INTRODUCTION TO SOFTWARE ENGINEERING

Dr.Lê Hùng Tiến

VLU

Introduction To Software Engineering

Carnegie Mellon University
The Practical Software Engineering Series

The Software Process Models

Course Objective

- Upon completion of this course, students will have the ability to:
  - Understand the discipline and principles of Software Engineering.
  - Understand the evolution of software in industry and the global competitive trends.
  - Understand software process, product and services.
  - Understand software modeling & techniques.
  - Demonstrate an appreciation for the breadth of software engineering.

Lecture Learning Objectives

- Upon completion of this lecture, students will be able to:
  - Understand why process is important in Software Engineering.
  - Appreciate the disciplines of Software Engineering.

- Outcomes:
  - Be able to describe the software process models and associated engineering practices.
  - Demonstrate an understanding of the relationship between software product and process.

The Problems with Software

- Software construction is human-intensive.
- Software is intangible.
- Software problems are often complex and difficult to solve.
- Software solutions require rigor and hard work.
- Software requirements are often ambiguous, not well-defined.
- Software size and complexity grow exponentially over time.
- Large projects suffer more problems than smaller ones, due to team size, communication, and skills.

Software Engineering Definitions

- The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software.
- An engineering discipline that is concerned with all aspects of software development.
- The establishment and use of good engineering principles to economically obtain software that is reliable and works efficiently.
- The use of techniques, methods, and methodologies to develop quality software products under constraints of limited time, budget and resources.

Software Engineering

- Software Engineering covers all aspects of software development and maintenance from concept initiation, analysis, architecture, design, construction, verification, validation and management of software activities to answer the following questions:
  - What is the problem to be solved?
  - What are the characteristics of the software that is used to solve the problem?
  - How will the software be realized?
  - How will the software be designed?
  - How will the software be constructed?
  - What approach will be used to uncover errors that were made during design and construction of the software?
  - How will the software be supported over time, when corrections, adaptations, and enhancements are requested?

Software Engineering Phases - 1

- Definition Phase (What)
  - What is the problem to be solved?
  - What are the characteristics of the software that is used to solve the problem?
- Development Phase (How)
  - How will the software be realized?
  - How will the software be designed?
  - How will the software be constructed?
- Maintenance Phase (Change)
  - What approach will be used to uncover errors that were made during design and construction of the software?
  - How will the software be supported over time, when corrections, adaptations, and enhancements are requested?

Software Engineering Phases - 2

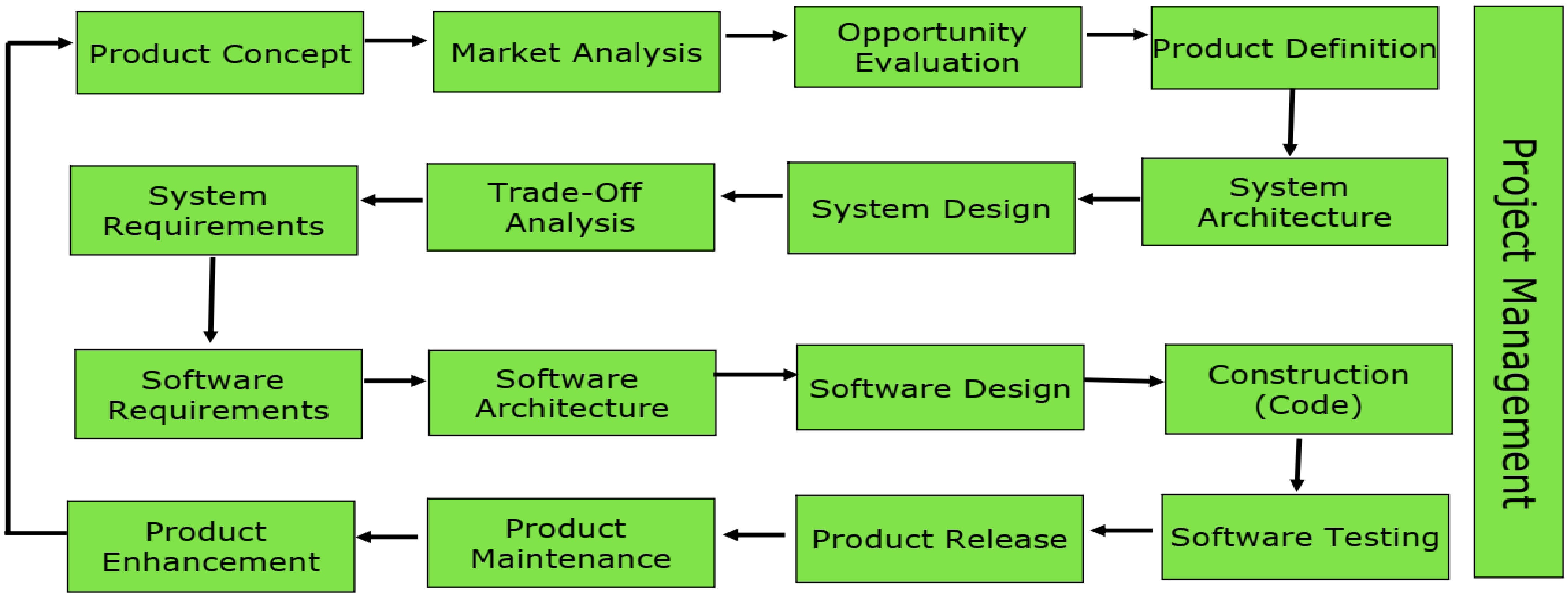
- <u>Definition Phase (What):</u>
  - Product Concept (What is the software, what information)
  - Market Analysis (Who has what, what benefits, what market)
  - Opportunity Evaluation (What should we build or buy)
  - Product Definition (What function and performance)

