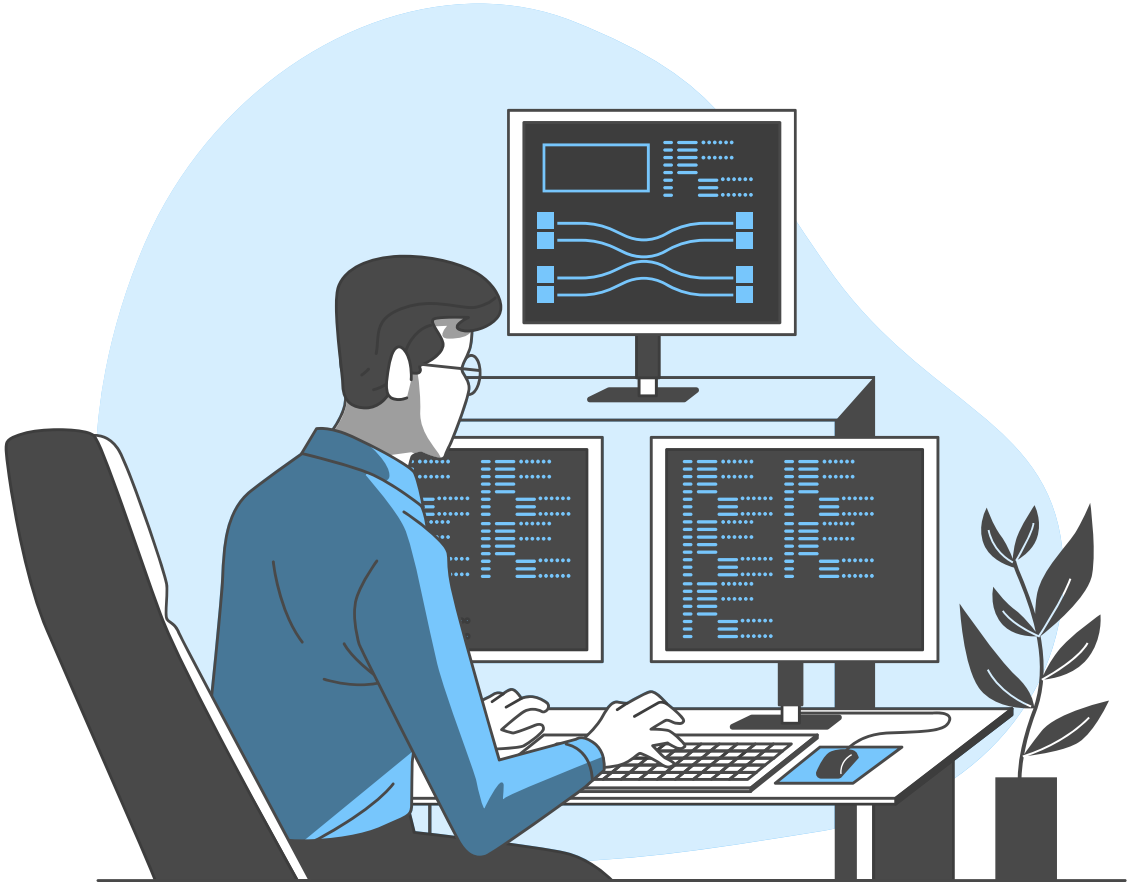
CDIP
Center for Development of IT Professionals



Professional Software Testing & Quality Assurance

Instructor
Parvez Hossain

Software Testing Techniques

Software testing techniques comprise the various ways and angles from which any software can be verified to ensure that it works and appears (UI elements, design) as expected.

- ❑ Equivalence Partitioning
- ❑ Boundary Value Analysis
- ❑ Decision Table Testing

Equivalence Partitioning

In equivalence partitioning, the input of a program is divided into classes. An equivalence class consists of Valid and Invalid states. It reduces the number of test cases.

The concept behind this Test Case Design Technique is that test case of a representative value of each class is equal to a test of any other value of the same class. It allows you to Identify valid as well as invalid equivalence classes.

Input conditions are valid between: 1 to 10 and 20 to 30

Hence there are five equivalence classes:

1. --- to 0 (invalid)
2. 1 to 10 (valid)
3. 11 to 19 (Invalid)
4. 20 to 30 (valid)
5. 31 to --- (invalid)

Select values from each class: -2, 3, 15, 25, 45

Boundary Value Analysis

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.

This software testing technique base on the principle that, if a system works well for these particular values then it will work perfectly well for all values which comes between the two boundary values.

For each variable we check:

❖ **Valid Test Case**

- Minimum value.
- Just above the minimum.
- Nominal Value.
- Just below Max value.
- Max value.

❖ **Invalid Test Case**

- Bellow minimum
- Above maximum

Boundary Value Analysis

Single Fault Assumption: When more than one variable for the same application is checked then one can use a single fault assumption.

Problem: Consider a Program for determining the Previous Data.
Input: *Day, Month, Year with valid ranges as-*
1 ≤ Month≤12
1 ≤ Day ≤31
1900 ≤ Year ≤ 2000

- **Design Boundary Value Test Cases.**

Boundary Value Analysis

Solution: Taking the **year** as a Single Fault Assumption i.e. year will be having values varying from 1900 to 2000 and others will have nominal values.

