UNIT V

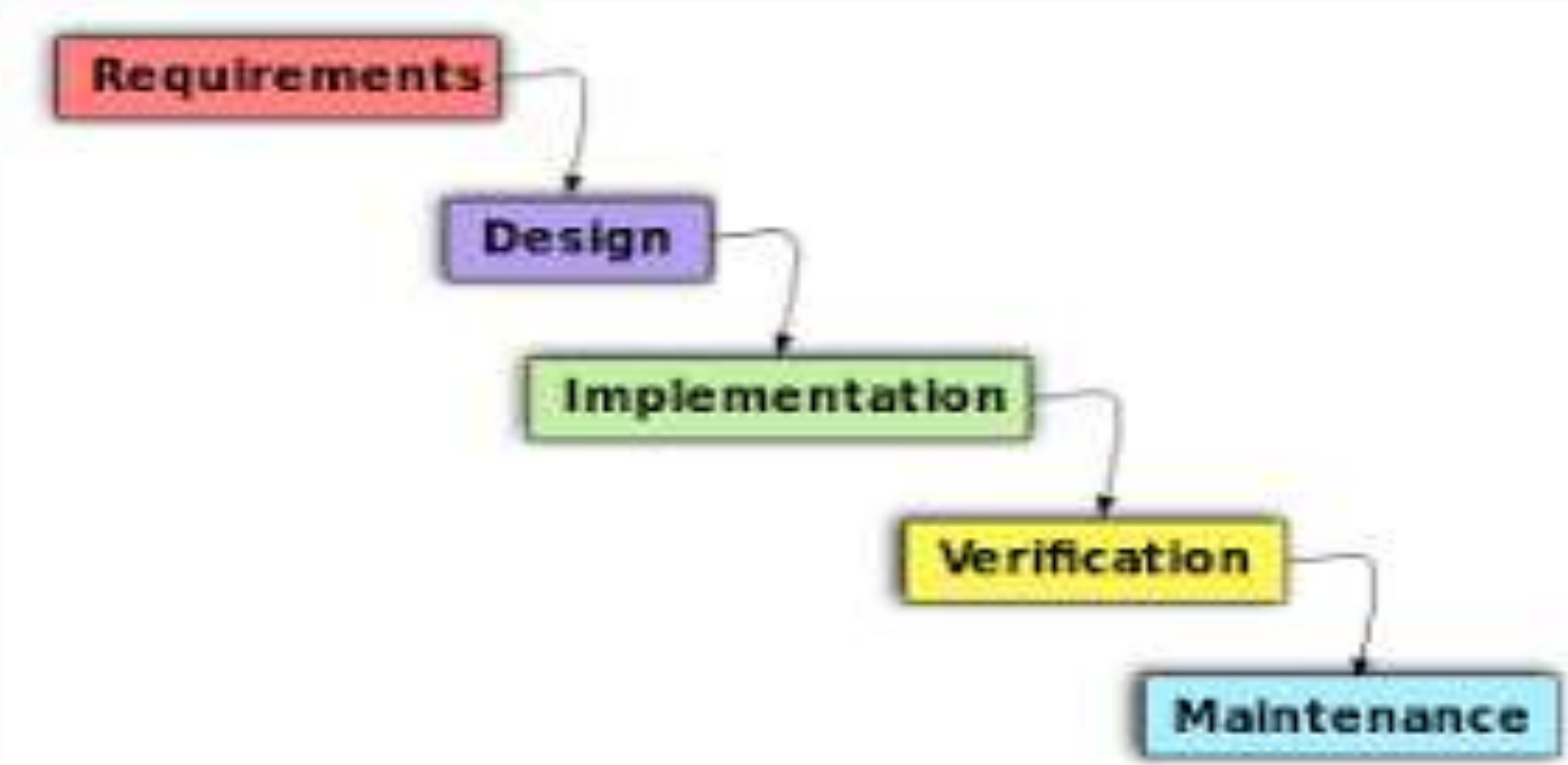
CODING AND TESTING

Objectives

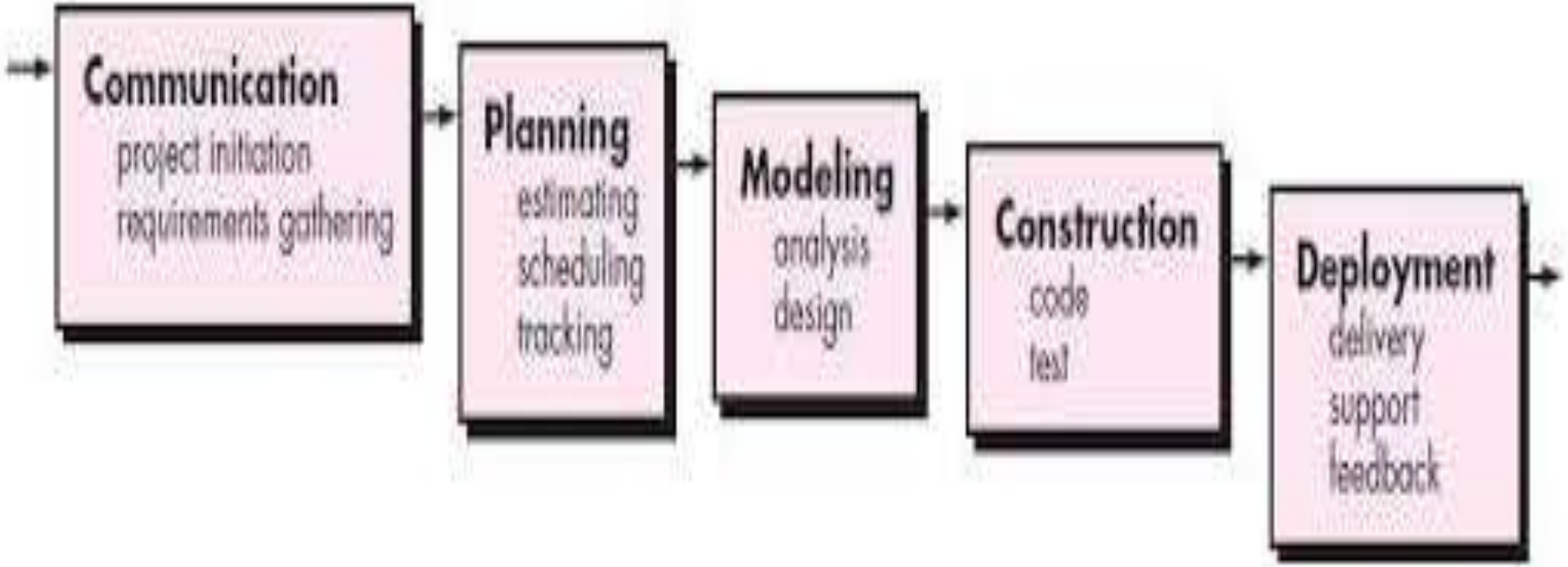
- To cover the strategies and tools associated with object oriented testing
  - Analysis and Design Testing
  - Unit/Class Tests
  - Integration Tests
  - System Tests
- Analysis and Design:
  - Testing begins by evaluating the OOA and OOD models
  - How do we test OOA models (requirements and use cases)?
  - How do we test OOD models (class and sequence diagrams)?
  - Structured walk-throughs, prototypes
  - Formal reviews of correctness, completeness and consistency

- **Programming:**
  - How does OO make testing different from procedural programming?
  - Concept of a unit‘ broadens due to class encapsulation
  - Integration focuses on classes and their execution across a thread or in the context of a use case scenario
  - Validation may still use conventional black box methods

