Session 2: System and software engineering process

- What is a system.
- Socio technical system.
- Characteristics and properties of a socio-technical system.
- System Reliability Engineering.
- Differences in disciplines and Interdisciplinary involvement.
- What is a software process.
- What is process pattern and types of pattern.
- Software process models.
- Types of software process models and comparisons.
- Phases and activities of a model.

What is a system?

- A purposeful collection of inter-related components working together to achieve some common objective.
- A system may include software, mechanical, electrical and electronic hardware and be operated by people.
- System components interdependent.
- The properties and behaviour of system components are extremely inter-mingled.

Socio-technical systems

- "Sociotechnical refers to the interrelatedness of **social and technical** aspects of an organization or the society as a whole". –Wiki.

- Systems that include technical systems but also operational processes and people who use and interact with the technical system.

- Socio-technical systems are governed by organisational policies and rules.

Socio-technical system characteristics

- **Emergent properties**
  - Properties of the system as a whole that depend on the system components and their relationships.

- **Non-deterministic**
  - They do not always produce the same output when presented with the same input because the systems' behaviour is partially dependent on human operators.

- **Complex relationships with organisational objectives**
  - The extent to which the system supports organisational objectives does not just depend on the system itself.

Emergent properties

- Properties of the system as a whole rather than properties that can be derived from the properties of components of a system.
- Emergent properties are a consequence of the relationships between system components.
- They can therefore only be assessed and measured once the components have been integrated into a system.

Examples of emergent properties

Property	Description
Volume	The volume of a system (the total space occupied) varies depending on how the component assemblies are arranged and connected.
Reliability	System reliability depends on component reliability but unexpected interactions can cause new types of failure and therefore affect the reliability of the system.
Security	The security of the system (its ability to resist attack) is a complex property that cannot be easily measured. Attacks may be devised that were not anticipated by the system designers and so may defeat built-in safeguards.
Repairability	This property reflects how easy it is to fix a problem with the system once it has been discovered. It depends on being able to diagnose the problem, access the components that are faulty and modify or replace these components.
Usability	This property reflects how easy it is to use the system. It depends on the technical system components, its operators and its operating environment.

Types of emergent property

- **Functional properties**
  - These appear when all the parts of a system work together to achieve some objective.
  - For example, a bicycle has the functional property of being a transportation device once it has been assembled from its components.
- **Non-functional emergent properties**
  - Examples are reliability, performance, safety, and security.
  - These relate to the behaviour of the system in its operational environment.
  - They are often critical for computer-based systems as failure to achieve some minimal defined level in these properties may make the system unusable.

System reliability engineering

- Because of component inter-dependencies, faults can be **propagated** through the system.
- System failures often occur because of **unforeseen inter-relationships** between components.
- It is probably impossible to **anticipate** all possible component relationships.
- Software reliability measures may give a **false picture** of the system reliability.

Influences on reliability

- **Hardware reliability:**
  - What is the probability of a hardware component failing and how long does it take to repair that component?
- **Software reliability:**
  - How likely is it that a software component will produce an incorrect output. Software failure is usually distinct from hardware failure in that software does not wear out.
- **Operator reliability:**
  - How likely is it that the operator of a system will make an error?

Reliability relationships

- Hardware failure can generate signals that are outside the range of inputs expected by the software.
- Software errors can cause alarms to be activated which cause operator stress and lead to operator errors.
- The environment in which a system is installed can affect its reliability.

The ‘shall-not’ properties

- Properties that the system should not exhibit
  - **Safety** - the system should not behave in an **unsafe way.**
  - **Security** - the system should not permit **unauthorised use.**
- Measuring or assessing these properties is very hard.

Systems engineering

- Specifying, designing, implementing, validating, deploying and maintaining a systems.

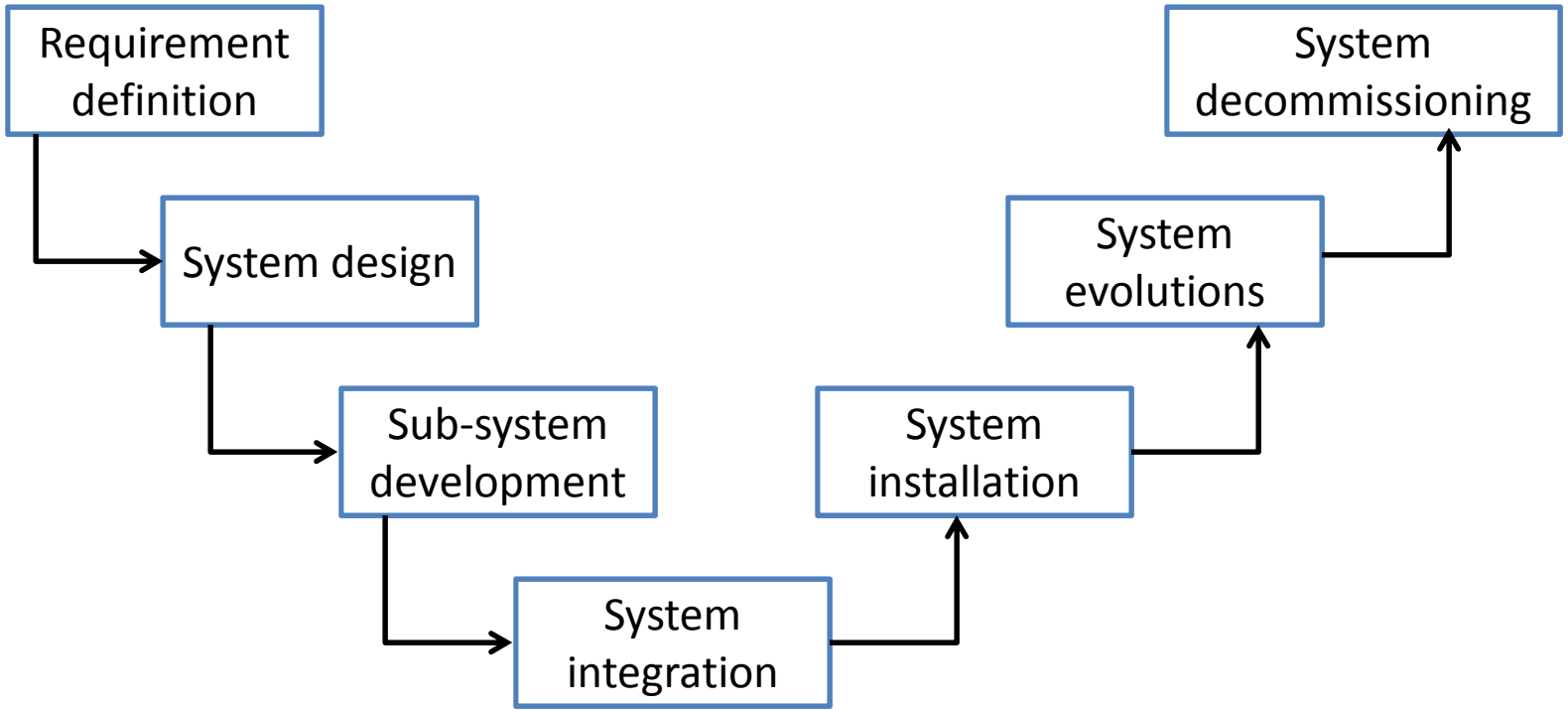
Concerned with:

