What is Equivalence Class Partitioning?

Equivalence partitioning is a Test Case Design Technique to divide the input data of software into different equivalence data classes. Test cases are designed for equivalence data class. The equivalence partitions are frequently derived from the requirements specification for input data that influence the processing of the test object. A use of this method reduces the time necessary for testing software using less and effective test cases.

Equivalence Partitioning = Equivalence Class Partitioning = ECP

It can be used at any level of software for testing and is preferably a good technique to use first. In this technique, only one condition to be tested from each partition. Because we assume that, all the conditions in one partition behave in the same manner by the software. In a partition, if one condition works other will definitely work. Likewise we assume that, if one of the condition does not work then none of the conditions in that partition will work.

Equivalence partitioning is a testing technique where input values set into classes for testing.

- Valid Input Class = Keeps all valid inputs.
- Invalid Input Class = Keeps all Invalid inputs.

Example of Equivalence Class Partitioning?

- A text field permits only numeric characters
- Length must be 6-10 characters long

Partition according to the requirement should be like this:

While evaluating Equivalence partitioning, values in all partitions are equivalent that's why 0-5 are equivalent, 6 – 10 are equivalent and 11- 14 are equivalent.

At the time of testing, test 4 and 12 as invalid values and 7 as valid one.

It is easy to test input ranges 6–10 but harder to test input ranges 2-600. Testing will be easy in the case of lesser test cases but you should be very careful. Assuming, valid input is 7. That means, you belief that the developer coded the correct valid range (6-10).

What is Boundary value analysis:

Boundary value analysis is a test case design technique to test boundary value between partitions (both valid boundary partition and invalid boundary partition). A boundary value is an input or output value on the border of an equivalence partition, includes minimum and maximum values at inside and outside boundaries. Normally Boundary value analysis is part of stress and negative testing.

Using Boundary Value Analysis technique tester creates test cases for required input field. For example; an Address text box which allows maximum 500 characters. So, writing test cases for each character once will be very difficult so that will choose boundary value analysis.

Example for Boundary Value Analysis:

Example 1

Suppose you have very important tool at office, accepts valid User Name and Password field to work on that tool, and accepts minimum 8 characters and maximum 12 characters. Valid range 8-12, Invalid range 7 or less than 7 and Invalid range 13 or more than 13.

Write Test Cases for Valid partition value, Invalid partition value and exact boundary value.

- Test Cases 1: Consider password length less than 8.
- Test Cases 2: Consider password of length exactly 8.
- Test Cases 3: Consider password of length between 9 and 11.
- Test Cases 4: Consider password of length exactly 12.
- Test Cases 5: Consider password of length more than 12.

Example 2

Test cases for the application whose input box accepts numbers between 1-1000. Valid range 1-1000, Invalid range 0 and Invalid range 1001 or more.

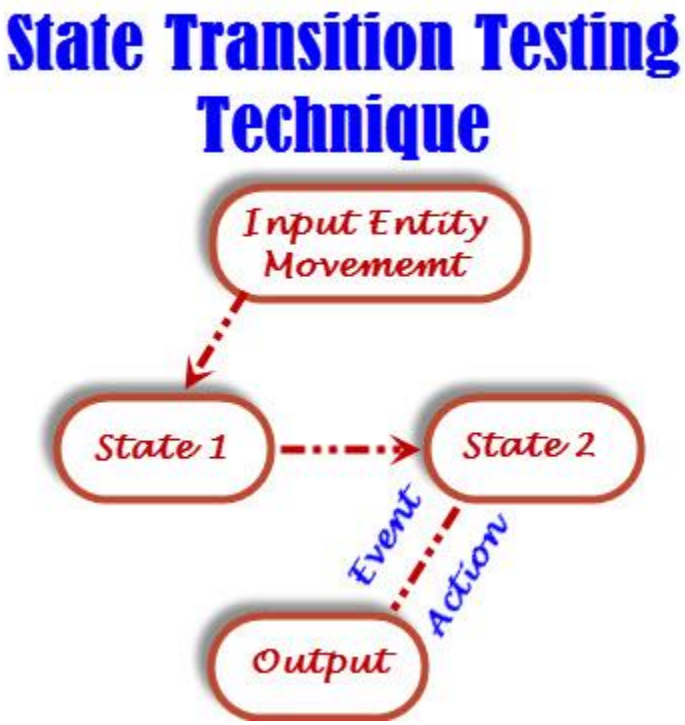
Write Test Cases for Valid partition value, Invalid partition value and exact boundary value.

