SOFTWARE TESTING AND QUALITY ASSURANCE

ICT304

Course Lecturer: Dr. KIMBI Xaveria

Course Objective and Development Attitude

Objectives:

- At the end of this course, the students should know how to use software testing concepts, methodologies and techniques to perform software verification and validation in real life projects.
- Student will know how write a Test Script and perform JUnit, in Java (NetBeans)

- **Development attitude**:
- A collaborative research-oriented approach in groups of 5

Evaluation Method

A three-phased evaluation method will be adopted:

- TD evaluation at the end of four lessons (Quiz/Practicals) = CC

- TP evaluation of two use cases in a chosen software product. (Unit test realization with any familiar tool of your choice: JUnit, CppUnit, PyUnit, Nunit, etc). (TP in a group on two topics: Library Management System and Student Results Processing System)

- General Examination:

- CC = 20% TP = 30% EE = 50%

Course Content

Module 1

Introduction to agile software development methodologies, using a specific real live case study:

- The rudiments of Software Testing

Module 2

- Static Testing Methodology and Techniques
- A real life case study

Module 3

- Dynamic Testing Methodology and Techniques
- A real life case study

GENERAL INTRODUCTION TO SOFTWARE TESTING

- What is Software Testing?
- The impact of Software Development in all Sectors
  (our day-to-day activities)
- A conceptual Map to Software Testing
- Relationship between Program Fault, Program Errors and Program Failure
- Why do we Need Software Testing?
- The Goals of Software Testing
- Testing and Debugging
- Limitation of software testing
- Software Verification and Software Validation
- The goals of Software Verification and Validation
- Software Verification and Validation Techniques
- Test Case Creation: What is a Test Case? Sources of Information for Test Case Selection
- SDLC Vs STLC

Test Oracle

What are you Verifying?

What are you validating?

- Fraudulent transactions in Banks, Finance,
- Malfunctioning of software controlled gadgets e.f X-ray Machines, aircraft, electricity bills with fake meters reading,

What is Software Testing?

Software development is risky venture.

- How can bugs be minimized in the SDLC ?
- Software Testing is executing a program with the intent (purpose) of finding errors. – To ensures the system is works according to the design specification
- Software Testing involves examining the behaviour of a system in order to discover potential faults

SRS is the building block in software development: **The customer knows what to expect, the developer understands what to code and the tester gets what to test**

- Since software applications are critical in modern life, it is very important to provide adequate quality and minimize the probability of faults in software products.

- **The impact of Software Development in all Sectors (our day-to-day activities)**

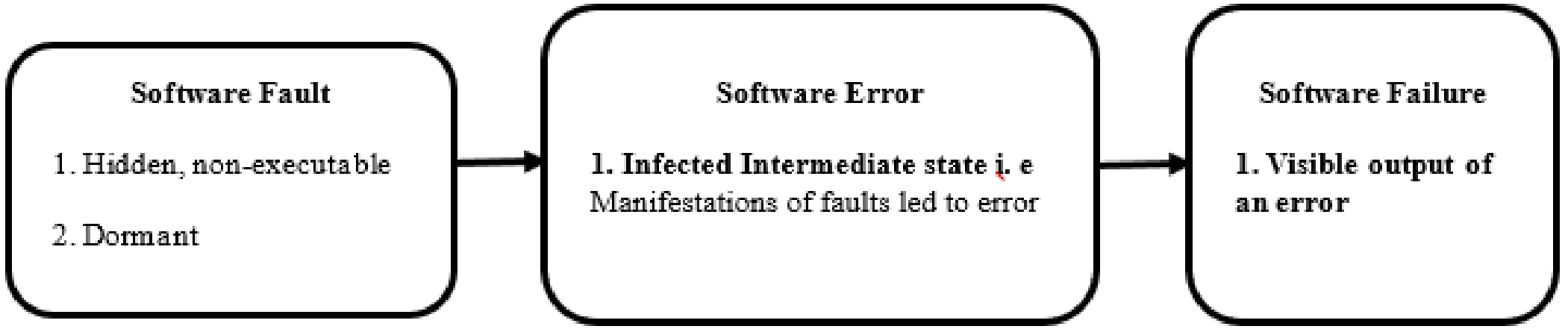
- **A conceptual Map to Software Testing**