- The services provided by the system.
- Constraints on its construction and operation.
- The ways in which it system is used.

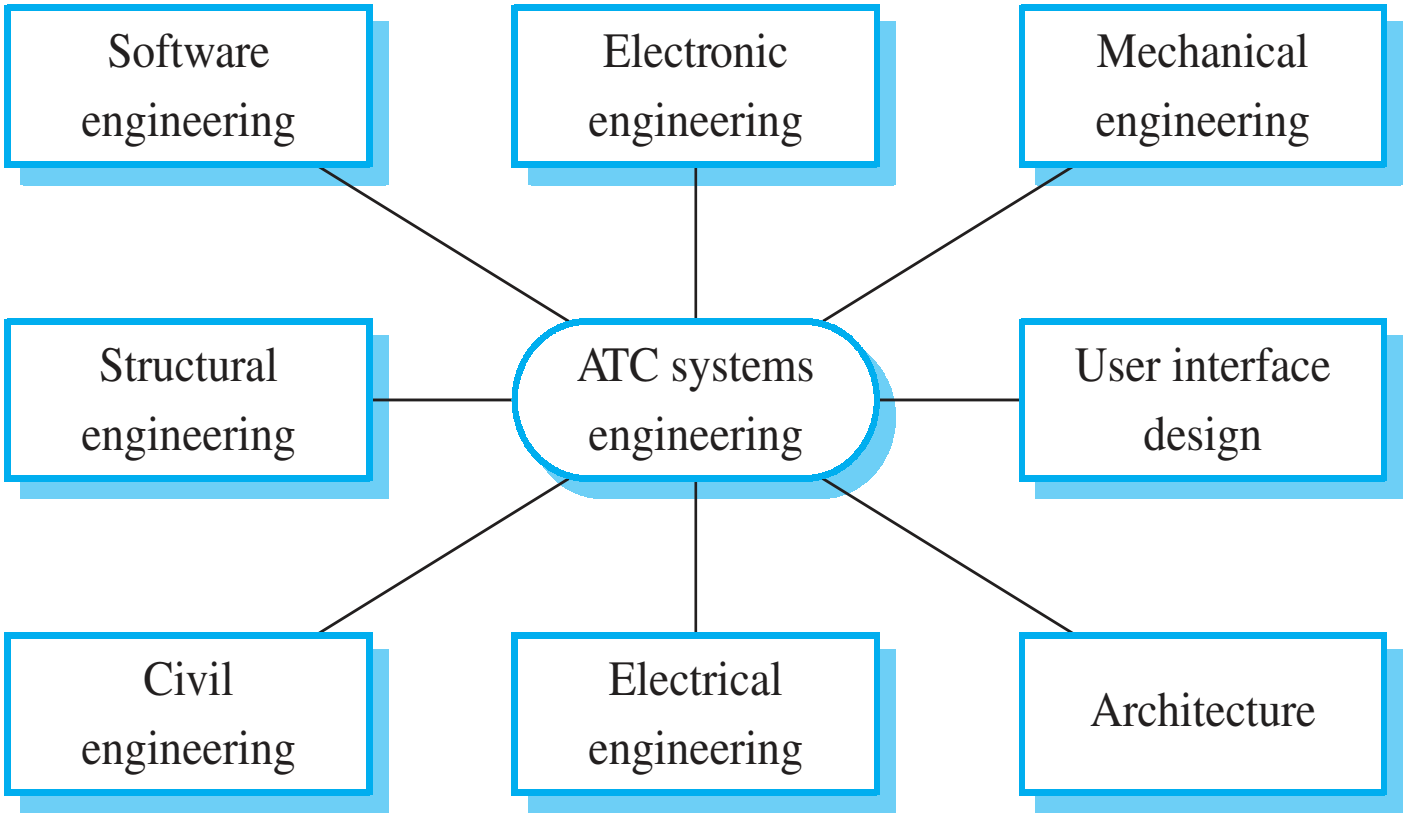
The system engineering process

- Usually follows a ‘waterfall’ model because of the need for parallel development of different parts of the system.
  - Little scope for iteration between phases because hardware changes are very expensive.
  - Software may have to compensate for hardware problems.
- Involves engineers from different disciplines who must work together.
  - Much scope for misunderstanding here.
  - Different disciplines use a different vocabulary and much negotiation is required.
  - Engineers may have personal agendas to fulfil.

The systems engineering process



Inter-disciplinary involvement



Difference between Disciplines

Software engineering and computer science?

- Computer science is concerned with theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.
- Computer science theories are still insufficient to act as a complete underpinning for software engineering (unlike e.g. physics and electrical engineering).

Software engineering and system engineering?

- System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering. Software engineering is part of this process concerned with developing the software infrastructure, control, applications and databases in the system.
- System engineers are involved in system specification, architectural design, integration and deployment.

What is a software process?

- A software process is a road map that helps you create a timely, high quality result.
- It is the way we produce software
- Provides stability and control

Produces:

- Programs, documents, and data produced as a consequence of the software engineering activities.

Process Patterns

- **A process pattern**
  - describes a process-related **problem** that is encountered during software engineering work.
  - identifies the **environment** in which the problem has been encountered.
  - suggests one or more proven **solutions** to the problem.

