SE2250 – Software Construction

Lecture #6

Software Development Process

February 25, 2019

Outline

- Introduction - problem slowing
- Development activities
- Software development process

Problem solving

Engineering is a **problem-solving** activity.

1. Formulate the problem.
2. Analyze the problem.
3. Search for solutions.
4. Decide on the appropriate solution.
5. Specify the solution.
6. Validate and verify the solution

Development activities

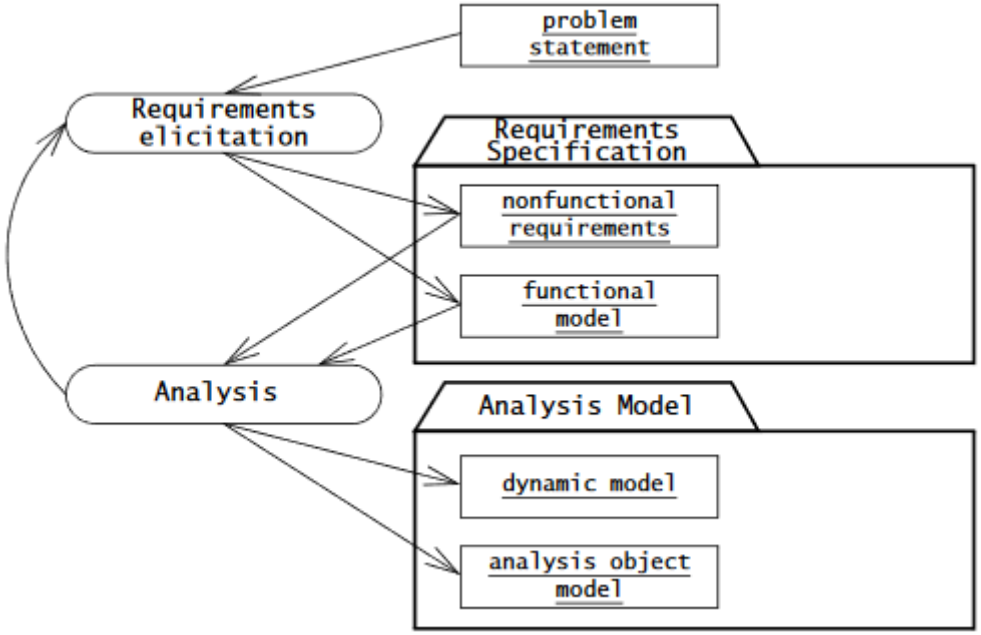
- Software development activities include
  1. Requirements Elicitation
  2. Analysis
  3. System Design
  4. Object Design
  5. Implementation
  6. Testing

Development activities

- **Requirement?**
- **A requirement** is a feature that the system must have or a constraint that it must satisfy to be accepted by the client.
- **Requirements engineering** aims at defining the requirements of the system
  - ***Requirements elicitation (A1)*** results in the specification of the system that the client understands
  - ***Analysis (A2)*** results in an analysis model that the developers can unambiguously interpret (class diagrams, sequence diagrams, state machines…)

Development activities

- **Requirements engineering**



Development activities

- ***Analysis (A2)*** focuses on producing a model of the system (the analysis model), which is correct, complete, consistent, and verifiable.
- Models?
  - **functional model**
    - represented by use cases and scenarios
  - **analysis object model**
    - represented by class and object diagrams
  - **dynamic model**
    - represented by state machine and sequence diagrams

Development activities

- **System design (A3)** is the transformation of an analysis model into a system design model
- Results
  - ***design goals***, describing the qualities of the system to optimize
  - ***software architecture***, describing the subsystem decomposition, dependencies among subsystems, subsystem mapping to hardware, control flow, access control, and data storage
  - ***boundary use cases***, describing the system configuration, start-up, shutdown, and exception handling issues

Development activities

- **System design (A3)**
  - ***design goals***, describing the qualities of the system to optimize
    - Derived from non-functional requirements
    - Trade-offs are commonly needed
  - ***boundary*** *use cases*
    - Configuration
      - may add some use cases to create objects
    - Start-up and shutdown
      - for example starting and shutting down a Web server
    - Exception handling
      - how should system handle component failures

Development activities

- **System design (A3)**
- ***Software architecture***: describing the subsystem decomposition, dependencies among subsystems, subsystem mapping to hardware, control flow, access control, and data storage
  - Divide the system into manageable pieces to deal with complexity: each subsystem is assigned to a team and realized independently
  - Address system-wide issues when decomposing the system
  - A ***subsystem*** is a replaceable part of the system with well-defined interfaces that encapsulates the state and behavior of its contained classes.

