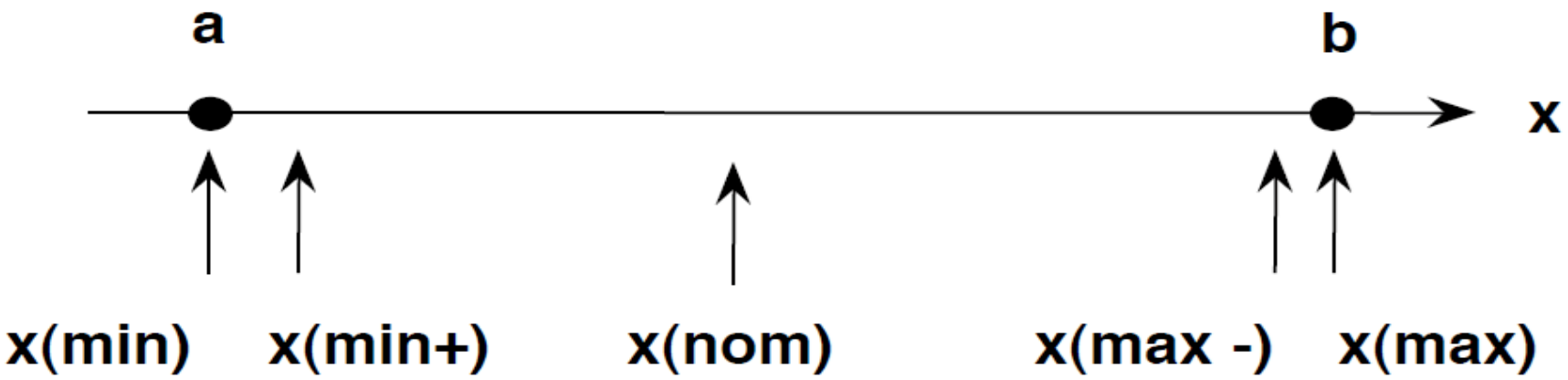
Test case design by BVA proceeds into 3 steps:

- Determine the range of values (usually it is equivalence class)
- Determine boundary values
- Check input variable value at the minimum, just above minimum, just below minimum, normal, at the maximum, just below maximum, just above maximum



Cyclomatic Complexity

- Cyclomatic Complexity=4n+1 yields test cases.



Example 1

- An exam has a pass boundary at 50 percent, merit at 75 percent and distinction at 85 percent. The Valid Boundary values for this scenario will be as follows:
- 49, 50 - for pass
- 74, 75 - for merit
- 84, 85 - for distinction
- Boundary values are validated against both the valid boundaries and **invalid boundaries**.
- The Invalid Boundary Cases for the above example can be given as follows:
- 0 - for lower limit boundary value
- 101 - for upper limit boundary value

Example 2

Example: Password field can not be shorter than 4 and longer than 28 (including) characters (numeric and alphabetic)

Equivalance classes	Boundary Values
0-3	{-1;0;1}, {2;3;4}
4-28	{3;4;5}, {27;28;29}
28+1	

Example 4 on Boundary Value Analysis Test Case Design Technique:

- Assume, we have to text a field which accepts Age 18 – 56

BOUNDARY VALUE ANALYSIS		
Invalid (min -1)	**Valid (min, +min, -max, max)**	**Invalid (max +1)**
17	18, 19, 55, 56	57

TEST CASES

- Minimum boundary value is 18
- Maximum boundary value is 56
- Valid Inputs: 18,19,55,56
- Invalid Inputs: 17 and 57
- Test case 1: Enter the value 17 (18-1) = Invalid
- Test case 2: Enter the value 18 = Valid
- Test case 3: Enter the value 19 (18+1) = Valid
- Test case 4: Enter the value 55 (56-1) = Valid
- Test case 5: Enter the value 56 = Valid
- Test case 6: Enter the value 57 (56+1) =Invalid

Example 2:

- Assume we have to test a text field (Name) which accepts the length between 6-12 characters.