- <u>Development Phase (How):</u>
  - System Architecture (How to structure functions, data)
  - System Design (How procedural details)
  - Trade-Off analysis (How to select alternatives)
  - System Requirements (How components interface, relate)
  - Software Requirements (How many functions, procedures)
  - Software Architecture (How to organize components)
  - Software Design (How to design components)
  - Software Construction (How to construct these components)

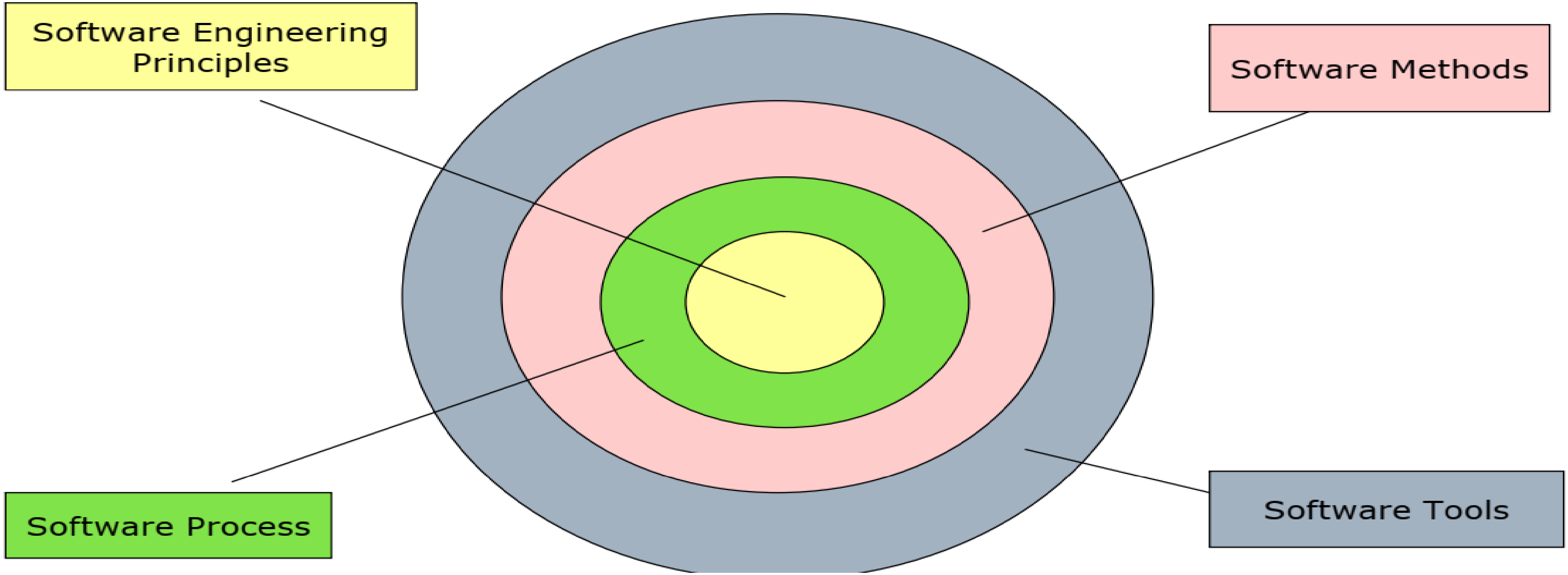
Software Engineering Phases - 3

- <u>Maintenance Phase (Change):</u>
  - Software verification (Unit tests, Module tests, System tests)
  - Software validation (Integration tests, Acceptance tests)
  - Product Release
  - Product Correction (Bug fixes, patches etc.)
  - Product Maintenance (Maintain operation, correct defects)
  - Product Enhancement (Improvement, add new functions)
  - Product Adaptation (Modification, re-hosting, re-engineering)

Product Development In Software Industry



Software Engineering Layered Architect



Software Engineering Disciplines

- The core of Software Engineering is its **principles** which is a commitment to quality and continuous improvement that ultimately leads to better approaches to solve problems.

- The next layer is the **process** that defines a set of disciplines or A sequence of actions that must be followed to effectively deliver the end-results (a solution or software products).

- Next to the process layer is the **methods** layer which provides the technical "How to" for constructing software. Methods consist of many tasks such as modeling and techniques.

- The outer layer is the **tools** that provide automated support for the process and methods.

Definitions

- **Process** (What, what's next)

  A series of actions, changes, or functions organized in a structured step-by-step way to achieve an end-result.

- **Method** (How to)

  A complete set of rules that establishes a precise and repeatable way of performing a task and arriving at a desired result**.**

- **Methodology**

  A collection of methods, procedures, and standards that defines an integrated synthesis of software engineering approaches to the development and maintenance of a product.

- **Tools** (automated)

  An automated or semi-automated device to support process, method and methodologies.

The Process Model

- To develop software products, software engineers must establish a strategy consisting of the process, methods and tools, organizing them into a structure that guides the development called "Process Model" or "Development Life-Cycle".

- There are several models, each describing approaches to a variety of development tasks or activities:
  - The Waterfall Life Cycle
  - The Prototyping Life Cycle
  - The Spiral Life Cycle
  - The Incremental Development Life Cycle
  - Others

The Basic Process - 1

1. **Requirements**:
   - Define the system to be developed
   - Set the project scope
   - Develop project plan

2. **Analyze**:
   - Review problem the organization wants to solve
   - Identify requirements for the proposed system
   - Prioritize and document the requirements

3. **Design**
   - Develop a “technical blueprint” or architecture
   - Identify system output, input, and user interfaces
   - Select system components (hardware, software, databases, telecommunications, and procedures)
   - Determine how these components are integrated

The Basic Process - 2

4. <u>Implement (Code)</u>:

- Translate the design into code
- Compile and check the code against syntax errors

5. <u>Test</u>:

