Sri Lanka Institute of Information Technology

SOFTWARE ENGINEERING PROCESS AND QUALITY MANAGEMENT

Lecture 2 – Specification Based Test Case Design Techniques

Introduction to Specification Based Testing

Key Characteristics:

- Focuses on **what** the system should do (not how).
- Independent of internal structure (**Black Box Testing/ Behaviour Based Testing**).
- Uses **functional requirements** as a reference.

Why Use Specification-Based Testing?

- Ensures correctness.
- Covers diverse inputs and outputs.
- Helps catch missing or unclear requirements early.
- Doesn’t require coding knowledge.

Types of Specification Based Test Case Design Techniques

- Equivalence partitioning
- Boundary value analysis
- Decision Table

Equivalence Partitioning

- Divide inputs into groups that are expected to behave the same way.
- Reduces the number of test cases while ensuring sufficient coverage.
- Example: Testing an age-based system that allows users aged **18-60**.
  - **Invalid Partition**: Age < 18
  - **Valid Partition**: Age 18-60
  - **Invalid Partition**: Age > 60

Equivalence Partitioning

Equivalence partitioning works on certain assumptions;

- The system will handle all the test input variations within a partition in the same way.
- If one of the input conditions passes, then all other input conditions within the partition will also pass.
- If one of the input conditions fails, then all other input conditions within the partition will also fail

Example;

Fitness 1st gym provides membership for the clients between the age of 16 to 60. It has the following online application form for getting the gym membership In this form

- User has to fill the age first
- User can go further only if the age is between 16 to 60
- Otherwise, there will be a message saying that you cannot get a membership

Form has to be tested with values

- Less than 16
- Between 16 to 60
- More than 60

How many combinations are required to say that the functionality works safely?

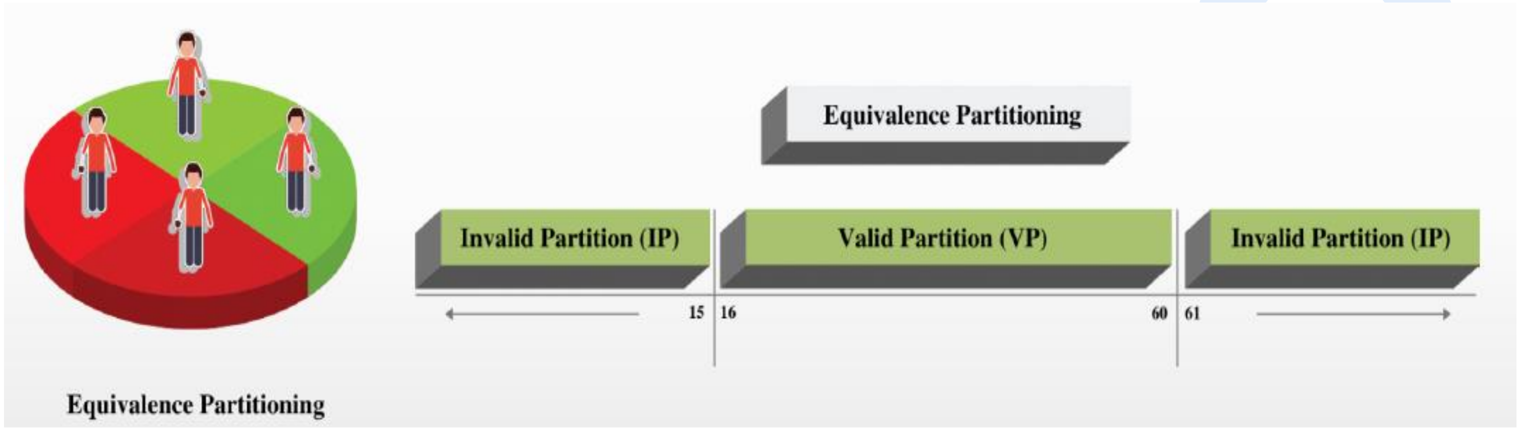
- Less than 16 has 16 combinations from 0 – 15
- 16 to 60 has 45 combinations
- Greater than 60 has 40 combinations (if only consider till 100)

Is it possible to test all these 100 combinations?

