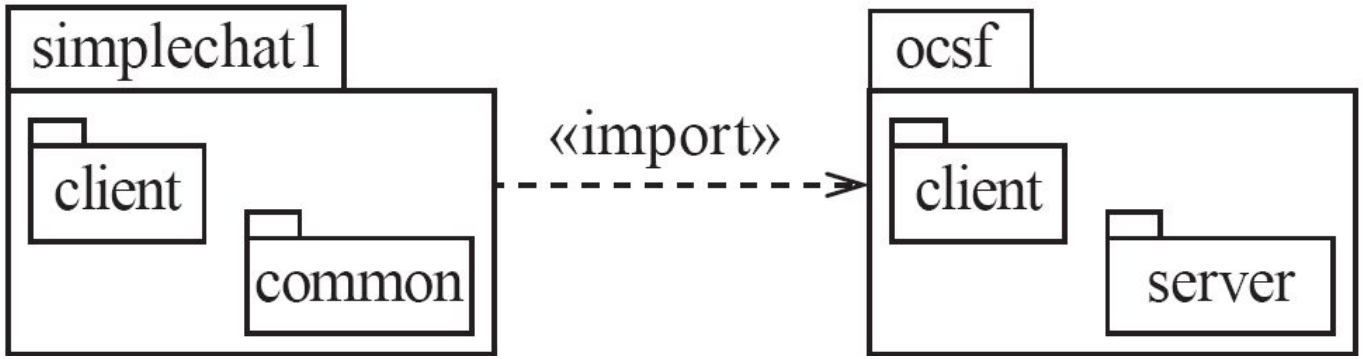
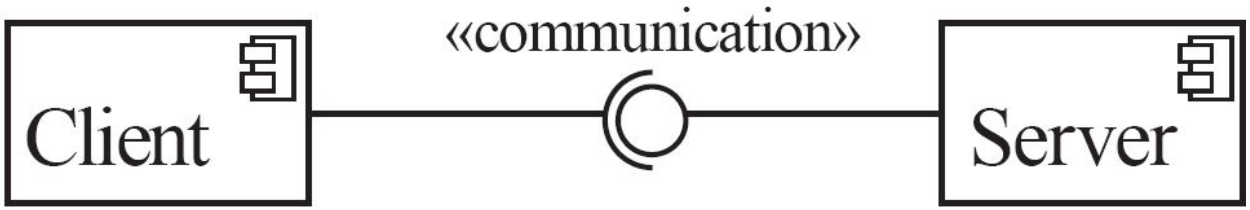
UML Architectural Diagrams

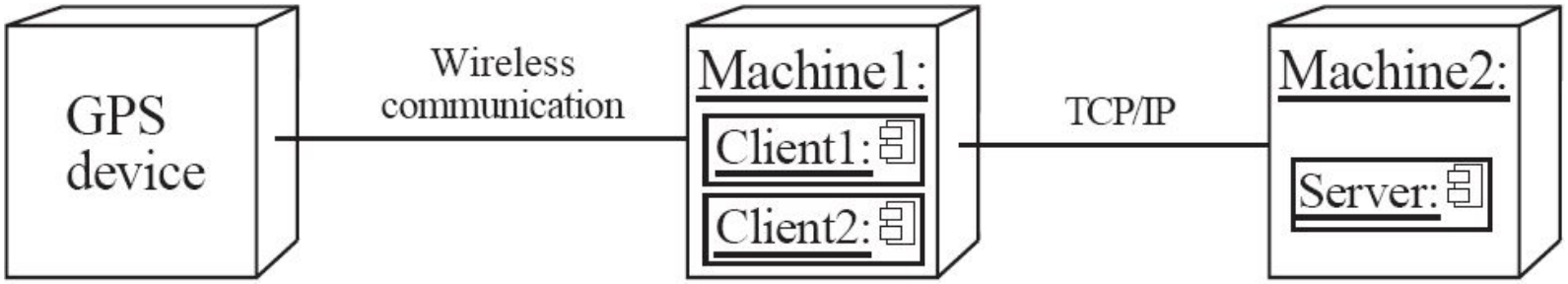
Package diagrams



Component diagrams



Deployment diagrams



Software Testing

Introduction

- What is it?
- Why is it important?

What is the work product?

- Need specification to know what to test

- Need expected output to know whether a particular test has ‘passed’

