





Software Testing

Lecture 18

Use Case Based Testing

**Chapter 5:
Dynamic Analysis-Test Design Techniques**



Use Case Based Testing

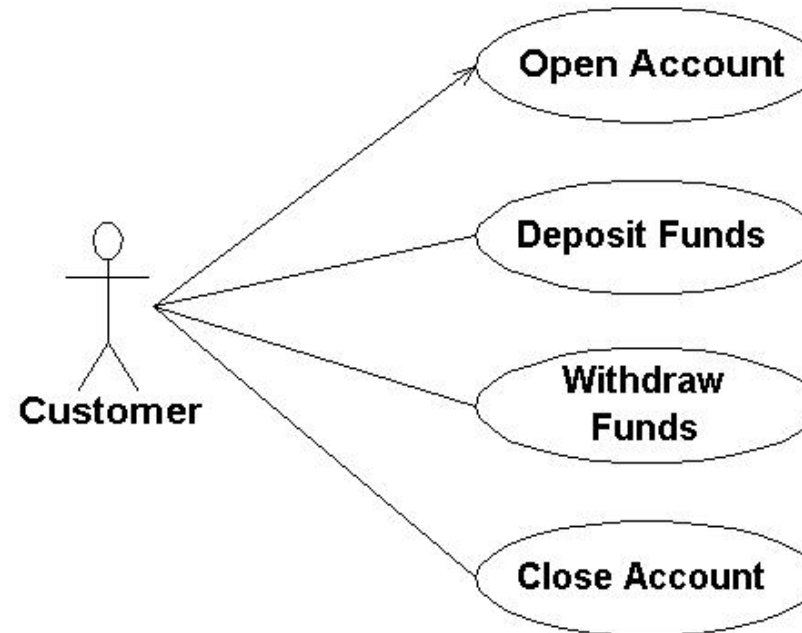
Use Case:

- ❖ **List of steps to achieve a goal**
- ❖ **Interaction between the actor and system**
- ❖ **Captures functional requirements of the system**
- ❖ **Document describing end to end behavior of the system from user perspective**
 - **User Action**
 - **System behavior**
- ❖ **It defines the main scenarios (and optional exceptional scenarios)**
 - **Scenario is sequence of events or workflow**



Use Case Elements

- ❖ **Brief description**
- ❖ **Actor**
- ❖ **Precondition**
- ❖ **Basic Flow**
- ❖ **Alternate flow**
- ❖ **Exception flow**
- ❖ **Post Conditions**





Use Case Based Testing

- ❖ **Black Box Testing**
- ❖ **Tests are designed to execute user scenarios**
- ❖ **Helps us to identify test cases that exercise the whole system.**
- ❖ **Testing of typical and exceptional workflow scenarios for the system**
- ❖ **Functional Testing**
- ❖ **Done from two side**
 - **Actors**
 - **Stakeholders**
- ❖ **Helps to uncover integration defects**



Deriving Test Cases with Use Cases

- ❖ Tests check the typical use of the system
 - Test case table for normal scenarios
- ❖ Every extensions must be tested by a test case
 - Test case table for exceptions
- ❖ If more than one extension exists test the functionality of the important one
- ❖ Important for acceptance of system



Example 1:

Consider the 'Show Student Marks' case, in a School Management System.

- *Use case Name: Show Student Marks*
- *Actors: Students, Teachers, Parents*
- *Pre-Condition:*
 - *1) The system must be connected to the network.*
 - *2) Actors must have a 'Student ID'.*

Main Scenario	Serial Number	Steps
A: Actor/ S: System	1	Enter Student Name
	2	System Validates Student Name
	3	Enter Student ID
	4	System Validates Student ID
	5	System shows Student Marks
Extensions	3a	Invalid Student ID S: Shows an error message
	3b	Invalid Student ID entered 4 times. S: Application Closes



Corresponding Test Case for 'Show Student Marks' case:

Test Cases	Steps	Expected Result
A	View Student Mark List 1 -Normal Flow	
1	Enter Student Name	User can enter Student name
2	Enter Student ID	User can enter Student ID
3	Click on View Mark	System displays Student Marks
B	View Student Mark List 2-Invalid ID	
1	Repeat steps 1 and 2 of View Student Mark List 1	
2	Enter Student ID	System displays Error message



Example 2: Flight Reservation Application

- ❖ Consider the first step of an end to end scenario for a login functionality for our Flight Reservation application where the Actor enters Agent Name and password to login into the Flight Reservation application.
- ❖ 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
- ❖ Or if Password, not valid four times system will close the application



Example 2: Flight Reservation Application

Use Case Name	Login	
Use case Description	A user login to System to access the functionality of the system.	
Actors	Customer, Admin	
Pre-Condition	System must be connected to the network.	
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	1a	<u>Invalid username</u> S :Display error message
	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