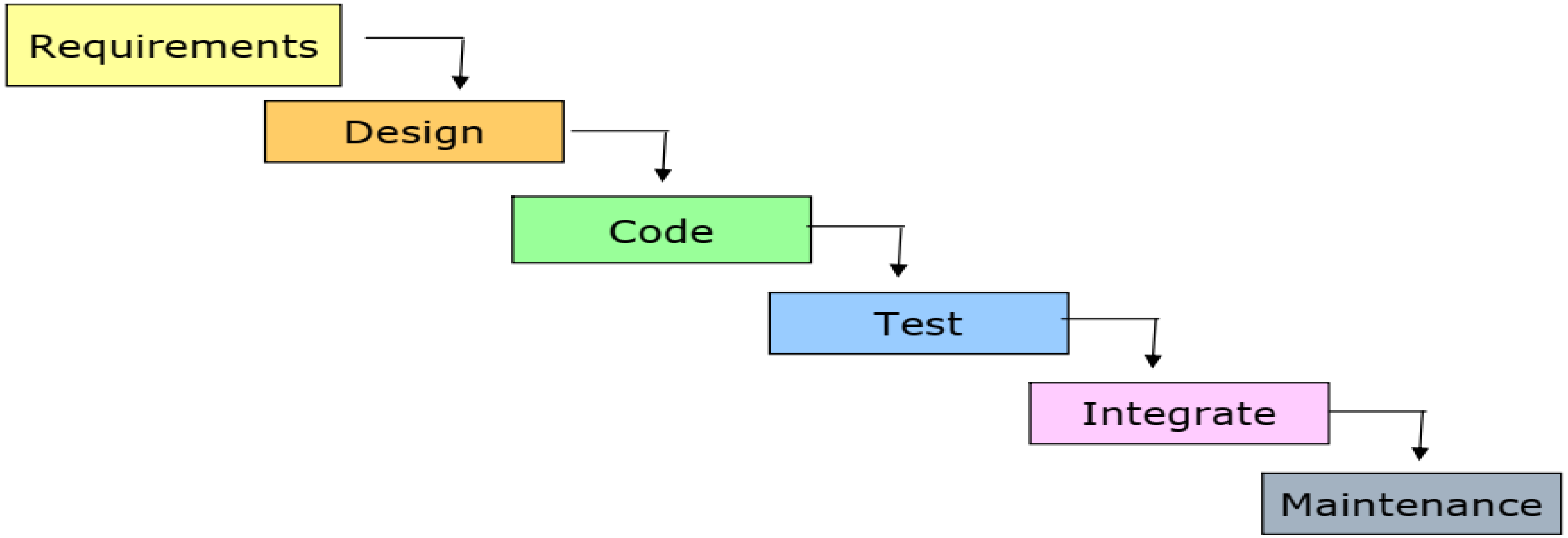
- Check the code to see if it meets the expected results
- Test the system implementation in system hardware
- Test the system in the business environment

6. <u>Maintain</u>:

- Continue to monitor and support the system to ensure it meets the business goals
- Provide support (debug, fix, enhance)

The Waterfall Model