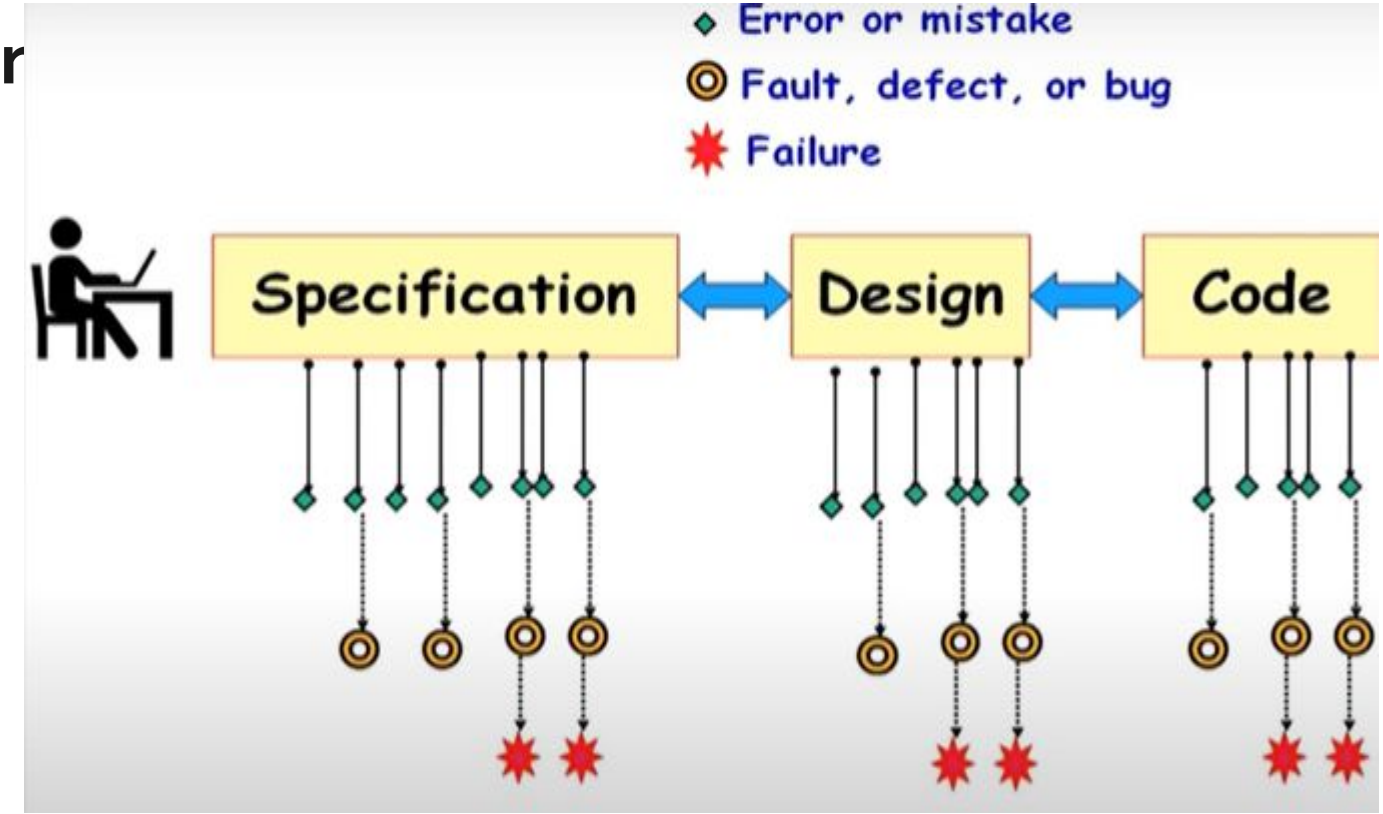
A Test case

Describe input, action and an expected response,
in order to determine if a feature of an application is working correctly.

TEST CASE ID	TEST SCENARIO	TEST CASE	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/ FAIL)
TC_LOGIN_001	Verify the login of Gmail	Enter valid User Name and valid Password	1. Need a valid Gmail Account to do login	1. Enter User Name	<Valid User Name>	Successful login	Gmail inbox is shown		
				2. Enter Password	<Valid Password>				
				3. Click "Login" button					
TC_LOGIN_001	Verify the login of Gmail	Enter valid User Name and invalid Password	1. Need a valid Gmail Account to do login	1. Enter User Name	<Valid User Name>	A message "The email and password you entered don't match" is shown			
				2. Enter Password	<Invalid Password>				
				3. Click "Login" button					
TC_LOGIN_001	Verify the login of Gmail	Enter invalid User Name and valid Password	1. Need a valid Gmail Account to do login	1. Enter User Name	<Invalid User Name>	A message "The email and password you entered don't match" is shown			
				2. Enter Password	<Valid Password>				
				3. Click "Login" button					
TC_LOGIN_001	Verify the login of Gmail	Enter invalid User Name and invalid Password	1. Need a valid Gmail Account to do login	1. Enter User Name	<Invalid User Name>	A message "The email and password you entered don't match" is shown			
				2. Enter Password	<Invalid Password>				
				3. Click "Login" button					

Fundamer

- Error
- Defect
- Failure



Test effectively and efficiently.


Program testing can be used to show the presence of bugs, but never to show their absence!
(Edsger Dijkstra)
lzquotes.com

What happens after Test? - Debugging

- Debugging is not testing but often occurs as a consequence of testing
- When a test case uncovers an error, debugging is the **process that results in the removal of the error**
- Attempts to match symptom with cause, thereby leading to error correction
- The activity must track down the cause of an error
- Debugging is difficult

Fundamentals

- Exhaustive Testing - Not Practical
- Software Testing is an attempt to test a small possible subset

