



Module 4

Blackbox Testing

Index

- ❖ **Blackbox testing**
 - ❖ **Equivalence class testing**
 - ❖ **Boundary value testing**
 - ❖ **Decision table testing**
 - ❖ **Pairwise testing**
 - ❖ **State transition testing**
 - ❖ **Use-case testing**
- 

❖ Blackbox testing

- ▶ Black Box Testing is a software testing method in which the functionalities of software applications are tested **without having knowledge of internal code structure, implementation details and internal paths**
 - ▶ Black Box Testing **mainly focuses on input and output of software applications**
 - ▶ It is entirely based on software requirements and specifications
 - ▶ It is also known as Behavioral Testing
- 



- ▶ The above Black-Box can be any software system you want to test or even your custom application
- ▶ Under Black Box Testing, you can test these applications **by just focusing on the inputs and outputs without knowing their internal code implementation.**

➤ How to do Blackbox testing

Here are the generic steps followed to carry out any type of Black Box Testing.

- ▶ Initially, the **requirements and specifications** of the system are examined
- ▶ Tester **chooses valid inputs (positive test scenario)** to check whether SUT (System Under Test) processes them correctly. Also, some **invalid inputs (negative test scenario)** are chosen to verify that the SUT is able to detect them
- ▶ Tester **determines expected outputs** for all those inputs
- ▶ Software tester **constructs test cases with the selected inputs**
- ▶ The **test cases are executed**
- ▶ Software tester **compares the actual outputs with the expected outputs**
- ▶ Defects if any are **fixed and re-tested**

➤ Types of Blackbox testing

There are many types of Black Box Testing but the following are the prominent ones :-

- ▶ **Functional testing** - This black box testing type is related to the **functional requirements** of a system; it is done by software testers
- ▶ **Non-functional testing** - This type of black box testing is not related to testing of specific functionality, but **non-functional requirements** such as performance, scalability, usability.
- ▶ **Regression testing** - It is done after code fixes, upgrades or any other system maintenance to check **the new code has not affected the existing code**


➤ Tools used for Blackbox testing

- ▶ Tools used for Black box testing largely depends on the type of black box testing you are doing
- ▶ For Functional/ Regression Tests you can use - QTP(Quick Test Professional), Selenium
- ▶ For Non-Functional Tests, you can use - LoadRunner, Jmeter

➤ **Blackbox Testing Techniques**

- 1) **Equivalence class testing**
- 2) **Boundary value testing**
- 3) **Decision table testing**
- 4) **Pairwise testing**
- 5) **State transition testing**
- 6) **Use-case testing**

1) Equivalence class testing

- ▶ **Equivalence Partitioning** or **Equivalence Class Partitioning** is type of black box testing technique which can be applied to all levels of software testing like unit, integration, system, etc.
 - ▶ In this technique, **input data units are divided into equivalent partitions** that can be used to derive test cases which reduces time required for testing because of small number of test cases
 - ▶ It **divides the input data of software into different equivalence data classes**
 - ▶ You can apply this technique, where there is a range in the input field
- 

Example

- ▶ Let's consider the behavior of Order Pizza Text Box Below
- ▶ Pizza values 1 to 10 is considered valid. A success message is shown.
- ▶ While value 11 to 99 are considered invalid for order and an error message will appear, **"Only 10 Pizza can be ordered"**