Program Errors/ Software Errors/Software Bugs

- An error in a program, whether due to design errors or coding errors, are known as **bugs.**
- **Error:** An error is said to occur whenever the behaviour of a system does not conform to that prescribed in the requirement specification document.
- Program Errors are classified as:
- Syntax Errors
- Logical Errors: e.g (Requirement errors, design errors, implementation errors)
- Run-time Errors
- Sematic Errors
- Etc
- ***From the above explanation, what are the differences between Software Fault, Software Error and Software Failure?***

- **Question(1) List three categories of Program Error with five(5) examples each**
- **Question(2) State three causes of Program Error**

Relationship between
Program Fault, Program Errors and Program Failure



Software Defects Relationships

Goals of Software Testing

So really, why do we test? Software-Controlled devices

- The general aim of software testing is to affirm the **quality of software systems** by systematically exercising the software in carefully controlled circumstances to:
- **Reveal faults** (finding errors: Syntax Errors and Logical errors)
- **Establish confidence in the software -Clarification of user's requirement specification** (from requirement analysis phase – functional and non-functional requirements)

Can these goals be achieved without challenges?

Challenges/Limitation of software testing

- Software testing can only shows the presence of errors not their absence.
- ***Example****: Some systems are required to demonstrate very high reliability (e.g., an aircraft should only fail completely once in 10 exponential 11 hours of flying). • So aircraft components have to be pretty reliable (but think about how many single points of failure a car has).*
- *Exhaustive testing is impossible*
- Incomplete, informal and changing requirement specifications or incomplete users' requirements.
- Testing effectiveness and software quality is hard to measure
- Despite these dramatic improvements in software development (Web design software, mobile application, Artificial Intelligent software, operating systems, etc), we are yet to develop a software technology that does it all, and the likelihood of one arising in the future is slime.

Quick Exercise

(1) List an example of social Networking software that is bug free

(2) Identify the functional / non functional requirements (if any) for a banking system and hence translate the functional requirements into non-functional requirements.

a) Verification of account balance.

b) Withdrawing money from bank.

c) Completion of transactions in less than one second.

d) Extending the system by providing more tellers for customers.

- Is there any difference between Software Testing and SoftwareDebugging

Testing and Debugging

What is the difference between testing and debugging?

- ***Testing:*** It involves the identification of bug/error/defect in the software without correcting it.

- ***Debugging:*** It involves the Identifying, Isolating and Fixing (I^2F) the problems/bug.

Software Verification and Software Validation

- **Verification and validation (V&V)** is the process of checking that a software system meets specifications and that it fulfils its intended purpose. It may also be referred to as **software Quality Control**
- **Verification:** The process of determining whether the products of a given phase of the software development process fulfil the requirements established during the previous phase.

- Validation: The process of evaluating software at the end of each phase development to ensure compliance with intended usage
- Therefore, at **each phase** of the software development life cycle, both verification and validation **MUST** be performed.

Verification

- It answers the questions like:
- **Are we building the product right?** i.e The software should conform to its specification.
- Am I accessing the data right (in the right place; **in the right way**).
- It is a Low level activity? (Conceptual Design)
- Performed during development on key artifacts, like walkthroughs, reviews and inspections, mentor feedback, training, checklists and standards.
- ? Demonstration of consistency, completeness, and correctness of the software at each stage and between each stage of the development life cycle.

Validation

It answers the question like:

- **Am I building the right product?**
- Am I accessing the right data (in terms of the data required to satisfy the requirement).
- It is a High level activity.
- Performed after a work product is produced against established criteria ensuring that the product integrates correctly into the environment.
- Determination of correctness of the **final software product** by a development project with respect to the user needs and requirements.

Diagrammatical representation of V& V

Question (1): List two Software Development Methodologies with two examples in each case and hence with the aid of a diagram define Software Verification and Software Validation and state explicitly at which phase in the Software Development Life Cycle is each apply.
Question (2): State the goal of Software Verification and Software Validation

The goals of Software Verification and Validation

- The goal of **software verification** is to find as many hidden defects in the software before delivery.
- The goal of **software validation** is to gain confidence in the software by showing that it meets its specifications. i.e To demonstrate to the developer and the client (customer) that the software meets its requirements

Verification and Validation Techniques

There are four main Verification and Validation Techniques.