Development activities

- **System design (A3)**
  - *Coupling*
    - The number of dependencies between two subsystems
  - *Cohesion*
    - The number of dependencies within a subsystem

Development activities

System design (3)

Architecture styles:

- **Repository:** subsystems access and modify a single data structure - central **repository**.
- **Model/View/Controller** (MVC): subsystems are classified into:
  - **model** subsystems maintain domain knowledge,
  - **view** subsystems display it to the user, and
  - **controller** subsystems manage the sequence of interactions with the user.

Development activities

More architecture styles:

- **Client-server:** The server provides services to the clients
- ***Three-tier*** (interface/presentation, application logic, storage)
- **Peer-to-peer:** subsystems can act as servers and as clients
- ***Pipe and filter -*** subsystems process data received from a set of inputs and send results to other subsystems via a set of outputs

Development activities

- **Object Design (A4)**
- Includes:
  - ***Reuse* -** off-the-shelf components, class libraries, design patterns are adapted
  - ***Interface specification* -** a complete interface specification for each subsystem
  - ***Restructuring*** – manipulate the system to increase reuse
  - ***Optimization* -** address performance requirements

Development activities

- **Implementation (A5)** - realize the system by translating the solution domain model into an executable representation (source code)
- Mapping concepts:
  - *Model transformations*
    - operate on object models (an attribute to a class)
  - *Refactorings*
    - operate on source code, improve a single aspect of the system without changing its functionality
  - *Forward engineering*
    - produces a source code template that corresponds to an object model
  - *Reverse engineering*
    - produces a model that corresponds to source code

Development activities

- **Testing** (A6) - find differences between the system and its models by executing the system or parts of it. Testing examples:
  - Unit testing
  - Integration testing
  - Acceptance testing
  - Reliability testing
  - …
- **Test case**
  - A set of input data and expected results that exercises a component with the purpose of causing failures and detecting faults

Development activities

What we do in SE2250:

- How to write specifications [little]
- How to write test plans [little]
- How to design [little]
- How to implement [lot]
- How to manage implementation [moderate]
- How to test [moderate]

Software development process

- A **software development process** is the process of dividing software development work into distinct phases to improve design, product management, and project management.

- A general model of the software development process is called a **software life cycle**.

Software processes in IEEE 1074 standard

Process group	Processes
Life Cycle Modeling	Selection of a Life Cycle Model
Project Management	Project Initiation Project Monitoring and Control Software Quality Management
Pre-Development	Concept Exploration System Allocation
Development	Requirements Design Implementation
Post-Development	Installation Operation and Support Maintenance Retirement
Integral	Verification and Validation Software Configuration Management Documentation Development Training