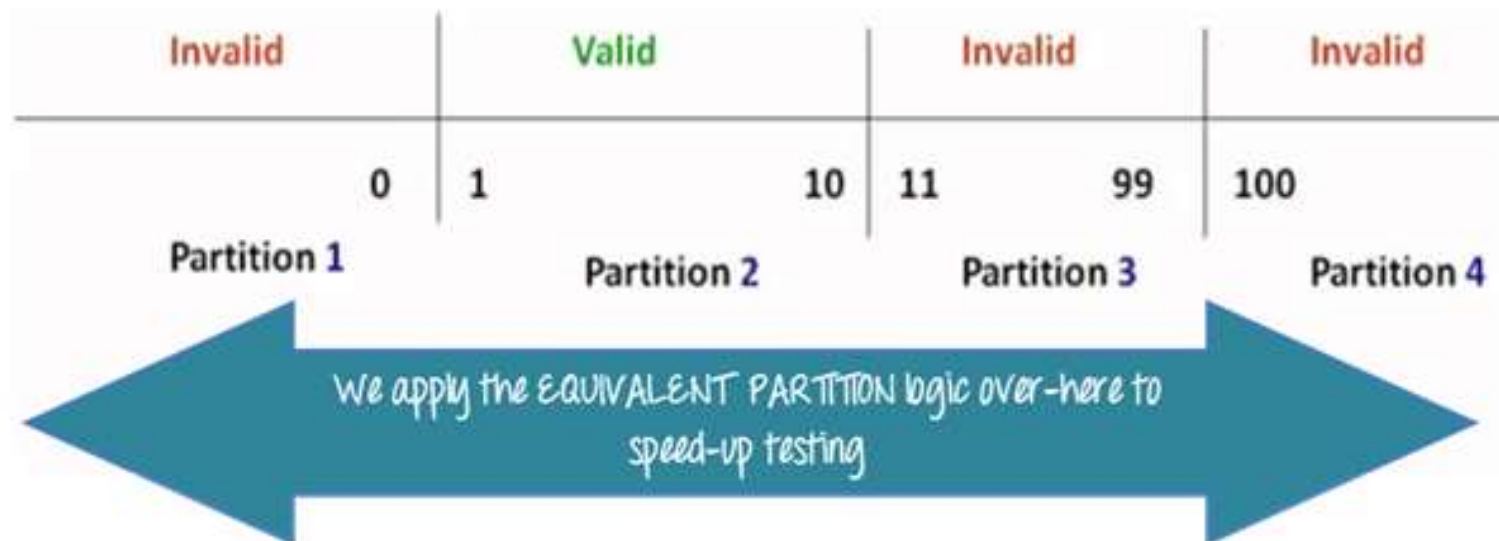
Order Pizza:

Submit

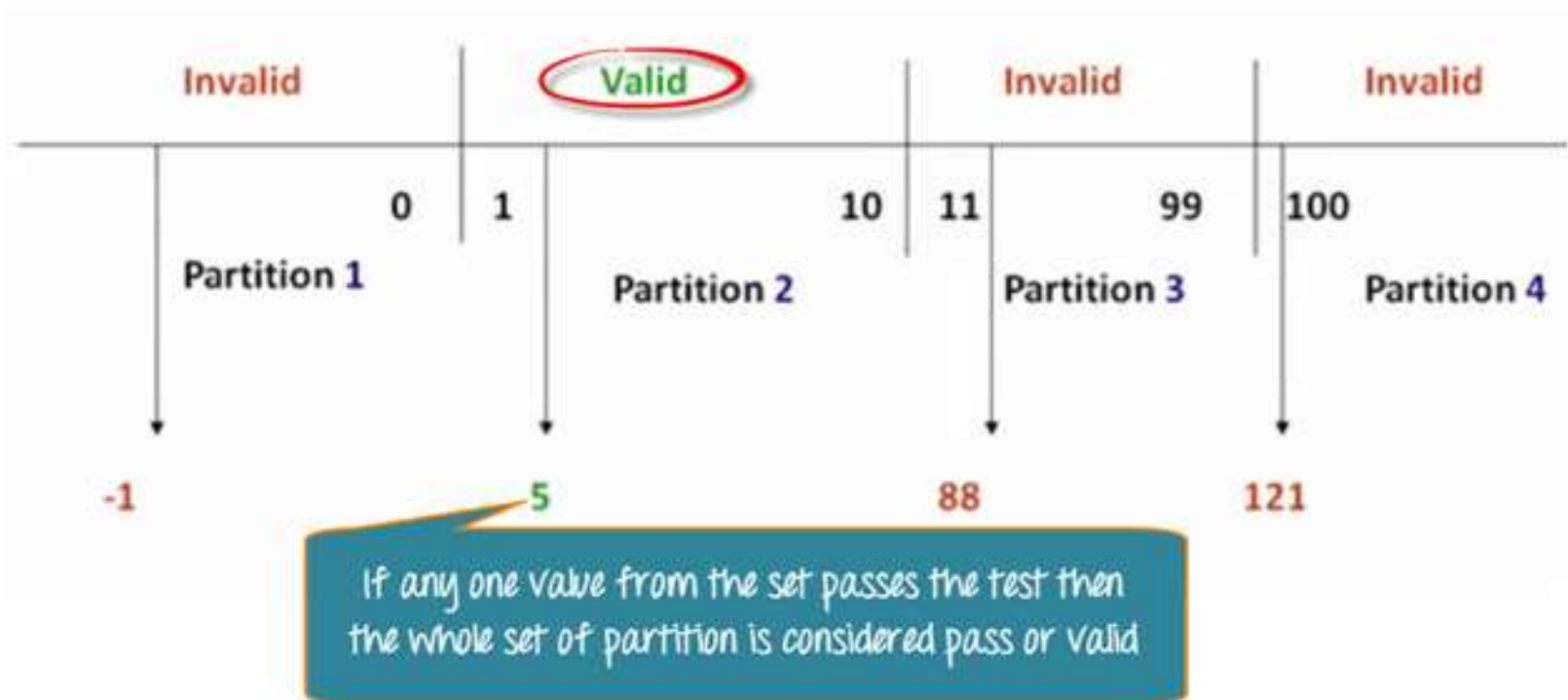
Here is the test condition

1. Any Number greater than 10 entered in the Order Pizza field(let say 11) is considered invalid.
2. Any Number less than 1 that is 0 or below, then it is considered invalid.
3. Numbers 1 to 10 are considered valid
4. Any 3 Digit Number say -100 is invalid.


We cannot test all the possible values because if done, the number of test cases will be more than 100. To address this problem, we use equivalence partitioning hypothesis where we divide the possible values of tickets into groups or sets as shown below where the system behavior can be considered the same.



The divided sets are called Equivalence Partitions or Equivalence Classes. Then we pick only one value from each partition for testing. The hypothesis behind this technique is **that if one condition/value in a partition passes all others will also pass**. Likewise, **if one condition in a partition fails, all other conditions in that partition will fail**.

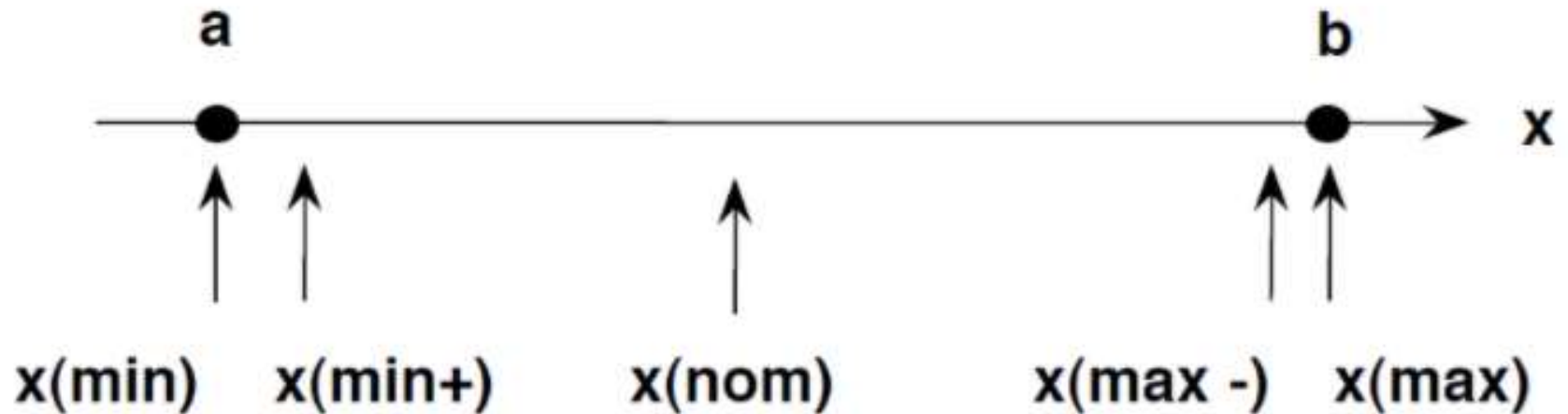


2) Boundary value testing

- ▶ Boundary testing is the process of testing between extreme ends or boundaries between partitions of the input values
 - ▶ So these extreme ends like Start - End, Lower - Upper, Maximum - Minimum, Just Inside - Just Outside values are called boundary values and the testing is called "boundary testing"
 - ▶ In Boundary Testing, Equivalence Class Partitioning plays a good role
 - ▶ Boundary Testing comes after the Equivalence Class Partitioning
- 