So, the basic questions

- What to test? TEST CASE SELECTION
- How to test? **STOPPING CRITERION**

Test Characteristics

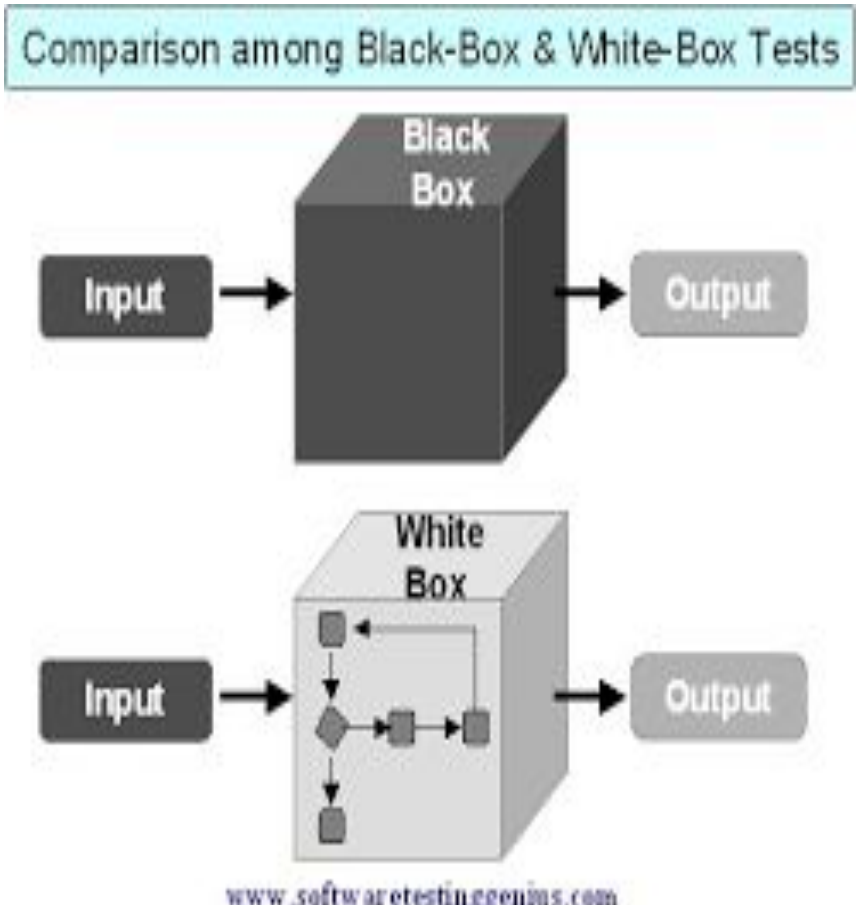
- *A good test has a high probability of finding an error.*
- *A good test is not redundant*
- *A good test should be "best of breed"*
- *A good test should be neither too simple nor too complex*

"It does not guarantee absence of errors"

Testing Principles

- All tests should be *traceable to customer/user requirements*
- Tests should be ***planned long before* testing begins**
- The *Pareto principle* applies to software testing
- Testing *should begin "in the small" and progress toward testing "in the large"*
- *Exhaustive testing* is not possible
- To be most effective, testing should be conducted by an ***independent third party***

Two Types of Testing



What criteria to use?

- Should the testing be done based on externally observable behavior ?

Or

- Should the code be seen to design testcases?

External Vs Internal View

-Black box testing

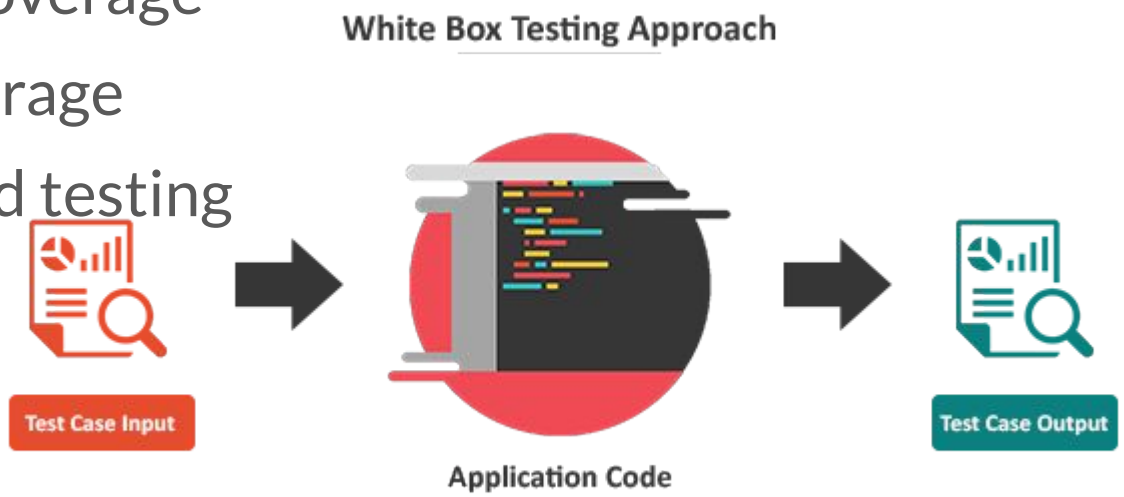
- White box testing

Black box testing

- Also called as *functional testing* or *behavioral testing*,
- Tests in the external point of view
- **Specifications are used to generate testcases**
- Tests for **absence of features**
- **No programming knowledge** is required

White box testing

- Also called as *glass box testing* or *structural testing*
- Tests the internal point of view or implementation
- Cannot detect absence of features
- Coverage measures are used
  - Statement coverage
  - Branch Coverage
  - Path oriented testing



