TestLeaf

Always Ahead

Test Design Techniques

TestLeaf
Always Ahead

Test design technique

- A test design technique is used to select a good set of tests from all possible tests for a given system.

- Exhaustive Testing is not possible, so we need to use Test Design Techniques in order to reduce the size of the input.
  - Exhaustive Testing is a Test approach in which the test suite comprises all combinations of input values and preconditions.

There are two main categories of Test Design Techniques, They are:

a. Static Techniques
b. Dynamic Techniques

Static Techniques

Testing of the software documents manually or with a set of tools but without executing the Software.

Two types of static testing techniques
1. Reviews (Manual Examination)
2. Static Analysis (Automated Analysis)

1. Reviews

Types of Reviews
- a) Informal Review
- b) Walkthrough
- c) Technical Review
- d) Inspection

2. Static Analysis

Static analysis tools are typically used by developers, Compilers offer some support for Static analysis,

Reviews

Informal Review - During informal review, the work product is given to a domain expert and the feedback/comments are reviewed by the owner/author.

Walkthrough - performed on any kind of requirements, design, or project plan and is generally moderated by the owner of the work product. Walkthrough can be also done for training people and establishing consensus.

Technical Review - performed on any kind of requirements, design, code, or project plan and is generally moderated by the technical lead. Technical review is a formal type of static test and done for observing whether or not the work product meets the technical specifications/standards.

Inspection – Inspections are a formal type of review that involves checking the documents thoroughly before a meeting and is carried out mostly by moderators. A meeting is then held to review the code and the design. Inspection meetings can be held both physically and virtually.

TestLeaf
Always Ahead

Dynamic techniques

- Dynamic test design techniques involve testing by running the system under test
- In this technique, the tester provides input data to the application and executes it. This is done to verify its different functional and non-functional requirements.

Categories of Dynamic Test Design Techniques

1. Specification-based or Black-box Techniques
    I. Equivalence Partitioning (EP)
    II. Boundary Value Analysis (BVA)
    III. Decision Table Testing
    IV. State Transition Testing
    V. Use Case Testing

2. Structure-based or White-box Techniques
    I. Statement Testing and coverage
    II. Decision Testing and Coverage
    III. Condition Testing, Multi Condition Testing

3. Experience-based Techniques
    I. Error Guessing
    II. Exploratory Testing

Equivalence Class Partitioning

Equivalent Class Partitioning allows you to divide set of test condition into a partition which should be considered the same.

Your task is to pick one condition out of each partition, which covers all possible scenarios, to execute test cases.

If a condition of a partition is valid, other conditions are valid too. Likewise, if a condition in a partition is invalid, other conditions are also invalid. This helps reduce the number of test cases.

TestLeaf
Always Ahead

Equivalence Class Partitioning

Example

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

Hence there are five equivalence classes

- Negative values to 0 (invalid)
- 1 to 10 (valid)
- 11 to 19 (invalid)
- 20 to 30 (valid)
- 31 to Other positive values (invalid)

You select values from each class, i.e., -2, 3, 15, 25, 45

Boundary Value Analysis

- The goal is to select test cases to execute boundary values. It includes maximum, minimum, inside or outside boundaries, typical values and error values.
- If the input is within the boundary value, it is considered ‘Positive testing.’ If the input is outside of the boundary value, it is considered ‘Negative testing.’
- The behavior of Negative testing is more likely to be incorrect than the behavior of Positive testing; and boundaries are an area in which testing is more likely to yield defects.
- 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.

Example:

Input condition is valid between 1 to 10

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

TestLeaf
Always Ahead

Decision Table Testing

- A Decision Table is a tabular representation of conditions versus test actions. Conditions are considered as inputs, while actions are considered as outputs.
- 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 every function, you need to create a table and list down all types of combinations of inputs and its respective outputs. This helps to identify a condition that is overlooked by the tester.

Decision Table Testing

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.

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

State Transition

- In State Transition technique changes in input conditions change the state of the Application Under Test (AUT). This testing technique allows the tester to test the behavior of an AUT.

- The tester can perform this action by entering various input conditions in a sequence.

- In State transition technique, the testing team provides positive as well as negative input test values for evaluating the system behavior.