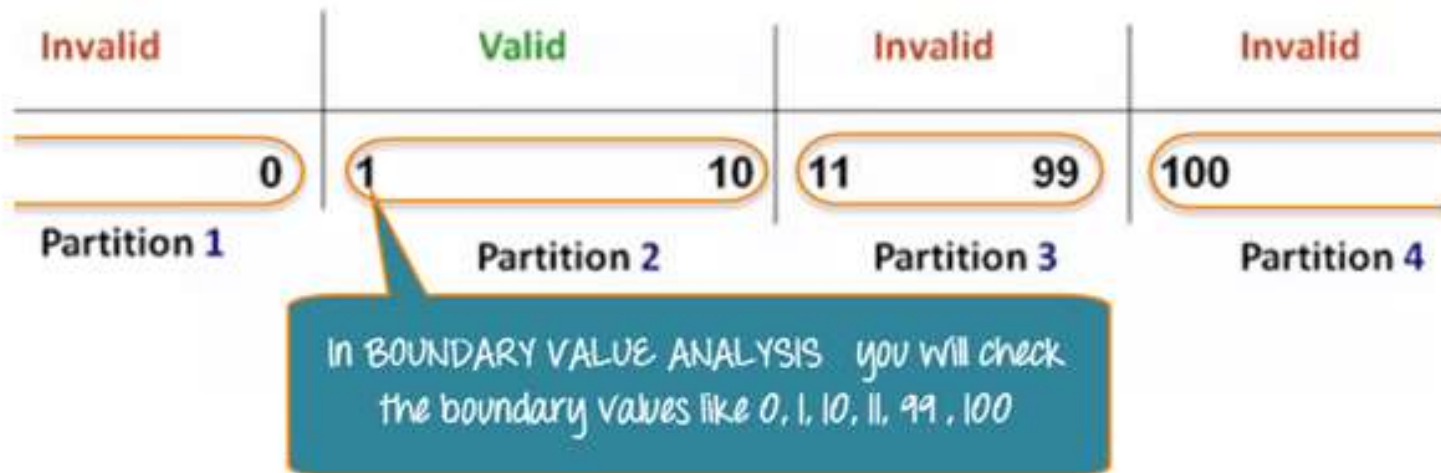
- The basic idea in normal boundary value testing is to select input variable values at their:

1. Minimum
2. Just above the minimum
3. A nominal value
4. Just below the maximum
5. Maximum



Example

Boundary Value Analysis- in Boundary Value Analysis, you test boundaries between equivalence partitions



In our earlier equivalence partitioning example, instead of checking one value for each partition, you will check the values at the partitions like 0, 1, 10, 11 and so on. As you may observe, you test values at **both valid and invalid boundaries**. Boundary Value Analysis is also called **range checking**.

Equivalence partitioning and boundary value analysis(BVA) are closely related and can be used together at all levels of testing.


Why Equivalence & Boundary Analysis Testing


1. This testing is used to reduce a very large number of test cases to manageable chunks.
2. Very clear guidelines on determining test cases without compromising on the effectiveness of testing.
3. Appropriate for calculation-intensive applications with a large number of variables/inputs

Summary:

- Boundary Analysis testing is used when practically it is impossible to test a large pool of test cases individually
- Two techniques - Boundary value analysis and equivalence partitioning testing techniques are used
- In Equivalence Partitioning, first, you divide a set of test condition into a partition that can be considered.
- In Boundary Value Analysis you then test boundaries between equivalence partitions
- Appropriate for calculation-intensive applications with variables that represent physical quantities


3) Decision table testing


- ▶ A Decision Table is a tabular representation of inputs versus rules/cases/test conditions
 - ▶ It is a very effective tool used for both complex software testing and requirements management
 - ▶ Decision table helps to check all possible combinations of conditions for testing and testers can also identify missed conditions easily
 - ▶ The conditions are indicated as True(T) and False(F) values
- 

- ▶ Decision table testing is a software testing technique used to test system behavior for different input combinations
 - ▶ This is a systematic approach where the different input combinations and their corresponding system behavior (Output) are captured in a tabular form
 - ▶ That is why it is also called as a **Cause-Effect** table where Cause and effects are captured for better test coverage
- 

Example

Let's create a decision table for a login screen.

