Course for Software Testing

BS (Software Engineering)
Spring Semester
February 2014
Foundation University Rawalpindi Campus

Instructor: Sohaib Altaf

sohaib@hybriditservices.com

http://www.hybriditservices.com/course/FU-BSSE8-ST

Lecture 7

Test Plan: How? Testing Types?

The Test Plan(Cont..)

- How (Manual, Tool-based, Both partial, Types: Black or While or Both, How much tests)
- Where (in-house or independent, or alpha, or beta, bugs propagated from past)
- When (Tentative Plan, Model: Waterfall or Incremental, Unit or More Units, Integration of Units)
- What? Testing Types

Classification: Types of testing

- More defects uncovering with little test, more cost will reduce
  - To minimize the costs associated with testing and with software failures
  - a goal of testing must be to uncover as many defects as possible with as little testing as possible.

- To fulfill this goal
  - Write test cases that have a high likelihood of uncovering the faults that are the most likely to be observed as a failure in normal use.

  - It is simply impossible to test every possible input-output combination of the system

  - There are simply too many permutations and combinations. (A, [8, 2, 4, 2]), (D, [9, 0, 5, 3]), (R, [7, 3, 6, 2])…….

  - Strive to write test cases that will uncover as many faults in as few test cases. (A, [8, 2, 4, 2]) is enough for all +ve integers that are > 0 & < 33000

Classification: Types of testing

- To Coverage/ Achieve Goal:
  - Test the Features & Characteristics
  - Test Functional & Non-Functional Requirements
  - Test all quality attributes

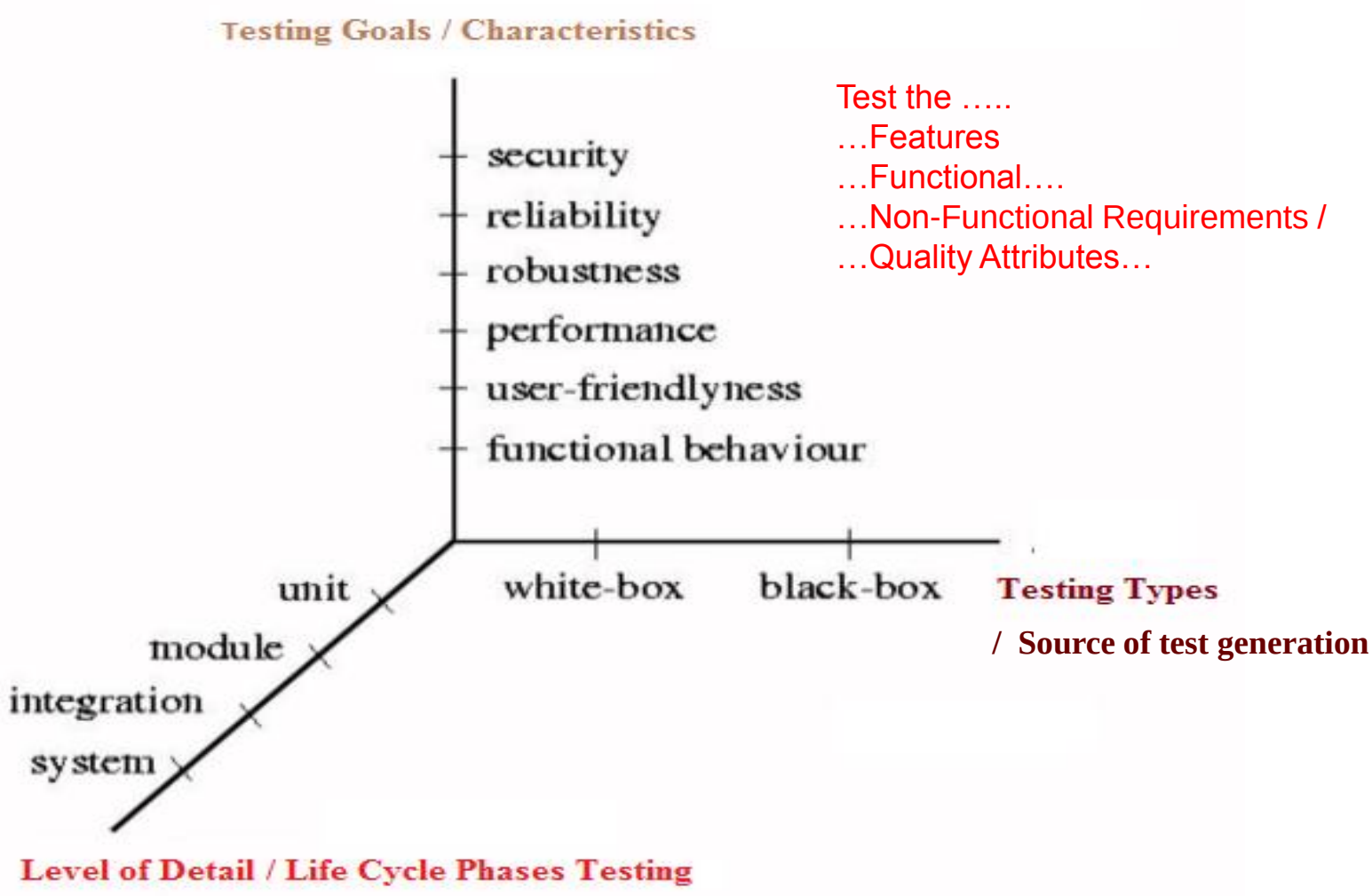
- To Reduce Defects on Increments:
  - Test the unit, test it after integration, Test the System

- To Test each aspects, to discourage accumulation of test:
  - Test Artifacts

- To Test with Time Dimension:
  - Process Models

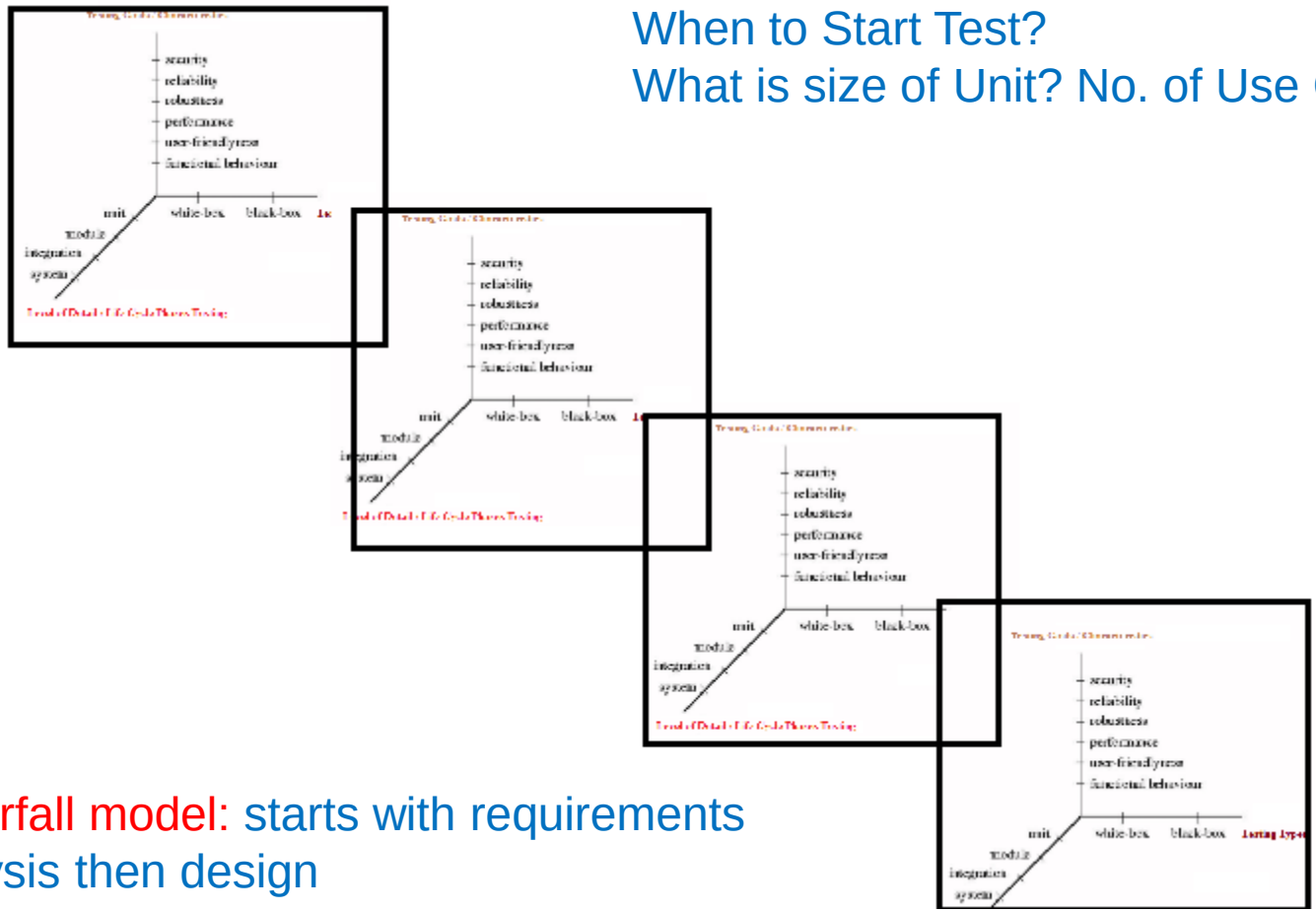
- To test with reduced effort:
  - White & Black Box Testing

