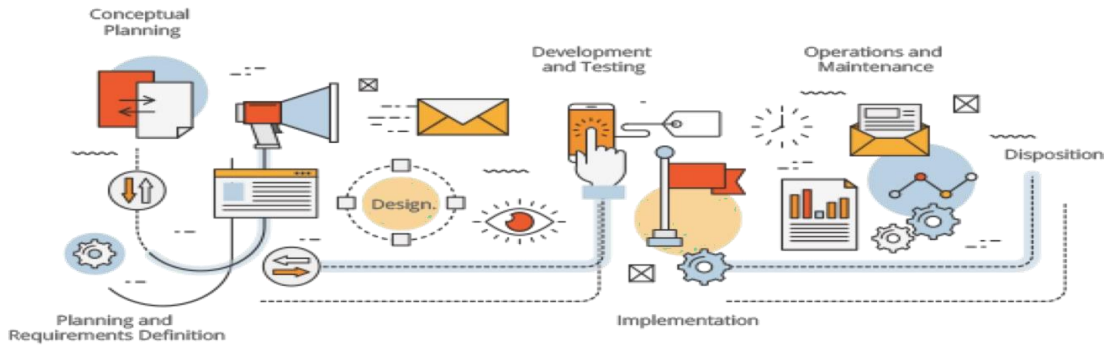
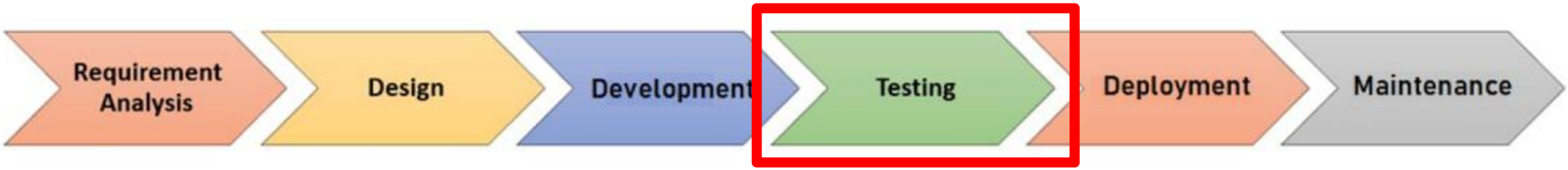
Software Engineering

Requirement Engineering



Software Development Life Cycle (SDLC)



Functional Testing

- Functional testing is a type of software testing in which the system is tested against the functional requirements of the system.
- Ensures that the requirements are properly satisfied by the application.
- Functional testing verifies that each function of the software application works in conformance with the requirement and specification.
- Boundary Value Analysis(BVA) is one of the functional testing.

Equivalence Classes and Boundary value analysis

Equivalence Classes:

- **Purpose:** Equivalence Classes aim to simplify testing by grouping inputs that are expected to produce similar results.
- **Example:** Consider a function that accepts a numerical input representing age. Equivalence Classes might **partition ages into groups** such as minors (0-17), adults (18-64), and seniors (65+). **Testing within each group should yield similar outcomes.**

Boundary Value Analysis (BVA):

- **Purpose:** Boundary Value Analysis **targets the edges or boundaries of Equivalence Classes** to identify potential errors.
- **Example:** Using the age example, if the minimum age requirement for a service is 18, BVA would test **inputs just below (17) and just above (19)** this boundary to ensure the system behaves correctly.

Boundary Value Analysis (BVA)

- Boundary value analysis is one of the widely used case design technique for black box testing. It is used to test boundary values because the input values near the boundary **have higher chances of error.**

- Whenever we do the testing by boundary value analysis, the **tester focuses** on, while entering **boundary value** whether the software is producing correct output or not.

- The basic assumption of boundary value analysis is, the **test cases that are created using boundary values are most likely to cause an error**.

- The boundaries of the partition covering all the testing defects where equivalence testing alone was difficult to handle those defects.

