Recall The Team Skills

1. Analyzing the Problem
2. Understanding User and Stakeholder Needs
3. Defining the System
4. Managing Scope
5. Refining the System Definition
6. **Building the Right System**
   - From Use Cases to Implementation
   - **From Use Cases to Test Cases**
   - Tracing Requirements
   - Managing Change
   - Assessing Requirements Quality

Chapter 26

From Use Cases to Test Cases

- The tester perspective
- Testing terms
- Test cases from use cases
- Black Box vs. White Box testing

A Tester's Perspective: Musings on the Big Black Box

- Traditionally, testers come late in the development process.
- They see the system as a black box because they know little about it.
- They may ask the following
  - What is the system supposed to do and in what order?
  - What are the things that may go wrong?
  - How can we create test scenarios?
  - How could I know that the system is tested completely?
  - Anything else about the system?
  - Is there a way to start testing earlier?

A Tester's Perspective: Musings on the Big Black Box

- This well be different if we have use cases.
- Testers will have black box +
  - Comprehensive use case model showing how the system behave, actors, and system-user interaction.
  - each use case has basic and alternative flow of events, pre-conditions, post-conditions
  - Supplementary nonfunctional requirements

- Thus use-case technique can derive the testing process.

Use case = test case ??

- Not really.
- We still need to make a lot of analysis to derive test cases from the use cases.

Common Testing Terms

- Test Plan: contains information about the purpose and goals of testing within the project, the strategies and resources needed to execute the testing process.

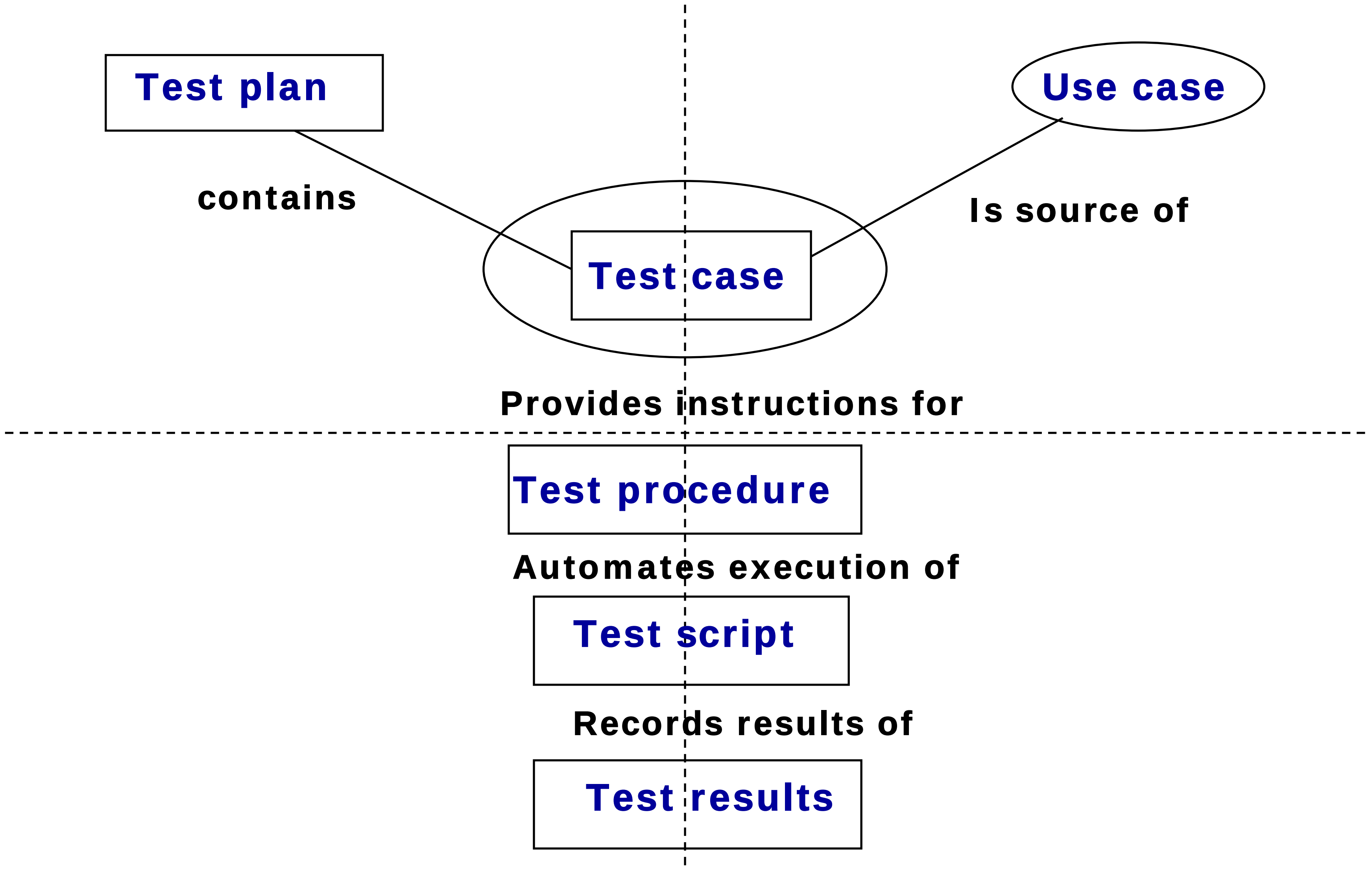
- Test case: set of test inputs, execution conditions and expected results developed for a particular objective (like satisfying a requirement)