- A consistent method for describing problem solutions within the context of the software process (defined at different levels of abstraction).
  1. Problems and solutions associated with a complete process model (e.g. prototyping).
  2. Problems and solutions associated with a framework activity (e.g. planning).
  3. A action with a framework activity (e.g. project estimating).

Process Pattern Types

- **Stage patterns**—defines a problem associated with a framework activity for the process. It includes multiple task patterns as well. For example, **Establishing Communication** would incorporate the task pattern **Requirements Gathering** and others.

- **Task patterns**—defines a problem associated with a software engineering action or work task and relevant to successful software engineering practice.

- **Phase patterns**—define the sequence of framework activities that occur with the process, even when the **overall flow of activities is iterative in nature**. Example includes Spiral Model or Prototyping.

An Example of Process Pattern

Describes an approach that may be applicable when stakeholders have a general idea of what must be done but are unsure of specific software requirements.

- **Pattern name.** Requirements Unclear
- **Intent.** This pattern describes an approach for building a model that can be assessed iteratively by stakeholders in an effort to identify or solidify software requirements.
- **Type.** Phase pattern
- **Initial context.** Conditions must be met (1) stakeholders have been identified; (2) a mode of communication between stakeholders and the software team has been established; (3) the overriding software problem to be solved has been identified by stakeholders ; (4) an initial understanding of project scope, basic business requirements and project constraints has been developed.
- **Problem.** Requirements are hazy or nonexistent. stakeholders are unsure of what they want.
- **Solution.** A description of the prototyping process would be presented here.
- **Resulting context.** A software prototype that identifies basic requirements. (modes of interaction, computational features, processing functions) is approved by stakeholders. Following this, **1.** This prototype may evolve through a series of increments to become the production software or **2.** the prototype may be discarded.
- **Related patterns.** Customer Communication, Iterative Design, Iterative Development, Customer Assessment, Requirement Extraction.

Process Assessment and Improvement

Software process **(SP)** cannot guarantee that software will be delivered on time, meet the needs, or has the desired technical characteristics. However, the process can be assessed to ensure that it meets a set of basic process criteria that have been shown to be essential for a successful software engineering.

Standard CMMI Assessment Method for Process Improvement (SCAMPI): Provides a five step process assessment model that incorporates five phases: initiating, diagnosing, establishing, acting and learning.

CMM-Based Appraisal for Internal Process Improvement (CBA IPI): Provides a diagnostic technique for assessing the relative maturity of a software organization.

SPICE—The SPICE (ISO/IEC15504): Standard defines a set of requirements for software process assessment. The intent of the standard is to assist organizations in developing an objective evaluation of the efficacy of any defined software process.

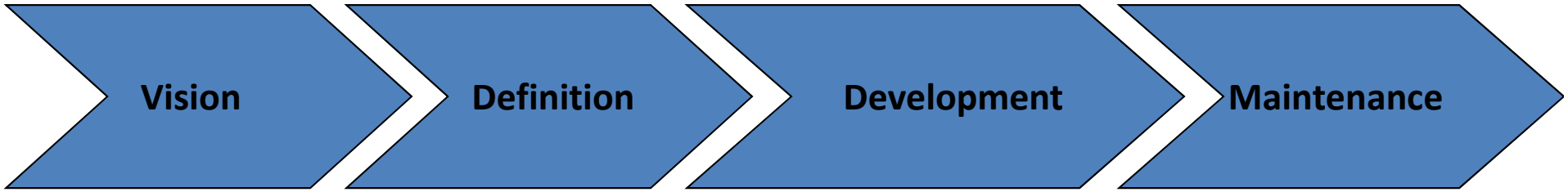
ISO 9001:2000 for Software: A generic standard that applies to any organization that wants to improve the overall quality of the products, systems, or services that it provides. Therefore, the standard is directly applicable to software organizations and companies.

Generic activities in all software processes are:

- **Specification** - what the system should do and its development constraints.
- **Development** - production of the software system.
- **Validation** - checking that the software is what the customer wants.
- **Evolution** - changing the software in response to changing demands.

Software Engineering Phases

- Vision – focus on ***why***
- Definition – focus on ***what***
- Development – focus on ***how***
- Maintenance – focus on ***change***



Software Lifecycle Models

- The way you **organize** your activities.
- Lifecycle model is a **series of steps** through which the product progresses.