How to do Equivalence Partitioning

- Valid partitions are values that the component or system should accept under test. This partition is called a "**Valid Equivalence Partition**."
- Invalid partitions are values that should be rejected by the component or system under test. This partition is called the "**Invalid Equivalence Partition**."

How to do Equivalence Partitioning

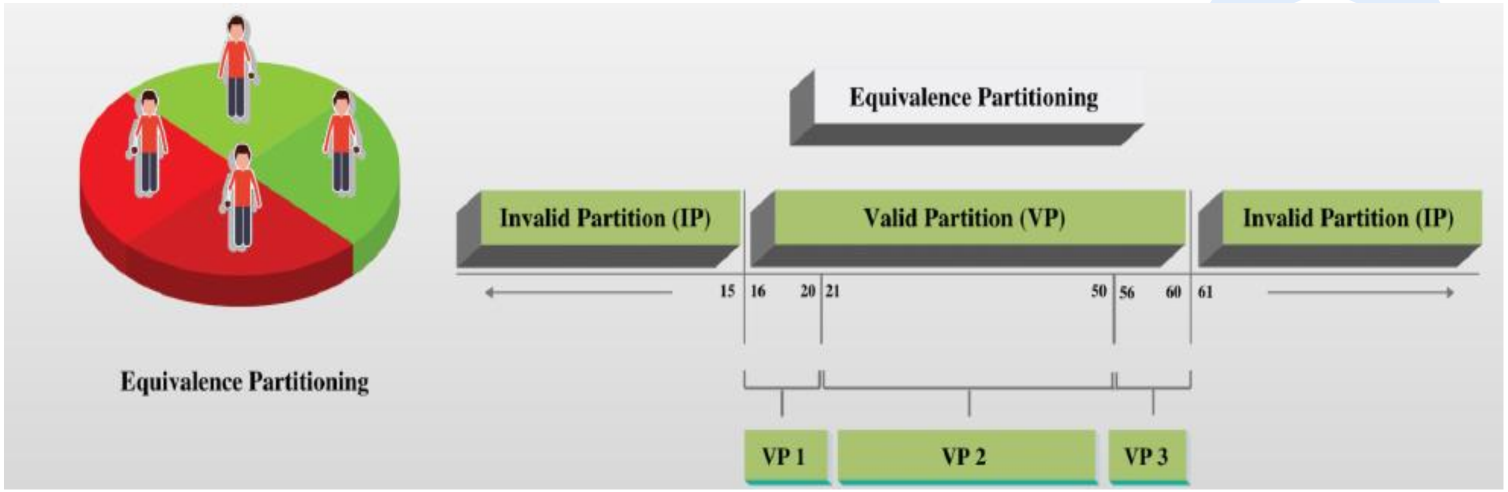


Equivalence Partitioning

These partitions can be further divided into sub partitions if required.

Assume that if the age is 16 to20 years old or 55 to 60 years old, there is an additional requirement to attach a proof of the age, while submitting the membership form In this case, how will the partitions look like?

Equivalence Partitioning



Equivalence Partitioning

VP 1, VP 2, and VP 3 are all valid sub partitions based on the additional requirements.