They include:

- Informal Analysis/Methods/–- human reasoning and subjectivity without stringent mathematical formalism. Inspection, etc
- Static Analysis/Methods – SRSD, Agorithm analysis
- Dynamic Analysis/Methods – Unit Testing, Integration Testing, Mutation Analysis, Symbolic Execution,
- Formal Analysis/Methods - Formal mathematical proofs using Z-notation, B-Notation,

- **Informal V&V techniques:** Informal V&V techniques are among the most commonly used. They are called informal because their tools and approaches rely heavily on human reasoning and subjectivity without stringent mathematical formalism.

- **Static Analysis**: As the term “static” suggests, it is based on the examination of a number of documents, namely requirements documents, software models, design documents, and source code. Static techniques do not require machine execution of the model. Traditional static analysis includes , inspection, walk-through, algorithm analysis, and proof of correctness.
- **Dynamic analysis** of a software system involves actual program execution in order to expose possible program failures.
- **Formal V&V techniques:** Formal V&V techniques (or formal methods) are based on formal mathematical proofs or correctness

How to Write Test Cases

- This is one major responsibility of a **Test Analysis**
- A Test Analyst writes high quality test cases in order to improve the quality of the software

Questions:

(1) What is the most important activity of a Test Analyst?

(2) Can a Test Analyst write effective Test Cases without understanding the application domain?

(3) Why do Test Analyst write Test Cases

Test Case Creation/How to Write Test Cases

- **The basic objective of writing test cases is to validate the test coverage of an application**
- **What is a Test Case?**
  Test Case (TC) is simply an input data to a software product to validate a particular test objective/target. IT describes the input and the expected outcome, in order to determine if a feature of an application works correctly.

  Test Suit: A collection of Test Cases
-

- **Sources of Information for Test Case Selection**

  - Test case design from Requirements Specification Document

  - Test case design from design document
- **Criteria for Selecting Test Cases**
- There are three criteria for the selection of test cases:
- Specifying a criterion for evaluating a set of input values.
- Generating a set of input values that satisfy a given criterion.

Steps to write Test cases

- (i) Generation of scenario diagrams/(ii) Creation of use case scenario matrix
- (iii) Identification of variables in a use case
- (iv) Identification of different input states of available variables
- (v) Design of test case matrix
- (vi) Assigning actual values to variables

Test Case Format (Template) 1

Test Case ID	Test Case Description	Test Input (Test Case Value)	Expected Results	Test Case Status

Table 1: Test Case Format Example

Test ID	Test Case Description	Test Input (Test Case Value)	Expected Result

- Testing must be defensive (Defensive Testing) i.e Testing which includes tests under both normal and abnormal conditions. (valid and invalid inputs)
- For any application, you need cover: Performance Testing, Security and Regression Testing
- **Therefore each test case must be linked to a specific requirement**

Requirement Traceability Matrix (RTM)

- **Requirement Traceability Matrix (RTM) is a document that maps and traces user requirement with test cases. It captures all requirements proposed by the client and requirement traceability in a single document, delivered at the conclusion of the Software development life cycle.**
- **The main purpose of Requirement Traceability Matrix is to validate that all requirements are checked via test cases such that no functionality is unchecked during Software testing.**
- **RTM is use to track requirements and their corresponding test cases.**

Types of Traceability Test Matrix

- In Software Engineering, traceability matrix can be divided into three major component as mentioned below:

- **Forward traceability:** This matrix is used to check whether the project progresses in the desired direction and for the right product. It makes sure that each requirement is applied to the product and that each requirement is tested thoroughly. **It maps requirements to test cases.**
- **Backward or reverse traceability:** It is used to ensure whether the current product remains on the right track. The purpose behind this type of traceability is to verify that we are **not expanding the scope of the project** by adding code, design elements, test or other work that is not specified in the requirements. **It maps test cases to requirements.**
- **Bi-directional traceability (Forward+Backward):** This traceability matrix ensures that all **requirements are covered by test cases.** It analyzes the impact of a change in requirements affected by the Defect in a work product and vice versa.