Name Enter Name *Accepts characters length (6 - 12)

BOUNDARY VALUE ANALYSIS		
Invalid (min -1)	**Valid (min, +min, -max, max)**	**Invalid (max +1)**
5 characters	6, 7, 11, 12 characters	13 characters

Test cases

- Minimum boundary value is 6
- Maximum boundary value is 12
- Valid text length is 6, 7, 11, 12
- Invalid text length is 5, 13
- Test case 1: Text length of 5 (min-1) = Invalid
- Test case 2: Text length of exactly 6 (min) = Valid
- Test case 3: Text length of 7 (min+1) = Valid
- Test case 4: Text length of 11 (max-1) = Valid
- Test case 5: Text length of exactly 12 (max) = Valid
- Test case 6: Text length of 13 (max+1) = Invalid

- There are two independent considerations that apply to input domain testing.

  1) Normal boundary value testing

  2) Considering Single Fault Assumption

- Normal Boundary Value Testing is concerned only with valid values of the input variables.
- Robust boundary value testing = (Invalid Variable + Valid Variable) values together.
- The second consideration is whether we make the "single fault" assumption common to reliability theory. This assumes that faults are due to incorrect values of any single variable.

Normal Boundary Value analysis

- Input domain testing is the most commonly taught (and perhaps the most commonly used) software testing technique
- We will see a number of approaches to boundary value analysis
- We will then study some of the limitations of domain testing

Boundary Value Analysis

- Many programs can be viewed as a function *F* that maps values from a set *A* (its domain) to values in another set *B* (its range)
- The input variables of F will have some (possibly unstated) boundaries:

$$F : A \rightarrow B$$

$$a \leq x_1 \leq b$$

$$c \leq x_2 \leq d$$

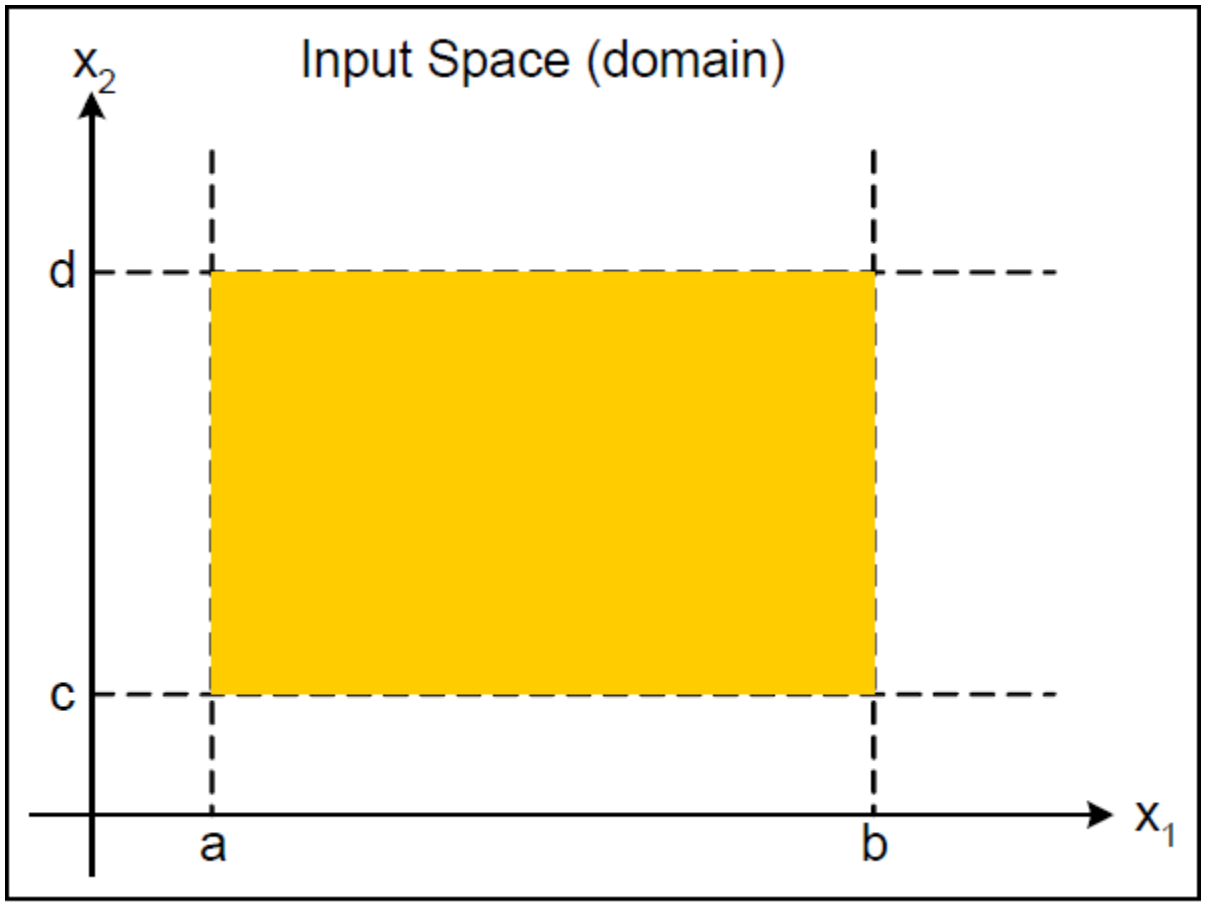
- Lets define two variables ,X^1 and X^2. Where X^1 lies between A and B and X^2 lies between C and D.