Email

Password


Log in

The condition is simple if the user provides correct username and password the user will be redirected to the homepage. If any of the input is wrong, an error message will be displayed.

Conditions	Rule 1	Rule 2	Rule 3	Rule 4
Username (T/F)	F	T	F	T
Password (T/F)	F	F	T	T
Output (E/H)	E	E	E	H

- T – Correct username/password
- F – Wrong username/password
- E – Error message is displayed
- H – Home screen is displayed


Interpretation:

- Case 1 – Username and password both were wrong. The user is shown an error message.
 - Case 2 – Username was correct, but the password was wrong. The user is shown an error message.
 - Case 3 – Username was wrong, but the password was correct. The user is shown an error message.
 - Case 4 – Username and password both were correct, and the user navigated to homepage
- 

While converting this to test case, we can create 2 scenarios ,

- Enter correct username and correct password and click on login, and the expected result will be the user should be navigated to homepage

And one from the below scenario

- Enter wrong username and wrong password and click on login, and the expected result will be the user should get an error message
 - Enter correct username and wrong password and click on login, and the expected result will be the user should get an error message
 - Enter wrong username and correct password and click on login, and the expected result will be the user should get an error message
- 

➤ Why Decision Table Testing is Important?

- ▶ Provides **good coverage**
- ▶ **Representation is simple** so it is easy to interpret and use

➤ Advantages of Decision Table Testing

- ▶ Any complex business conditions can be easily turned into decision tables
- ▶ In a case we are going for 100% coverage typically when the input combinations are low, this technique can ensure the coverage

➤ Disadvantages of Decision Table Testing

- ▶ When the **number of input increases** the table will become more **complex**

4) Pairwise testing

- ▶ Pairwise Testing also known as **All-pairs testing** is a testing approach taken for testing the software using **combinatorial method**
- ▶ It's a method to **test all the possible discrete combinations of the parameters involved**
- ▶ Assume we have a piece of software to be tested which has got 10 input fields and 10 possible settings for each input field, then there are 10^{10} possible inputs to be tested
- ▶ In this case, **exhaustive testing is impossible even if we wish to test all combinations**

Example

- ▶ An application with simple **list box** with 10 elements (Let's say 0,1,2,3,4,5,6,7,8,9) along with a **checkbox, radio button, Text Box and OK Button**
- ▶ The Constraint for the Text box is **it can accept values only between 1 and 100**
- ▶ Below are the values that each one of the GUI objects can take :

List Box - 0,1,2,3,4,5,6,7,8,9

Check Box - Checked or Unchecked

Radio Button - ON or OFF

Text Box - Any Value between 1 and 100

Exhaustive combination of the product B is calculated.

List Box = 10

Check Box = 2

Radio Button = 2

Text Box = 100

Total Number of Test Cases using Cartesian Method : $10 * 2 * 2 * 100 = 4000$

Total Number of Test Cases including Negative Cases will be > 4000

Now, the idea is to bring down the number of test cases. We will first try to find out the number of cases using the conventional software testing technique. We can consider the list box values as 0 and others as 0 is neither positive nor negative. Radio button and check box values cannot be reduced, so each one of them will have 2 combinations (ON or OFF). The Text box values can be reduced into three inputs (Valid Integer, Invalid Integer, Alpha-Special Character).

Now, we will calculate the number of cases using software testing technique, $2*2*2*3 = 24$ (including negative cases).

Now, we can still reduce the combination further into All-pairs technique.

Step 1: Order the values such that one with most number of values is the first and the least is placed as the last variable.

Step 2: Now start filling the table column by column. List box can take 2 values.

Step 3: The Next column under discussion would be check box. Again Check box can take 2 values.

Step 4: Now we need to ensure that we cover all combinations between list box and Check box.

Step 5: Now we will use the same strategy for checking the Radio Button. It can take 2 values.