- **Advantage of Requirement Traceability Matrix**
- It confirms 100% test coverage
- It highlights any requirements missing or document inconsistencies
- It shows the overall defects or execution status with a focus on business requirements
- It helps in analyzing or estimating the impact on the QA team's work with respect to revisiting or re-working on the test cases

Software Testing Life Cycle Process

- **Software Testing Life Cycle (STLC) process** is a sequence of steps use to test software products.
- STLC Phases
- There are following six major phases in every Software Testing Life Cycle:

- Requirement Analysis
- Test Planning
- Test Case Development
- Test Environment setup
- Test Execution
- Test Results – **Test Oracle**
- **What is Entry and Exit Criteria in STLC?**
- **Entry Criteria**: Entry Criteria gives the prerequisite items that must be completed before testing can begin.
- **Exit Criteria:** Exit Criteria defines the items that must be completed before testing can be concluded
-

- **Requirement Phase Testing**
- Requirement Phase Testing also known as Requirement Analysis in which test team studies the requirements from a **testing point of view** to identify testable requirements and the QA team may interact with various stakeholders to understand requirements in detail. Requirements could be either functional or non-functional. Automation feasibility for the testing project is also done in this stage.

- **Activities in Requirement Phase Testing**

- Identify types of tests to be performed.
- Gather details about testing priorities and focus.
- Prepare Requirement Traceability Matrix (RTM).
- Identify test environment details where testing is supposed to be carried out.
- Automation feasibility analysis (if required).

- **Deliverable of Requirement Phase Testing**
- - RTM

- **Test Case Development Phase**
- The Test Case Development Phase involves the creation, verification and rework of test cases & test scripts after the test plan is ready. Initially, the Test data is identified then created and reviewed and then reworked based on the preconditions. Then the QA team starts the development process of test cases for individual units.

- **Test Case Development Activities**

- Create test cases, automation scripts (if applicable)
- Review and baseline test cases and scripts
- Create test data (If Test Environment is available)

- **Deliverables of Test Case Development**

- Test cases/Test scripts

- **Test Environment Setup**
- Test Environment Setup decides the software and hardware conditions under which a work product is tested. It is one of the critical aspects of the testing process and can be done in parallel with the Test Case Development Phase. Test team may not be involved in this activity if the development team provides the test environment. The test team is required to do a readiness check (smoke testing) of the given environment.

- **Test Environment Setup Activities**

- Understand the required architecture, environment set-up and prepare hardware and software requirement list for the Test Environment.
- Setup test Environment and test data
- Perform smoke test on the build

- **Deliverables of Test Environment Setup**
- Environment ready with test data set up
- Smoke Test Results.

- Test Execution Activities
-
- Execute tests as per plan
- Document test results, and log defects for failed cases
- Map defects to test cases in RTM
- Retest the Defect fixes
- Track the defects to closure
- Deliverables of Test Execution
-
- Completed RTM with the execution status
- Test cases updated with results
- Defect reports

- Test Planning in STLC
- Test Planning in STLC is a phase in which a Senior QA manager determines the test plan strategy along with efforts and cost estimates for the project. Moreover, the resources, test environment, test limitations and the testing schedule are also determined. The Test Plan gets prepared and finalized in the same phase.

- **Test Planning Activities**

- Preparation of test plan/strategy document for various types of testing
- Test tool selection
- Test effort estimation
- Resource planning and determining roles and responsibilities.
- Training requirement
- **Deliverables** of Test Planning

- Test plan /strategy document.
- Effort estimation document.

LECTURE FOUR:
SOFTWARE TESTING METHODOLOGIES

A) Structural Testing or White-Box Testing (Dynamic Testing)

B) Functional Testing or Black-Box Testing (Static Testing)

C) Grey Box Testing

Oracle Testing (Oracle Problem), Model Checking, Random Testing, etc

SOFTWARE TESTING METHODOLOGIES

Two main testing methodologies:

- Dynamic Testing: Testing by executing the program with real inputs.

(evaluates the code and the Internal Structure , developers' view point, knowledge of Programming Language used, Test Cases - Source Code)

- Static Testing: Testing without executing the program. (No Internal Structure, Users' view point. No knowledge of Programming Language used Test Cases - requirement specification)

Goals/Significance of White Box Testing: *focuses on the internal structure of the system under test.*