Classification: Types of testing



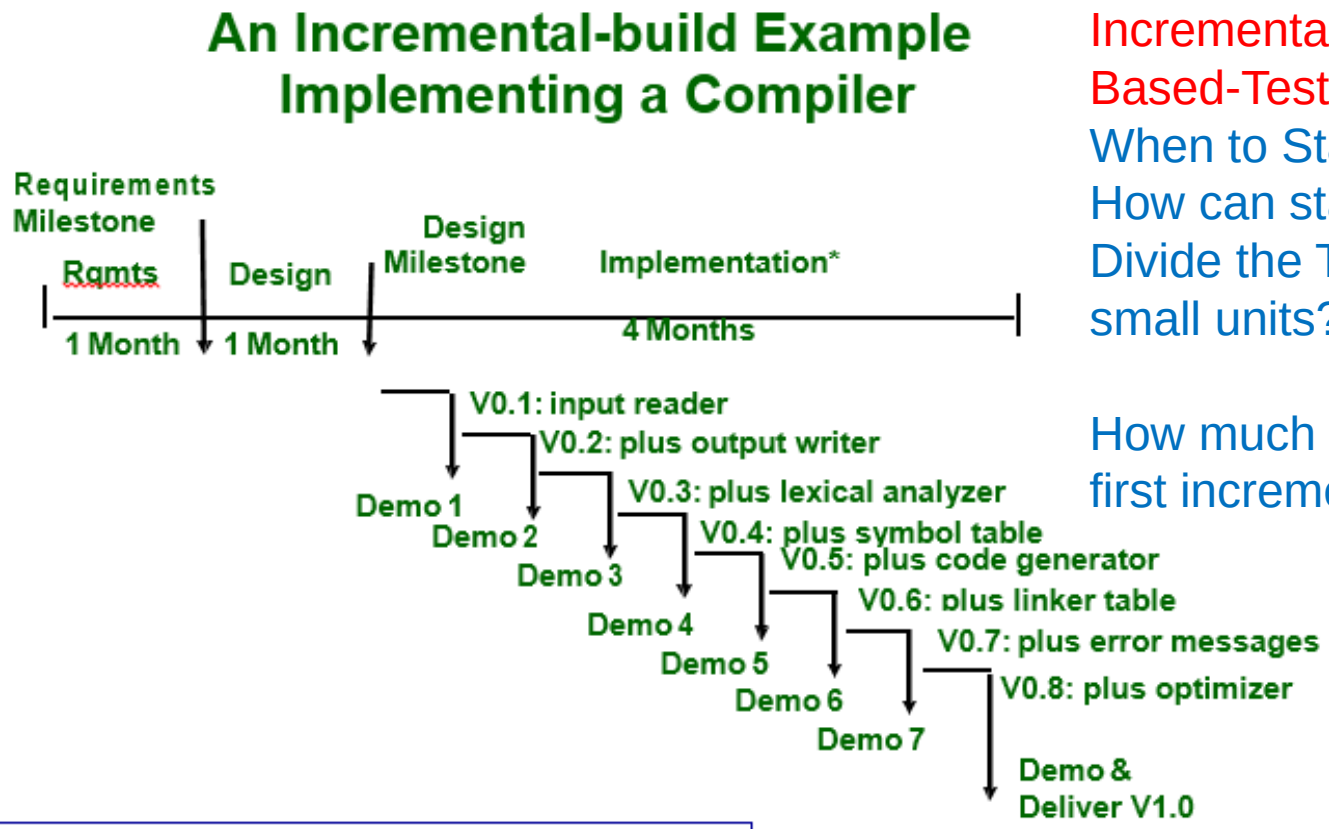
Classification: Types of testing

Waterfall Model Based Testing..?
When to Start Test?
What is size of Unit? No. of Use Case?



Waterfall model: starts with requirements analysis then design

Classification: Incremental Model



*implementation of each increment includes detailed design, coding, review, integration, testing, and demonstration

Incremental Model Based-Testing?

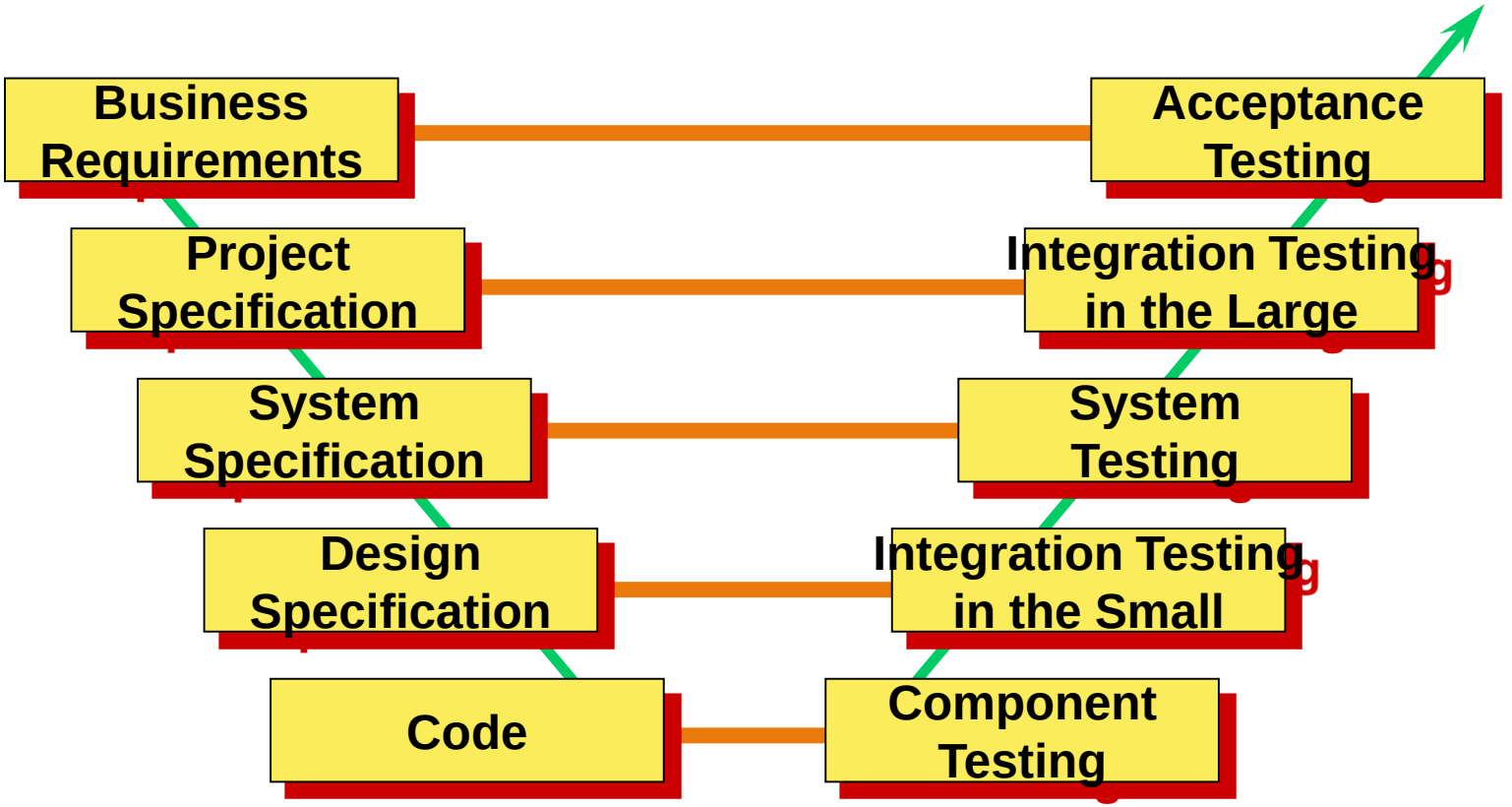
When to Start Test?