Step 6: Verify if all the pair values are covered as shown in the table below.

Text Box	List Box	Check Box	Radio Button
Valid Int	0	check	ON
Valid Int	others	uncheck	OFF
Invalid Int	0	check	ON
Invalid Int	others	uncheck	OFF
AlphaSpecialCharacter	0	check	ON
AlphaSpecialCharacter	others	uncheck	OFF

Result of Pair-Wise Testing:

Exhaustive Combination results in > 4000 Test Cases.

Conventional Software Testing technique results in 24 Test Cases.

Pair Wise Software Testing technique results in just 6 Test Cases.

➤ **Advantages of Pairwise Testing**

- ▶ Pairwise testing reduces the number of execution of test cases
- ▶ Pairwise testing increases the test coverage
- ▶ Pairwise testing increases the defect detection ratio
- ▶ Pairwise testing takes less time to complete the execution of the test suite
- ▶ Pairwise testing reduces the overall testing budget for a project

➤ **Disadvantages of Pairwise Testing**

- ▶ Pairwise testing is not beneficial if the values of the variables are inappropriate
- ▶ In pairwise testing it is possible to miss the highly probable combination while selecting the test data
- ▶ Pairwise testing is not useful if combinations of variables are not understood correctly


5) State transition testing

- ▶ **State Transition Testing** is a black box testing technique in which changes made in input conditions cause state changes or output changes in the Application under Test(AUT)
- ▶ State transition testing helps to analyze behaviour of an application for different input conditions
- ▶ Testers can provide positive and negative input test values and record the system behavior
- ▶ **State Transition Testing Technique** is helpful where you need to **test different system transitions.**

➤ When to Use State Transition?

- ▶ This can be used when a tester is testing the application for a **finite set of input values**
- ▶ When the tester is trying to test sequence of events that occur in the application under test

➤ When to Not Rely On State Transition?

- ▶ When the testing is not done for sequential input combinations
 - ▶ If the testing is to be done for different functionalities like exploratory testing
- 

Four Parts Of State Transition Diagram

There are 4 main components of the State Transition Model as below

1) **States** that the software might get



1st Try

2) **Transition** from one state to another




3) **Events** that origin a transition like closing a file or withdrawing money



Incorrect Pin

4) **Actions** that result from a transition (an error message or being given the cash.)




Access
Granted

State Transition Diagram and State Transition Table

There are two main ways to represent or design state transition, State transition diagram, and state transition table.

In state transition diagram the states are shown in boxed texts, and the transition is represented by arrows. It is also called State Chart or Graph. It is useful in identifying valid transitions.


In state transition table all the states are listed on the left side, and the events are described on the top. Each cell in the table represents the state of the system after the event has occurred. It is also called State Table. It is useful in identifying invalid transitions.



