**Boundary value analysis:**

Boundary Value Analysis is a popular technique for black box testing. It is used to identify defects and errors in software by testing input values on the boundaries of the allowable ranges. Find the errors at the input domain boundary, identify the issues in the beginning can helps to prevent those causing problems later in the software development program. Boundary Value Analysis improves the accuracy in testing the system limit. It helps us to enhancing customer satisfaction by delivering high-quality software.

Improved accuracy is one of the best benefits of boundary value analysis and also catching issues at the earlier stages, Boundary value analysis improves the software [quality](https://www.qable.io/blog/top-10-chatgpt-prompts-for-quality-testing)

**Example:**

Consider a developer write code for an amount text field, which will accept and transfer value only from 100 to 5000. Here we check it by entering 99 in to the amount field and click on the transfer button. It will show an error message as 99 is an invalid amount, because the boundary values are already set as 100 to 5000, 99 is less than 100 so the text field will not transfer the amount.

**Advantages:**

Efficiency

Thoroughness

Coverage

Simplicity

Early Detection of Defects

**Disadvantages:**

Incomplete Coverage

Limited Applicability

Dependency on Specifications

Difficulty in Identifying Boundaries

Overemphasis on Boundaries

**Decision table testing**

Decision table testing is one of the software testing method, it’s used to test the system behavior for different input combination

For ex take login screen, here we have to put user id and password for login process. If we enter correct user id password the user will be redirected to the home page, if anyone of the input (user id r password) is wrong an error message will be displayed.

**Advantages**

When the system behavior is different for different inputs and not the same for a range of inputs, but a decision table can be used.

The representation is simple so that it can be easily interpreted and is used for development and business as well.

This table will help to make effective combinations and can ensure better coverage for testing

Any complex business conditions can be easily turned into decision tables

**Disadvantages**

The main disadvantage is that when the number of inputs increases the table will become more complex

**Use case testing**

Use case testing are the specific way of designing interaction with software items. Its helps to identify Test cases that cover full system on step by step basis from start to end. Test cases are the interaction between the user and software components. Use cases are made on basis of the user actions and the response of the software application to those users action. A collection of variables that will be used to determine whether an application, software system, or one of its features is working as intended.

**Example:**

Here we consider user as an actor ‘A’ and system by ‘S’ here we have to login on website, its need user id and password word.

Success scenario:

Actor ‘A’

System ‘S’

Step 1- user name & password

Step 2- validate password

Step 3- access given

If password not valid

Step 2a- Display message and ask for re try 4 times

Step 2b — close application

**LCSAJ testing**

This testing involves testing the linear code sequences and ensuring that all possible jumps or decisions are completed during testing. The objective is to cover each possible path through the code, including both true and false outcomes of conditional statements.

It’s worth noting that while LCSAJ testing is effective for achieving code coverage, it doesn’t guarantee that all functional requirements or edge cases have been tested. Therefore, it is often used in combination with other testing techniques, such as boundary value analysis, equivalence partitioning, and use case testing, to ensure both code coverage and functional correctness.