Test Design techniques used by QA predominantly.

Since exhaustive testing is not possible; Manual Testing Techniques help reduce the number of test cases to be executed while increasing test coverage. They help identify test conditions that are otherwise difficult to recognize.

## **Boundary Value Analysis (BVA)**

Boundary value analysis is based on testing at the boundaries between partitions. It includes maximum, minimum, inside or outside boundaries, typical values and error values.

If an input condition is restricted between values x and y, then the test cases should be designed with values x and y as well as values which are above and below x and y

Input condition is valid between 1 to 10

Boundary values 0,1,2 and 9,10,11

## **Equivalence Class Partitioning**

Equivalent Class Partitioning allows you to divide a set of test conditions into a partition. And from each partition/class a value is picked for validation.

Input conditions are valid between

1 to 10 and 20 to 30

Hence there are five equivalence classes

--- to 0 (invalid)

1 to 10 (valid)

11 to 19 (invalid)

20 to 30 (valid)

31 to --- (invalid)

You select values from each class, i.e.,

-2, 3, 15, 25, 45

## **Decision Table Based Testing**

A decision table is also known as a Cause-Effect table. This software testing technique is used for functions which respond to a combination of inputs or events. For example, a submit button should be enabled if the user has entered all required fields.

| **Conditions** | **Rule 1** | **Rule 2** | **Rule 3** | **Rule 4** |
| --- | --- | --- | --- | --- |
| **Username (T/F)** | W | V | W | V |
| **Password (T/F)** | W | W | V | V |
| **Output (E/H)** | E | E | E | H |

**Legend:**

* **V** – Valid or Correct username/password
* **W** – Wrong username/password
* **E** – Error message is displayed
* **H** – Home screen is displayed

## **State Transition**

It is used to capture the behavior of the software application when different input values are given to the same function.

|  | Correct Password | Incorrect Password |
| --- | --- | --- |
| Attempt 1- S1 | S5 | S2 |
| Attempt 2 - S2 | S5 | S3 |
| Attempt 3 - S3 | S5 | S4 |
| Attempt 4 - S4 | S5 | S6 |
| Access Given - S5 |  |  |
| Blocked - S6 |  |  |

**Error Guessing**

Error guessing is a technique in which there is no specific method for identifying the error. It is based on the experience of the test analyst, where the tester uses the experience to guess the problematic areas of the software. It is a type of black box testing technique which does not have any defined structure to find the error.

The main purpose of this technique is to identify common errors at any level of testing by exercising the following tasks:

* Enter blank space into the text fields.
* Enter invalid parameters.
* Divide by zero.
* Use the maximum limit of files to be uploaded.
* Check buttons without entering values.

**Use Case**

It is a description of how a specific user uses the system. This method is widely used in developing tests at systems or acceptance levels.

A use case is a pictorial representation of how the user interacts with a system or product. It is a user-driven approach instead of a system-driven one. It describes the success scenarios, error scenarios, and any critical variations or exceptions to be handled. It can be written in a document or made as a visual with the help of a use case model.

### **Conclusion**

* Test Case Design Technique allows you to design better cases. There are five primarily used techniques.
* Boundary value analysis is testing at the boundaries between partitions.
* Equivalent Class Partitioning allows you to divide a set of test conditions into a partition which should be considered the same.
* Decision Table software testing technique is used for functions which respond to a combination of inputs or events.
* In State Transition technique changes in input conditions change the state of the Application Under Test (AUT)
* Error guessing is a software testing technique which is based on guessing the error which can prevail in the code.

## **What are the types of test case design techniques?**

These test case design techniques can be classified into three major groups: :

* **Specification-based (Black Box)**
* **Structure-based (White Box)**
* **Experience-based (Grey Box)**
* **Specification(Requirement) -based Test Design Techniques**:
  + **Equivalence Partitioning**: Involves dividing input data into partitions or groups to ensure that the test cases are representative of each partition.
  + **Boundary Value Analysis**: Focuses on testing values at the boundaries of input domains, as these often represent areas where defects are more likely to occur.
  + **Decision Table Testing**: Uses decision tables to model complex business rules and conditions, helping to identify different combinations of inputs and their expected outcomes.
  + **State Transition Testing**: Suitable for systems with different states, this technique involves testing transitions between states to ensure the system behaves as expected.
* **Structure-based Test Design Techniques**:
  + **Statement Coverage**: Aims to execute all individual statements in the code at least once during testing.
  + **Branch Coverage**: Ensures that every branch or decision in the code is executed and tested.
  + **Path Coverage**: Involves testing all possible paths through a program to ensure that every logical branch is executed.
  + **Control Flow Testing**: Focuses on testing the flow of control within a program, including loops, conditionals, and other control structures.
* **Experience-based Test Design Techniques**:
  + **Error Guessing**: Testers use their intuition, experience, and domain knowledge to anticipate where defects might be found in the software.
  + **Exploratory Testing**: Testers explore the application, learn about it on the fly, and design and execute tests based on their understanding and observations.
  + **Ad-hoc Testing**: Testing without predefined test cases, where testers use their experience and creativity to identify and execute tests on-the-fly.
  + **Checklist-based Testing**: Involves using a predefined checklist of items to ensure that important aspects of the application are tested.

These techniques are often used in combination to create comprehensive test strategies that address various aspects of software testing. The choice of technique depends on factors such as project requirements, available resources, and the nature of the software being tested.