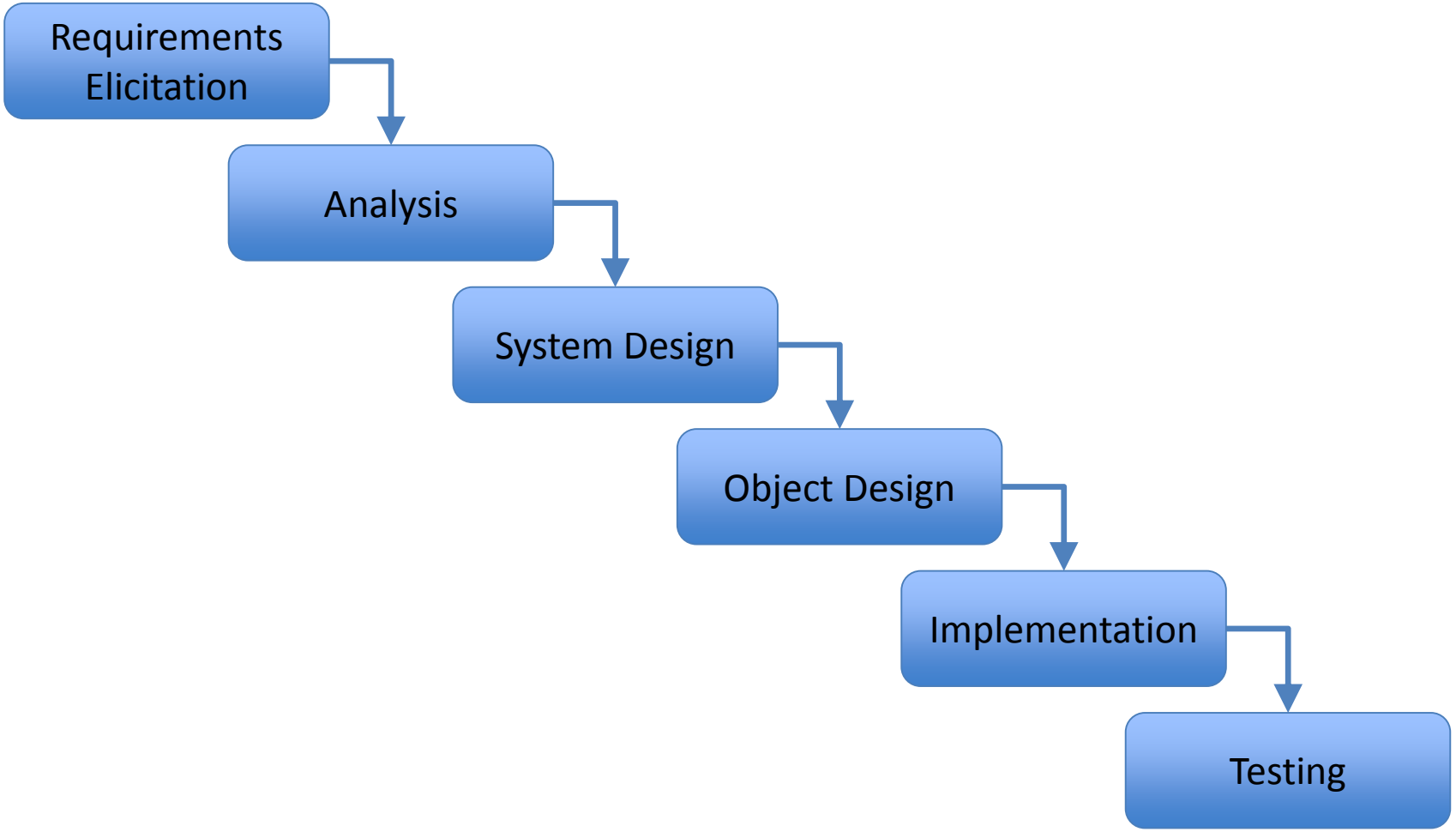
Software development process

- **The waterfall model**
  - Plan-driven model. Separate and distinct phases of specification and development.
- **Incremental development**
  - Specification, development and validation are interleaved. May be plan-driven or agile.
- **Integration and configuration**
  - The system is assembled from existing configurable components. May be plan-driven or agile.