Why Equivalence & Boundary Analysis Testing

1. This testing is used to reduce a very large number of test cases to manageable chunks.

2. Very clear guidelines on determining test cases without compromising on the effectiveness of testing.

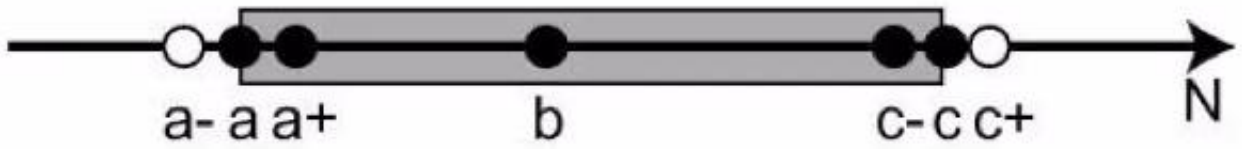
3. Appropriate for calculation-intensive applications with a large number of variables/inputs

- 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.

- Use of special techniques that can select test cases intelligently from the pool of test-case, such that all test scenarios are covered.
  - **Equivalence Partitioning**
  - **Boundary Value Analysis**

Test case design by BVA proceeds into 3 steps:

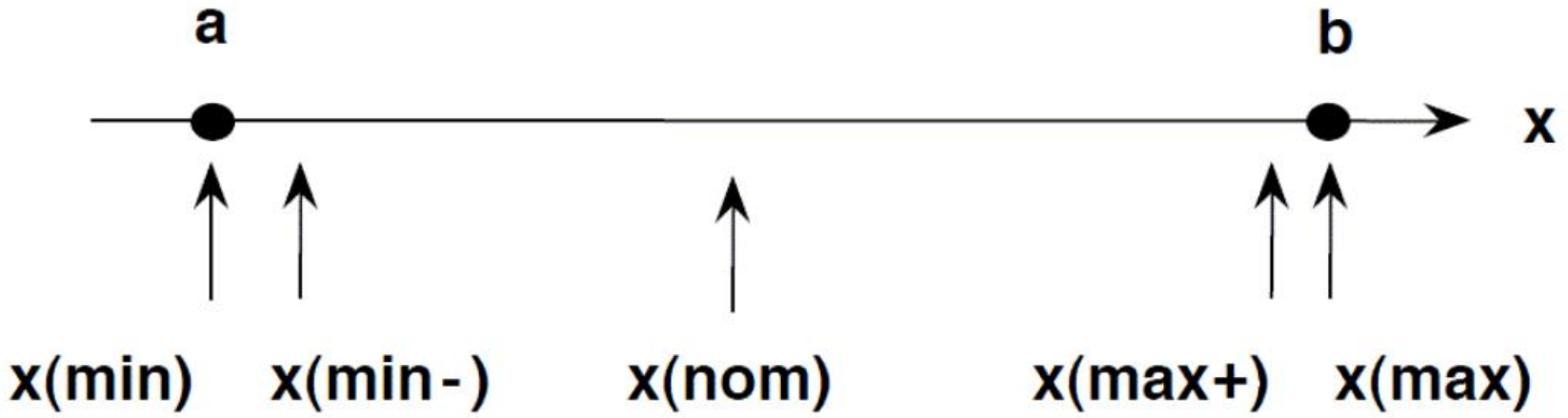
- Determine the range of values (usually it is equivalence class)
- Determine boundary values
- Check input variable value at the minimum, just above minimum, just below minimum, normal, at the maximum, just below maximum, just above maximum



Boundary Value Testing

For each variable we check-

- Minimum value.
- Just above the minimum.
- Nominal Value: values that are within the valid range of the domain
- Just below Max value.
- Max value.



Example: Consider a system that accepts ages from 18 to 56.

Boundary Value Analysis(Age accepts 18 to 56)		
Invalid (min-1)	Valid (min, min + 1, nominal, max – 1, max)	Invalid (max + 1)
17	18, 19, 37, 55, 56	57

Valid Test cases:

- Enter the value- 18.
- Enter the value- 19.
- Enter the value- 37.
- Enter the value- 55.
- Enter the value- 56.