Test Cases	Month	Day	Year	Output
1	6	15	1900	14 June 1900
2	6	15	1901	14 June 1901
3	6	15	1960	14 June 1960
4	6	15	1999	14 June 1999
5	6	15	2000	14 June 2000

Boundary Value Analysis

Taking **Day** as Single Fault Assumption i.e. Day will be having values varying from 1 to 31 and others will have nominal values.

Test Cases	Month	Day	Year	Output
6	6	1	1960	31 May 1960
7	6	2	1960	1 June 1960
8	6	15	1960	15 June 1960
9	6	30	1960	29 June 1960
10	6	31	1960	Invalid Day

Boundary Value Analysis

Taking **Month** as Single Fault Assumption i.e. Month will be having values varying from 1 to 12 and others will have nominal values.

Test Cases	Month	Day	Year	Output
11	1	15	1960	15 Jan 1960
12	2	15	1960	15 Feb 1960
13	6	15	1960	15 June 1960
14	11	15	1960	15 Nov 1960
15	12	15	1960	15 Dec 1960

Decision Table

A decision table is also known as to 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.

The first task is to identify functionalities where the output depends on a combination of inputs. If there are large input set of combinations, then divide it into smaller subsets which are helpful for managing a decision table.

For an effective testing strategy, it is necessary to thoroughly examine these areas of the software. The defect clustering method relies on the teams' knowledge and experience to identify which modules to test. You can identify such risky modules from your experience. Therefore, the team only has to focus on those "sensitive" areas, saving both time and effort.

Following are steps to create a decision table:

- Enlist the inputs in rows
- Enter all the rules in the column
- Fill the table with the different combination of inputs
- In the last row, note down the output against the input combination.

Decision Table

Example: A submit button in a contact form is enabled only when all the inputs are entered by the end user.

	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7	Rule 8
Input								
Name	F	T	F	T	F	T	F	T
Email	F	F	T	T	F	F	T	T
Message	F	F	F	F	T	T	T	T
Output								
Submit	F	F	F	F	F	F	F	T

Requirement Traceability Matrix

CDIP
Center for Development of IT Professionals

Requirement Traceability Matrix (RTM) is a document that maps and traces user requirement with test cases. It captures all requirements proposed by the client and requirement traceability in a single document, delivered at the conclusion of the Software development life cycle.

The main purpose of Requirement Traceability Matrix is to validate that all requirements are checked via test cases such that no functionality is unchecked during Software testing.

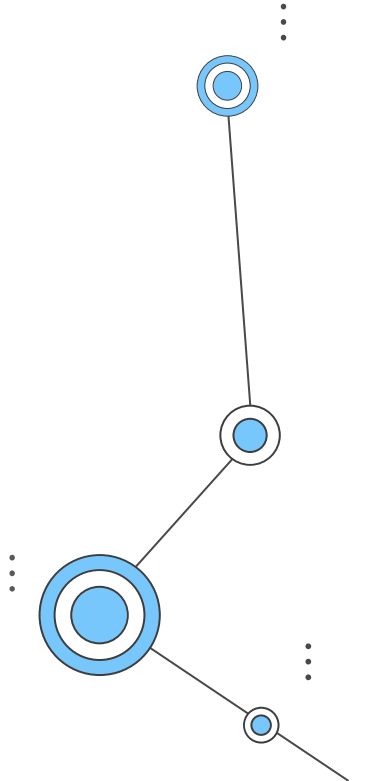
- **Forward Traceability:** In 'Forward Traceability' Requirements to the Test cases. It ensures that the project progresses as per the desired direction and that every requirement is tested thoroughly.
- **Backward Traceability:** The Test Cases are mapped with the Requirements in 'Backward Traceability'. Its main purpose is to ensure that the current product being developed is on the right track. It also helps to determine that no extra unspecified functionalities are added and thus the scope of the project is affected.
- **Bi-Directional Traceability (Forward + Backward):** A Good Traceability matrix has references from test cases to requirements and vice versa (requirements to test cases). This is referred to as 'Bi-Directional' Traceability. It ensures that all the Test cases can be traced to requirements and each and every requirement specified has accurate and valid Test cases for them.

Requirement Traceability Matrix

Requirement	Test Scenario	Test Case	Defects
R1	TS1	TS1.TC1 TS1.TC2	D01
R2	TS2	TS2.TC1 TS2,TC2 TS2.TC3	D02 D03
R3	TS3	TS1.TC1 TS2.TC1 TS3.TC1 TS3.TC2	NIL

Requirement Traceability Matrix

Requirement	Req Description	Test Case	Status
R1	Login to the application	TC1 TC2	TS1 – Pass TC2 – Pass
R2	Document Creation	TC3 TC4 TC5	TC3 – Pass TC4 – Pass TC5 – Fail
R3	Search Document	TC6 TC7 TC8 TC9	TC6 – Pass TC7 – Pass TC8 – Pass TC9 – No Run



Thanks!

Do you have any questions?

Email: parvez9605@gmail.com
WhatsApp: +88 01772880239