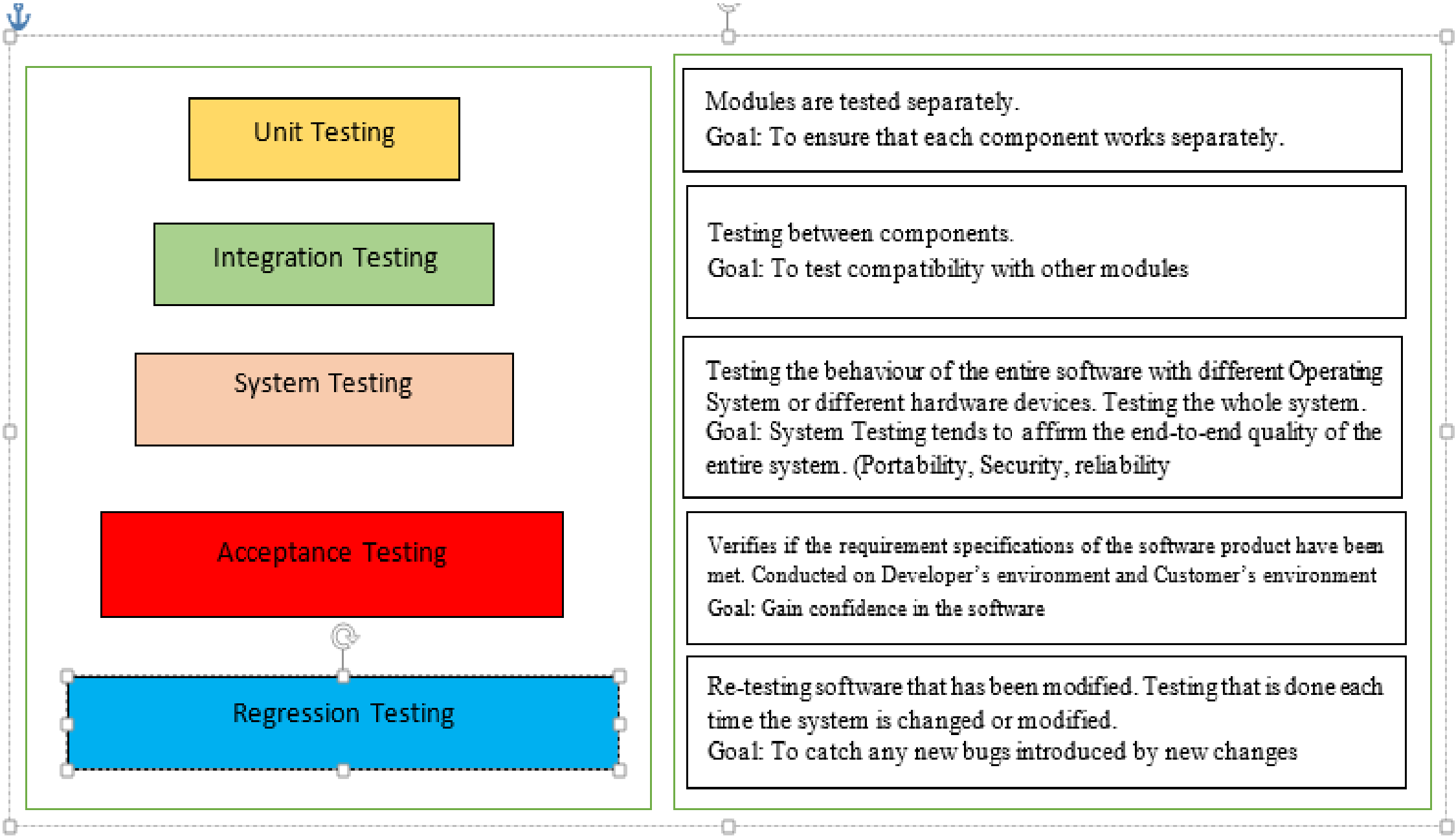
Goals/Significance of Black Box Testing: *Ensure that all requirement has been captured in the SRS document. And overall quality of the software*