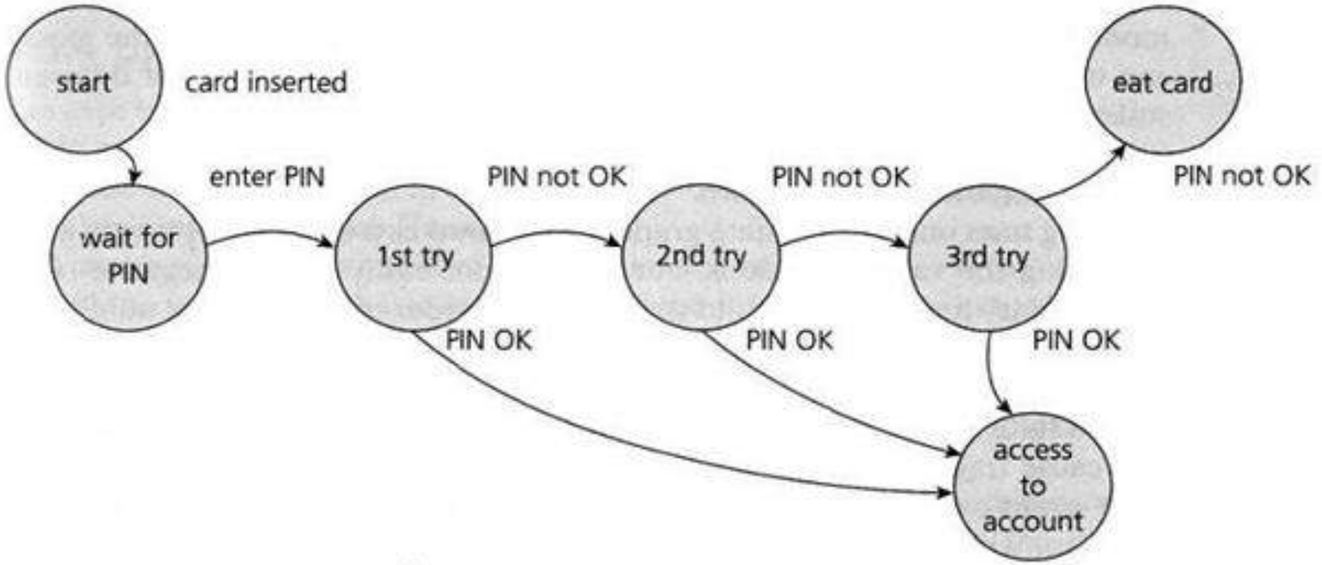
Guideline for State Transition:

- State transition should be used when a testing team is testing the application for a limited set of input values.

- The technique should be used when the testing team wants to test sequence of events which happen in the application under test.

TestLeaf
Always Ahead

State Transition



TestLeaf
Always Ahead

Use Cases

- A Use Case is used to define the system that how to use the system for performing a specific task. and A Test Case is defined as a group of test inputs, execution condition, and expected results which further lead to developing a particular test objective.

- A use case is not a part of execution it is only a diagrammatic presentation of a document that specifies how to perform a certain task. If we talk about test case it is used to validate the software which is developed by testers for validating that the software is in working as per requirement or not.

Main Success Scenario	Step	Description
A:Actor S:System	1	**A: Enter Agent Name & Password**
	2	**S: Validate Password**
	3	**S: Allow Account Access**
Extensions	2a	**Password not valid** S: Display Message and ask for re-try 4 times
	2b	**Password not valid 4 times** S: Close Application

TestLeaf
Always Ahead

Experience-based technique

Exploratory Testing:
Usually, this process is carried out by domain experts. They perform testing just by exploring the functionalities of the application without having the knowledge of the requirements. Testers can explore and learn the system while using these techniques. High severity bugs are found very quickly in this type of testing.

Error Guessing:
Error guessing is one of the testing techniques used to find bugs in a software application based on the tester's prior experience. In Error guessing, no specific rules are applied.

Structure-based or White-Box techniques

Statement Coverage or Line Coverage:
In this technique, every statement in the source code is executed at least once. Thereby, we can check what the source code is and is not expected to do. However, we cannot test the false condition in the source code.
Statement coverage = (No. of statements Executed/Total no. of statements in the source code)*100

Condition Coverage or Predicate Coverage:
Condition coverage is seen for Boolean expression. Condition coverage ensures whether all the Boolean expressions have been covered and evaluated to both TRUE and FALSE.

Decision Coverage or Branch Coverage:
Test coverage criteria require enough test cases so that each condition in a decision takes on all possible outcomes at least once, and each point of entry to a program or subroutine is invoked at least once. That is, every branch (decision) is either true and false. It is helpful to invalidate all branches in the code to make sure that no branch leads to any abnormal behavior.

Multiple Condition Coverage:
Every combination of 'true' or 'false' for the conditions related to a decision has to be tested in this technique.

TestLeaf
Always Ahead