- Test Cases 1: Consider test data exactly as the input boundaries of input domain i.e. values 1 and 1000.
- Test Cases 2: Consider test data with values just below the extreme edges of input domains i.e. values 0 and 999.
- Test Cases 3: Consider test data with values just above the extreme edges of input domain i.e. values 2 and 1001

State transition Test case design technique

State transition testing is a form of **Dynamic Testing Technique** that comes in use when the system explained as a finite number of states and the evolutions between the states is ruled by the rules of the system. Another use of this technique when features of a system are characterized as states that converts to other state, this transition is explained by the method of the software

The diagrammatic explanation is shown below,



So, the diagram shows that, the input condition has came the reason of an entity transitions from State 1 to State 2 that guides to an event and results to an action and finally gives the output.

Four major parts of state transition model:

- States that the software might get (open/closed or sufficient/insufficient funds)
- The transition from one state to another (with single transitions)
- The events that origin a transition (closing a file or withdrawing money)
- Actions that result from a transition (an error message or being given the cash)

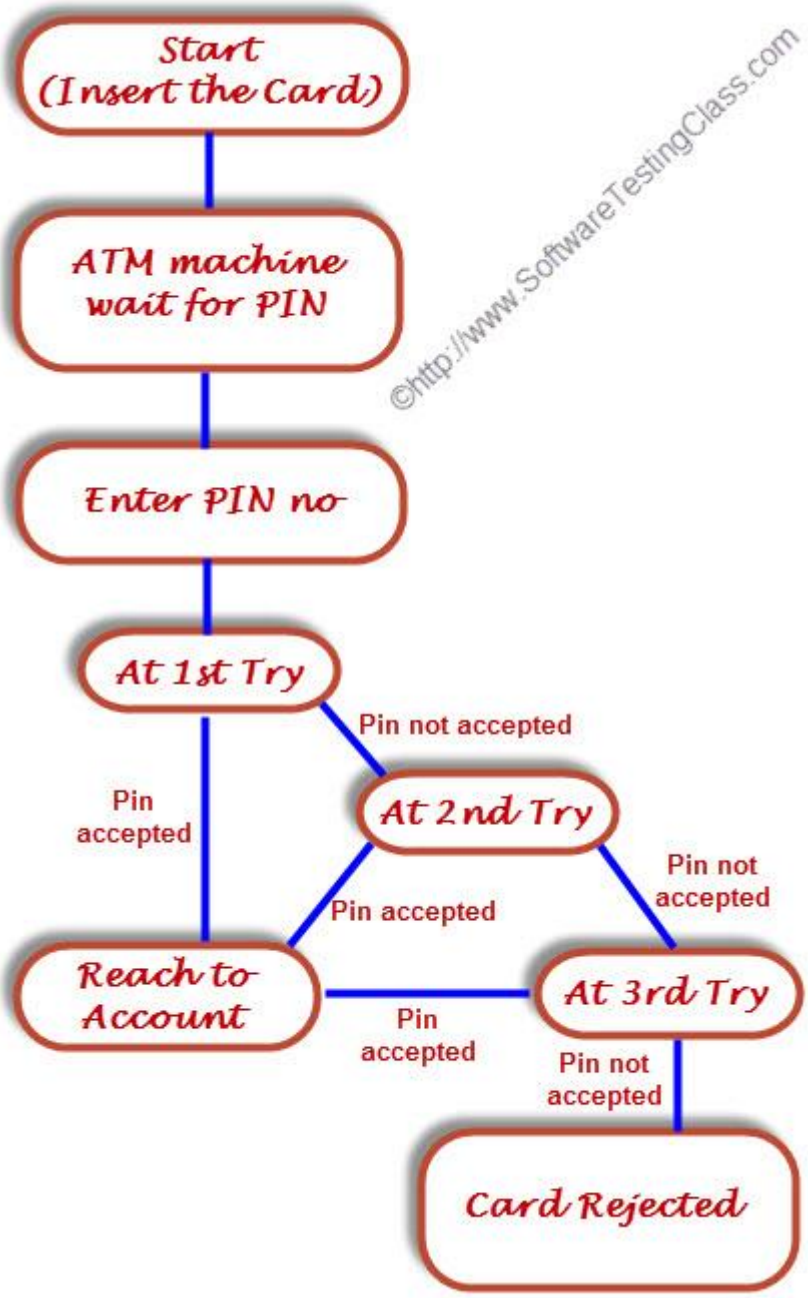
For any given state, one event origins only one action, but that the same event from a different state might origin a different action and a different end state.