Invalid Test cases:

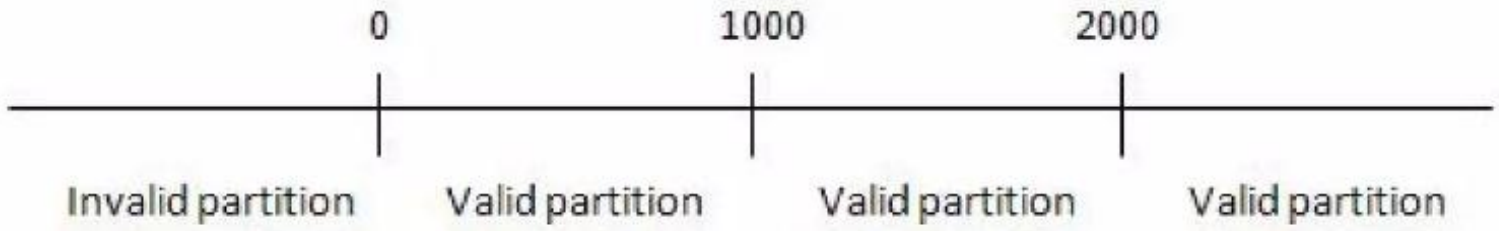
- Enter the value- 17.
- Enter the value- 57.

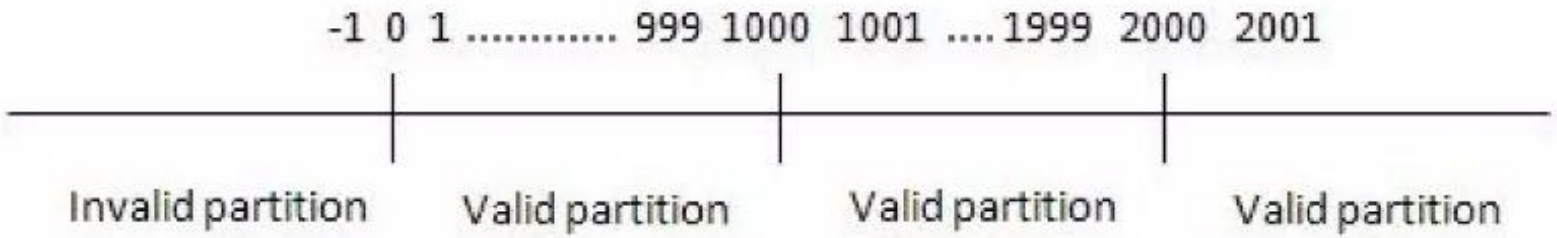
Example: Password field can not be shorter than 4 and longer than 28 (including) characters (numeric and alphabetic)

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

Example: bank has different charges depending on the transaction done.

- 5% of the amount for transaction less than or equal to 1000
- 6% of the amount for transaction more than 1000 and less than or equal to 2000
- 7% of the amount for transaction more than 2000





Example: The program selects candidates for military service focusing only on age (not less 18 and no more 25)

Conscription in the army

Not conscription	Conscription	Not conscription
17	18 25	26

Boundary values: {17;18,19} {24;25;26}

Example: In a system designed for postal services' payment: Letters up to 100g are called as 'light'.Postal rates for sending the light letters up to 10g are $25. The next 40g should be played by $35. Each next 25g up to 100g should be played by an extra $10. Partitions should be designed for "grams".

Equivalance classes:	Boundary Values
0 – 9g: 25$	{-1, 0, 1}, {8, 9, 10},
10-49g: 35$	{9, 10, 11}, {48,49,50}
50-74g:10$	{49, 50, 51}, {73,74,75}
75-99g: 10$	{74,75,99}, {98,99,100}
100+1g: not a light letters	{99,100,101}

Example: we test module for HR, which determines to hire the candidate or not, based on the age of the candidate:

- 0-16 (Do not Hire); 16-18
- (Can hire only part time);
- 18-55 (Can hire on full time);
- 55-99(Do not Hire)

Equivalance classes:	Boundary Values
0-15: Do not Hire	{-1, 0, 1}, {14, 15, 16},
16-17: Can hire only part time	{17, 18, 19},
18-54: Can hire on full time	{54, 55, 56},
55-99: Do not Hire	{98, 99, 100}