The waterfall life cycle is a sequential development model in which development is seen as flowing downwards (like a waterfall) through the phases of requirements, design, code, test, integrate and maintenance.



The Waterfall Model

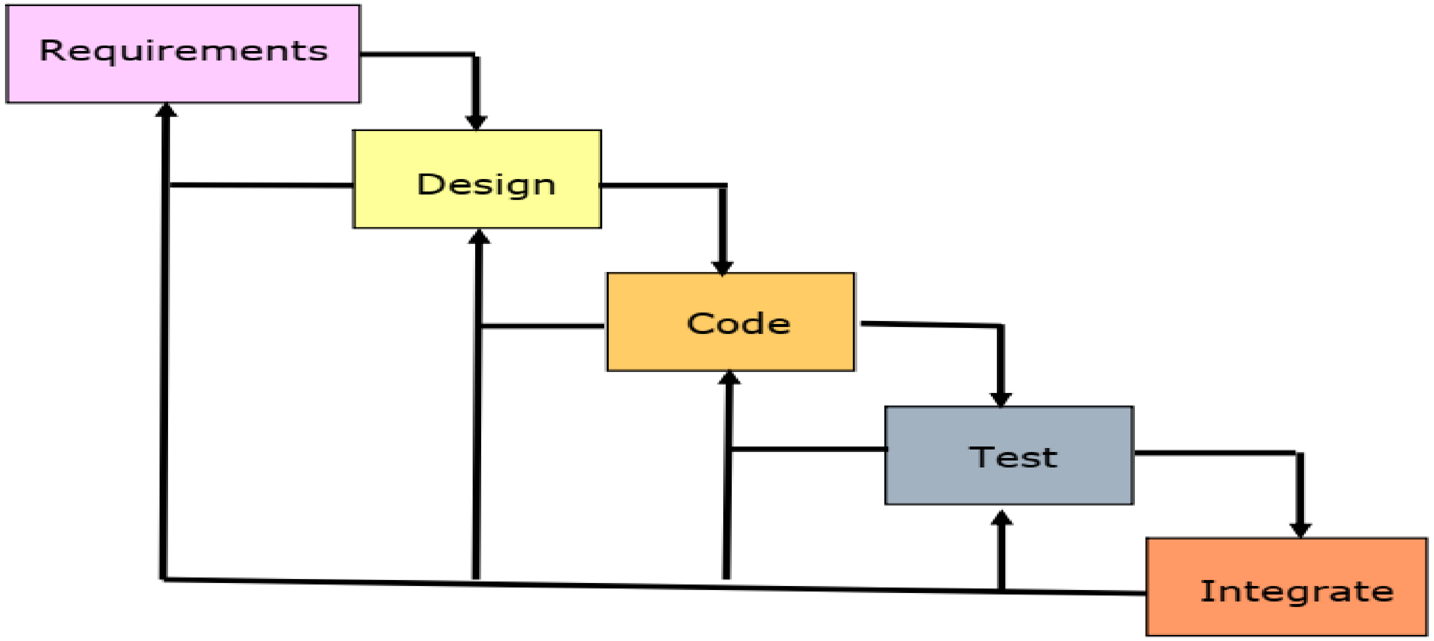
- In the waterfall model, proceed from one phase to the next in a sequential manner. Thus, only move to the next phase when the preceding phase is completed.