Therefore, test conditions could be,
Enter Age 5
Enter Age 18
Enter Age 30
Enter Age 58
Enter Age 65

Equivalence Partitioning

- Unique Partitions – Ensure that each test value belongs to only one partition (no overlap).
- Complete Coverage – Test cases should cover all identified partitions.
- Coverage Measurement – Calculated as:

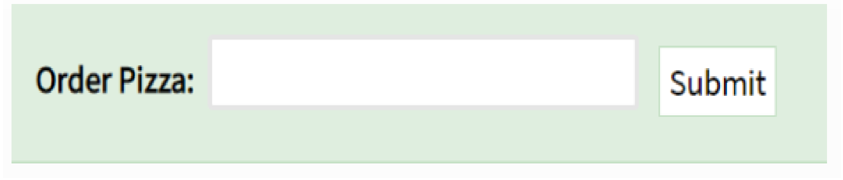
$$\text{Test Coverage} = \frac{\text{Partitions Tested}}{\text{Total Recognized Partitions}} \times 100\%$$

Drawbacks of Equivalence Partitioning

- Depends on Correct Partitioning
- Limited to Stated Requirements
- No Insight into Code Implementation

Equivalence Partitioning - Exercise

Consider the behavior of “Order Pizza” text box given below. Requirement of the above function



- Pizza values 1 to 10 are considered valid. A success message is shown
- Pizza values less than 1 are considered invalid and an error message will appear, “Please enter a valid count“
- Pizza values grater than 10 are considered invalid and an error message will appear, "Only 10 Pizzas can be ordered at a time“.

Perform equivalence partitioning and identify the test conditions with sample values for the above function

Equivalence Partitioning - Answer

Test conditions:

Any number less than 1 that is 0 or below is considered invalid.
Enter Pizza Value = -1

Numbers 1 to 10 are considered valid.
Enter Pizza Value = 5

Any Number greater than 10 is considered invalid.
Enter Pizza Value = 15

Boundary Value Analysis

- A software testing technique in which tests are designed to include representatives of boundary values in a range.
- An extension of equivalence partitioning.
- Testing the boundaries of partitions.
- Useable only when the partition is ordered, consisting of numeric or sequential data.
- The minimum and maximum values of a partition are its boundary values.

Boundary Value Analysis

- There are high chances of finding the defects at the boundaries of a partition
- Equivalence partitioning alone is not sufficient to catch such defects
- Boundary Value Analysis was designed to detect anomalies at the boundaries of a partition

How to do Boundary Value Analysis

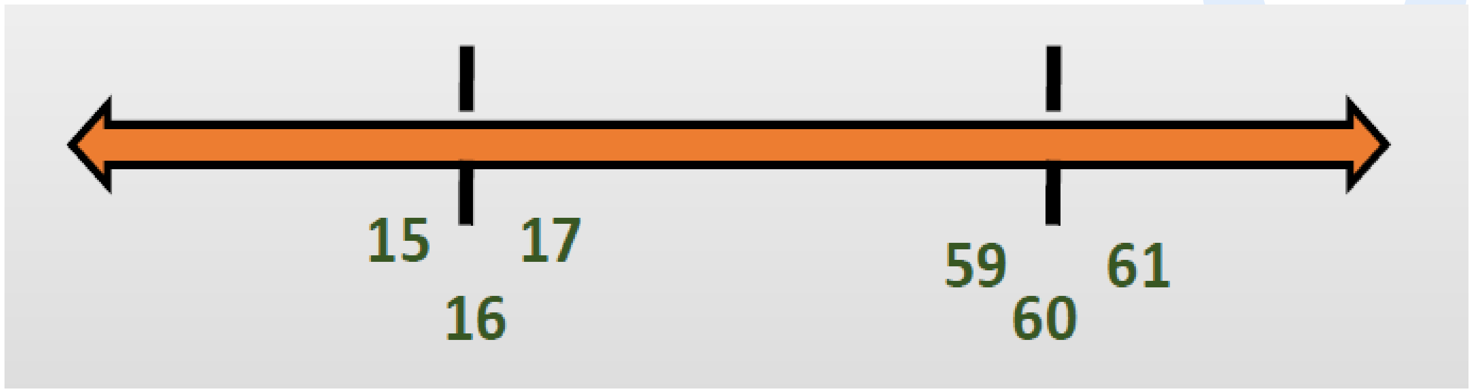
Refer to the same example of Fitness 1st gym form (on equivalence partitioning) where the user is required to enter the age.