- $A \leq X^1 \leq B$

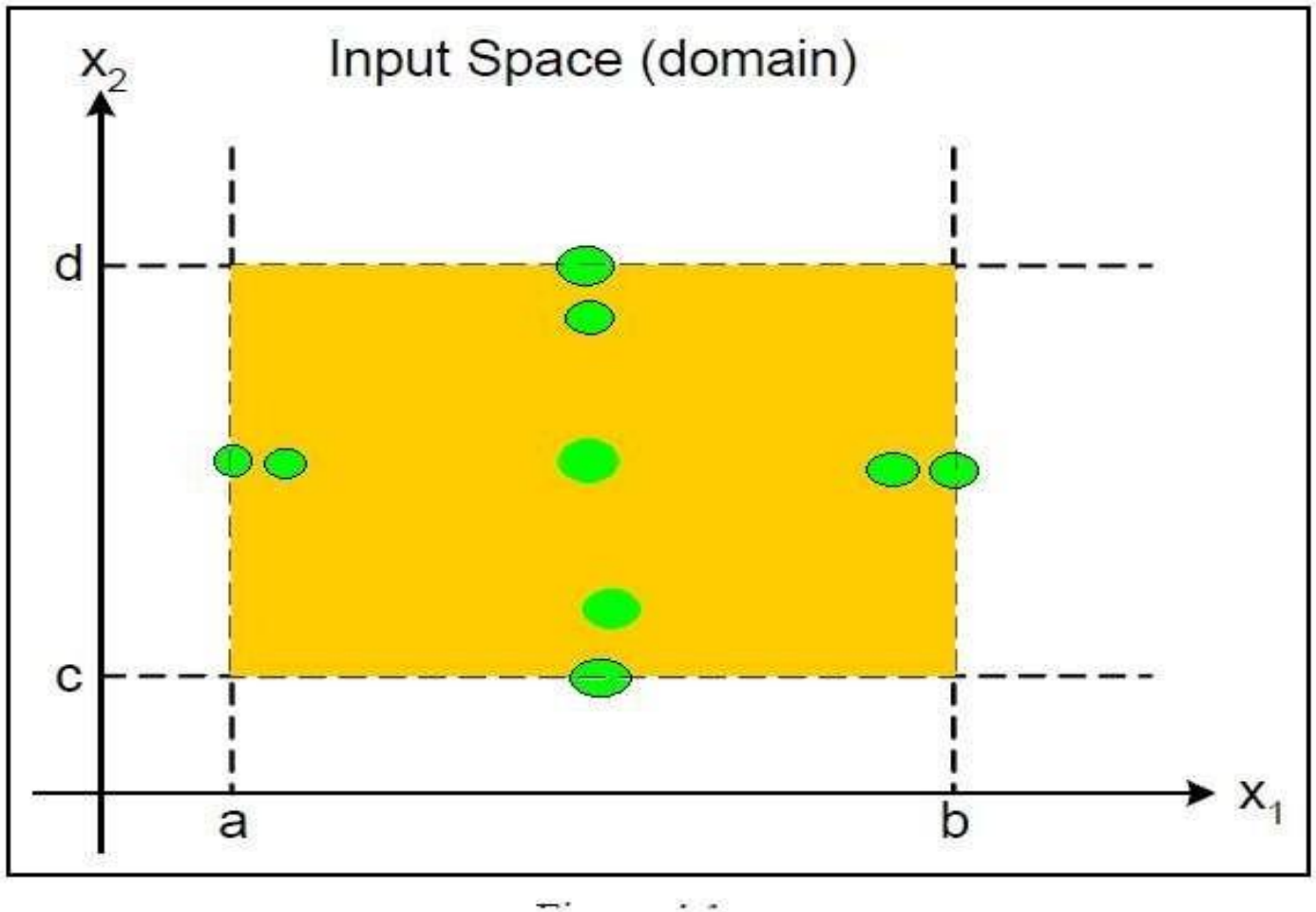
- $C \leq X^2 \leq D$

- The values of A, B, C and D are the extremities of the input domain. These are best demonstrated by the figure.

The yellow shaded area is the input domain.



The green dots represent the boundary values.



The values used to test the extremities are:-

- Min ------------------------------ Minimal
- Min+ ----------------------------Just above Minimal
- Nom ------------------------------ Average
- Max- ----------------------------Just below Maximum
- Max ----------------------------- Maximum

Single fault assumption

- Failures are only rarely the result of the simultaneous occurrence of two (or more) faults
- Generate test cases as such for all i
  - Values of all but one variable x_i at nominal
  - x_i assumes all 5 values from previous slide

Two-variable function test cases

$<x_{1nom}, x_{2min}>$

$<x_{1nom}, x_{2min+}>$

$<x_{1nom}, x_{2nom}>$

$<x_{1nom}, x_{2max-}>$

$<x_{1nom}, x_{2max}>$

$<x_{1min}, x_{2nom}>$

$<x_{1min+}, x_{2nom}>$