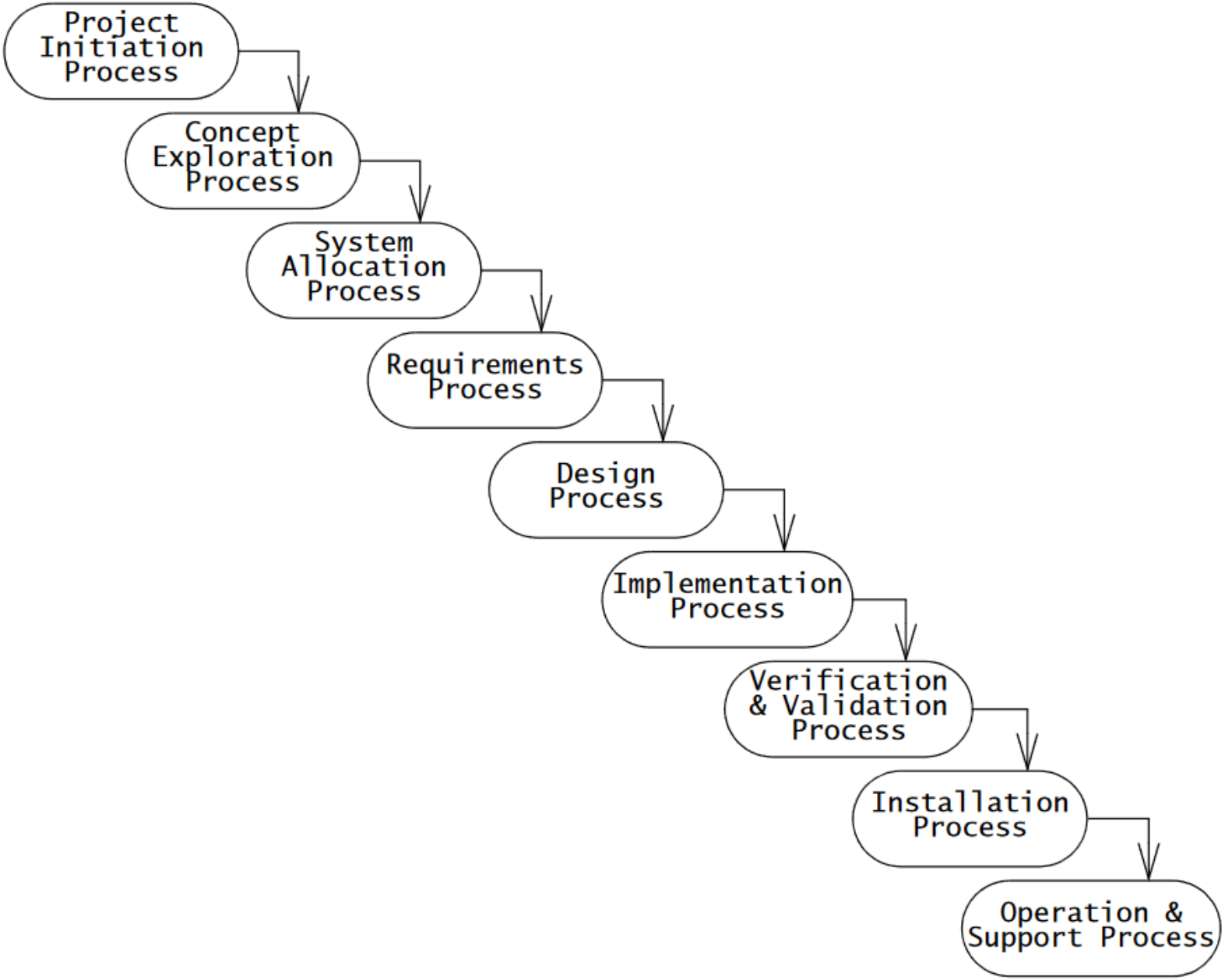
Waterfall

- Activity-centered
- Oldest
- Software development activities are performed in sequence