- The first step of boundary value analysis is to create the equal partitions
- Concentrate on the valid partition, which ranges from 16 60

Boundary Value Analysis

- There is a three-step approach to identify boundaries
  1. Identify the exact boundary value of the partition Class - 16 and 60
  2. Get the boundary value which is one less than the exact boundary - 15 and 59
  3. Get the Boundary Value which is one more than the precise Boundary - 17 and 61

Boundary Value Analysis



Boundary Value Analysis

- Following are the combinations for boundary value for the age criteria.

- Valid Boundary Conditions - Age 16, 17, 59, 60
- Invalid Boundary Conditions - Age 15, 61

- Valid boundary conditions fall under valid partition class and invalid boundary conditions fall under invalid partition class.

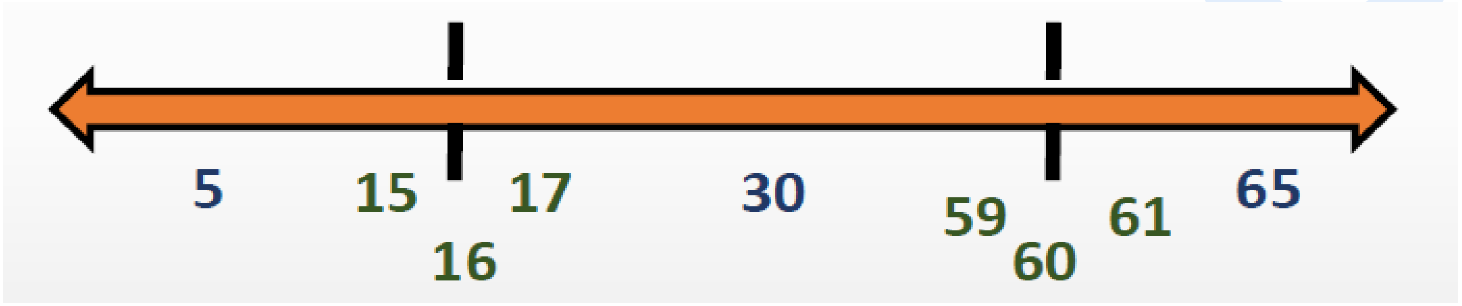
Boundary Value Analysis with Equivalence partitioning

Boundary Value Analysis is combined with equivalence partitioning to get a full set of test conditions refer to the previous example.

The range is from 16 to 60;

- Equivalence partition analysis gives test conditions as 5, 30, 65
- Boundary value analysis gives test conditions as 15, 16, 17, 59, 60, 61.

Boundary Value Analysis with Equivalence partitioning



All test conditions: 5, 15, 16, 17, 30, 59, 60, 61, 65

Drawbacks of Boundary Value Analysis

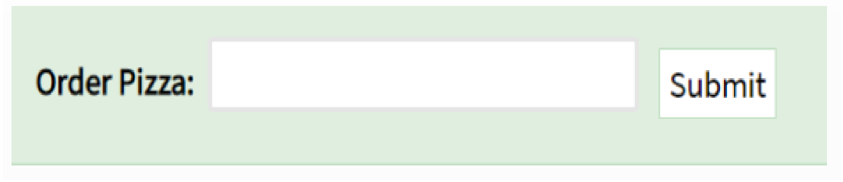
- Boundary value and equivalence partitioning assume that the application will not allow you to enter any other characters or values This assumption is, however, not valid for all applications
- Boundary value analysis cannot handle situations where the
decision depends on more than one input values

Example

If the Gym form has another field Male and Female, and the age limit will vary according to that selection

Boundary Value Analysis - Exercise

Consider the behavior of “Order Pizza” text box given below. Requirement of the above function