$<x_{1nom}, x_{2nom}>$

$<x_{1max-}, x_{2nom}>$

$<x_{1max}, x_{2nom}>$

Let's apply this to the Triangle problem

- The NextDate problem is a function of three variables: day, month and year. Upon the input of a certain date it returns the date of the day after that of the input.
- The input variables have the obvious conditions:
- $1 \leq Day \leq 31$. (**Total number of test cases=4(3)+1=13**)
- $1 \leq month \leq 12$.
- $1812 \leq Year \leq 2012$.

month	day	year
min = 1	min = 1	min = 1812
min+ = 2	min+ = 2	min+ = 1813
nom = 6	nom = 15	nom = 1912
max- = 11	max- = 30	max- = 2011
max = 12	max = 31	max = 2012

Boundary Value Analysis Test Cases

Case	month	day	year	Expected Output
1	6	15	1812	June 16, 1812
2	6	15	1813	June 16, 1813
3	6	15	1912	June 16, 1912
4	6	15	2011	June 16, 2011
5	6	15	2012	June 16, 2012
6	6	1	1912	June 2, 1912
7	6	2	1912	June 3, 1912
8	6	30	1912	July 1, 1912
9	6	31	1912	error
10	1	15	1912	January 16, 1912
11	2	15	1912	February 16, 1912
12	11	15	1912	November 16, 1912
13	12	15	1912	December 16, 1912