Example 3: ATM

Use Case Name: Cash Withdrawal

Actors: Customer, Bank

Description: This Use case describes how the customer uses the ATM system to withdrawal cash from his/her bank account

Pre-conditions:

- 1. ATM system is online**
- 2. The ATM system has sufficient cash balance.**



Example 3: ATM Withdraw Cash

Normal Work Flow for withdrawing cash:

1.	The Customer will insert their debit card.
2.	The System will prompt for the PIN.
3.	The Customer will enter the PIN.
4.	The System will display the option to "Withdraw Cash".
5.	The Customer will select the option to withdraw cash.
6.	The System will prompt for an amount.
7.	The Customer will enter an amount.
8.	The System will submit the amount to the Bank for approval.
9.	The Bank will confirm the transaction.
10.	The System will dispense the cash amount.
11.	The System will return the card.
12.	The use case ends with a success.



Example 3: ATM Withdraw Cash

Extensions:

4a. If the PIN is invalid,

1.	The System will return the card.
2.	The use case ends with a failure.

8a. If the amount is invalid (see additional business rules),

1.	The System will display an error message and prompt for another amount.
2.	The use case resumes at step 7.

10a. If the Bank declines the transaction,

1.	The System will return the card.
2.	The use case ends with a failure.

3a,5a,7a If the Customer selects the option to "Cancel",

1.	The System will return the card.
2.	The use case ends.



Example 3: ATM Withdraw Cash

Additional Business rule:

- ❖ **The system will dispense a maximum of \$100 in a single transaction**

Post conditions:

- ❖ **The system will receive their cash amount.**


Now do Use Case Testing

- 1. Review the use case whether the requirements are complete or not**
- 2. Review the Exceptions workflows whether they are complete or not.**



Example 3: ATM Withdraw Cash

Test Cases for Normal Workflow:

TEST CASE(S)	Steps	Expected Results
Test Case:	Cash withdrawal1 - Normal workflow	
1)	Insert debit card. 	The System will prompt for the PIN.
2)	Enter the valid PIN.	The System will display the option to "Withdraw Cash".
3)	Select the option to "Withdraw Cash".	The System will prompt for an amount.
4)	Enter amount as \$100 (valid amount).	The System will dispense the cash amount. The System will return the card.



Advantages and Disadvantages of Usecase based testing

Advantages:

- ❖ **User focused, because testing is done by user perspective**
- ❖ **This testing type helps in covering all the possible flows that the actors are likely to take. So better coverage.**
- ❖ **Reduces time in test case design.**

Disadvantages:

- ❖ **If any flow or use case is missing in the use case document, it will also have an impact on the testing process.**
- ❖ **Not suitable for non-functional requirements**



► Questions and Answers





بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ



Software Testing

Lecture 19

White Box Testing

Chapter 5:
Dynamic Analysis-Test Design Techniques



White Box Testing

Definition by ISTQB

- ❖ ***White-box testing:*** Testing based on an analysis of the internal structure of the component or system.
- ❖ ***White-box test design technique:*** Procedure to derive and/or select test cases based on an analysis of the internal structure of a component or system.

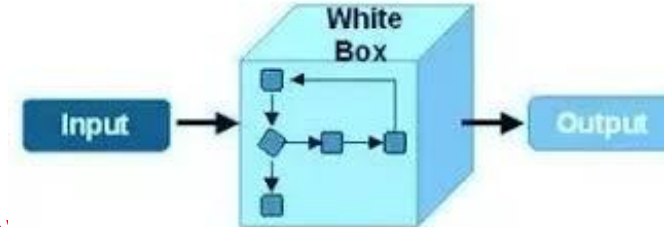
Test cases design:

- ❖ **Test cases are designed with the help of the program structure**



WHITE BOX TESTING

- ❖ Also called Structured, clear box, open box. Glass box testing
- ❖ Strategy in which testing is based on
 - Structure
 - Internal Paths
 - Implementation
- ❖ Goal is to verify working flow of an application.
- ❖ Programming knowledge is essential.