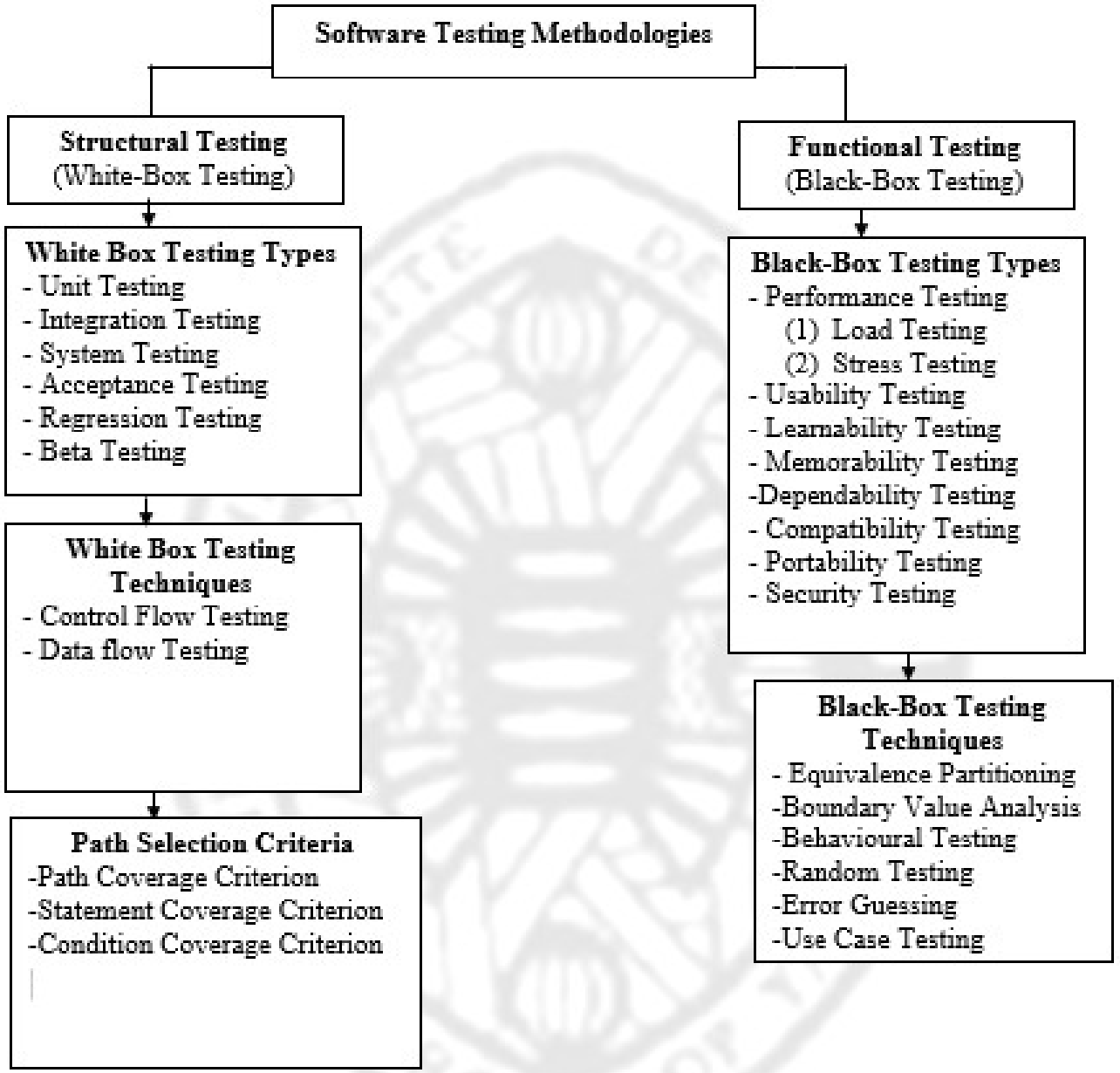
The limitation of white box testing

- Developers often test with the intent to prove that the code works rather than proving that it doesn't work.
- There are many kinds of errors that white box testing won't find:

  Performance problems

  Usability problems


Unit Testing
Integration Testing
System Testing
Acceptance Testing
Regression Testing
Modules are tested separately.
Goal: To ensure that each component works separately.
Testing between components.
Goal: To test compatibility with other modules
Testing the behaviour of the entire software with different Operating System or different hardware devices. Testing the whole system.
Goal: System Testing tends to affirm the end-to-end quality of the entire system. (Portability, Security, reliability
Verifies if the requirement specifications of the software product have been met. Conducted on Developer's environment and Customer's environment
Goal: Gain confidence in the software
Re-testing software that has been modified. Testing that is done each time the system is changed or modified.
Goal: To catch any new bugs introduced by new changes