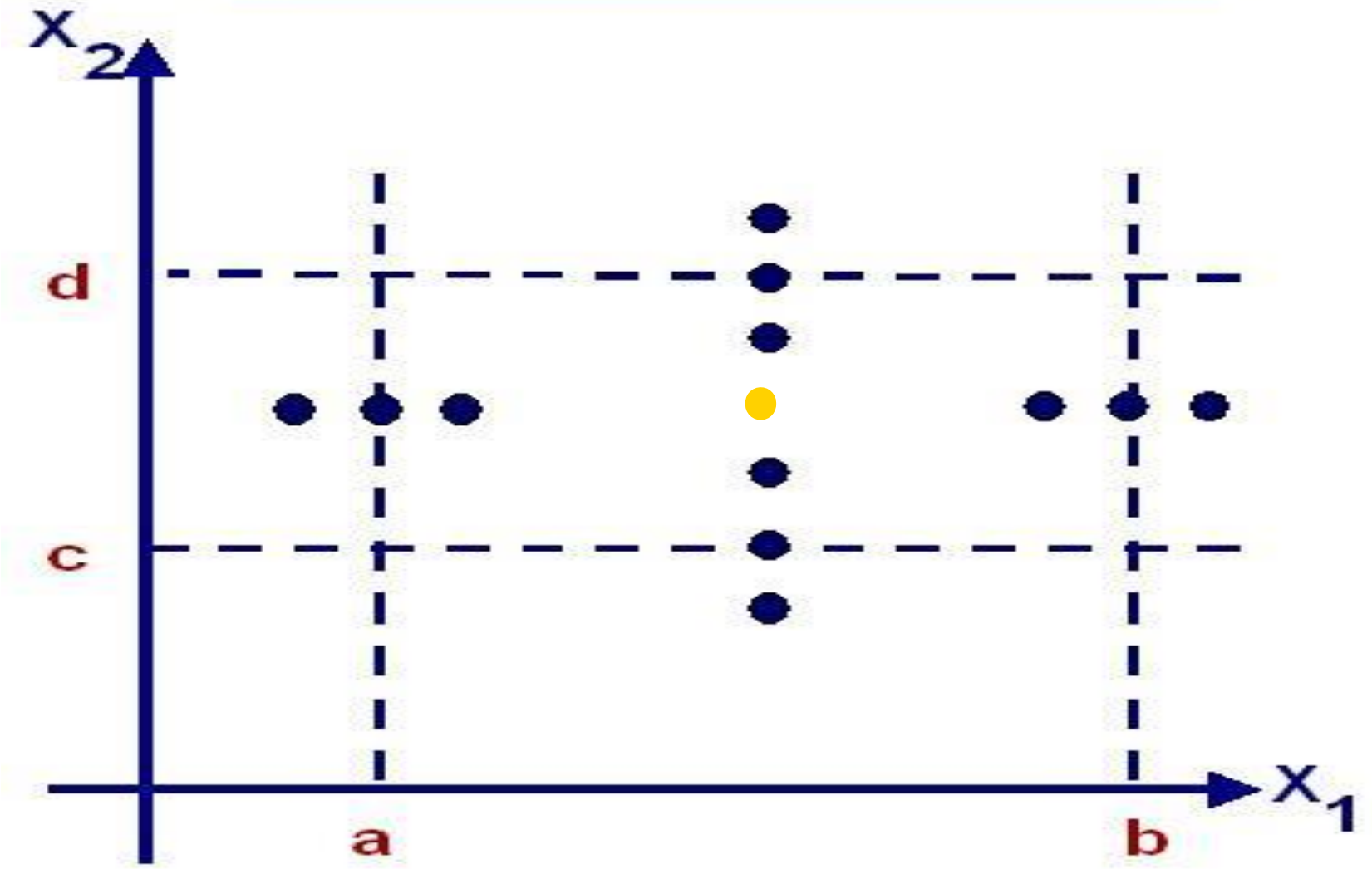
Limitations

- Does not work well for Boolean variables
  - We will see a more suitable approach next week
- Does not work well for logical variables
  - PIN, transaction type
- Assumes independent variables
  - NextDate test cases unsatisfactory

Robustness Testing

- In BVA, we remain within the legitimate boundary of our range i.e. for testing we consider values like (min, min+, nom, max-, max) whereas in Robustness testing, we try to cross these legitimate boundaries as well.
- Thus for testing here we consider the values like (min-, min, min+, nom, max-, max, max+)

Robustness Test Cases



Hierarchy

- Boundary Value testing of n inputs : $4n + 1$
- Robustness testing of n inputs : $6n + 1$
- "Worst case" for boundary value : 5^n
- "Worst case" for robustness : 7^n

1) Boundary Value is a subset of Robustness

2) Worst Case of boundary value is a subset of Worst Case of robustness

Worst case BVT

Worst Boundary value testing on 2 variables.