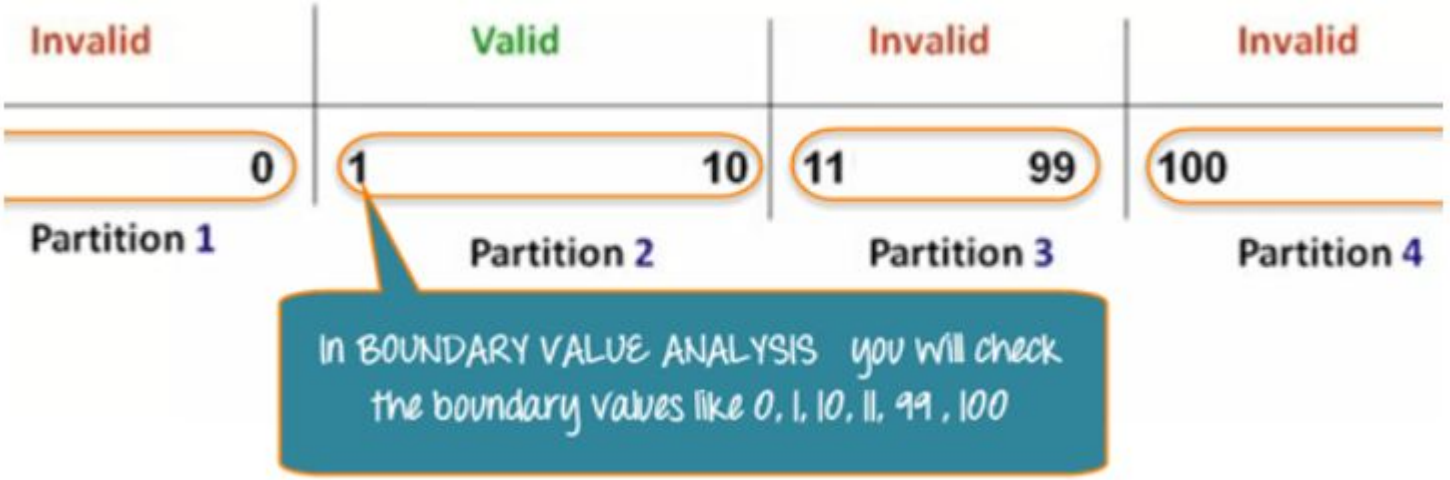
Black box testing techniques

- Equivalence Partitioning
  - *divides the input domain* of a program into classes of data

Equivalence Partitioning		
Invalid	Valid	Invalid
<=50	50-90	>=90

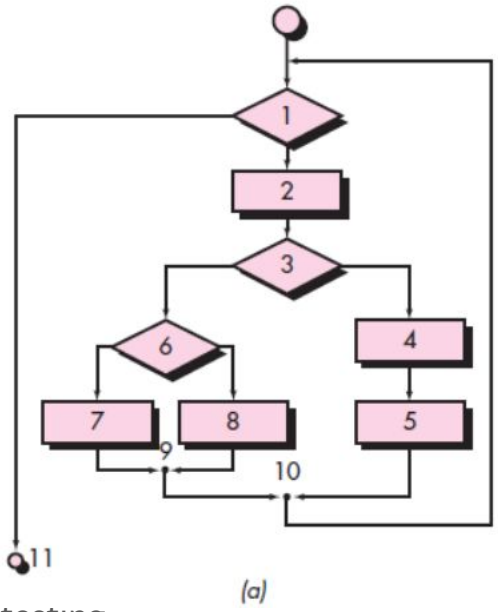
Black box testing techniques

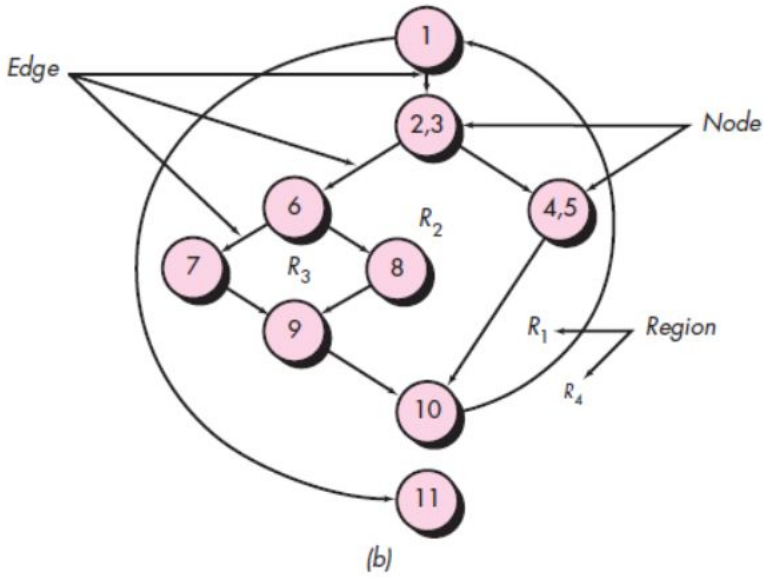
- Boundary Value Analysis
  - Boundaries are very good places for errors to occur.
    - For example – If valid range is 1 to 10 then test for 1; 10 also apart from valid and invalid inputs