How can start early?

Divide the Test in small units?

How much use case in first increments?

Classification: V- Model Testing



Classification: Types of testing

C1: Source of test generation.

C2: Lifecycle phase in which testing takes place

C3: Goal of a specific testing activity

C4: Characteristics of the artifact under test

C5: Test Process Models

C1: Source of test generation

Artifact	Technique	Example
Requirements (informal)	Black-box	Ad-hoc testing Boundary value analysis Category partition Classification trees Cause-effect graphs Equivalence partitioning Partition testing Predicate testing Random testing
Code	White-box	Adequacy assessment Coverage testing Data-flow testing Domain testing Mutation testing Path testing Structural testing Test minimization using coverage
Requirements and code	Black-box and White-box	
Formal model: Graphical or mathematical specification	Model-based Specification	Statechart testing FSM testing Pairwise testing Syntax testing
Component interface	Interface testing	Interface mutation Pairwise testing

C2: Lifecycle phase in which testing takes place

- To Reduce Defects on Increments:
  - Test the unit, test it after integration,
    Test the System

Phase	Technique
Coding	Unit testing
Integration	Integration testing
System integration	System testing
Maintenance	Regression testing
Post system, pre-release	Beta-testing

C3: Goal of specific testing activity

- To Coverage/ Achieve Goal:
  - Test the Features & Characteristics

Goal	Technique	Example
Advertised features	Functional testing	
Security	Security testing	
Invalid inputs	Robustness testing	
Vulnerabilities	Vulnerability testing	
Errors in GUI	GUI testing	Capture/plaback
		Event sequence graphs
		Complete Interaction Sequence
Operational correctness	Operational testing	Transactional-flow
Reliability assessment	Reliability testing	
Resistance to penetration	Penetration testing	
System performance	Performance testing	Stress testing
Customer acceptability	Acceptance testing	
Business compatibility	Compatibility testing	Interface testing
		Installation testing
Peripherals compatibility	Configuration testing	

C4: Artifact under test

- To Test each aspects, to discourage accumulation of test:
  - Test Artifacts

Characteristics	Technique
Application component	Component testing
Client and server	Client-server testing
Compiler	Compiler testing
Design	Design testing
Code	Code testing
Database system	Transaction-flow testing
OO software	OO testing
Operating system	Operating system testing
Real-time software	Real-time testing
Requirements	Requirement testing
Software	Software testing
Web service	Web service testing

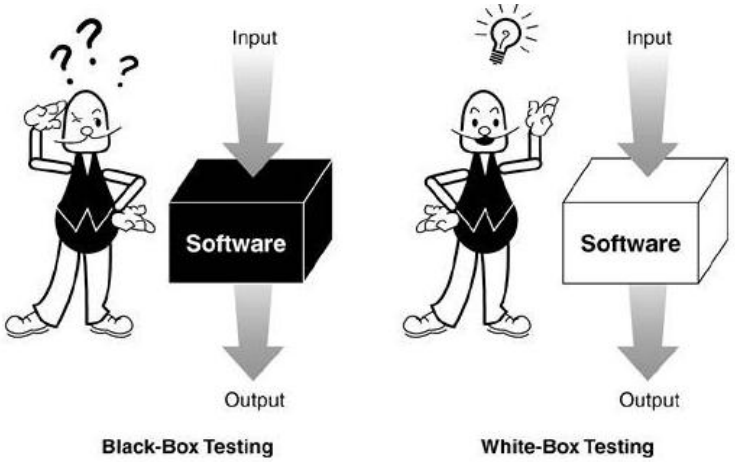
C5: Test Process Models

- To Test with Time Dimension:
  - Process Models

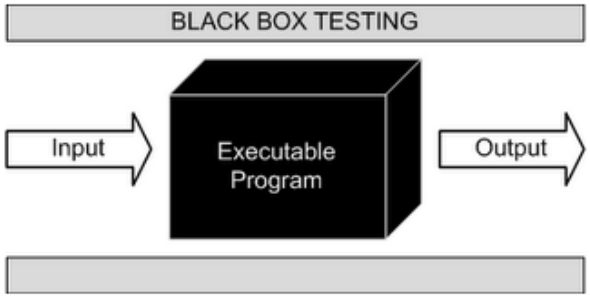
Process	Attributes
Waterfall	Testing Done at End
V-model	Specifies Testing Activities in each Phase
Incremental	Each Increment might be prototype, Demo or Release
Agile	Small Increments, Flexible

Classification: Types of testing

- One Classification
- Source of Test Generation
  - Black box testing
  - & White box testing.

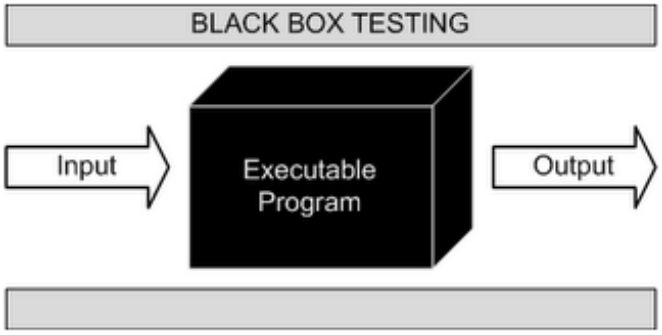


- Black box testing
  - also called functional testing
  - is *testing that ignores the internal mechanism of a system or component and focuses solely on the outputs generated in response to selected inputs and execution conditions*.
  - Observes Failure
    - When output is not observed as expected on given inputs
    - Report the failure
    - Reproduce the failure & note the steps