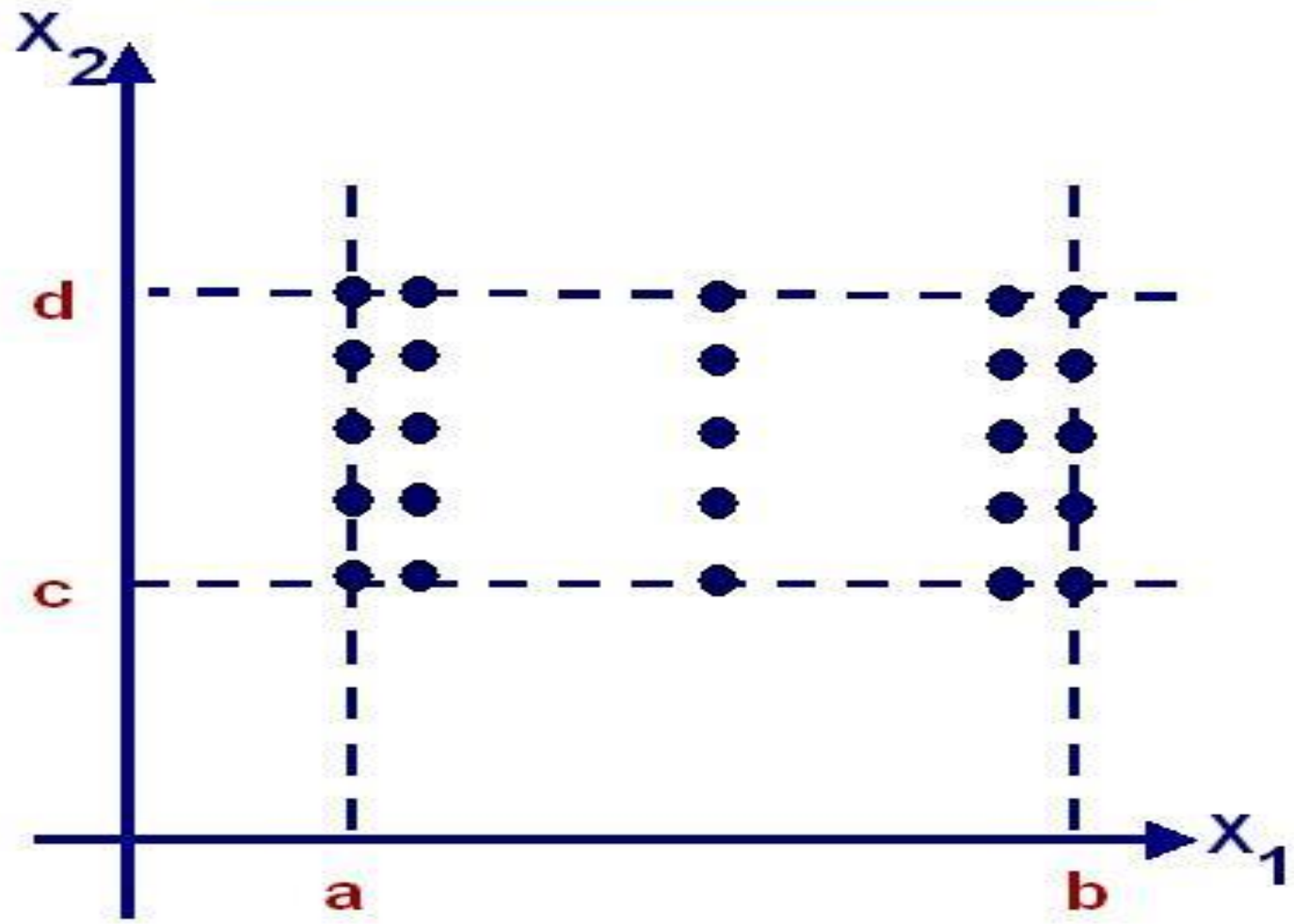
Suppose we have two variables x1 and x2 to test.