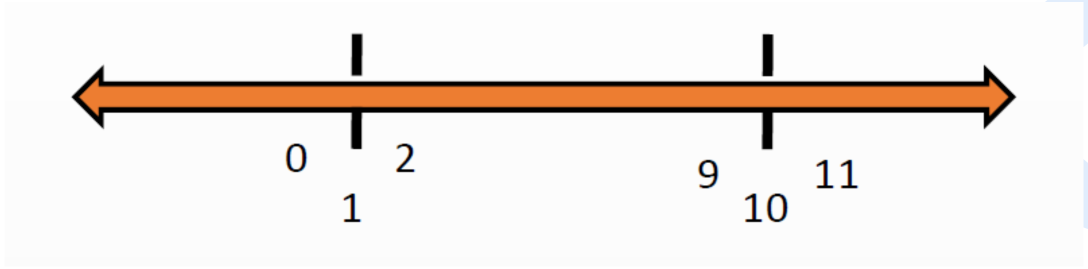


- Pizza values 1 to 10 are considered valid. A success message is shown
- Pizza values less than 1 are considered invalid and an error message will appear, “Please enter a valid count“
- Pizza values grater than 10 are considered invalid and an error message will appear, "Only 10 Pizzas can be ordered at a time“.

Perform boundary value analysis and identify the boundary value test conditions with sample values for the above function.

Boundary Value Analysis - Answer

Answer



Boundary value analysis test conditions:

Value = 0 - Invalid

Value = 1 - Valid

Value = 2 - Valid

Value = 9 - Valid

Value = 10 - Valid

Value = 11 - Invalid

Decision Table

- Software Testing technique in which tests are more focused on business logic or business rules
- A decision table is a good way to deal with combinations of inputs
- Decision tables provide a systematic way of stating complex business rules, which is useful for developers as well as for testers
- Decision tables can be used in test design whether or not they are used in specifications, as they help testers explore the effects of combinations of different inputs and other software states that must correctly implement business rules

How to use Decision Table to design test case designing?

1. Identify a suitable function or subsystem which reacts according to a combination of inputs or events. The system should not contain too many inputs otherwise the number of combinations will become unmanageable

Decision Table

Example

A gym membership application allows users to either enter the **total amount they are willing to pay** or the **number of months they want the membership for**.

If the user enters the **total amount**, the application calculates how many months the membership will last based on the monthly fee.

If the user enters the **number of months**, the application calculates the total cost of the membership.

Decision Table

2. Identify the conditions.

- Enter the total amount.
- Enter the number of months.

Decision Table

3. Identify all of the combinations (of True and False)

$$\text{Total Combinations} = 2^{\text{Number of Conditions}}$$

Conditions	Rule 1	Rule 2	Rule 3	Rule 4
Enter total amount				
Enter number of months				

Sri Lanka Institute of Information Technology

Decision Table

4. Identify the correct outcome for each combination. In this example, user can enter one or both of the two fields. Each combination is also referred to as a rule.

Conditions	Rule 1	Rule 2	Rule 3	Rule 4
Enter total amount	T	T	F	F
Enter number of months	T	F	T	F
Actions/ Outcomes				
Calculate how many months for membership	Y	Y		
Total cost of membership	Y		Y	

Decision Table

- What happens if the customer doesn't enter anything in either of the two fields?

- The table has depicted a combination that was not mentioned in the specification.

- Can assume that this combination should result in an error message.

Therefore, add another action to the decision table

Decision Table

Conditions	Rule 1	Rule 2	Rule 3	Rule 4
Enter total amount	T	T	F	F
Enter number of months	T	F	T	F
Actions/ Outcomes				
Calculate how many months for membership	Y	Y		
Total cost of membership	Y		Y	
Error message				Y

Decision Table

What if the customer is not allowed to enter both the total amount and the number of months?