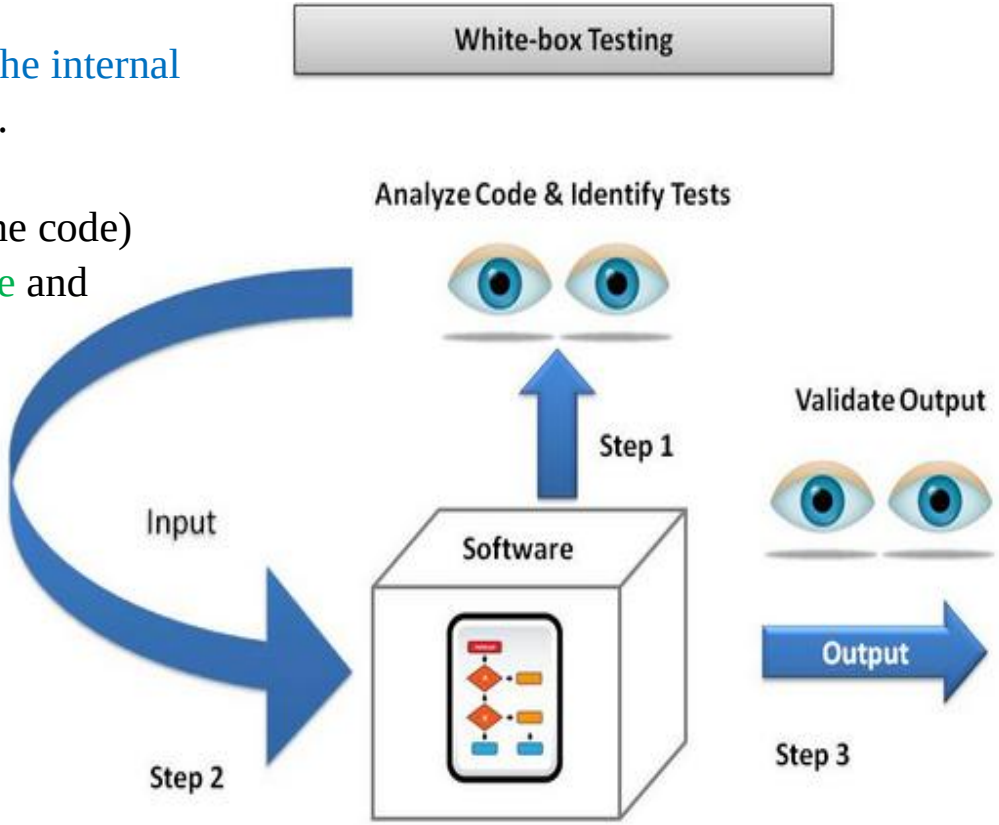
Classification: Types of testing

- Black box testing
  - No access to the source code itself.
    - The code is considered to be a “big black box” to the tester who can’t see inside the box.
    - The tester knows only that information can be input into to the black box, and the black box will send something back out.
    - Based on the requirements knowledge, the tester knows what to expect the black box to send out and tests to make sure the black box sends out what it’s supposed to send out.
    - Verification & Validation, black box testing is often used for validation (are we building the right software?)
    - Tester or (independent tester)



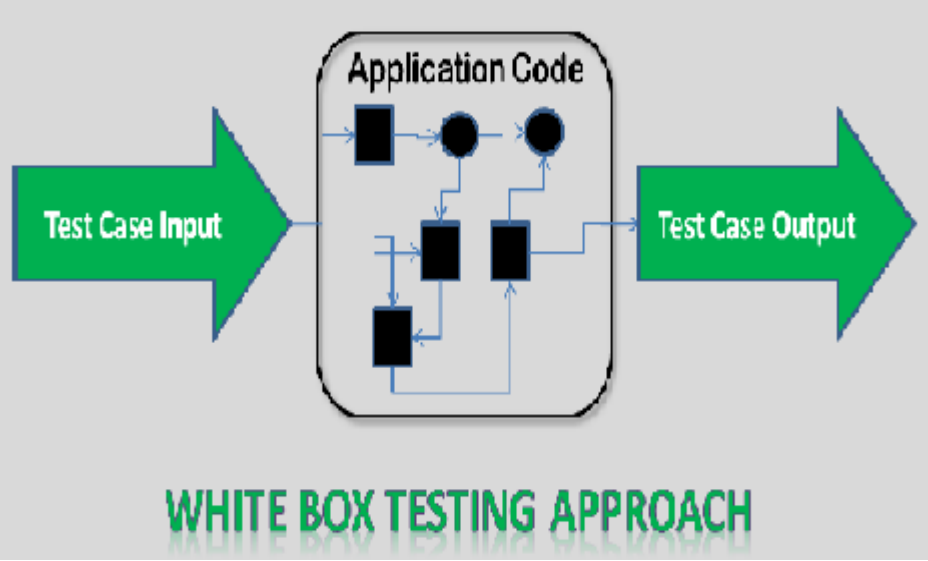
Classification: Types of testing

- White box testing
  - also called structural testing and glass box testing
  - is *testing that takes into account the internal mechanism of a system or component*.
  - *White box testing* focuses on the internal structure of the software code.
  - The white box tester (most often the developer of the code) knows what the code looks like and writes test cases by executing methods with certain parameters.



Classification: Types of testing

- White box testing
  - Verification & Validation, white box testing is often used for verification (are we building the software right?).
  - Based on knowledge of internal logic of an application's code
  - Based on coverage of code statements, paths, conditions
  - Tests are logic driven



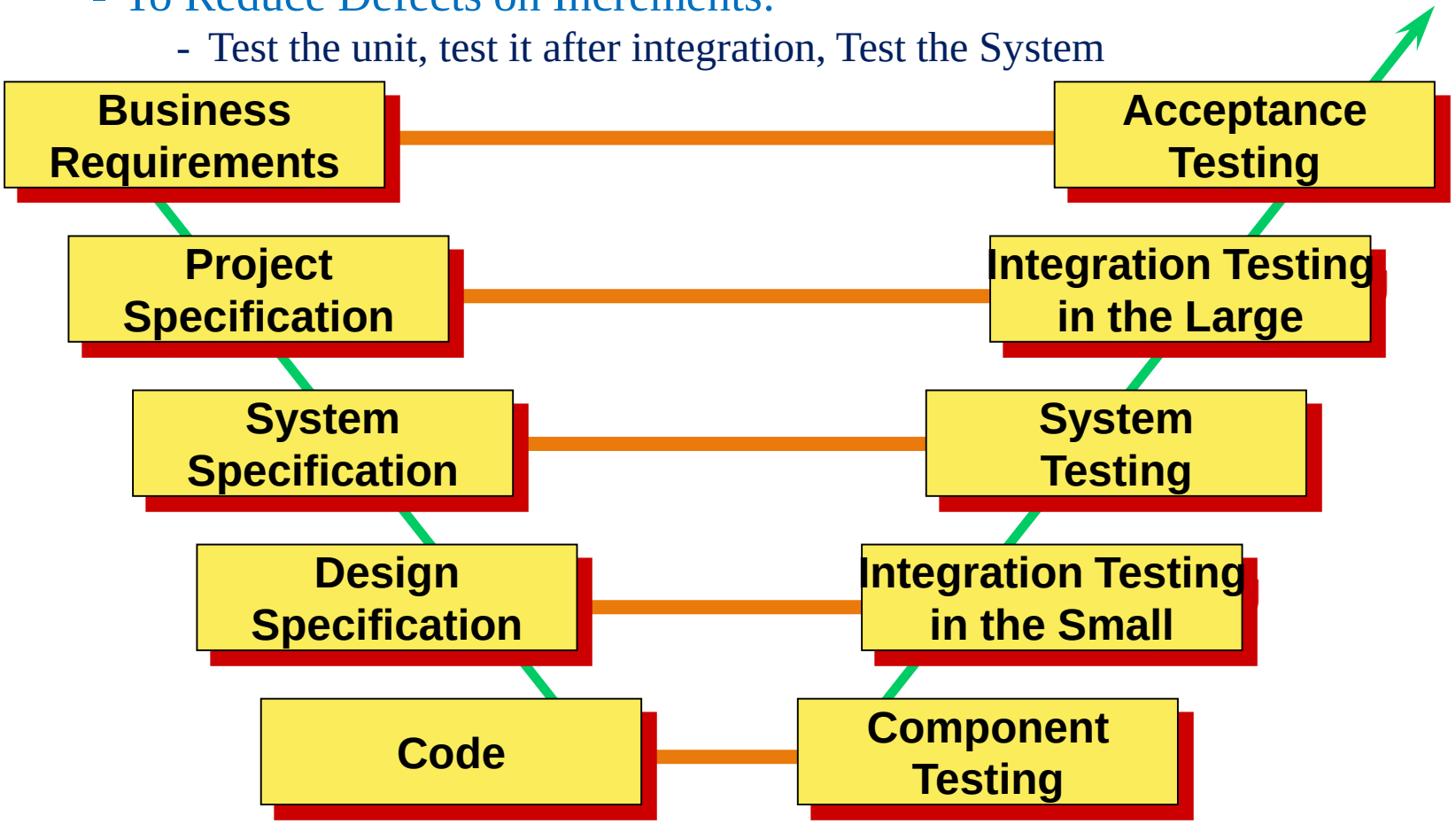
Classification: Types of testing

- "Black box"
  - Random Testing
  - Equivalence partitioning
  - Boundary value analysis
  - State transition testing
  - Cause-effect graphing
  - Decision Table
- "White box"
  - Statement testing
  - Branch / Decision testing
  - Data flow testing
  - Branch condition testing