The range of x1: 10 to 90

The range of x2: 20 to 70

	x1	x2
Min	10	20
Min+	11	21
Nominal	50	45
Max-	89	69
Max	90	70

Worst-Case Test Cases



Test cases

Total test cases = A*A

25 = 5*5

A= Number of testing points.

These testing points are min, min+, nominal, max- and max.

We can generate 25 test cases from both variables x1 and x2 by making a combination of each value of one variable with each value of another variable.

Test Case#	X1,X2	Test Case #	X1,X2	Test Case #	X1,X2
1	10,20	2	10,21	3	10,45
4	10,69	5	10,70	6	11,20
7	11,21	8	11,45	9	11,69
10	11,70	11	50,20	12	50,21
13	50,45	14	50,69	15	50,70
16	89,20	17	89,21	18	89,45
19	89,69	20	89,70	21	90,20
22	90,21	23	90,45	24	90,69
25	90,70				

Special Value Testing

- The testing executed by Special Value Testing technique is based on past experiences, which validates that no bugs or defects are left undetected.
- Moreover, the testers are extremely knowledgeable about the industry and use this information while performing Special Value testing.
- Another benefit of opting Special Value Testing technique is that it is Ad-hoc in nature. There are no guidelines used by the testers other than their "best engineering judgment".
- The most important aspect of this testing is that, it has had some very valuable inputs and success in finding bugs and errors while testing a software.

Special Value Test Cases

Case	*Month*	*Day*	*Year*	*Reason*
SV-1	2	28	2012	Feb. 28 in a leap year
SV-2	2	28	2013	Feb. 28 in a common year
SV-3	2	29	2012	Leap day in a leap year
SV-4	2	29	2000	Leap day in 2000
SV-5	2	28	1900	Feb. 28 in 1900
SV-6	12	31	2011	End of year
SV-7	10	31	2012	End of 31-day month
SV-8	11	30	2012	End of 30-day month
SV-9	12	31	2012	Last day of defined interval

Random Testing

- Random Testing is a form of functional black box testing that is performed when there is not enough time to write and execute the tests.

Random Testing

- The basic idea in Random Testing is that, rather than always choose the min, min+, nom, max–, and max values of a bounded variable, use a random number generator to pick test case values.
- This avoids any form of bias in testing. It also raises a serious question: how many random test cases are sufficient?
- This is usually done through the software application that uses the Random Number Generator function that picks values for a bounded variable $a \leq x \leq b$.
- For each type of the output several test cases bundled together, it can be shown in percent % of the total test cases. Just an example: 35% of the total 125 test cases used to check the Isosceles triangle, shown in tabular format.

- The table on next slide shows the results of randomly generated test cases. They are derived from a Visual Basic application that picks values for a bounded variable $a \leq x \leq b$.
- The formula used here is: $x = Int((b - a + 1) * Rnd + a)$
- Where x is output i.e. one of 4 columns (Nontriangle,scalance, Isosceles, Equilateral.
- Function Int returns the integer part of a floating point number, and the function Rnd generates random numbers in the interval [0, 1].
- The program keeps generating random test cases until at least one of each output occurs.

Random Testing Characteristics

- Random testing is performed where the defects are NOT identified in regular intervals.
- Random input is used to test the system's reliability and performance.
- Saves time and effort than actual test efforts.
- Other Testing methods Cannot be used to.

Random Test Cases for Triangle Program

Test Cases	Nontriangles	Scalene	Isosceles	Equilateral
1289	663	593	32	1
15,436	7696	7372	367	1
17,091	8556	8164	367	1
2603	1284	1252	66	1
6475	3197	3122	155	1
5978	2998	2850	129	1
9008	4447	4353	207	1
Percentage	49.83%	47.87%	2.29%	0.01%

of Random Test Cases Generated

Random Test Cases for Commission Program

Test Cases	10%	15%	20%
91	1	6	84
27	1	1	25
72	1	1	70
176	1	6	169
48	1	1	46
152	1	6	145
125	1	4	120
Percentage	1.01%	3.62%	95.37%

of Random Test Cases Generated