- The waterfall model is easy to understand and useful when you have well-defined and stable requirements for a large software system with large distributed teams. But in practice, it is unrealistic because requirements often change, making it difficult to fully complete one phase before moving to the next.

    - For example, customers may not know exactly what they want in a system to clearly define the requirements. They change their mind (requirements) as the project progresses. If the developers have completed the design, it must be modified to accommodate new requirements. This may require a lot of time and effort.

The Waterfall Model

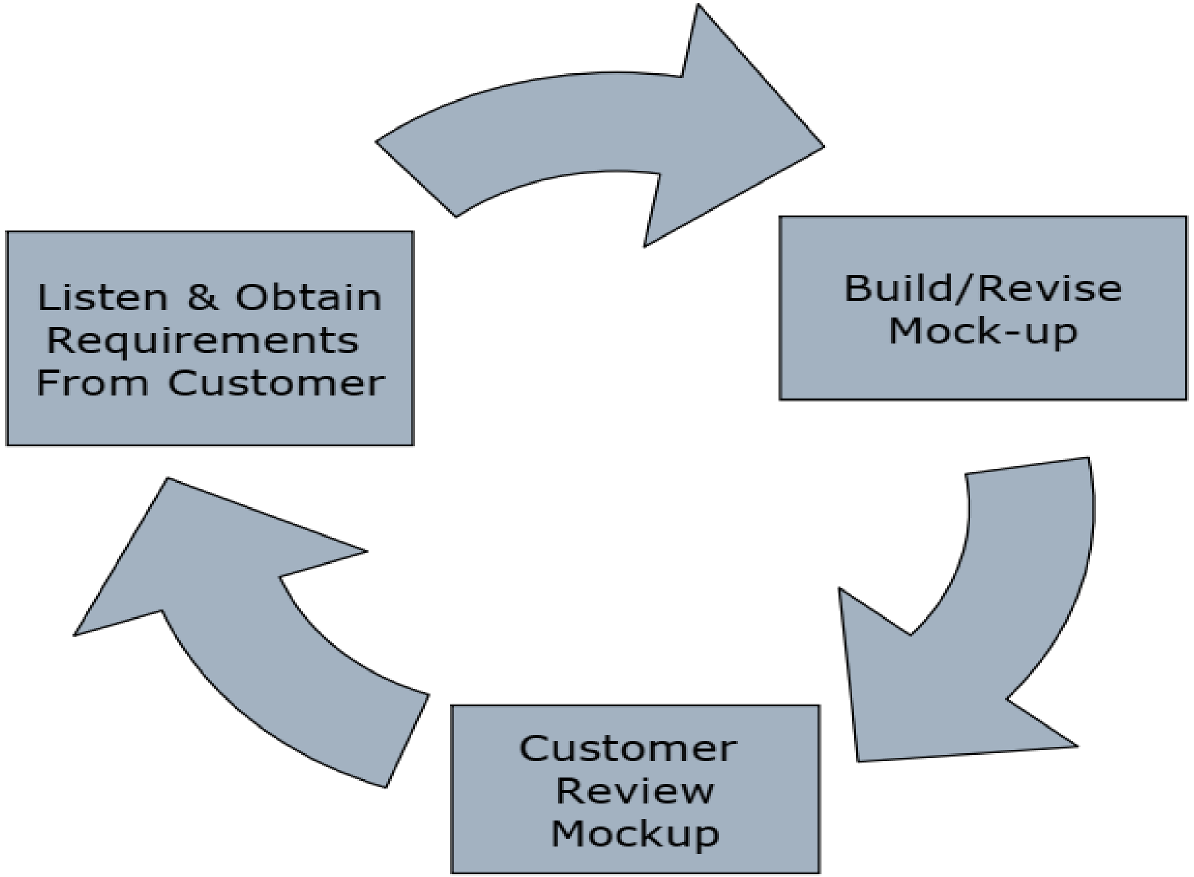
Most often, customers do not know the exact requirements for a project, and they are almost certain to change as the project progresses. Developers may have to go back to a previous phase for clarification. Very time consuming on rework.