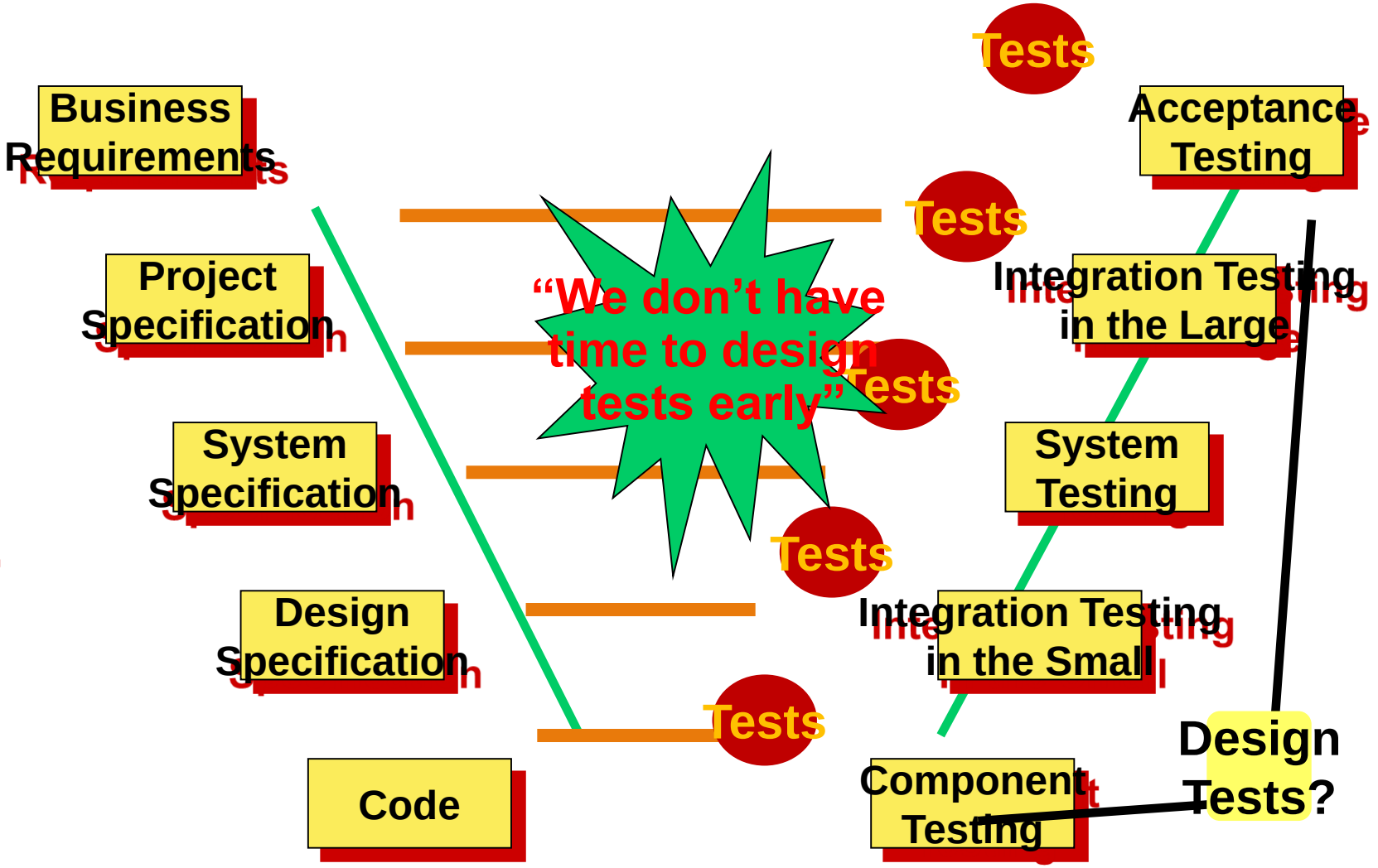
We study in more detail on 13th Week

Classification: Types of testing

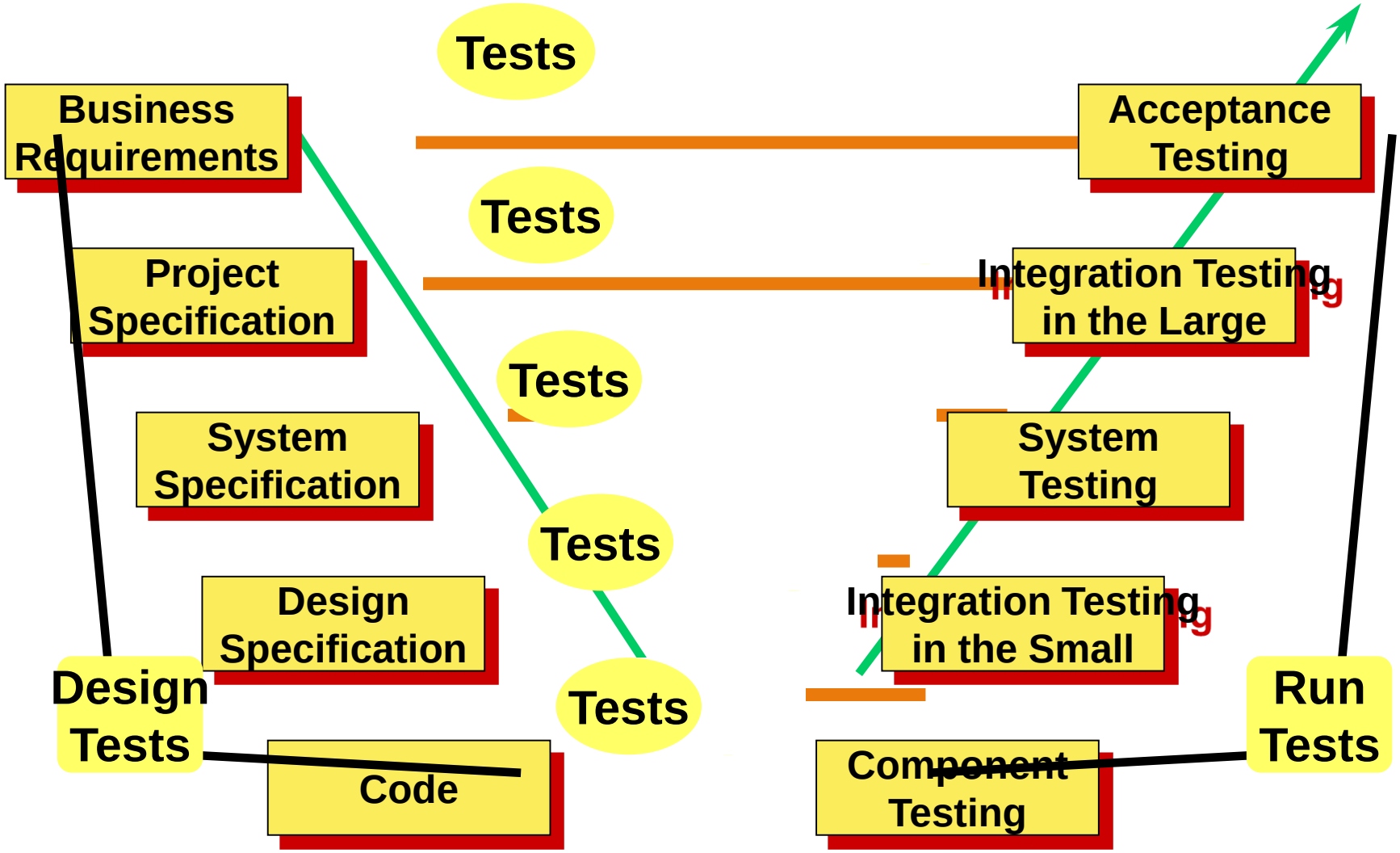
- To Reduce Defects on Increments:
  - Test the unit, test it after integration, Test the System