An example to explain State transition testing technique:

Go to ATM machine to withdraw $1000, requires to the ATM machine and get cash. Later on, go to the ATM and give the same request at same amount but this time ATM refuses the amount because of insufficient balance amount.

This refusal happened just because of bank account has been changed from one state (sufficient funds) to another state (insufficient funds). The transaction that caused the account to change its state was maybe the earlier withdrawal.

Below diagram shows that how ATM get process using Personal Identity Number (PIN).



The above state diagram shows several states and four events: Insert Card, Enter PIN, PIN accepted and PIN not accepted). There would also be a transition from the "Card Rejected" state back to the start state.

There should be a 'cancel' option at the time of 'wait for PIN' process and at the time of three tries that should also go back to the start state and eject the card.

Even think, still this state diagram is incomplete; it also gives us some more information that should be helpful in designing the test cases and explaining the state transition technique.

Let's take some scenario to develop test cases,

- The first test case should be very much sensible to enter the correct PIN at the first time
- The second test should be to enter an incorrect PIN each time, so that the system rejects the card
- To test all transition, firstly do the testing where the PIN was incorrect the first time but OK the second time and another test where the PIN was correct on the third try
- But, these tests are basically less important than the first two tests

The major advantages of the state transition technique is that the model can be as detailed or as abstract as you need it to be. Where a part of the system is more significant a larger depth of detail can be modelled. Where the system is less important, the model can use a single state to indicate what would otherwise be a series of different states.

The Coverage of single transition is also called as 0-switch coverage, the coverage of pair's transition is 1-switch coverage, coverage of triples transition is 2-switch coverage, etc.

Explaining test cases from the state transition model is a black-box approach. Determining how much test has been completed is a white-box perspective. However, state transition testing is regarded as a black-box technique. Determining tests simply from a state graph (also known as a state chart) is wonderful for looking the valid transitions, but we may not simply look the negative tests, where we try to make invalid transitions. In order to see the total number of mixtures of states and transitions, both valid and invalid, a state table is useful.

Table below shows the state table lists all the states down at one side of the table and all the events that origin transitions along the top (or vice versa). Every cell then symbolizes a state-event pair.

The information of each cell shows which state the system will shift to, when the resultant event arises while in the associated state. This will contain possible erroneous events – events that are not accepted to happen in certain states. These are negative test conditions.

	Insert Card	**Valid PIN**	**InValid PIN**

Start State	S2	–	–
Wait for PIN	–	S6	S3
1st try invalid	–	S6	S4
2nd try invalid	–	S6	S5
3rd try invalid	–	–	S7
Access Account	–	?	?
Card not excepted	S1 (for new card)	–	–

This Table shows different states in the first column and the respective inputs across the top row. For Example:

- Suppose the system is in state 1, after inserting the card it will go to state 2.
- If system is in state 2, after entering a valid PIN, it goes to State 6 to access the account. In State 2 if we enter an invalid PIN it goes to State 3.