The Prototyping Model

- The prototyping model can build software quickly and inexpensively for evaluation and demonstration.
- The development of an incomplete system to understand the system and the scale of the problem.
- Prototyping is an iterative design approach, where designs are created, evaluated, and refined until the desired performance is achieved.
- For example:
  - ***Prototype –*** a model of a proposed product, service, or system.
  - ***Proof-of-concept prototype –*** used to prove the technical feasibility of a proposed system.
  - ***Selling prototype*** - used to convince people of the worth of a proposed system.

The Prototyping Model



Listen & Obtain Requirements From Customer

Build/Revise Mock-up

Customer Review Mockup

The Rapid Application Development Model

- The Rapid Application Development (RAD) Model is a development sequence that focuses on a very "short development cycle" or rapid adaptation of the waterfall.

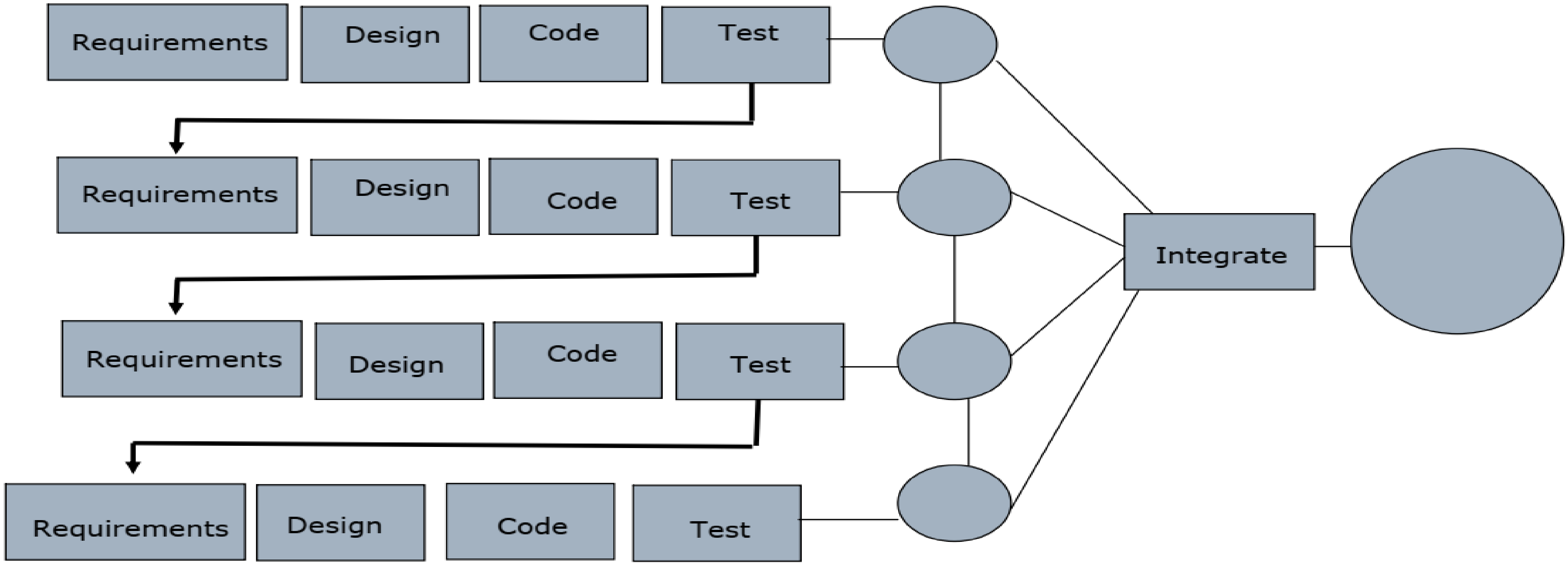
- The key to a rapid Application Development model is to start with a small "scalable scope" of what is to be built incrementally and keep evolving until the entire system is built.