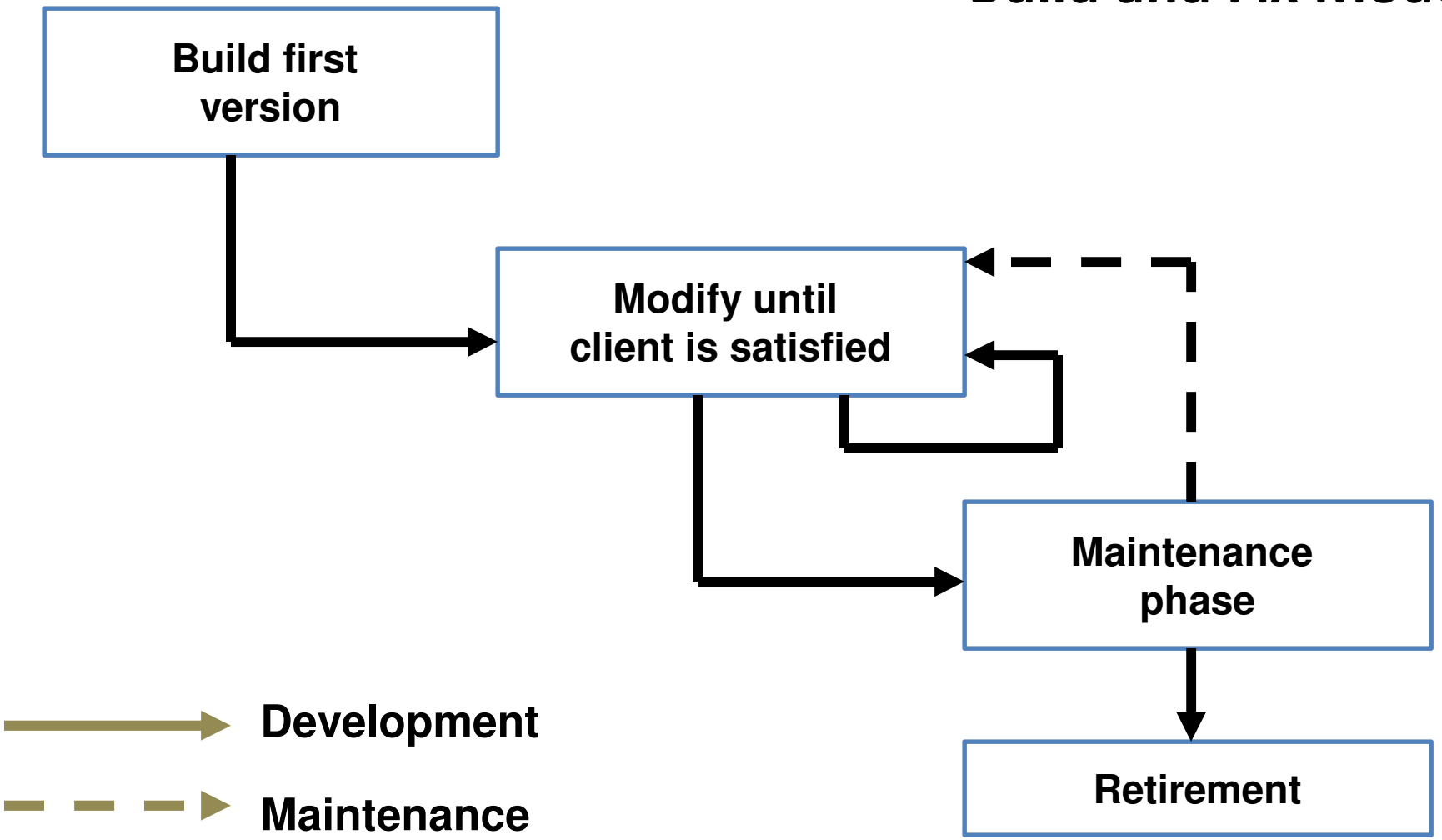
Software Life-Cycle phases

- Requirements phase
- Specification phase
- Design phase
- Implementation phase
- Integration phase
- Maintenance phase
- Retirement

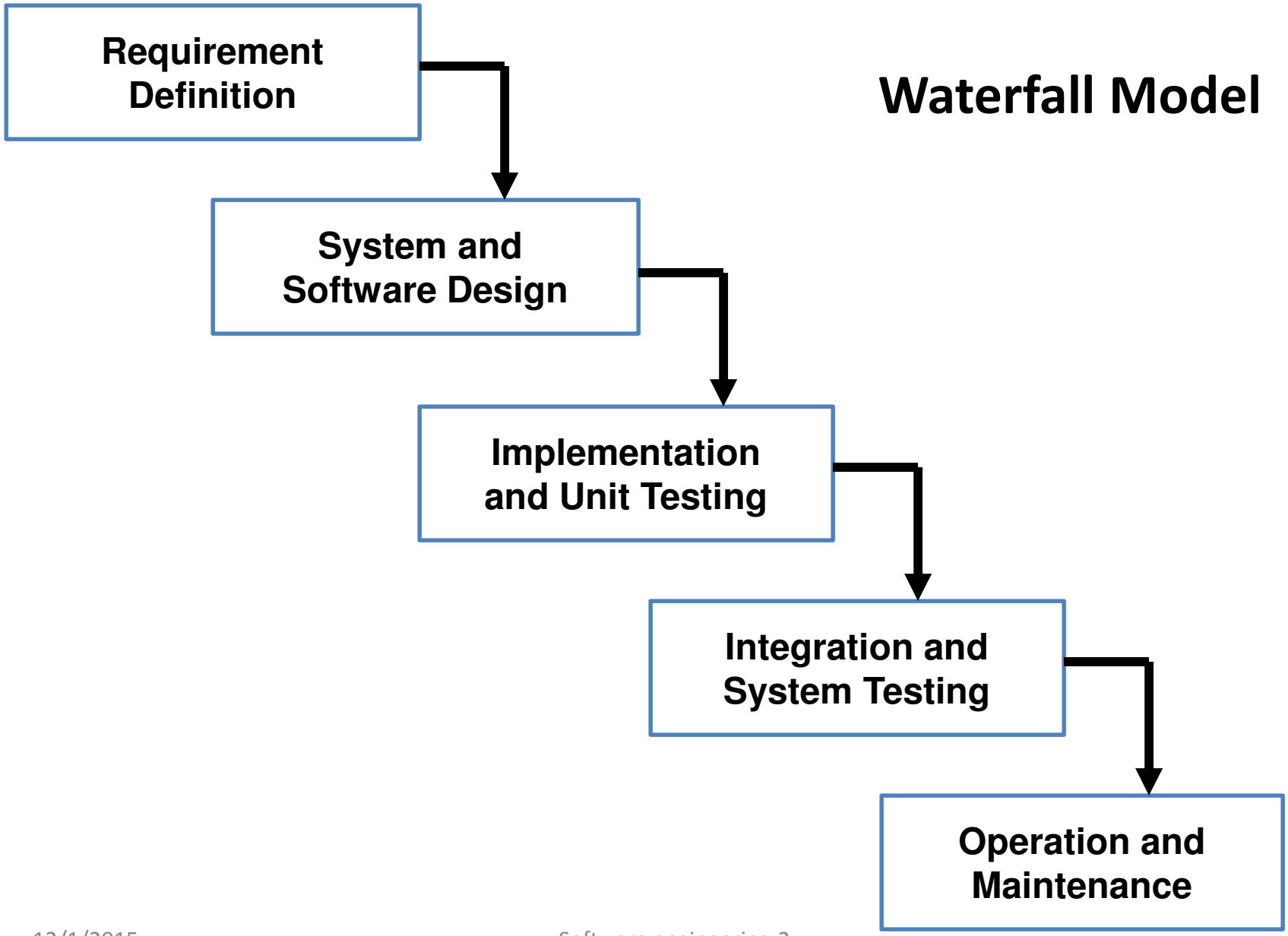
Different Lifecycle Models

- Build-and-fix model
- Waterfall model
- Rapid prototyping model
- Incremental model
- Extreme programming
- Synchronize-and-stabilize model
- Spiral model
- Object-oriented life-cycle models
- Comparison of life-cycle models