White box testing techniques

- Basis Path Testing



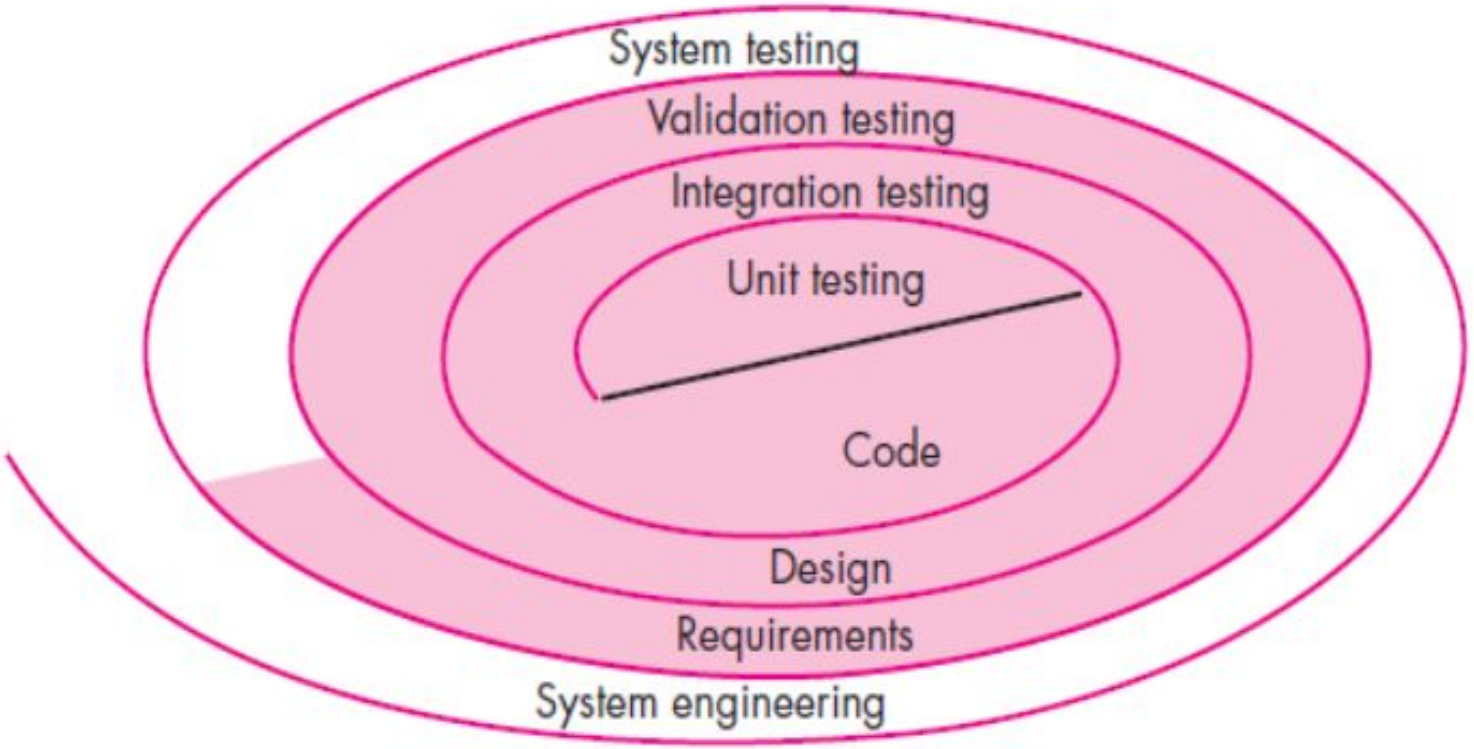


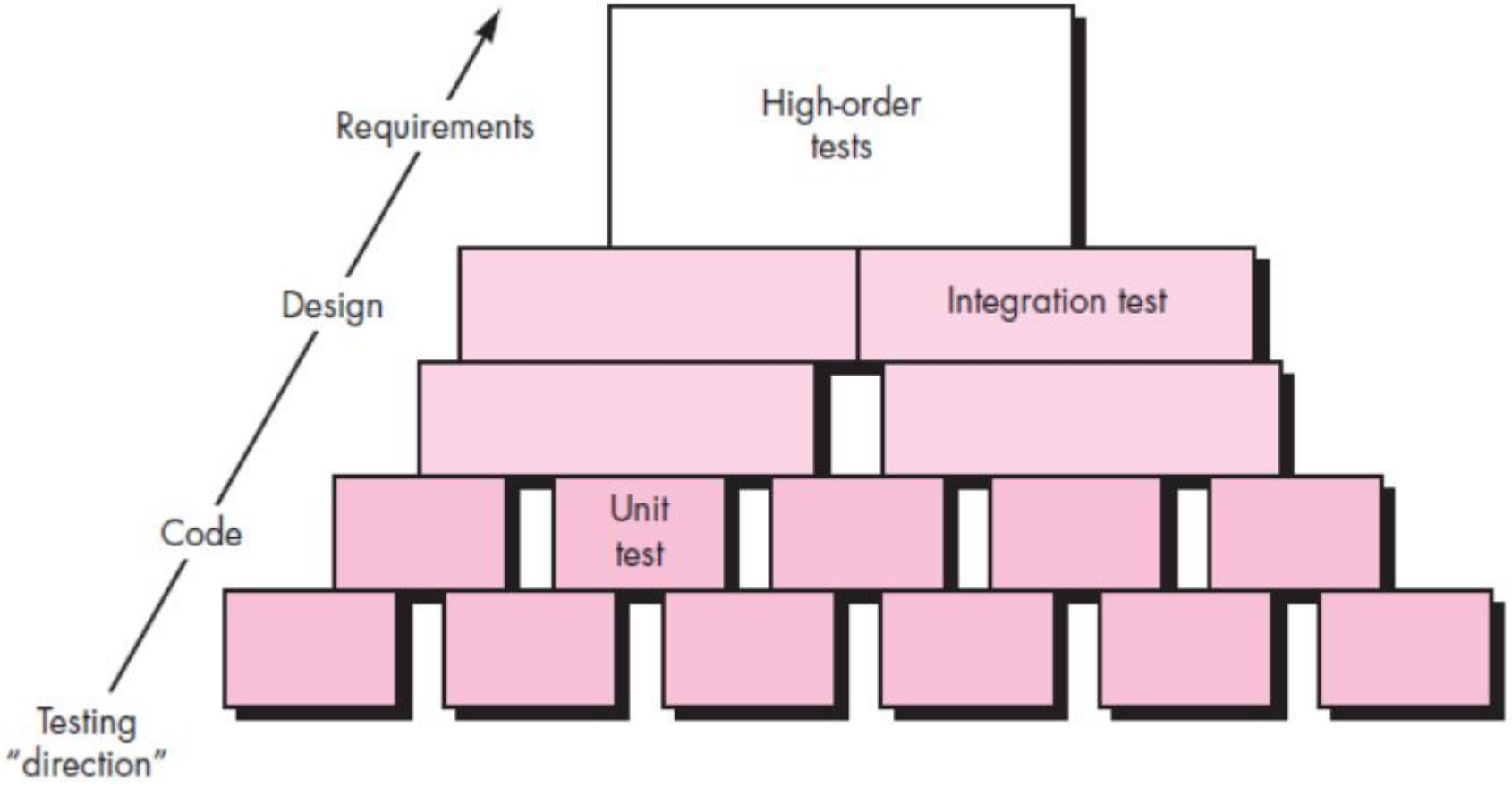
- Control structure testing
  - Condition testing
  - Loop testing