- This model requires strong commitment from developers and users to work together in a very constrained time frame to develop the system incrementally, otherwise it will not work.

- If the system can not be properly broken into small pieces or modularized, then this model will not work well.

The Rapid Application Development Model

Several iterations are built to help put the system in place.



The Incremental Model

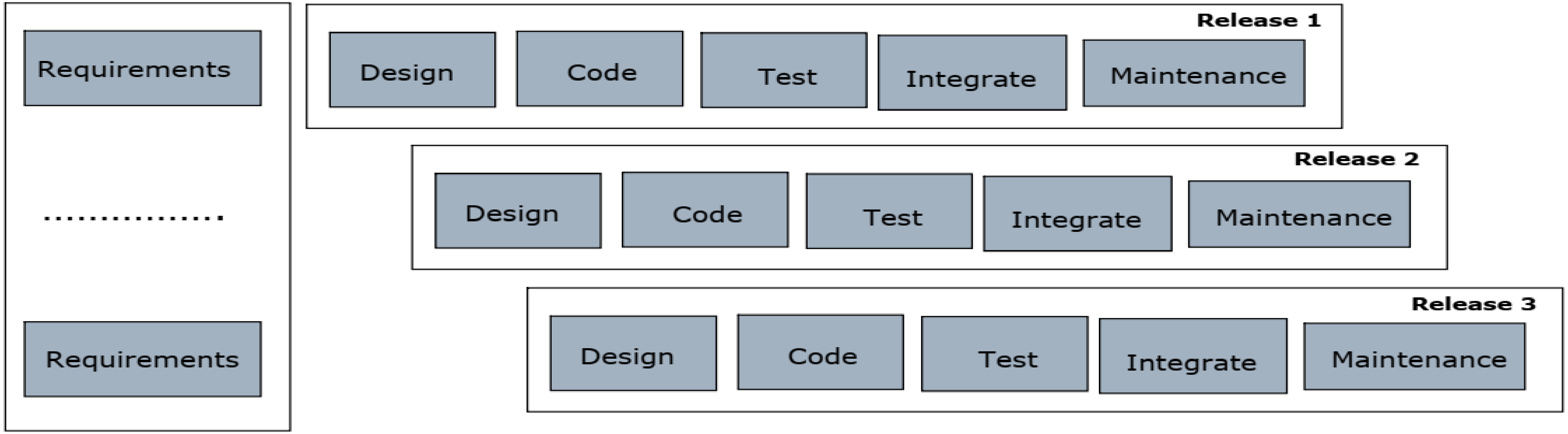
- □ Incremental development life cycle is a **staging model** in which various parts of the system are developed at different times and integrated as they are completed.

- □ Iterative development is a **rework scheduling strategy** in which time is set aside to revise and improve parts of the system.

- □ The basic idea behind iterative enhancement is to develop a software system **incrementally**, allowing the developer to take advantage of what was being learned during the development of earlier, incremental, deliverable versions of the system.

- □ Developers start with a simple implementation of a subset of the requirements and iteratively enhance the evolving sequence of versions until the full system is implemented. At each iteration, design modifications are made and new functional capabilities are added.

The Incremental Model



Requirements continue to be analyzed then allocated to a series of incremental releases.

The Spiral Model

- The development life cycle that combines elements of both design and prototyping concepts. This model combines the features of the prototyping model and the waterfall model.

- Estimates (i.e. budget, schedule, etc.) get more realistic as work progresses in each iteration because important issues are discovered earlier.