- Test Procedure: set of detailed instructions for the setup, execution, and evaluation of results for a given test case.

Common Testing Terms

- Test script: a software script that automates the execution of the test procedure.
- Test coverage: defines the degree to which a given test or a set of tests addresses all specified test cases for a given system or component.
- Test item: a build that is an object of testing
- Test results: set of data captured during the execution of a test

Relationships of test artifacts



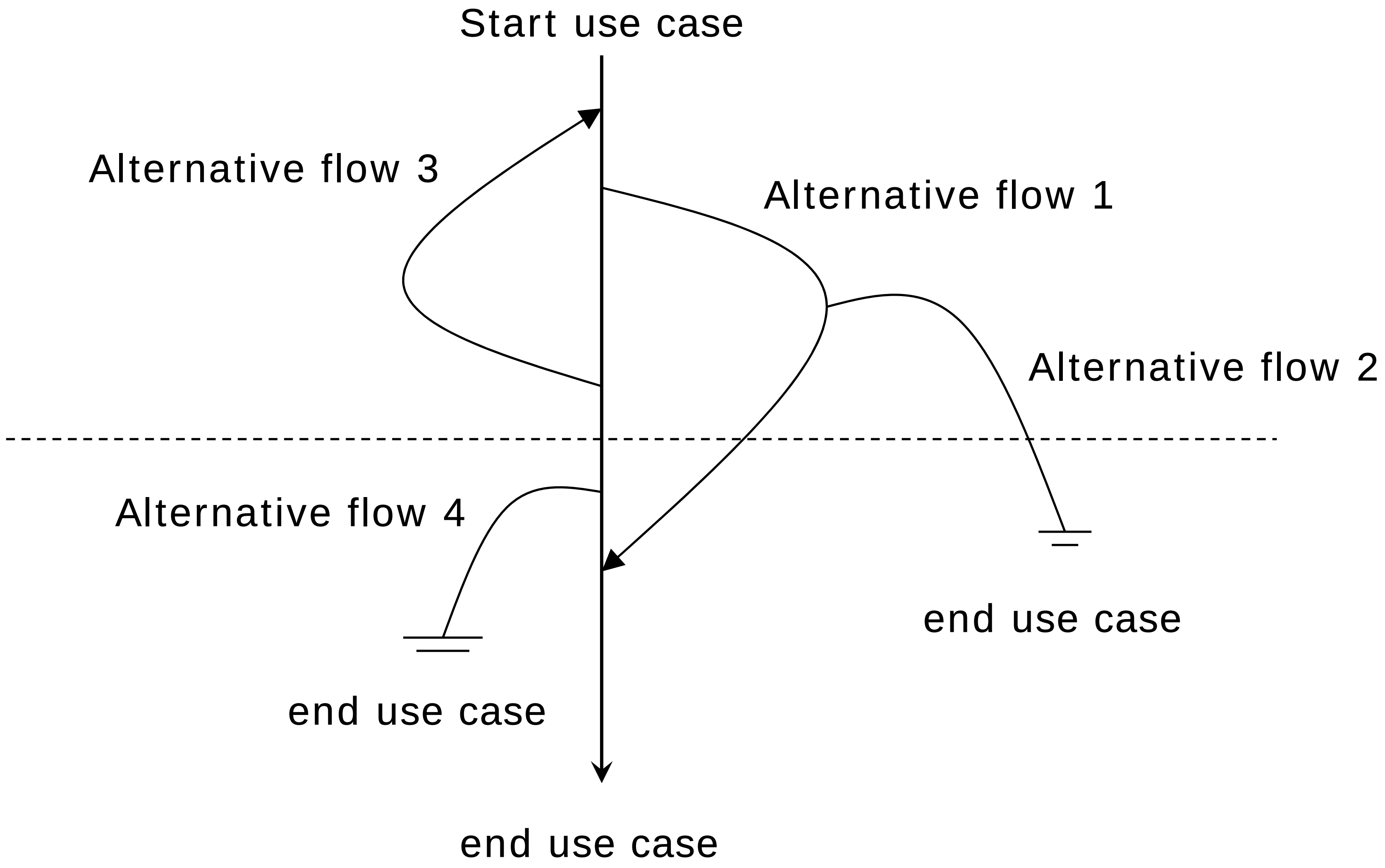
The role of test cases

- Test cases forms the foundation on which to design and develop test procedure
- Depth of testing is proportional to the number of test cases
- Scale of test effort is proportional to the number of test cases
- Test design, development and resources are governed by the test cases

Use case scenarios

- A scenario is an instance of a use case
- That is, it is a use case execution wherein a specific user executes the use case in a specific way

Use case scenarios



Deriving test cases from use cases: A four step processes

1. Identify the use case scenarios
2. For each scenario, identify one or more test cases
3. For each test case, identify the conditions that will cause it to execute.
4. Complete the test case by adding data values

Identify the use case scenarios

- Use simple matrix that can be implemented in a spreadsheet, database or test management tool.
- Number the scenarios and define the combinations of basic and alternative flows that leads to them.
- Many scenarios are possible for one use case
- Not all scenarios may be documented .. Use an iterative process
- Not all documented scenarios may be tested
  - Use cases may be at a level that is insufficient for testing
  - Team’s review process may discover additional scenarios

Identify the use case scenarios Example

Scenario number	Originating flow	Alternative flow	Next alternative	Next alternative
1	Basic flow			
2	Basic flow	Alt. flow 1		
3	Basic flow	Alt. flow 1	Alt. flow 2	