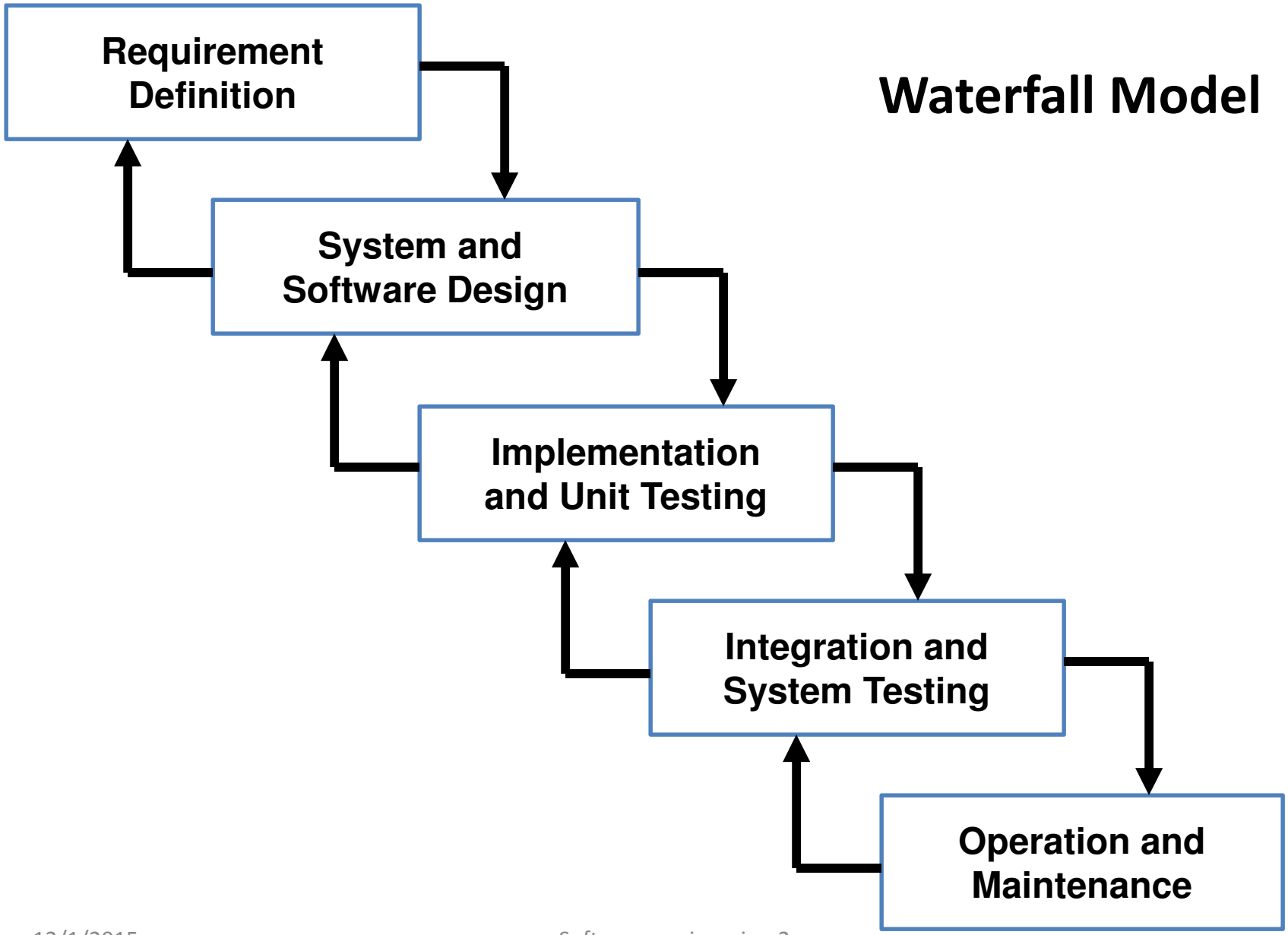
Build and Fix Model



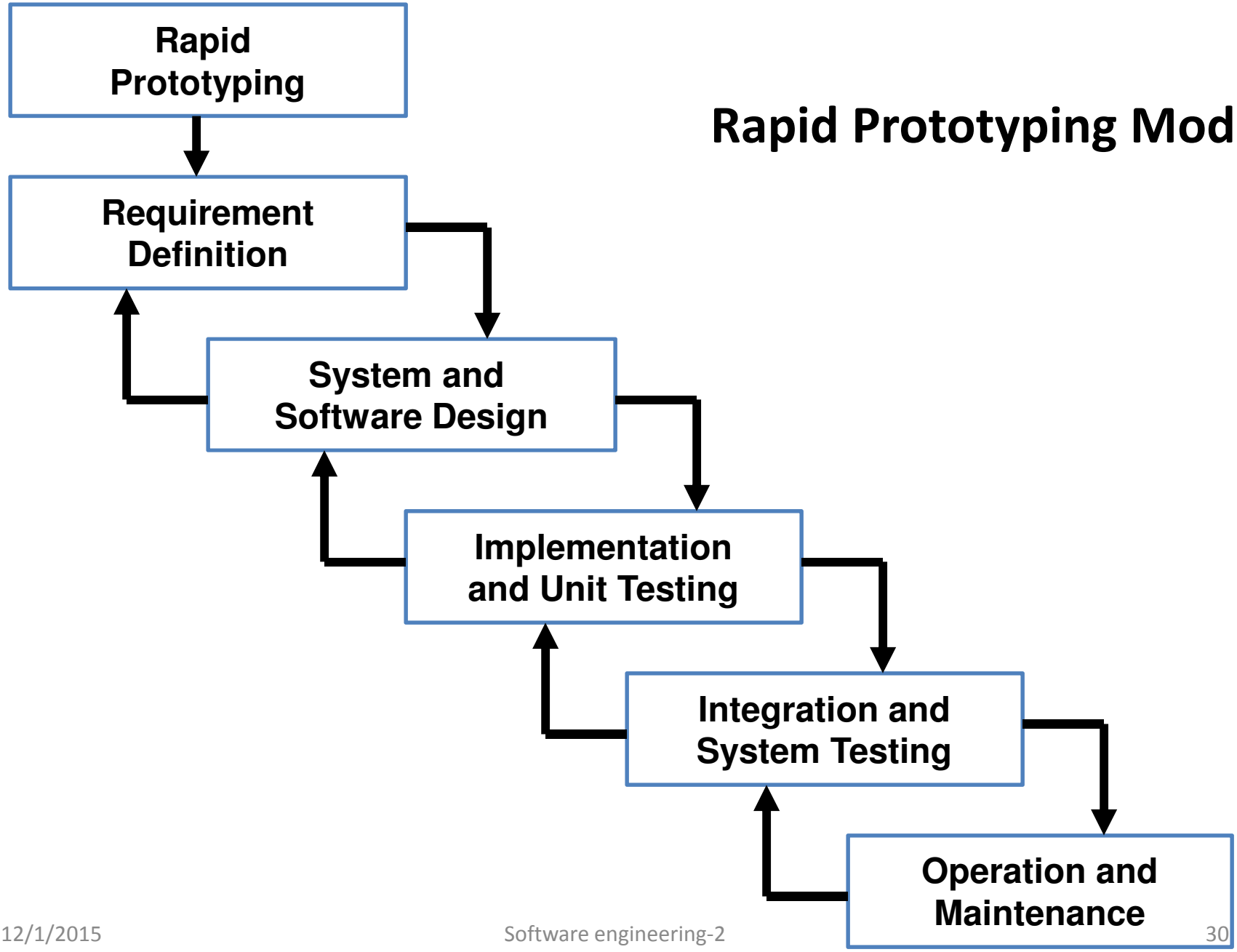
Waterfall Model



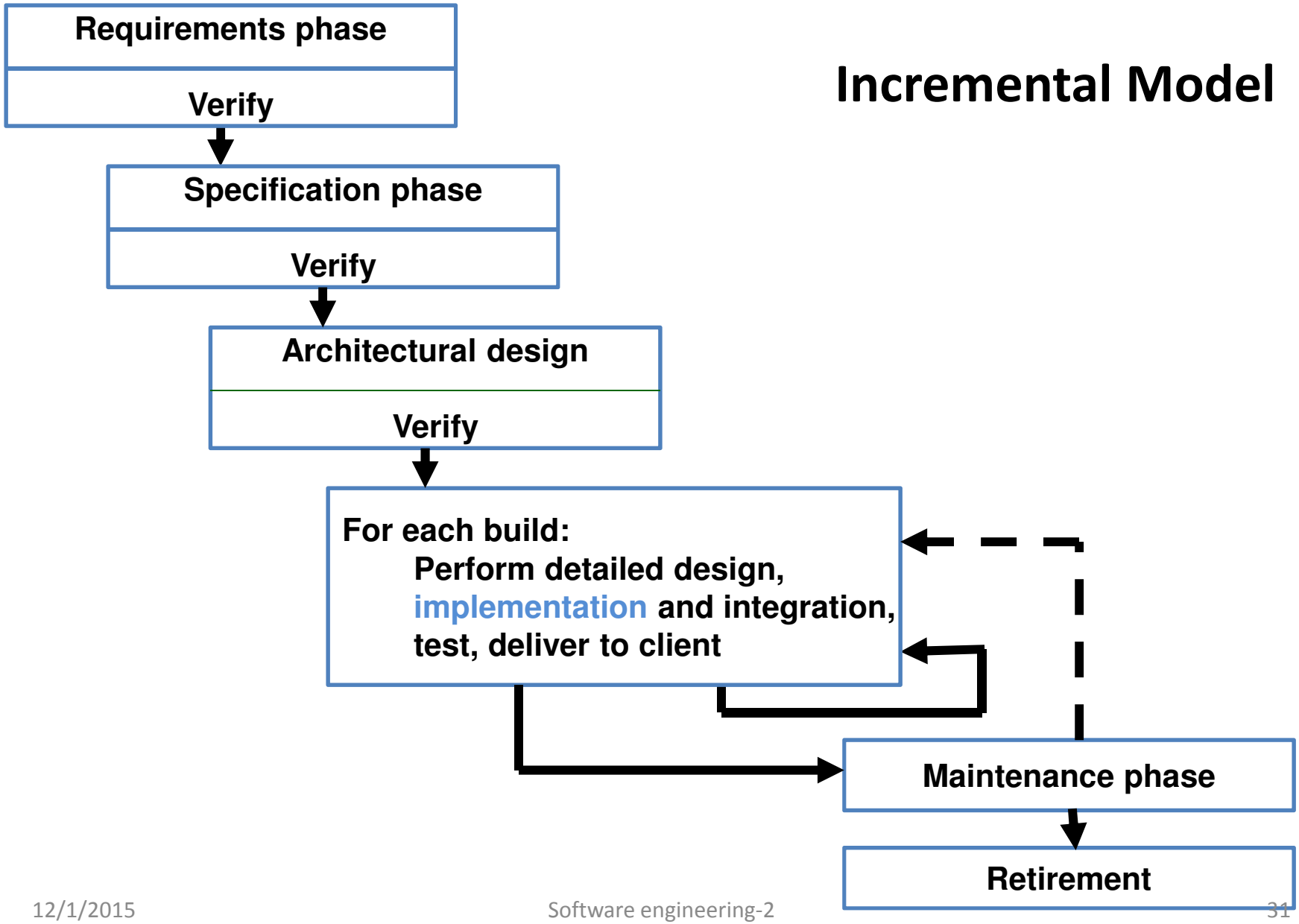
Waterfall Model

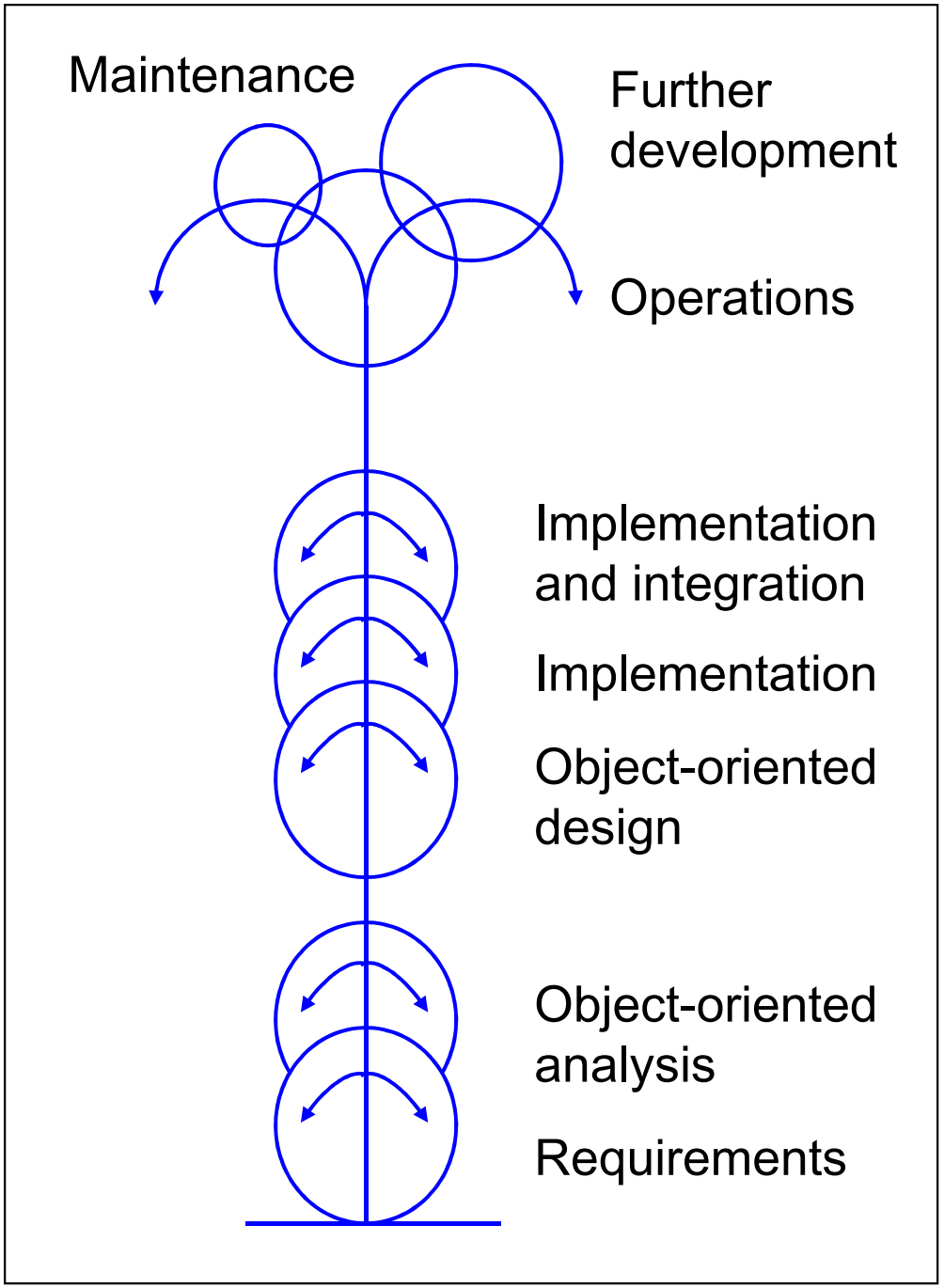


Rapid Prototyping Model



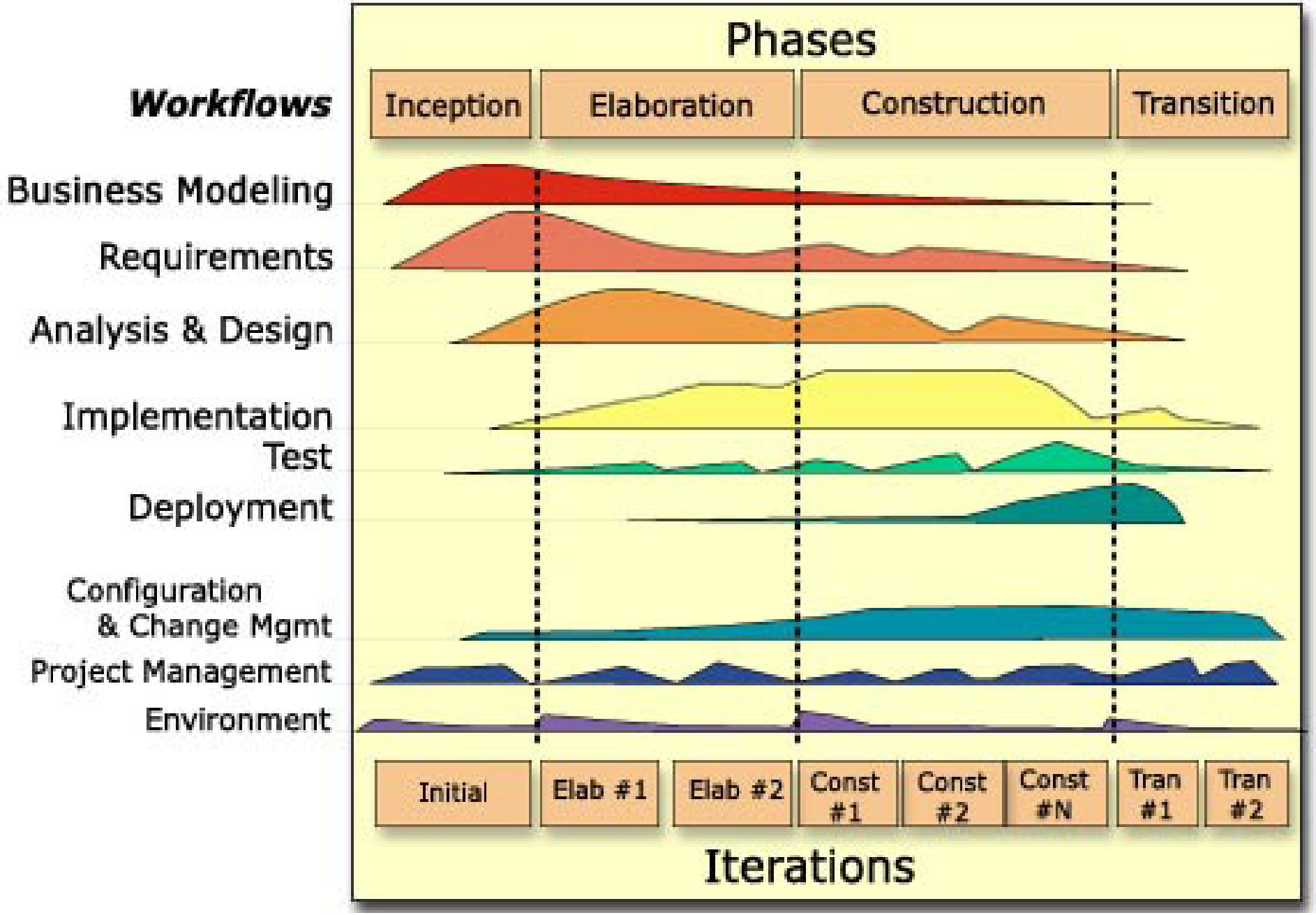
Incremental Model



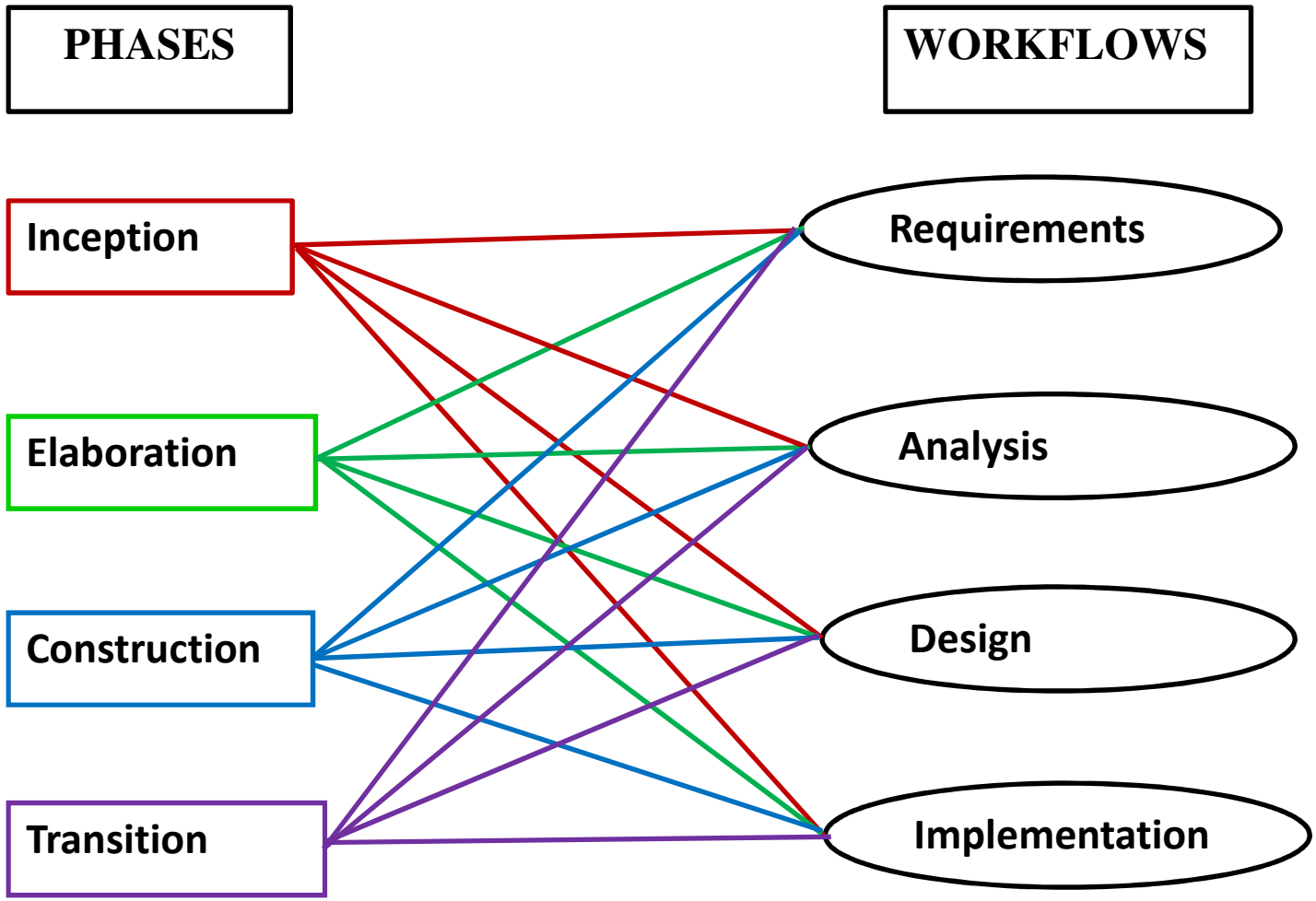


Fountain Model

Rational Unified Process



Phases and Workflows - activities



Comparison of Process Models

Process Model	Strengths	Weaknesses
Build-and-fix model	• Fine for short programs that not require maintenance	• Totally unsatisfactory for nontrivial programs
Waterfall model	• Disciplined approach • Document driven	• Delivered product may not meet client's needs