Testing Strategies for Software Process

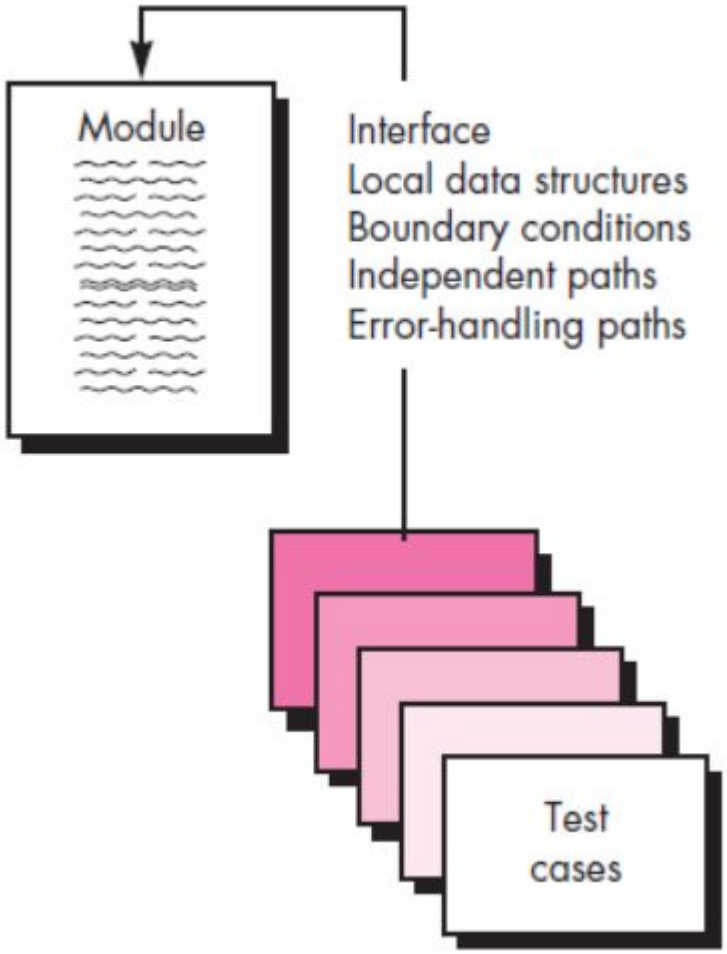
- Testing **begins at the component level** and **works "outward"** toward the **integration** of the entire computer-based system.
- Different testing techniques are appropriate for different software engineering approaches and at different points in time.