V-Model: late test design



V-Model: late test design



Classification: Types of testing

- **Testing Levels**
  - Unit / Component testing
  - Integration testing
  - System testing
  - Acceptance testing
  - Beta Testing
  - Alpha Testing
  - Regression Testing

Unit testing

- The most ‘micro’ scale of testing.
- Most thorough look at detail
  - error handling
  - interfaces
- Tests done on particular functions or code modules.
- Requires knowledge of the internal program design and code.
- Usually done by programmer (not by testers).
- also known as component, module, program testing

Unit testing

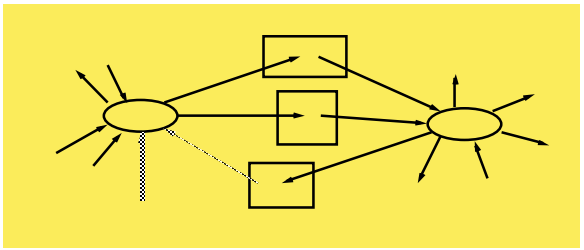
Objectives	• To test the function of a program or unit of code such as a program or module • To test internal logic • To verify internal design • To test path & conditions coverage • To test exception conditions & error handling
When	• After modules are coded
Input	• Internal Application Design • Master Test Plan • Unit Test Plan
Output	• Unit Test Report

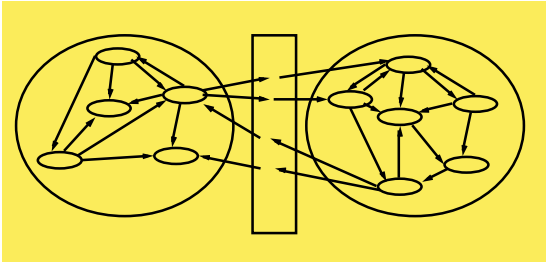
Unit testing

Who	•Developer
Methods	•White Box testing techniques •Test Coverage techniques
Tools	•Debug •Re-structure •Code Analyzers •Path/statement coverage tools
Education	•Testing Methodology •Effective use of tools

Integration testing

- more than one Unit / (tested) component
- communication between components / units
- what the set can perform that is not possible individually
- non-functional aspects if possible
- integration types: big-bang vs incremental (top-down, bottom-up, functional, strategy)
- done by designers, analysts, or independent testers



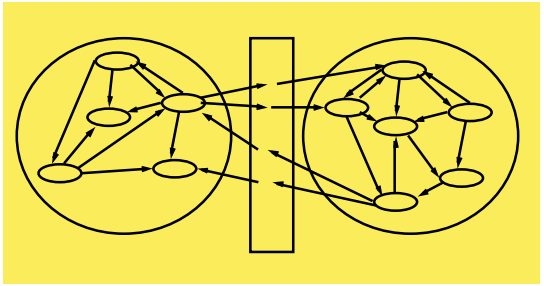


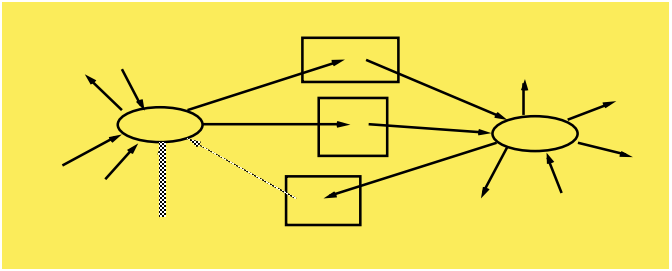
Incremental integration testing

- Continuous testing of an application as and when a new functionality is added.

- Application's functionality aspects are required to be independent enough to work separately before completion of development.

- Done by programmers or testers.





Incremental integration testing

- Testing of combined parts of an application to determine their functional correctness.

- ‘Parts’ can be
  - code modules
  - individual applications
  - client/server applications on a network.

Incremental integration testing

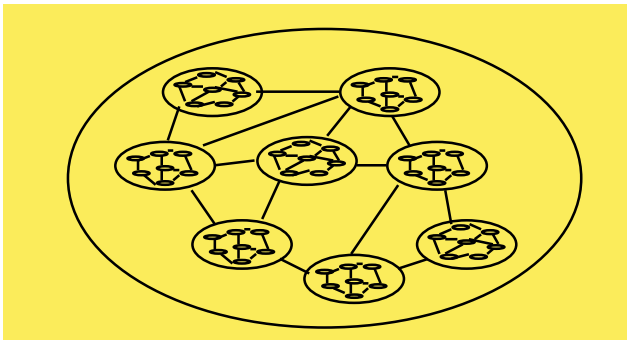
Objectives	• To technically verify proper interfacing between modules, and within sub-systems
When	• After modules are unit tested
Input	• Internal & External Application Design • Master Test Plan • Integration Test Plan
Output	• Integration Test report

Incremental integration testing

Who	•Developers
Methods	•White and Black Box techniques •Problem / Configuration Management
Tools	•Debug •Re-structure •Code Analyzers
Education	•Testing Methodology •Effective use of tools

System testing

- last integration step
- functional
  - functional requirements and requirements-based testing
  - business process-based testing
- non-functional
  - as important as functional requirements
  - often poorly specified
  - must be tested
- often done by independent test group



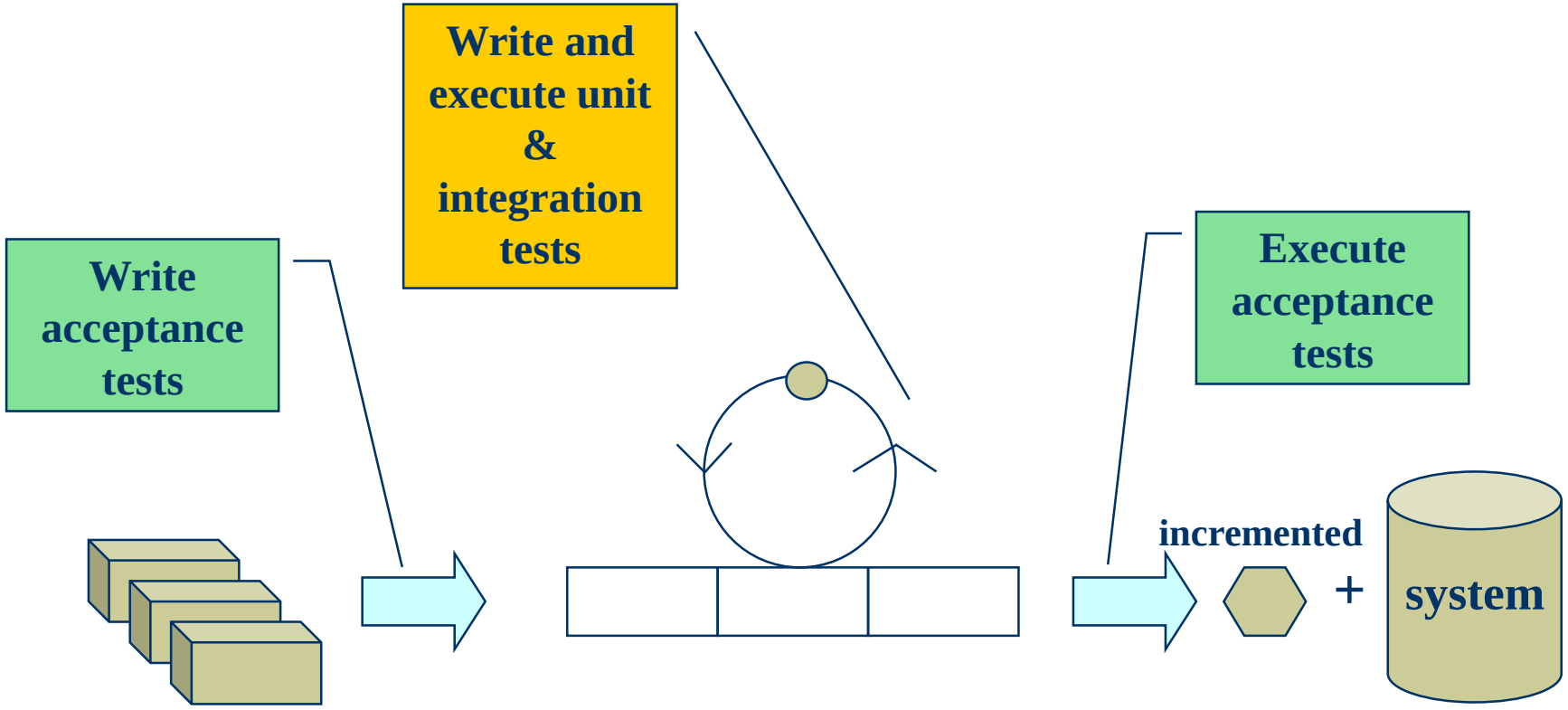
Systems Integration Testing

Objectives	• To test the co-existence of products and applications that are required to perform together in the production-like operational environment (hardware, software, network) • To ensure that the system functions together with all the components of its environment as a total system • To ensure that the system releases can be deployed in the current environment
When	• After system testing • Often performed outside of project life-cycle
Input	• Test Strategy • Master Test Plan • Systems Integration Test Plan
Output	• Systems Integration Test report

