Positive Negative Testing

Two approaches to testing software

- Test to pass
- Test to fail

Which to do first?

Positive and Negative testing

- behaves as expected

- Valid Input Data
- does what it is supposed to do so

- Invalid Input Data
- does not do anything that it is not supposed to do so

Equivalence Class Partitioning

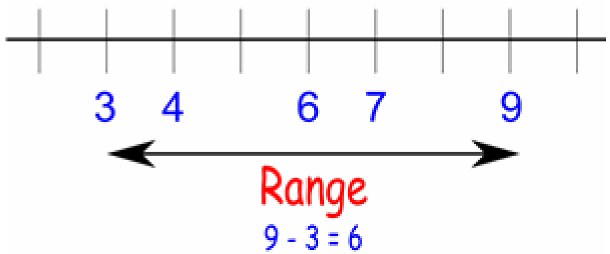
Equivalence Class Partitioning

- Partition
  - Set of inputs that generate one single expected output
- Equivalence class
  - Behavior of the software is same for a set of values.

Important points to consider

- Both valid and invalid Equivalence class
- May also be selected for output conditions
- No hard and fast rules for deriving equivalence class, it just gives the tester the guidelines for partitioning.

List of Conditions

- 

- Number of values
- 

- The first character of a part identifier **MUST BE** a letter

Example

- Function square_root
- message (x:real)
- when x >_0.0
- reply (y:real)
- where y >_0.0 & approximately (y*y,x)
- otherwise reply exception imaginary_square_root
- end function

EC1.The input variable x is real, valid.
EC2. The input variable x is not real, invalid.
EC3.The value of x is > 0.0, valid.
EC4.The value of x is <0.0, invalid.

- x and y are input/output variables to write equivalence class

Good Approach

- Unique identifier for each equivalence class
- Develop test cases for all valid equivalence classes until all have been covered by (included in) a test case.
- Develop test cases for all invalid equivalence classes until all have been covered individually.

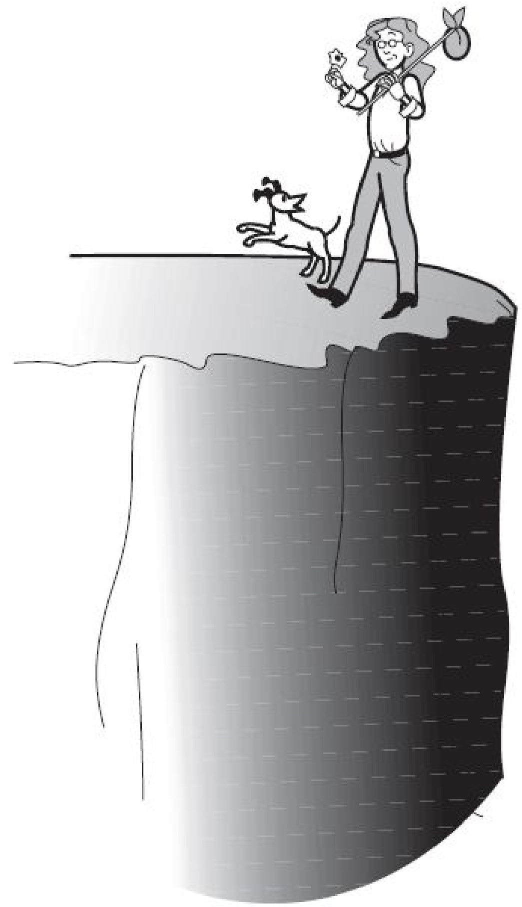
Advantages

- This technique has the following advantages:
  - It eliminates the need for exhaustive testing, which is not feasible.
  - It guides a tester in selecting a subset of test inputs with a high probability of detecting defect.
  - It allows a tester to cover a larger domain of inputs/outputs with a smaller subset selected from an equivalence class.

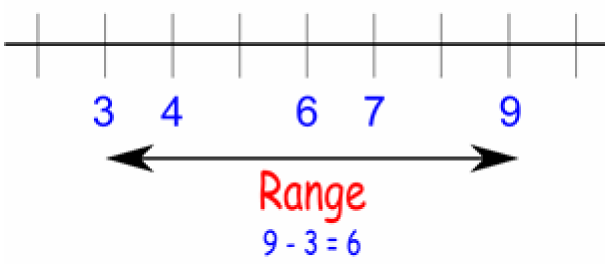
Boundary Value Analysis

Boundary Value Analysis

- Most of the errors hover around boundaries and conditions.



Conditions

- 

  - Valid test cases for the ends of the range, and
  - In-valid test cases for possibilities just above and below the ends of the range.
- Number of values
  - Valid test cases for the minimum and maximum numbers
  - Invalid test cases that include one lesser and one greater than the maximum and minimum.
- 

  - First and last elements of the set

Example

- Suppose we are testing a module that allows a user to enter new widget identifiers into a widget data base. We will focus only on selecting equivalence classes and boundary values for the inputs. The input specification for the module states that a widget identifier should consist of 3–15 alphanumeric characters of which the first two must be letters. We have three separate conditions that apply to the input: (i) it must consist of alphanumeric characters, (ii) the range for the total number of characters is between 3 and 15, and, (iii) the first two characters must be letters.

- First we consider condition 1, the requirement for alphanumeric characters. This is a "must be" condition. We derive two equivalence classes.
  - EC1. Part name is alphanumeric, valid.
  - EC2. Part name is not alphanumeric, invalid.
- Then we treat condition 2, the range of allowed characters 3–15.
  - EC3. The widget identifier has between 3 and 15 characters, valid.
  - EC4. The widget identifier has less than 3 characters, invalid.
  - EC5. The widget identifier has greater than 15 characters, invalid.
- Finally we treat the "must be" case for the first two characters.
  - EC6. The first 2 characters are letters, valid.
  - EC7. The first 2 characters are not letters, invalid

Condition	Valid equivalence classes	Invalid equivalence classes
1	EC1	EC2
2	EC3	EC4, EC5
3	EC6	EC7

- BLB—a value just below the lower bound
- LB—the value on the lower boundary
- ALB—a value just above the lower boundary
- BUB—a value just below the upper bound
- UB—the value on the upper bound
- AUB—a value just above the upper bound
- BLB—2 BUB—14
- LB—3 UB—15
- ALB—4 AUB—16

Module name: **Insert_Widget**
Module identifier: **AP62-Mod4**
Date: **January 31, 2000**
Tester: **Michelle Jordan**

Test case identifier	Input values	Valid equivalence classes and bounds covered	Invalid equivalence classes and bounds covered
1	abc1	EC1, EC3(ALB) EC6	
2	ab1	EC1, EC3(LB), EC6	
3	abcdef123456789	EC1, EC3 (UB) EC6	
4	abcde123456789	EC1, EC3 (BUB) EC6	
5	abc*	EC3(ALB), EC6	EC2
6	ab	EC1, EC6	EC4(BLB)
7	abcdefg123456789	EC1, EC6	EC5(AUB)
8	a123	EC1, EC3 (ALB)	EC7
9	abcdef123	EC1, EC3, EC6 (typical case)	

Thank you