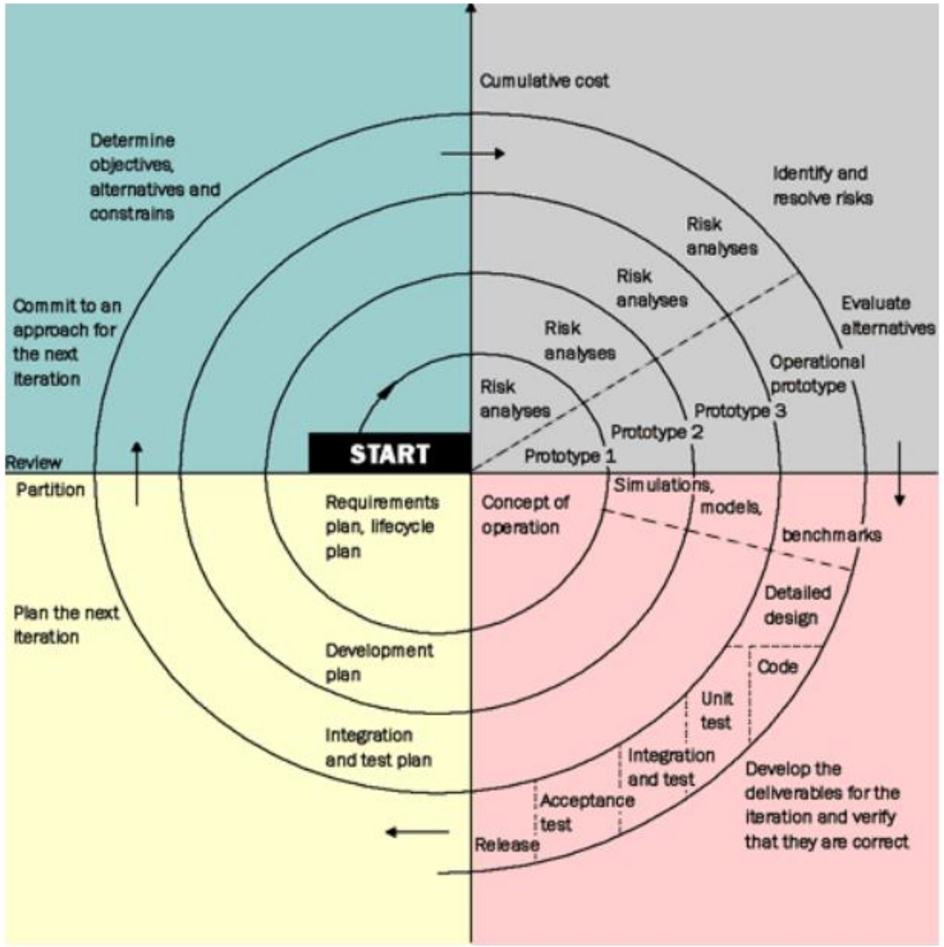
- It is better able to cope with the (nearly inevitable) changes that software generally faces.

- Software engineers can get their hands in and start working on a project earlier.

The Spiral Model

Requirements

Design



Review

Code

The Spiral Model

- Requirements are defined in as much detail as possible.
- A preliminary design is created for the system.
- A first prototype of the system is constructed from the preliminary design.
- Project teams evaluate the prototype for strengths, weaknesses, and risks.
- Project teams define the requirements of the second prototype.
- Project teams plan and design the second prototype, followed by constructing and testing.
- At the customer's request, the entire project can be aborted if the risk is deemed too great. Risk factors might involve development cost overruns, operating-cost miscalculation, or any other factor that could, in the customer's judgment, result in a less-than-satisfactory final product.
- The existing prototype is evaluated as was the previous prototype, and, if necessary, another prototype is developed from it according to the fourfold procedure outlined above.
- The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.
- The final system is constructed, based on the refined prototype.
- The final system is thoroughly evaluated and tested.

Summary

□ The quality of software products is governed by the quality of the process that created it.

□ Software Engineering is a discipline that integrates process, methods and tools for the creation of software products that have high quality and meet users’ needs.

□ There are different process models to organize the software development, each has strengths and weaknesses.