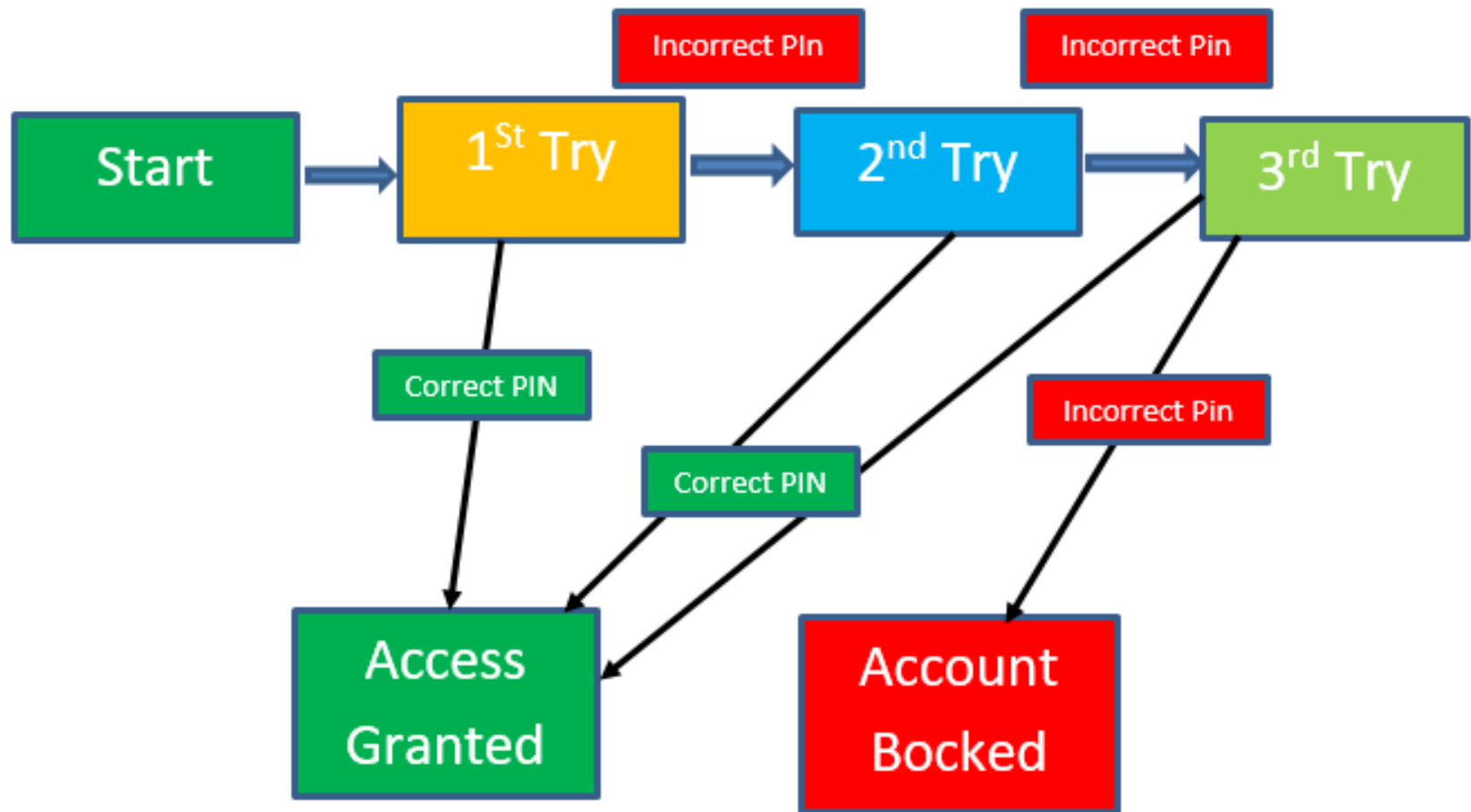
Example

Let's consider an ATM system function where if the user enters the invalid password three times the account will be locked.

In this system, if the user enters a valid password in any of the first three attempts the user will be logged in successfully. If the user enters the invalid password in the first or second try, the user will be asked to re-enter the password. And finally, if the user enters incorrect password 3rd time, the account will be blocked.



State transition diagram



In the diagram whenever the user enters the correct PIN he is moved to Access granted state, and if he enters the wrong password he is moved to next try and if he does the same for the 3rd time the account blocked state is reached.

State Transition Table

	Correct PIN	Incorrect PIN
S1) Start	S5	S2
S2) 1 st attempt	S5	S3
S3) 2 nd attempt	S5	S4
S4) 3 rd attempt	S5	S6
S5) Access Granted	-	-
S6) Account blocked	-	-

In the table when the user enters the correct PIN, state is transitioned to S5 which is Access granted. And if the user enters a wrong password he is moved to next state. If he does the same 3rd time, he will reach the account blocked state.

Advantages and Disadvantages of State Transition Technique

Advantages

This testing technique will provide a pictorial or tabular representation of system behavior which will make the tester to cover and understand the system behavior effectively.


By using this testing, technique tester can verify that all the conditions are covered, and the results are captured

Disadvantages

The main disadvantage of this testing technique is that we can't rely in this technique every time. For example, if the system is not a finite system (not in sequential order), this technique cannot be used.

Another disadvantage is that you have to define all the possible states of a system. While this is all right for small systems, it soon breaks down into larger systems as there is an exponential progression in the number of states.