Systems Integration Testing

Who	•System Testers
Methods	•White and Black Box techniques •Problem / Configuration Management
Tools	•Recommended set of tools
Education	•Testing Methodology •Effective use of tools

Acceptance: Iterative Software development



Prioritized functionalities

"Written before"

"Executed after the development"

Acceptance Testing

- are specified by the customer and analyst to test that the overall system is functioning as required (*Did developers build the right system?*).
- Acceptance tests typically test the entire system, or some large chunk of it.

Objectives	• To verify that the system meets the user requirements
When	• After System Testing
Input	• Business Needs & Detailed Requirements • Master Test Plan • User Acceptance Test Plan
Output	• User Acceptance Test report
Who	Users / End Users

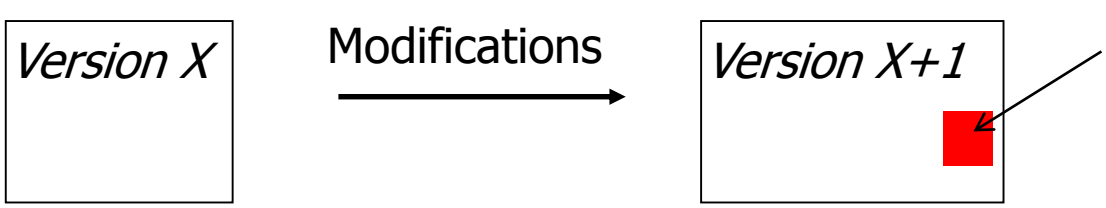
Acceptance Testing

Who	Users / End Users
Methods	•Black Box techniques •Problem / Configuration Management
Tools	Compare, keystroke capture & playback, regression testing
Education	•Testing Methodology •Effective use of tools •Product knowledge •Business Release Strategy

Regression testing

Selective re-testing of a system or component to verify that modifications have not caused unintended effects

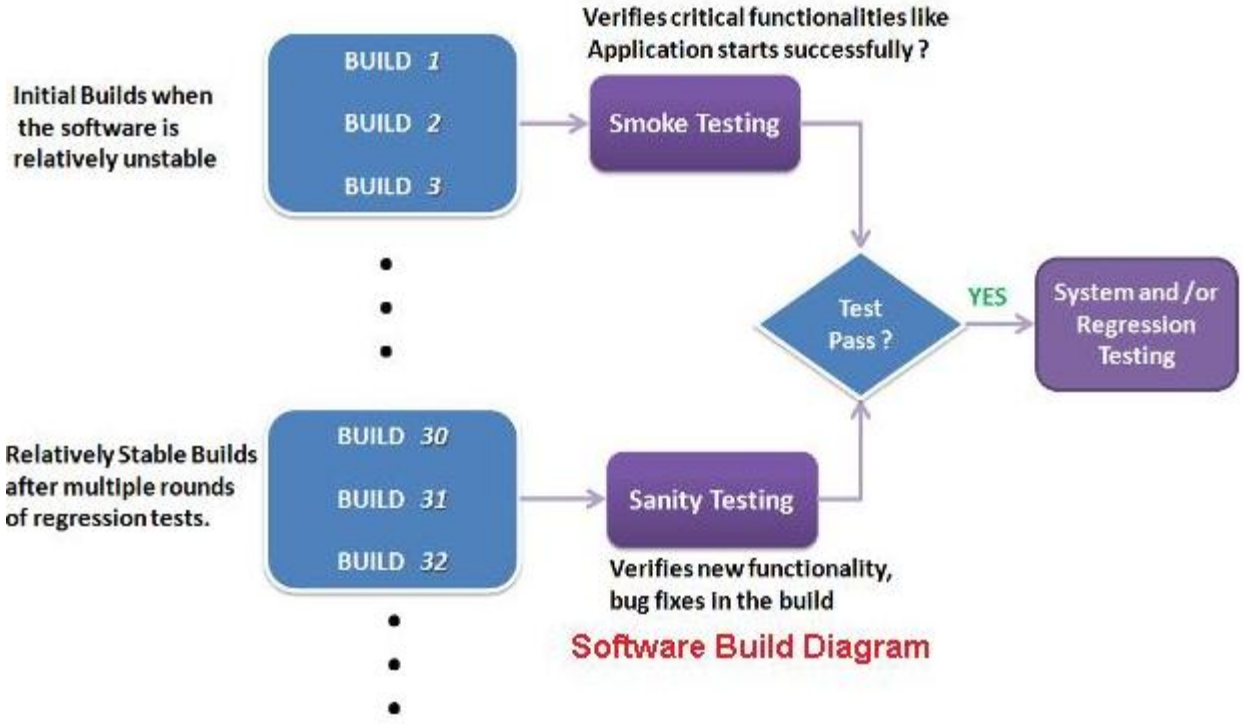
- Ripple effects
- Can be conducted at each of the test levels: unit, integration, system



- find affected testcases (red)
- change affected testcases (red)
- execute them
- define new test cases, if necessary

Testcases to modify

Smoke Testing…



Alpha Test….

- Smoke Test
  - group of test cases that establish that the system is stable and all major functionality is present and works under “normal” conditions
- Beta:
- Alpha
- Independent Testing before acceptance
- Parallel Testing after Delivery

- Life Cycle Testing…..

  Depends on Life Cycle in which software evolves….

  Depends on Software Process Models

4th Classification: Goal Based Testing

- Functional Based Testing
- Non-Functional Based Testing
- Requirement Based Testing
- Business Process Based Testing
  - what will be used most often?
  - what is critical to the business?

Goal: Non-functional system testing

- Different types of non-functional system tests:
  - usability - configuration / installation
  - security - reliability / qualities
  - documentation - back-up / recovery
  - storage - performance, load, stress
  - Volume - efficiency
  - Application specific features i.e. Web Application or OO Application or Embedded Application

Assignment: Select any Type of Application Domain and review Non-functional requirement that is specific to your project domain. Demonstrate the testing cycle of your project along with selected NFR Test. Or you can select from my assigned topics (find at selected pages at photocopy)

Goal-Based: Multi-User Tests

- Concurrency Tests
  - detect record locking problems
- Load Tests
  - the measurement of system behaviour under realistic multi-user load
- Stress Tests
  - testing conducted to evaluate a system or component at or beyond the limits of its specification or requirement
  - go beyond limits for the system - know what will happen