Decision Table

Conditions	Rule 1	Rule 2	Rule 3	Rule 4
Enter total amount	T	T	F	F
Enter number of months	T	F	T	F
Actions/ Outcomes				
Calculate how many months for membership		Y		
Total cost of membership			Y	
Error message	Y			Y

Decision Table

Notice that there is only one ‘Yes’ for the actions in each column

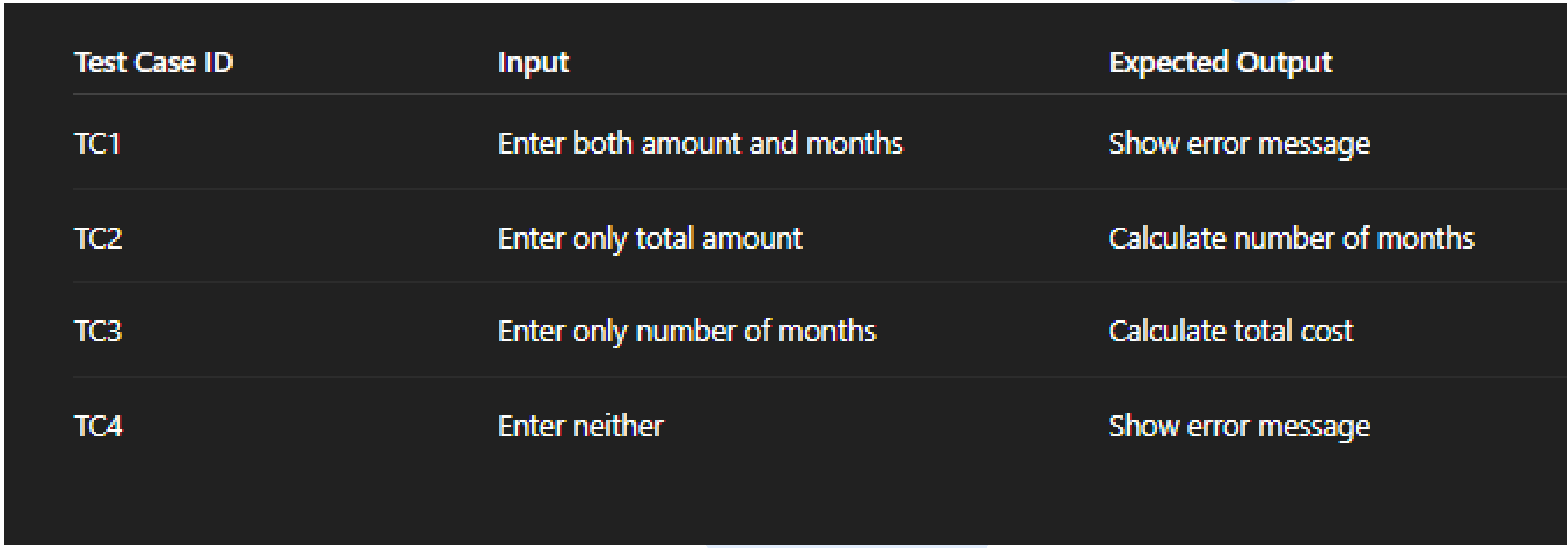
- This is called a mutually exclusive action - only one action occurs for each combination of conditions

Decision Table

Conditions	Rule 1	Rule 2	Rule 3	Rule 4
Enter total amount	T	T	F	F
Enter number of months	T	F	T	F
Actions/ Outcomes				
Result	Error message	Calculate number of months for membership	Calculate total cost for membership	Error message

Decision Table

5. The final step of this technique is to write test cases to exercise each of the four rules in our table



Test Case ID	Input	Expected Output
TC1	Enter both amount and months	Show error message
TC2	Enter only total amount	Calculate number of months
TC3	Enter only number of months	Calculate total cost
TC4	Enter neither	Show error message

Decision Table - Exercise

A customer can use a credit card with three conditions as follows

- New customers will get a 15% discount on all purchases today
- If the customer is an existing customer and holds a loyalty card, then he or she will get a 10 discount
- If the customer has a coupon, then he or she can get 20 off today (but it can't be used with the 'new customer' discount)

Create the decision table for the above scenario

THANK YOU!