4	Basic flow	Alt. flow 3		
5	Basic flow	Alt. flow 3	Alt. flow 1	
6	Basic flow	Alt. flow 3	Alt. flow 1	Alt. flow 2
7	Basic flow	Alt. flow 4		
8	Basic flow	Alt. flow 3	Alt. flow 4	

Identify the test cases

- Parameters of any test case:
  - Conditions
  - Input (data values)
  - Expected result
  - Actual result

Test case ID	Scenario / conditon	Data value 1	Data value 2	Data value N	Exp. results	Actual results
1	Scenario 1					
2	Scenario 2					
3	Scenario 3					

Identify the test conditions

- For each test case identify the conditions that will cause it to execute a specific events.
- Use matrix with columns for the conditions and for each condition state whether it is
  - valid (V): must be true for the basic flow to execute
  - Invalid (I): this will invoke an alternative flow
  - Not applicable (N/A): to the test case
- Read HOLIS example page 314-315

Add data values to complete the test cases

- Design real input data values that will make such conditions to be valid or invalid and hence the scenarios to happen.
- You may want to look at the use case constructs and branches.

Managing test coverage

- Select the most appropriate or critical use cases for the most thorough testing
- Choose the use cases based on a balance between the cost, risk, and necessity of verifying the use case.
- Determine the relative importance of your use cases by using a priority algorithm

Black-box vs. white-box testing

- White-box testing

  =internal inspection

  =design assurance

look inside the system and see how it does the things. Look at the architecture and the implementation of the system

Key Points

- One of the greatest benefits of the use case techniques is that it builds a set of assets that can be used to derive the testing process.
- Use cases can directly derive or seed the development of test cases
- The scenarios of a use case create templates for individual test cases
- Adding data values completes the test cases
- Testing non-functional requirements completes the testing process.