Software Testing Methodologies
Structural Testing
(White-Box Testing)
Functional Testing
(Black-Box Testing)
White Box Testing Types
- Unit Testing
- Integration Testing
- System Testing
- Acceptance Testing
- Regression Testing
- Beta Testing
Black-Box Testing Types
- Performance Testing
(1) Load Testing
(2) Stress Testing
- Usability Testing
- Learnability Testing
- Memorability Testing
-Dependability Testing
- Compatibility Testing
- Portability Testing
- Security Testing
White Box Testing Techniques
- Control Flow Testing
- Data flow Testing
Black-Box Testing Techniques
- Equivalence Partitioning
-Boundary Value Analysis
-Behavioural Testing
-Random Testing
-Error Guessing
-Use Case Testing
Path Selection Criteria
-Path Coverage Criterion
-Statement Coverage Criterion
-Condition Coverage Criterion

LECTURE FIVE:
Principles of Software Testing

White Box Testing

- Testing based on program source code
- Extent to which (source) code is executed, i.e. Covered
- Different kinds of coverage: (i.e the extent to which the code has been covered)

  - Path Coverage Criterion
  - Statement Coverage Criterion
  - Condition/Branch/Decision Coverage Criterion
  - Predicate Coverage Criterion

CONTROL FLOW GRAPH

- Control Flow Graph is a graphical representation of a program Unit.
- A Control Flow Graph for a program consists of a node for each statement in the program and edges representing the flow of control between statements.
- Some of the limitations of control flow testing are:
-
- (1) Control flow testing cannot catch all initialization mistakes.
- (2) Requirement Specification mistake are not caught by control flow testing.
- (3) **Therefore,** It's unlikely to find missing paths and features if the program and the model on which the tests are based are done by the same person.

Data Flow Testing

```
If ( Condition 1 ) {
     x = a;
}
Else {
     x = b;
}

If ( Condition 2 ) {
     y = x + 1;
}
Else {
     y = x - 1;
}
```

What test cases do we need?

Definitions: 1) x = a; 2) x = b;
Uses: 1) y = x + 1; 2) y = x – 1;

1. (x = a, y = x + 1)
2. (x = b, y = x + 1)
3. (x = a, y = x – 1)
4. (x = b, y = x – 1)

EXERCISE (15Marks)

(1) Assuming you are to develop a software for the **Management of Student Results** in University of Yaounde I with the following grading scheme:

Grades	Range
80≤x≤100	A^+
75≤x<80	A
70≤x<75	B^+
65≤x<70	B
60≤x<65	B^-
55≤x<60	C^+
50≤x<55	C
45≤x<50	C^-
40≤x<45	D^+
35≤x<40	D
0≤x<35	E

(a) Write an algorithm for the above Software Project and hence state the characteristics of Algorithms?

(b) Define Test Cases and hence explain the criteria for selecting Test Cases

(c) Present the Test Case Document of the chosen Project and hence product the corresponding Sequence diagram, Class Diagram and Object diagram of the project. .

(d) With respect to the above project, explain the concept of Disruptive Testing

(e) Using C++ or VB.Net or any other programming of your choice: write down the function for computing the grades of N students and hence depict the Control Flow Graph (CFG) of the System.

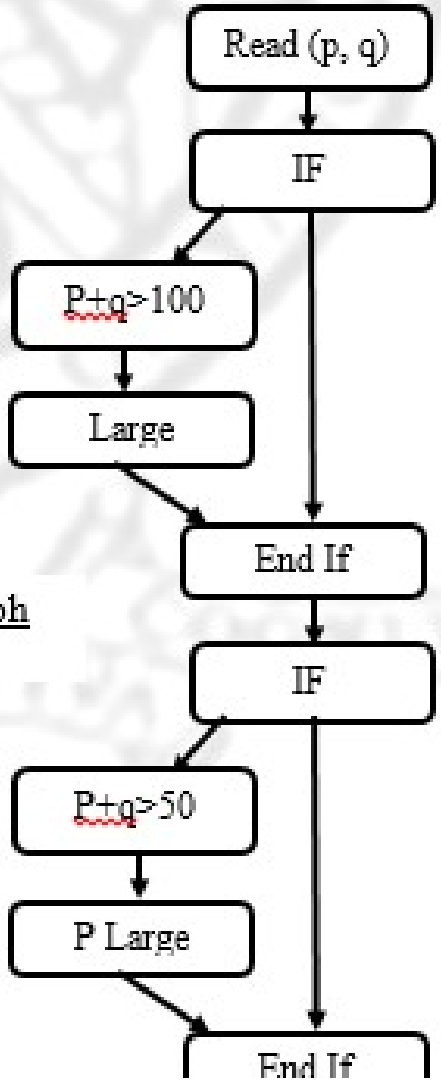
(e) Recommend three software testing approaches for the system giving your justification and the significant of each test.