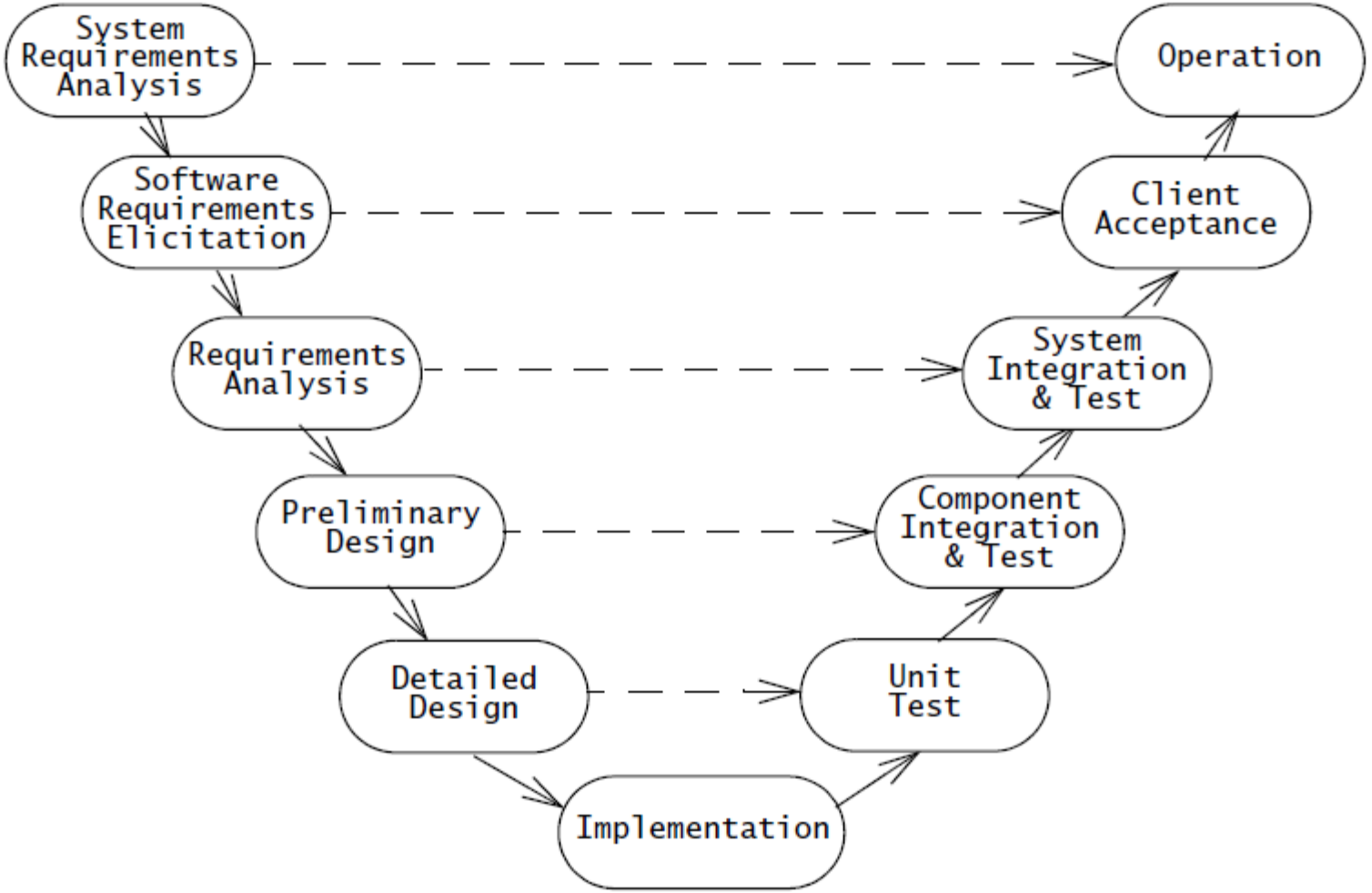
Waterfall


Requirements Elicitation
Analysis
System Design
Object Design
Implementation
Testing

Waterfall using IEEE 1074 names



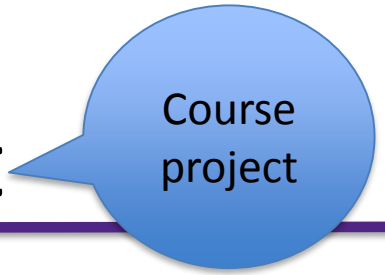
V-model (waterfall variation)



Waterfall problems

- Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.
- The waterfall model is mostly used for large systems engineering projects where a system is developed at several sites.