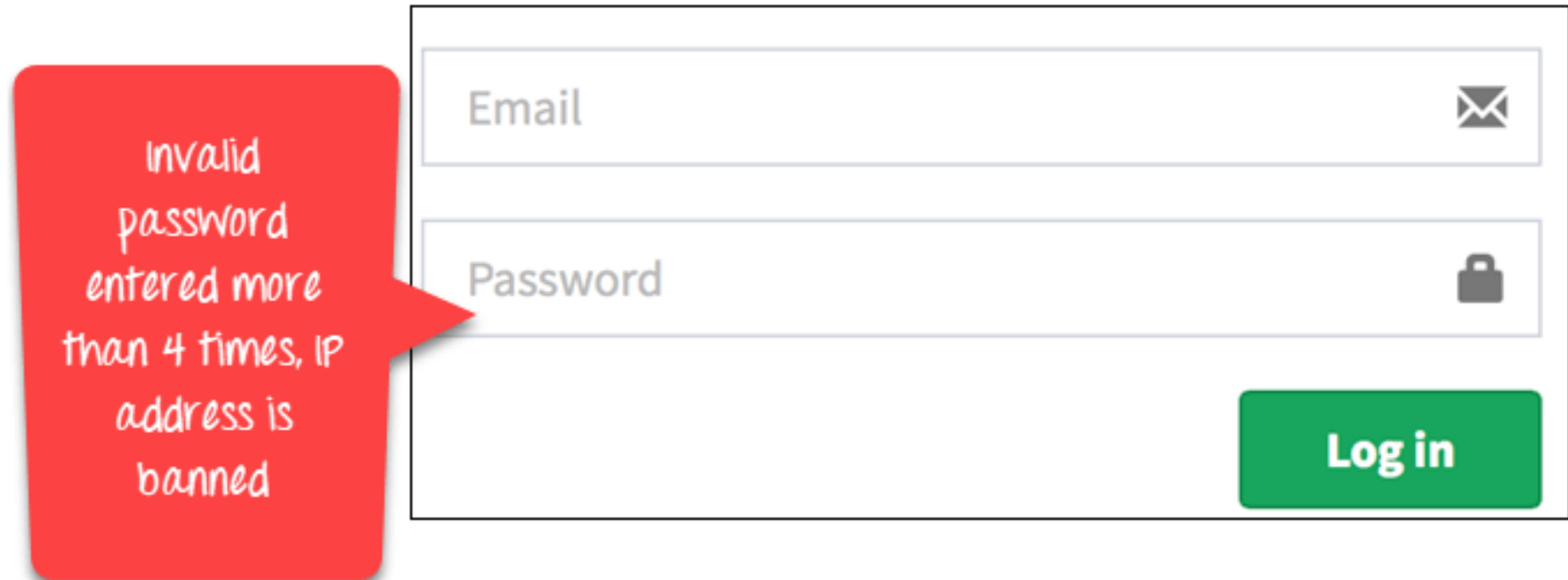
6) Use-case testing

- ▶ A Use Case in Testing is a brief description of a particular use of the software application by an actor or user
 - ▶ Use cases are made on the basis of user actions and the response of the software application to those user actions
 - ▶ It is widely used in developing test cases at system or acceptance level
- 

- ▶ Use Case Testing is a software testing technique that helps to identify test cases that cover entire system on a transaction by transaction basis from start to end
- ▶ Test cases are the interactions between users and software application
- ▶ Use case testing helps to identify gaps in software application that might not be found by testing individual software components

Example

In a use-case, an actor is represented by "A" and system by "S". We create Use for a login functionality of a Web Application as shown below



The image shows a login form with two input fields: "Email" and "Password". The "Email" field has an envelope icon on the right, and the "Password" field has a lock icon on the right. Below the fields is a green "Log in" button. A red callout bubble points to the "Password" field with the text: "Invalid password entered more than 4 times, IP address is banned".

Invalid password entered more than 4 times, IP address is banned

Email

Password

Log in

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	<u>Password not valid</u> S: Display Message and ask for re-try 4 times
	2b	<u>Password not valid 4 times</u> S: Close Application

- Consider the first step of an end to end scenario for a login functionality for our web application where the Actor enters email and password.
- In the next step, the system will validate the password
- Next, if the password is correct, the access will be granted
- There can be an extension of this use case. In case password is not valid system will display a message and ask for re-try four times
- If Password, not valid four times system will ban the IP address.