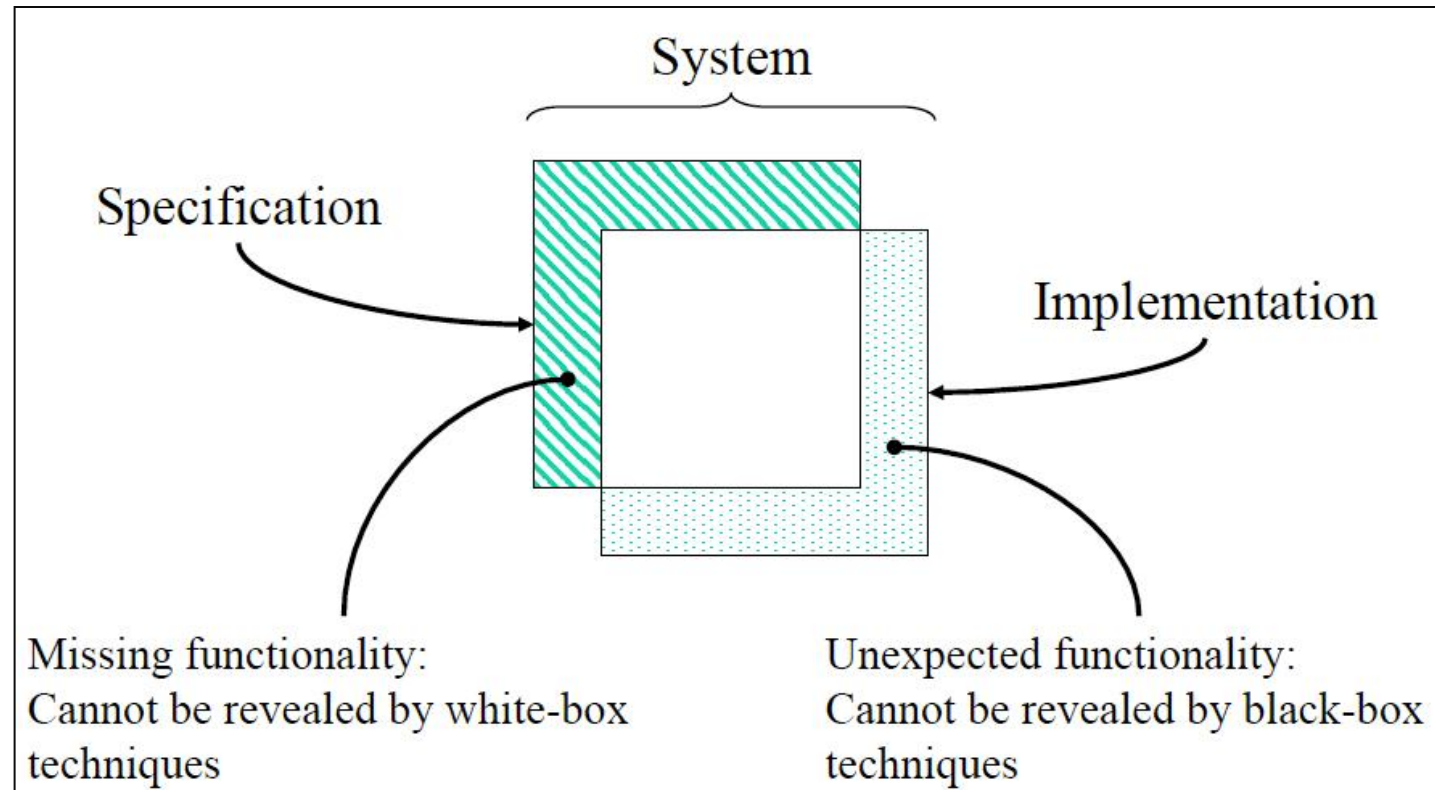


Example

- ❖ A tester, usually a developer as well, studies the implementation code of a certain field on a webpage
- ❖ Determines all legal (valid and invalid) AND illegal inputs
- ❖ Verifies the outputs against the expected outcomes, which is also determined by studying the implementation code.
- ❖ **WBT is like work of mechanic who examines the engine to see why the car is not moving**