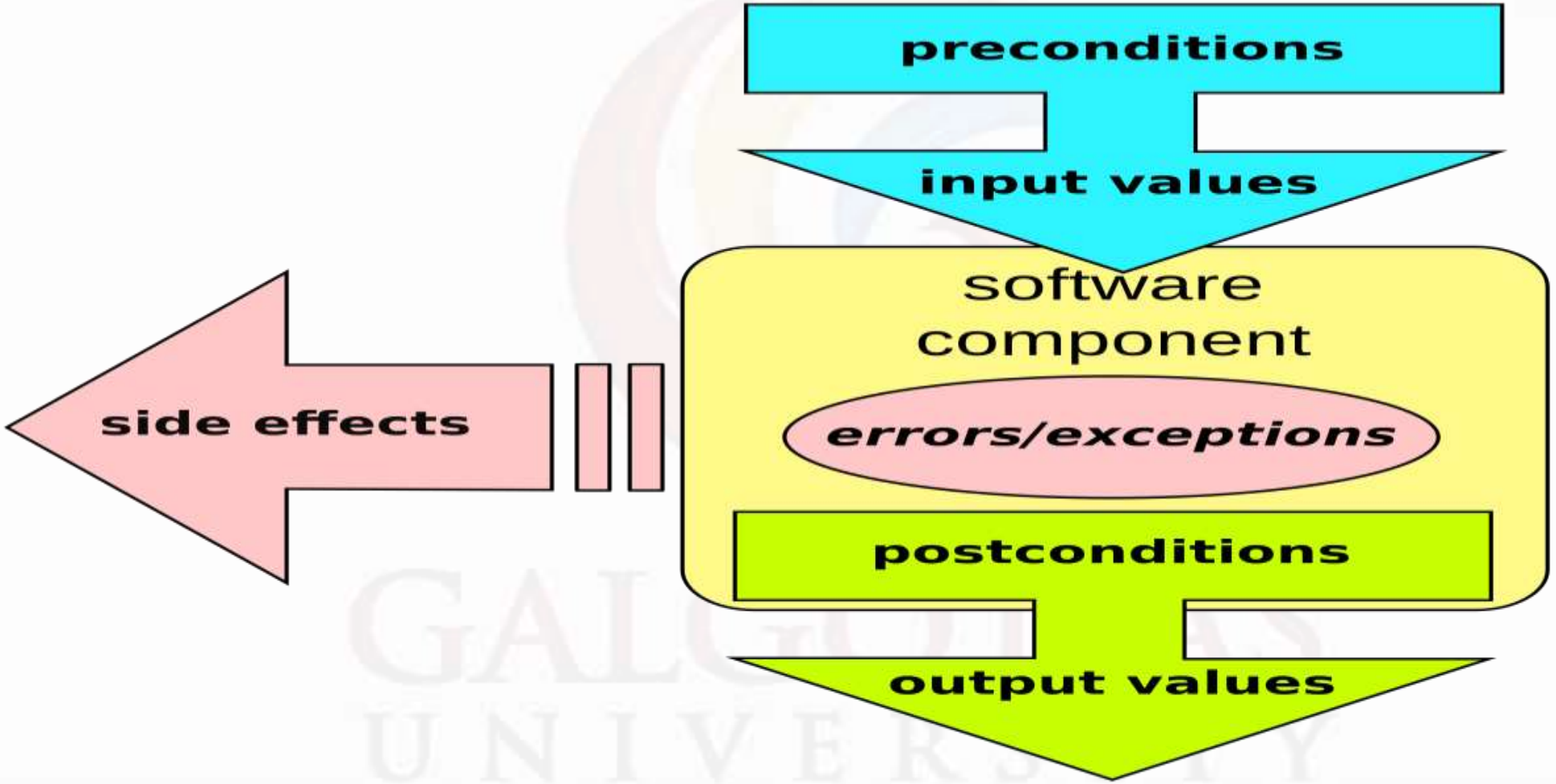
Object-Oriented Testing



Object-Oriented Testing



Object-Oriented Testing



Completion Criteria

- When are we done testing?
1. One view: testing is never done… the burden simply shifts from the developer to the customer
2. Testing is done when you run out of time or money
3. Use a statistical model:
   - Assume that errors decay logarithmically with testing time
   - Measure the number of errors in a unit period
   - Fit these measurements to a logarithmic curve

Strategic Issues

- Issues to address for a successful software testing strategy:
  - Specify product requirements long before testing commences. For example: portability, maintainability, usability.
  - Understand the users of the software, with use cases
  - Develop a testing plan that emphasizes rapid cycle testing. Get quick feedback from a series of small incremental tests
  - Build robust software that is designed to test itself Use assertions, exception handling and automated testing tools, Conduct formal technical reviews/inspections to assess test strategy and test cases.

Testing OOA and OOD Models

- The review of OO analysis and design models is especially useful because the same semantic constructs (e.g., classes, attributes, operations, messages) appear at the analysis, design, and code level.
- If the error is not uncovered during analysis and propagated further more efforts needed during design or coding stages.
- By fixing the number of attributes of a class during the first iteration of OOA, the following problems may be avoided:
- Creation of unnecessary subclasses. Incorrect class relationships.

- **Improper behavior of the system or its classes.**

  - Analysis and design models cannot be tested in the conventional sense, because they cannot be executed.
  - Formal technical reviews can be used to examine the correctness and consistency of both analysis and design models.
  - Correctness:

- **Syntax:** Each model is reviewed to ensure that proper modeling conventions have been maintained.

- **Semantic:** Must be judged based on the model's conformance to the real world problem domain by domain experts.

Consistency:

- May be judged by considering the relationship among entities in the model.

- Each class and its connections to other classes should be examined.
- The Class-responsibility-collaboration model and object-relationship diagram can be used.

Testing Models

- Criteria Correctness

. Completeness Consistency

- Early informal models are tested informally
- The criteria should be interpretive in the context of iterative incremental approach

Model Testing Approach

- Testing by comparison

- compares each model to its predecessor or to previous forms of the model

- Testing by inspection

- uses checklists to make sure that the model meets certain criteria

- Testing by verification

follows certain steps to assure completeness and consistency of one part of the model with another

- Examples of Analysis and Design Models to be tested
  - CRC cards
    - English text descriptions of a single class, its responsibilities, and it collaborators with other classes
  - Class specifications

Complete specification of a class including its data structure, method names, number and type of parameters, return values, pre and post-conditions.

Examples of Analysis and Design Models to be tested

- Use cases
  - A representation of the systems usage
- State-Transition Models
  - State transition diagrams for classes, clusters, and subsystems
- Object network
  - Message sequence between methods in classes
  - Transaction-Flow Models

Testing the Class Model

Revisit the Use Cases, CRC cards and UML class model.

Check that all collaborations are properly represented. Inspect the description of each CRC index card to determine if a delegated responsibility is part of the collaborator‘s definition

- Example: in a point of sale system. A *read credit card* responsibility of a *credit sale* class is accomplished if satisfied by a *credit card* collaborator

1. Invert connections to ensure that each collaborator asked for a service is receiving requests from a reasonable source
   - Example: a credit card being asked for a purchase amount
2. These steps are applied iteratively to each class and through each evolution of the OOA Model .

References:

- Craig Larman, “Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development”, Third Edition, Pearson Education, 2005.

- Paul C. Jorgensen, “Software Testing:- A Craftsman‟s Approach”, Third Edition, Auerbach Publications, Taylor and Francis Group, 2008.

Thank You