Software Testing Strategy—The Big Picture

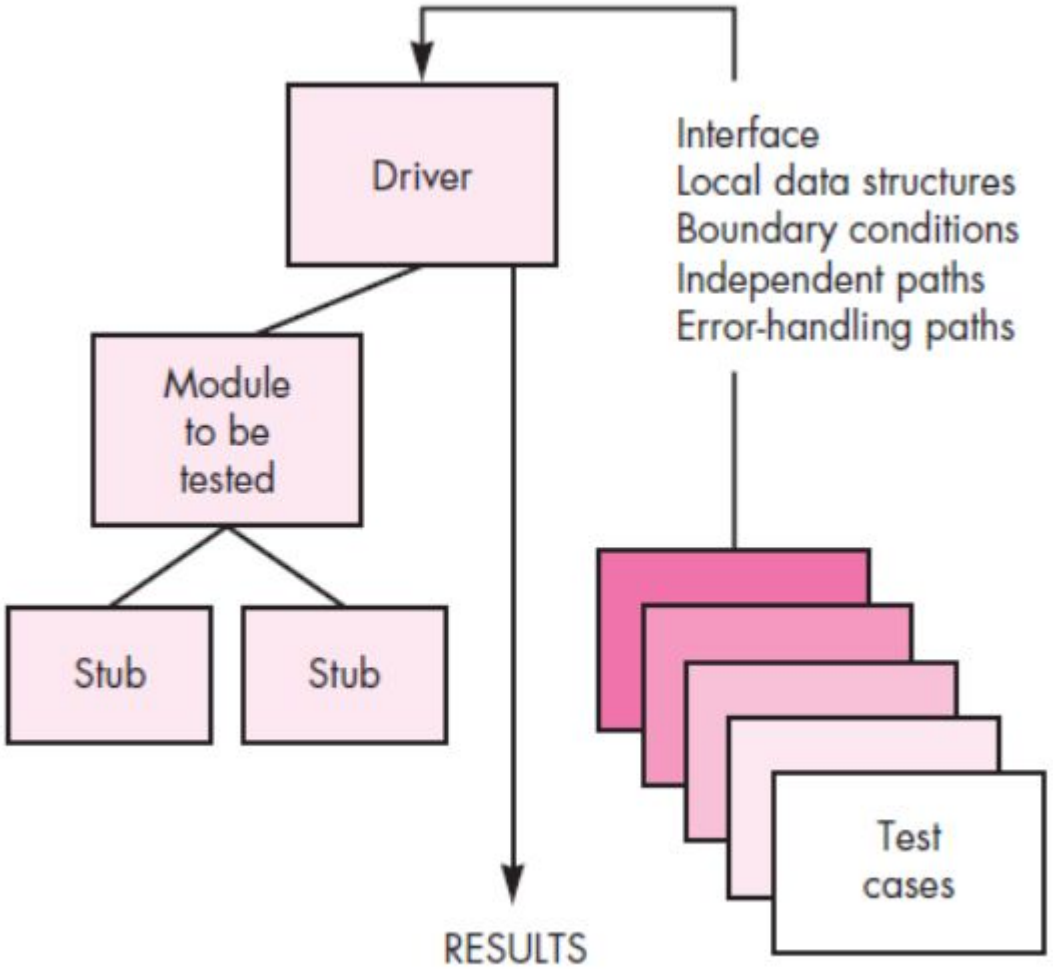



Requirements
Design
Code
Testing
"direction"
High-order
tests
Integration test
Unit
test

Unit Testing



Unit Test Procedure



Integration Testing

Once all modules have been unit tested:

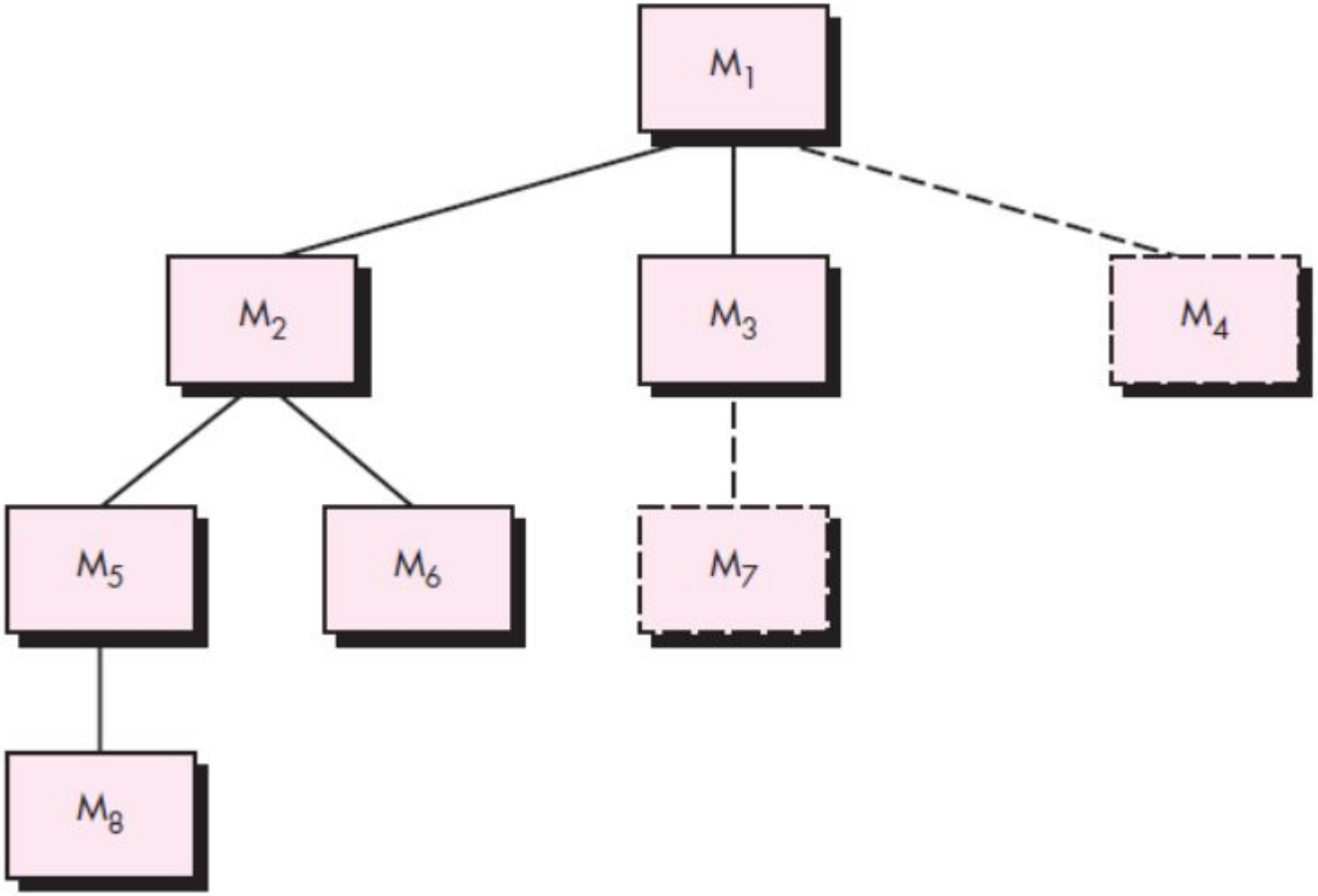
“If they all work individually, why do you doubt that they’ll work when we put them together?”