Incremental development

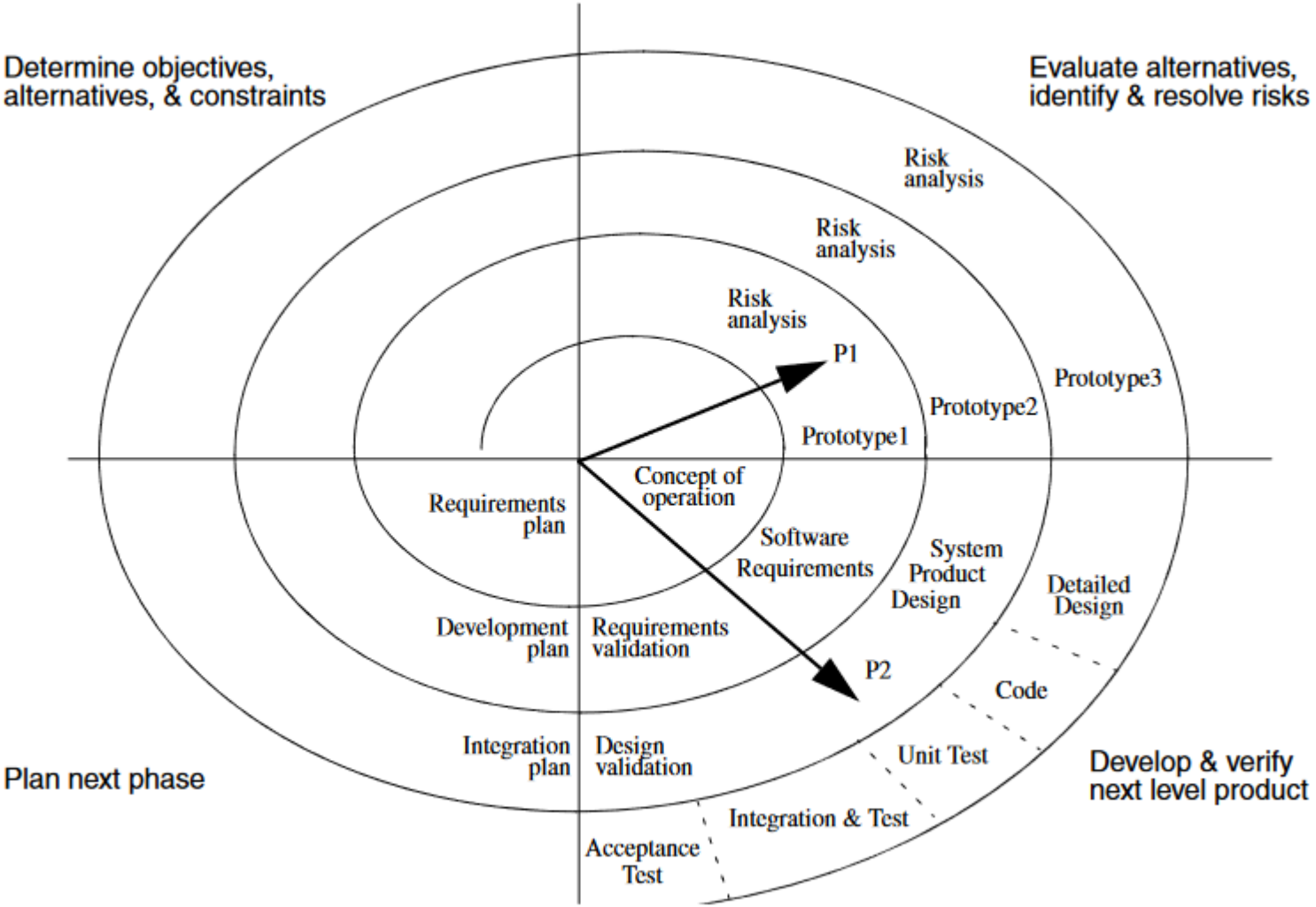


- Specification, development and validation are interleaved. May be plan-driven or agile.
- The cost of accommodating changing customer requirements is reduced.
- It is easier to get customer feedback on the development work that has been done.
- The process is not as visible as in the waterfall.
- System structure tends to degrade as new increments are added.

Spiral (Incremental development)

- Activity-centered
- Each round follows the waterfall model
- Subsequent rounds are represented as additional layers on the spiral.
- The distance from the origin is the cost accumulated by the project.
- The angular coordinate indicates the progress accomplished within each phase.

Spiral


Determine objectives, alternatives, & constraints
Evaluate alternatives, identify & resolve risks
Risk analysis
Risk analysis
Risk analysis
P1
Prototype3
Prototype2
Prototype1
Concept of operation
Requirements plan
Software Requirements
System Product Design
Detailed Design
Development plan
Requirements validation
P2
Code
Unit Test
Integration plan
Design validation
Integration & Test
Acceptance Test
Plan next phase
Develop & verify next level product

Agile development

- Refers to a group of software development methodologies based on iterative development
- Program specification, design and implementation are inter-leaved
- The system is developed as a series of versions or increments with stakeholders involved in version specification and evaluation
- Frequent delivery of new versions for evaluation
- Extensive tool support (e.g. automated testing tools) used to support development.
- Minimal documentation – focus on working code
- System structure tends to degrade as new increments are added.

Agile development

The Manifesto for Agile Software Development principles (only a few from 12):

- Customer satisfaction by early and continuous delivery of valuable software
- Welcome changing requirements, even in late development
- Continuous attention to technical excellence and good design
- Working software is the primary measure of progress

…

Agile development

Popular agile development frameworks include:

- Scrum
- Kanban
- Extreme programming (XP)
- Feature-driven development

Agile development

Practices:

- Pair programming
- Daily stand-up (daily scrum)
- Unit testing
- Test-driven development
- Continuous integration (CI)
- User stories for specification
- Velocity tracking
- Code refactoring
- Test automation
- …

Practical problems with agile development

- The informality of agile development is incompatible with the legal approach to contract definition that is commonly used in large companies.
- Agile methods are designed for small co-located teams yet much software development now involves worldwide distributed teams.
- Agile methods are most appropriate for new software development rather than software maintenance. Yet the majority of software costs in large companies come from maintaining their existing software systems.

Software development process

- Most large systems are developed using a process that incorporates elements from different models.