Lecture FIVE:
CONTROL FLOW TESTING

- The adequacy of the test cases is measured with a metric called coverage (Coverage is a measure of the completeness of the set of test cases) (i.e the extent to which the code has been covered)

```
Read p
Read q
IF p+q > 100
THEN
Print "Large"
ENDIF
IF p > 50 THEN
Print "p Large"
ENDIF
```

Control Flow Graph

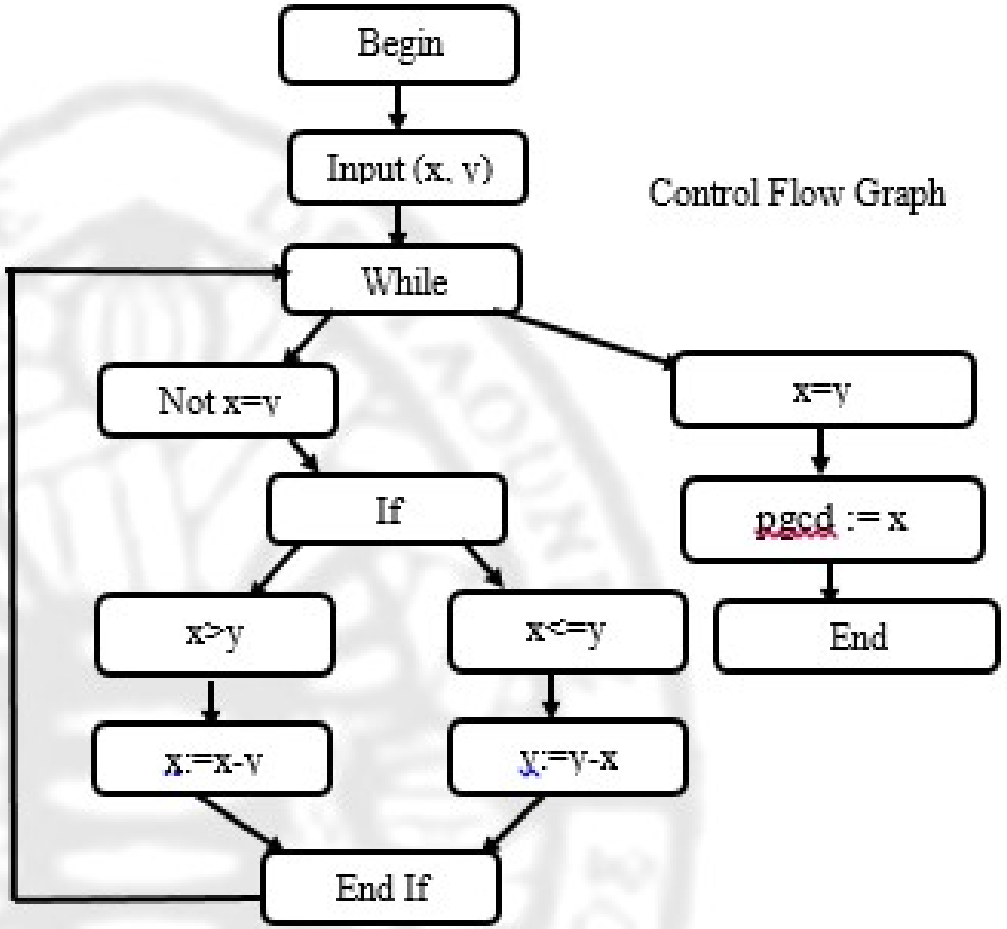


Using the pseudo Code below, produce the corresponding **Control Flow Graphs**

(i)

```
Begin
Read(x); read(y);
  While (not (x = y) ) loop
    If x > y Then
       x := x - y ;
    Else
       y := y - x ;
    End if ;
  End loop ;
  Print pgcd := x ;
End;
```

Control Flow Graph



LECTURE 4: Test Case Design
White-Box Testing Techniques

LECTURE 6: Test Case Design
Black-Box Testing Techniques

LECTURE 7: Security Testing