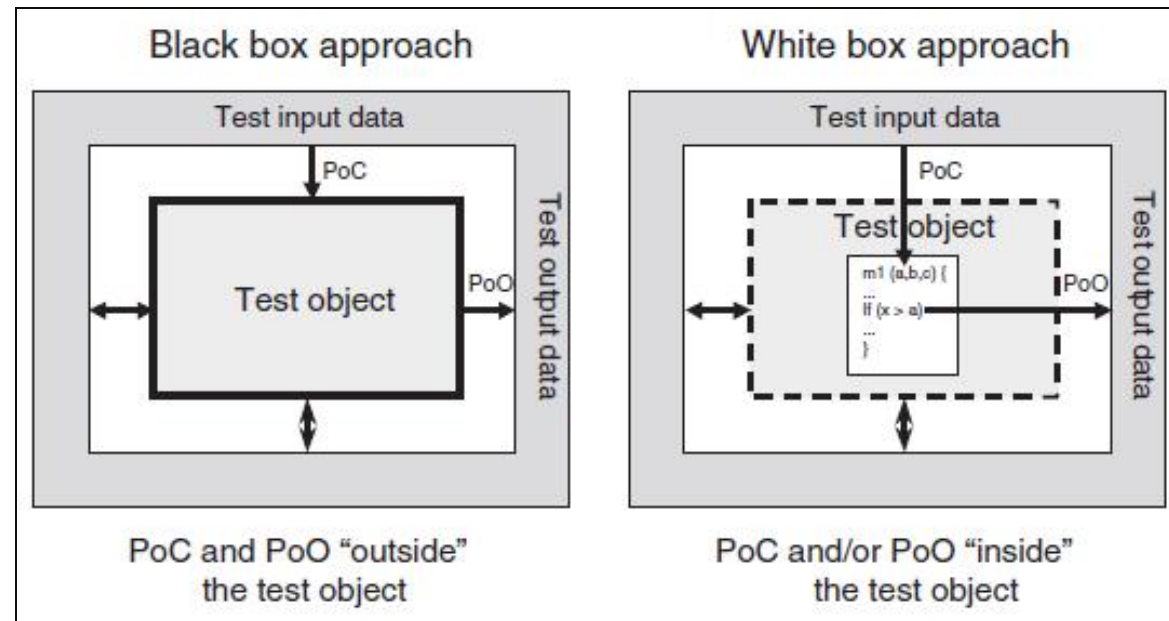


Black Box VS White Box





Black Box VS White Box



- ❖ **Black Box Example:** Search something on google by using keywords.
- ❖ **White Box Example:** By input to check and verify loops



White Box Testing - Applicability

It's applicable to the following levels of software testing:

- ❖ **Unit Testing**
 - For testing paths within a unit.
- ❖ **Integration Testing**
 - For testing paths between units.
- ❖ **System Testing**
 - For testing paths between subsystems.



What do we verify in White Box Testing?

- ❖ **Internal security holes**
- ❖ **Broken or poorly structured paths in the coding processes**
- ❖ **The flow of specific inputs through the code**
- ❖ **Expected output**
- ❖ **The functionality of conditional loops**
- ❖ **Testing of each statement, object and function on an individual basis**



How do you perform White Box Testing?

Two steps to do:

1: Understand the source code

2: Create test cases and execute

- **Test cases are designed and selected based on code.**
- **Why we perform WBT?**
- **To ensure:**
 - ❖ **all independent paths**
 - ❖ **All logical decisions**
 - ❖ **All loops executed at their boundaries**



White Box Testing Tools

Some of the prominent tools:

- ❖ **JUnit**
- ❖ **Veracode**
- ❖ **EclEmma**
- ❖ **RCUNIT**
- ❖ **Cfix**
- ❖ **Googletest**
- ❖ **EMMA**



White Box Testing Techniques

- ❖ **Basis for WBT is source code.**
- ❖ **Foundation of WBT is to execute every part of code at least once.**
- ❖ **WBT coverage criteria is**
 - ❖ **- Covering all statements, branches, conditions etc**
- ❖ **Different test cases => Different execution paths**
- ▶ **Techniques are:**
 - ❖ **Statement Coverage Testing**
 - ❖ **Decision Coverage Testing**
 - ❖ **Condition Coverage Testing**
 - ❖ **Multiple Condition Testing**
 - ❖ **Path Coverage Testing**
 - ❖ **Data Flow Based Testing**



Statement Coverage Testing:

- ❖ **Statement :**
 - ▶ - '**An entity** in a programming language, which is typically the smallest indivisible unit of execution' (ISTQB Def).
- ❖ **Statement coverage:**
 - ▶ - 'The percentage of executable statements that has been exercised by a test suite'. (ISTQB Def)
- ❖ **Test Coverage:**
 - **Test coverage measures the amount of testing performed by a set of tests.**
- ❖ **Coverage = (Number of coverage items exercised/Total number of coverage items) * 100**



White Box Testing Techniques

Statement Coverage Testing:

- ❖ White box test design technique
- ❖ Execute all or selected statements of the test object
- ❖ Traverse all statements at least once
- ❖ Covers only the true conditions.
- ❖ What the source code is expected to do and what it should not.
- ❖ Drawback of this technique is that we cannot test the false condition in it.



Statement Coverage Testing

- ❖ **Translate the source code into a control flow graph.**
- ❖ **Statements are represented as nodes (circles)**
- ❖ **Control flow between the statements is represented as edges (connections).**
- ❖ **Sequences of unconditional statements are illustrated as one single node**
- ❖ **Unreachable code can be detected**
- ❖ **Empty ELSE-parts are not considered**



Statement Coverage Testing

- ❖ **Test Requirements= Statements in the program**
- ❖
$$\text{Statement Coverage} = \frac{\text{Number of executed statements}}{\text{Total number of statements}} \times 100$$
- ❖ **Test Completeness Criteria (% of the statements in the software which were executed atleast once)**
- ❖ **How 100% statement coverage can be achieved?**
- ❖ **Aim is to achieve the maximum amount of statement coverage with the minimum number of test cases**
- ❖ **In some situation one test case can also be enough**
- ❖ **Statement coverage is also called C0- coverage/measure**



► Questions and Answers