Goal-Based: Multi-User Tests

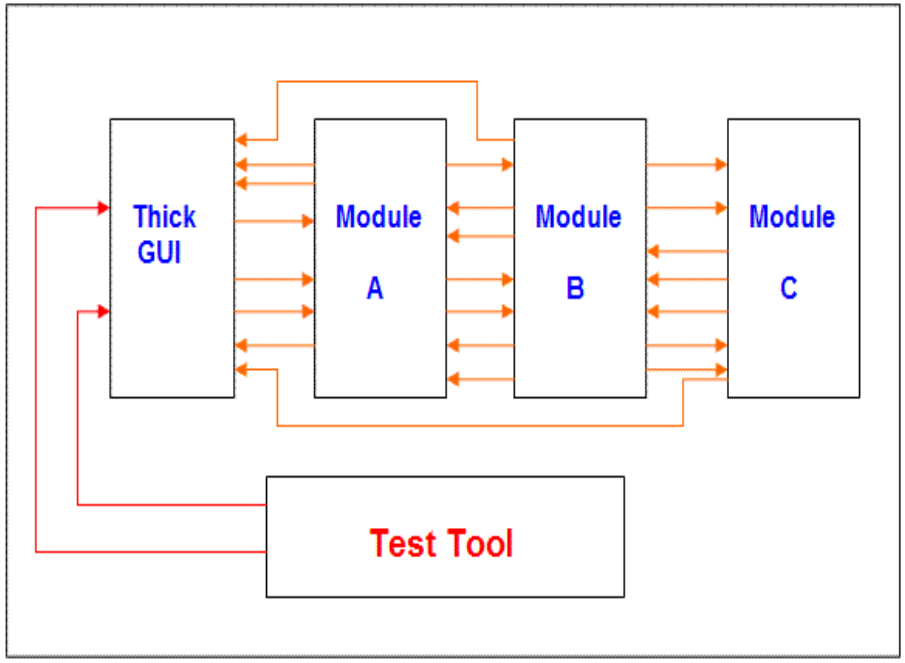
- Performance testing
  - testing conducted to evaluate the compliance of a system or component with specified performance requirements
- Robustness testing
  - Testing whereby test cases are chosen outside the domain to test robustness to unexpected, erroneous input
- Usability testing
  - testing conducted to evaluate the extent to which a user can learn to operate, prepare inputs for, and interpret outputs of a system or component

5th Classification: Artifacts Based Testing

Badly designed systems makes testing difficult

- We have a thick GUI that has program logic. The interfaces between the modules are not clearly defined.
- Testing of specific functions cannot be isolated.
- Testing has to be done through the GUI

"Badly designed system"



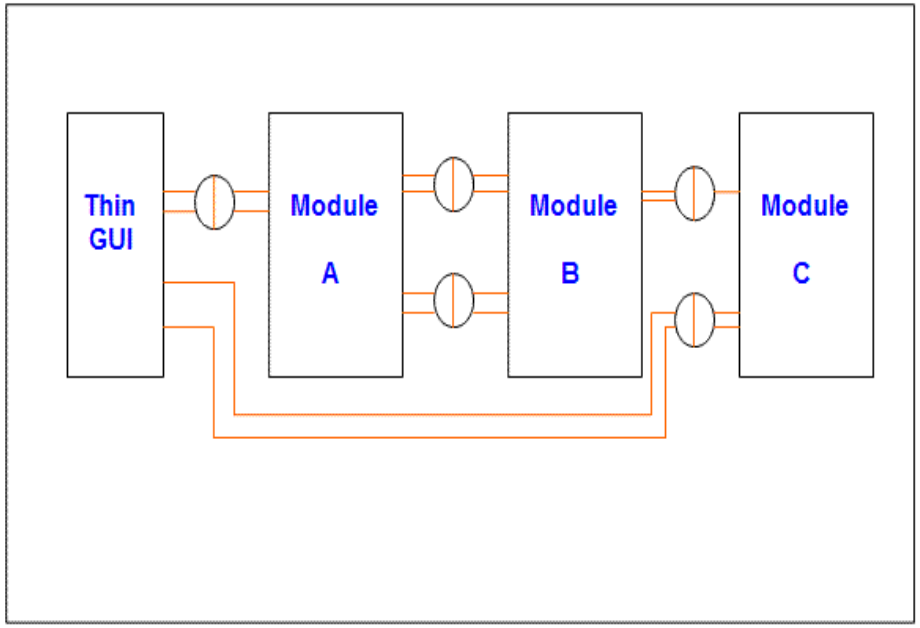
GUI-test drivers

5th Classification: Artifacts Based Testing

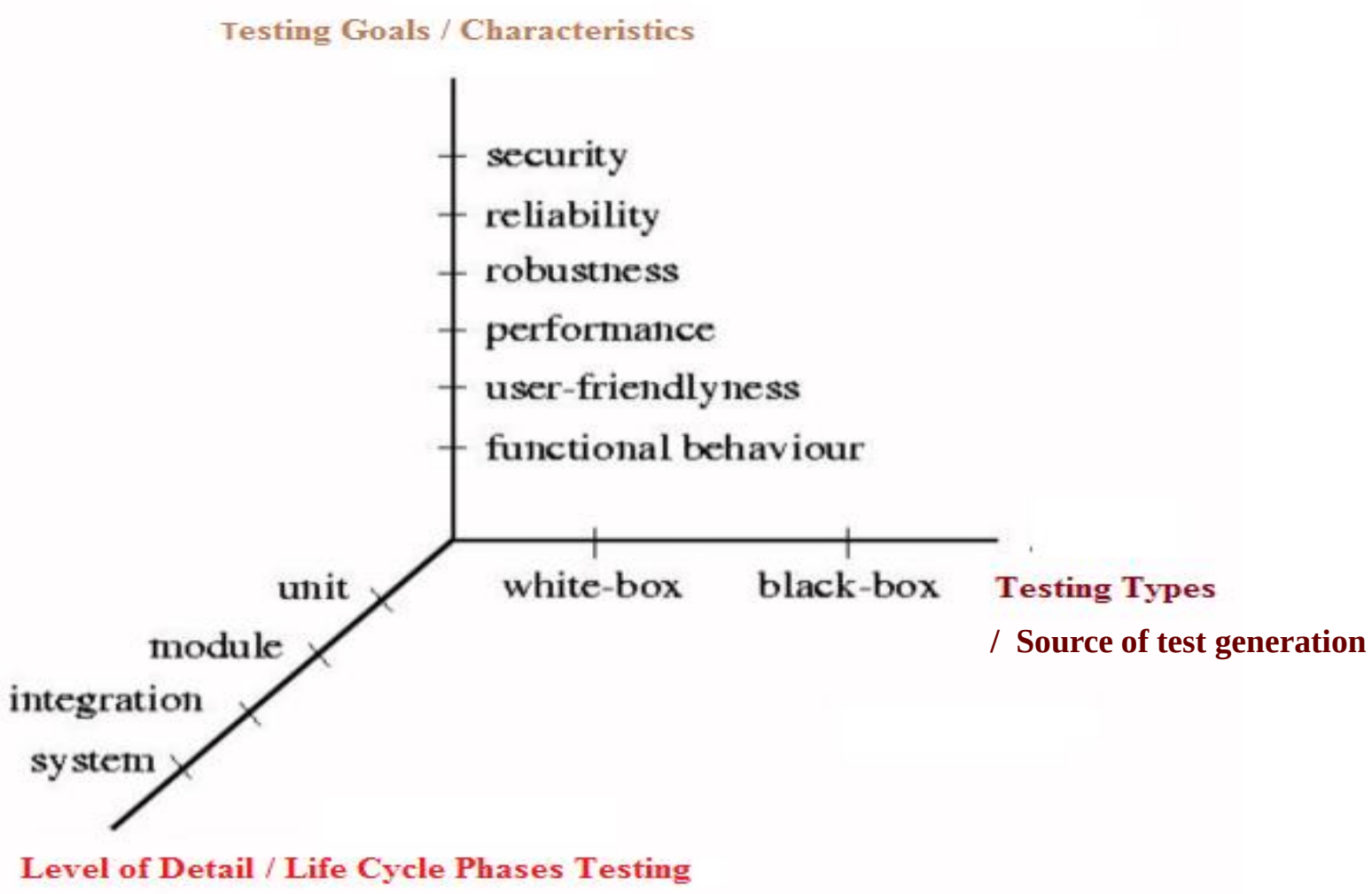
Well architected applications makes testing simple **Design for testability**

- The GUI does not contain any program logic other than dealing with presentation.
- The interfaces between the modules are well defined.

"*Well architected application*"



Classification: Types of testing



Classification: Types of testing

One possible classification is based on the following four classifiers:

C1: Source of test generation.

C2: Lifecycle phase in which testing takes place

C3: Goal of a specific testing activity

C4: Characteristics of the artifact under test

C5: Test Process Models