Rapid prototyping model	• Ensures that delivered product meets client's needs	• Not yet proven
Incremental model	• Maximizes early return on investment • Promotes maintainability	• Requires open architecture • May degenerate into build-and-fix
Extreme programming	• Maximizes early return on investment • Fine when requirements are vague	• Small projects, small teams • Lack of design documentation • Has not yet been widely used
Synchronize-and-stabilize model	• Components always work together • Early insights into operation of product	• Has not been widely used other than at Microsoft
Spiral model	• "How much to test ?" in terms of risk • Maintenance is another cycle	• Only for large-scale, in-house products
Object-oriented models	• Iteration within phases • Parallelism between phases	• May degenerate into CABTAB

Documentation Phase?

- There is NO documentation phase
- Every phase must be fully documented before starting the next phase
  - Postponed documentation may never be completed
  - The responsible individual may leave
  - The product is constantly changing—we need the documentation to do this
  - The design (for example) will be modified during development, but the original designers may not be available to document it

Phase	Documents	QA
Requirement Definition	• Rapid prototype, or • Requirements document	• Rapid prototype • Reviews
Functional Specification	• Specification document (specifications) • Software Product Management Plan	• Traceability • FS Review • Check the SPMP
Design	• Architectural Design • Detailed Design	• Traceability • Review
Coding	• Source code • Test cases	• Traceability • Review • Testing
Integration	• Source code • Test cases	• Integration testing • Acceptance testing
Maintenance	• Change record • Regression test cases	• Regression testing

Acknowledgement

- Thank you all for your time.

Next session

- Waterfall Model.
- Evolutionary Development.
- Exploratory development.
- Throwaway prototyping.
- Spiral Model.
- Advantages and disadvantages of Spiral Model.