In the table, a dash (-) sign represents invalid transitions of that state. A question mark in two cells – either a valid or invalid PIN, when we are using the account. Possibly the system will catch our PIN number as the amount of cash to withdraw? Other invalid cells would be actually not possible in this example. Invalid (negative) tests will try to make invalid transitions, transitions that shouldn't be possible.

Use case test case design Technique

Use Case Testing is a functional black box testing technique that helps testers to identify test scenarios that exercise the whole system on each transaction basis from start to finish.

A use case is a description of a particular use of the system by an actor or user. It is used widely in developing tests at system or acceptance level.

What is Use Case Testing?

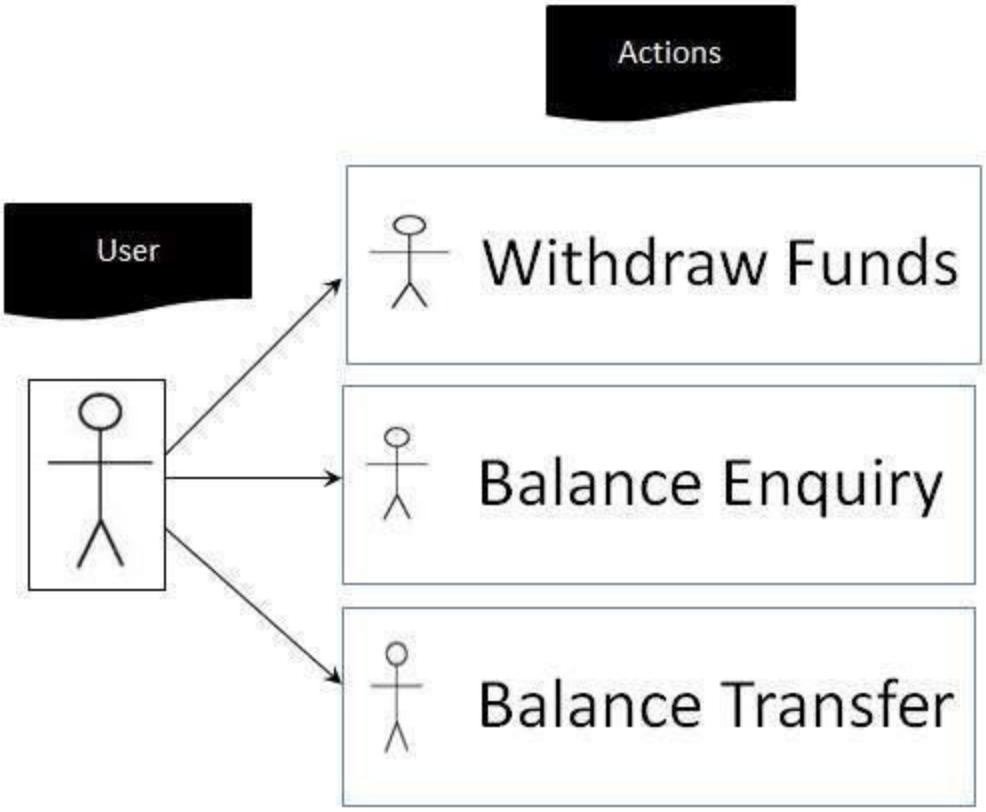
Use Case Testing, is a technique that helps identify test cases that cover the entire system, on a transaction by transaction basis from start to the finishing point.

Characteristics of Use Case Testing:

- Use Cases capture the interactions between 'actors' and the 'system'.
- 'Actors' represents user and their interactions that each user takes part into.
- Test cases based on use cases and are referred as scenarios.
- Capability to identify gaps in the system which would not be found by testing individual components in isolation.
- Very effective in defining the scope of acceptance tests.

Example:

The Below example clearly shows the interaction between users and possible actions.


Actions
User
Withdraw Funds
Balance Enquiry
Balance Transfer