The problem, of course, is

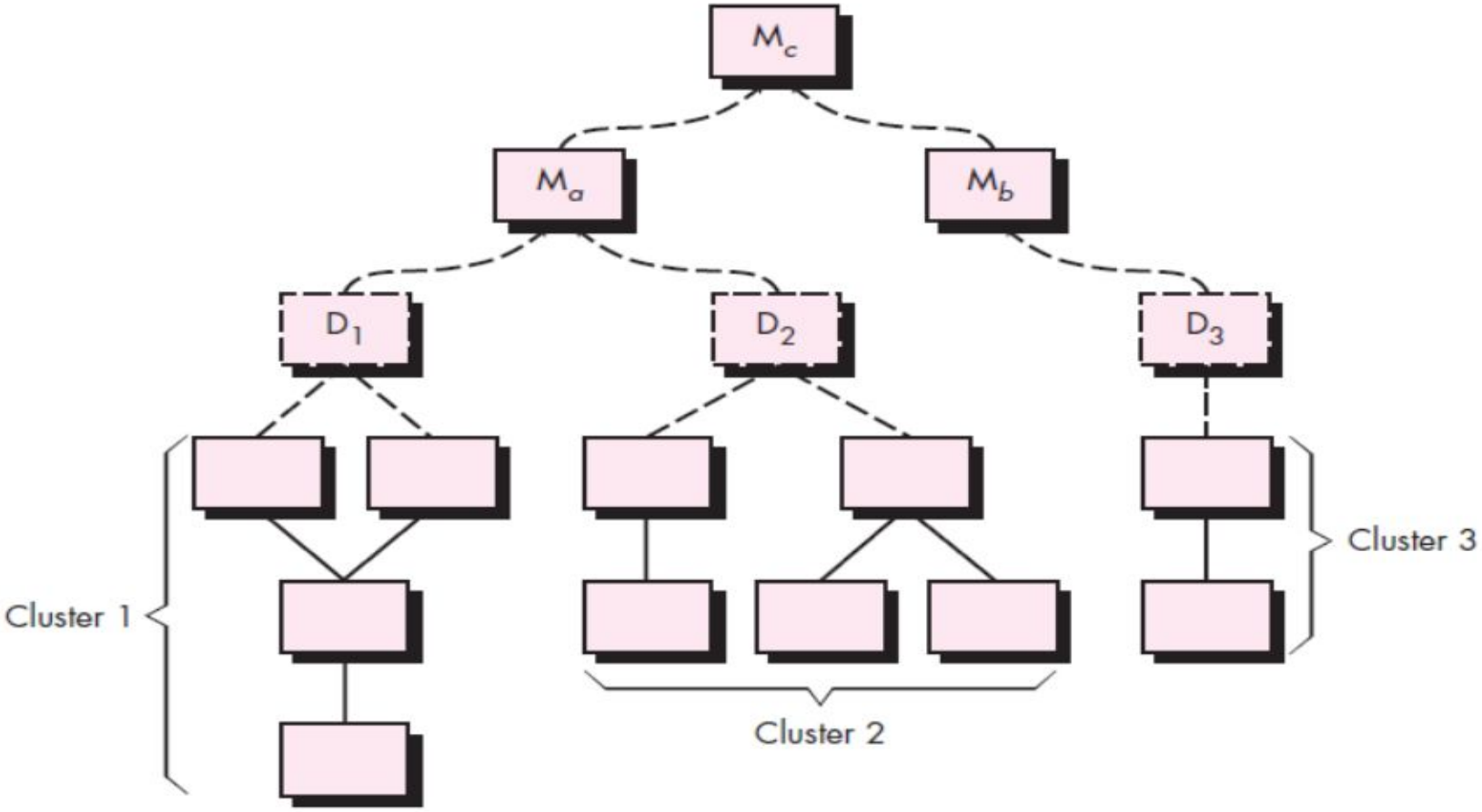
“putting them together”—interfacing.

- Incremental Approach is desirable
  - Top-Down Integration
  - Bottom-Up Integration

Top-down Integration



Bottom-up integration



Regression testing

- *Re-execution of some subset of tests that have already been* conducted to ensure that changes have not propagated unintended side effects.

- Whenever software is corrected, some aspect of the software configuration (the program, its documentation, or the data that support it) is changed.

- Regression testing helps to ensure that changes do not introduce unintended behavior or additional errors.

Validation Testing

- Testing focuses on user-visible actions and user-recognizable output from the system.
- Validation **succeeds when software functions in a manner that can be reasonably expected by the customer**
- If Software Requirements Specification has been developed, it forms the basis for a validation-testing approach

Acceptance Test

- Alpha Testing
  - Conducted at the developer's site by a representative group of end users in a controlled environment.
  - The software is used in a natural setting with the developer and records errors and usage problems.
- Beta Testing
  - Conducted at one or more end-user sites
  - The developer generally is not present.
  - It is a "live" application of the software in an environment that cannot be controlled by the developer.
  - The customer records all problems.

System Testing

- Recovery Testing
  - Systems must recover from faults and resume processing with little or no downtime.
  - It is a system test that forces the software to fail in a variety of ways and verifies that recovery is properly performed
- Security Testing
  - Security testing attempts to verify that protection mechanisms built into a system will, in fact, protect it from improper penetration
- Stress Testing
  - Stress testing executes a system in a manner that demands resources in abnormal quantity, frequency, or volume
- Performance Testing
  - It test the run-time performance of software within the context of an integrated system.

Code Inspection

- An inspection is an activity in which one or more people systematically examine source code or documentation, looking for defects.
- Both testing and inspection rely on different aspects of human intelligence
- Inspecting allows you to get rid of many defects quickly.

THANK YOU