Chapter 3

Software Processes

Software Processes

- Coherent sets of activities for specifying, designing, implementing and testing software systems

Objectives

- To introduce software process models
- To describe a number of different process models and when they may be used
- To describe outline process models for requirements engineering, software development, testing and evolution
- To introduce CASE technology to support software process activities

Topics covered

- Software process models
- Process iteration
- Software specification
- Software design and implementation
- Software validation
- Software evolution
- Automated process support

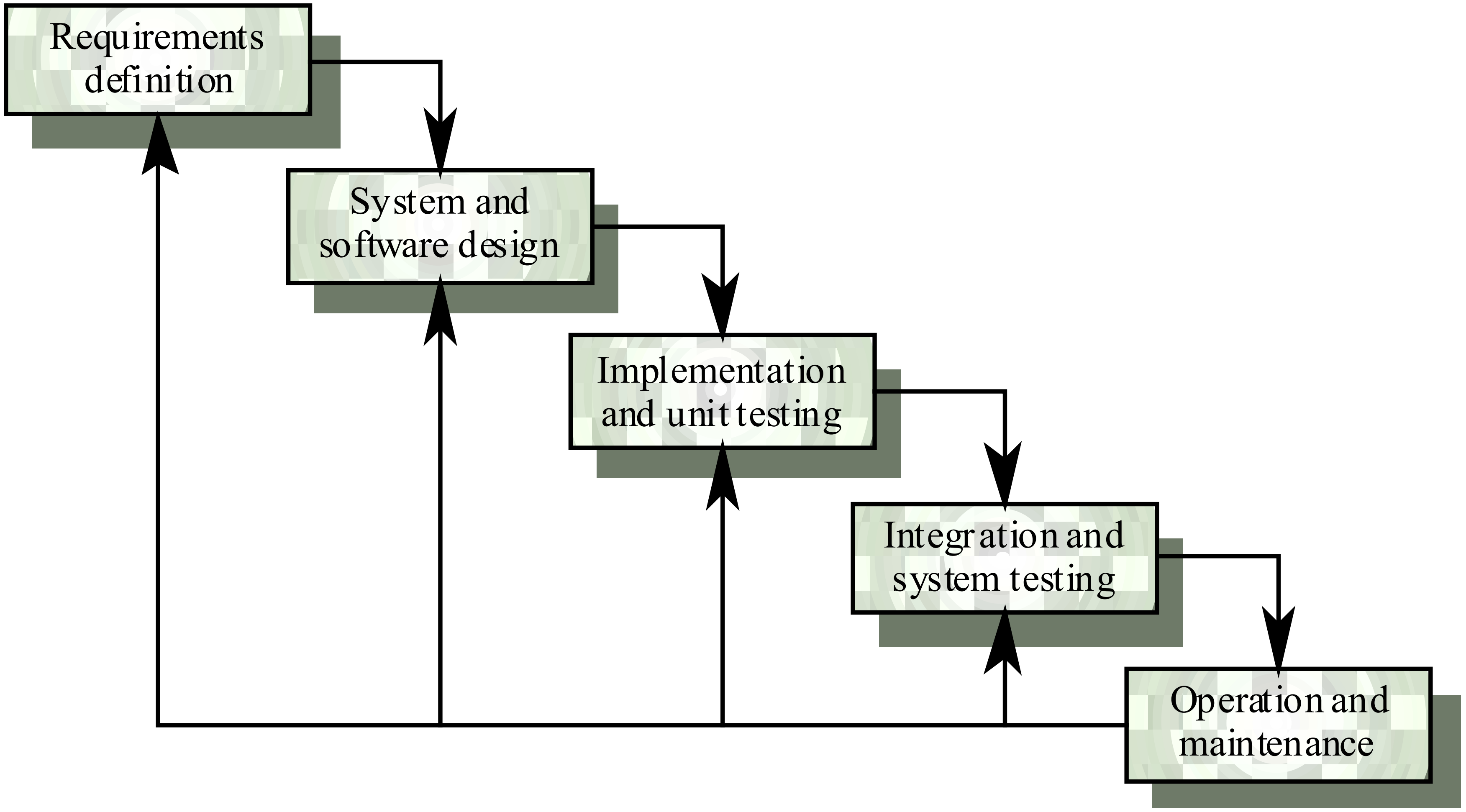
The software process

- A structured set of activities required to develop a software system
  - Specification
  - Design
  - Validation
  - Evolution
- A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective

Generic software process models

- The waterfall model and V model
  - Separate and distinct phases of specification and development
- Evolutionary development
  - Specification and development are interleaved
- Formal systems development
  - A mathematical system model is formally transformed to an implementation
- Reuse-based development
  - The system is assembled from existing components

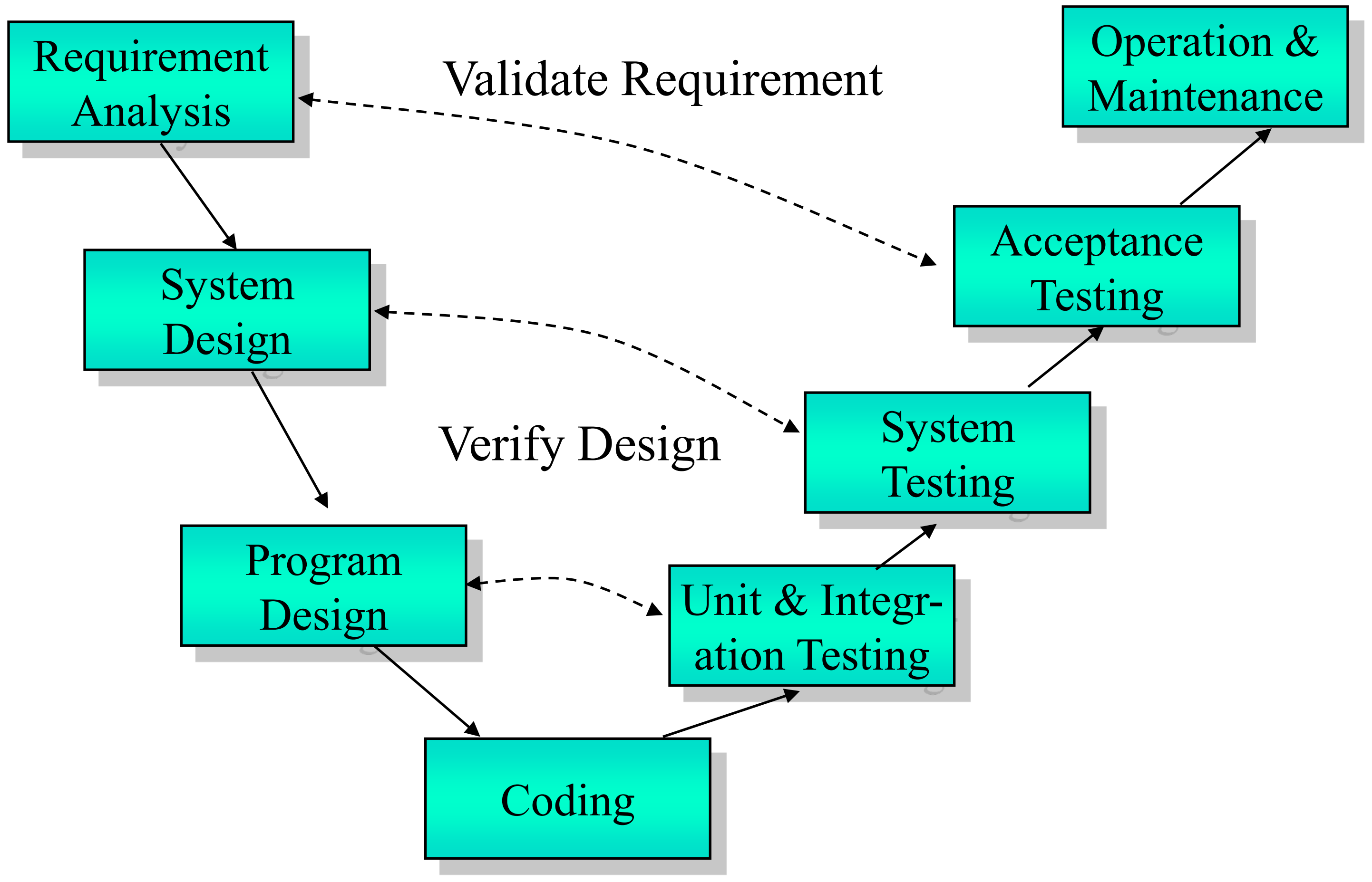
Waterfall model


Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

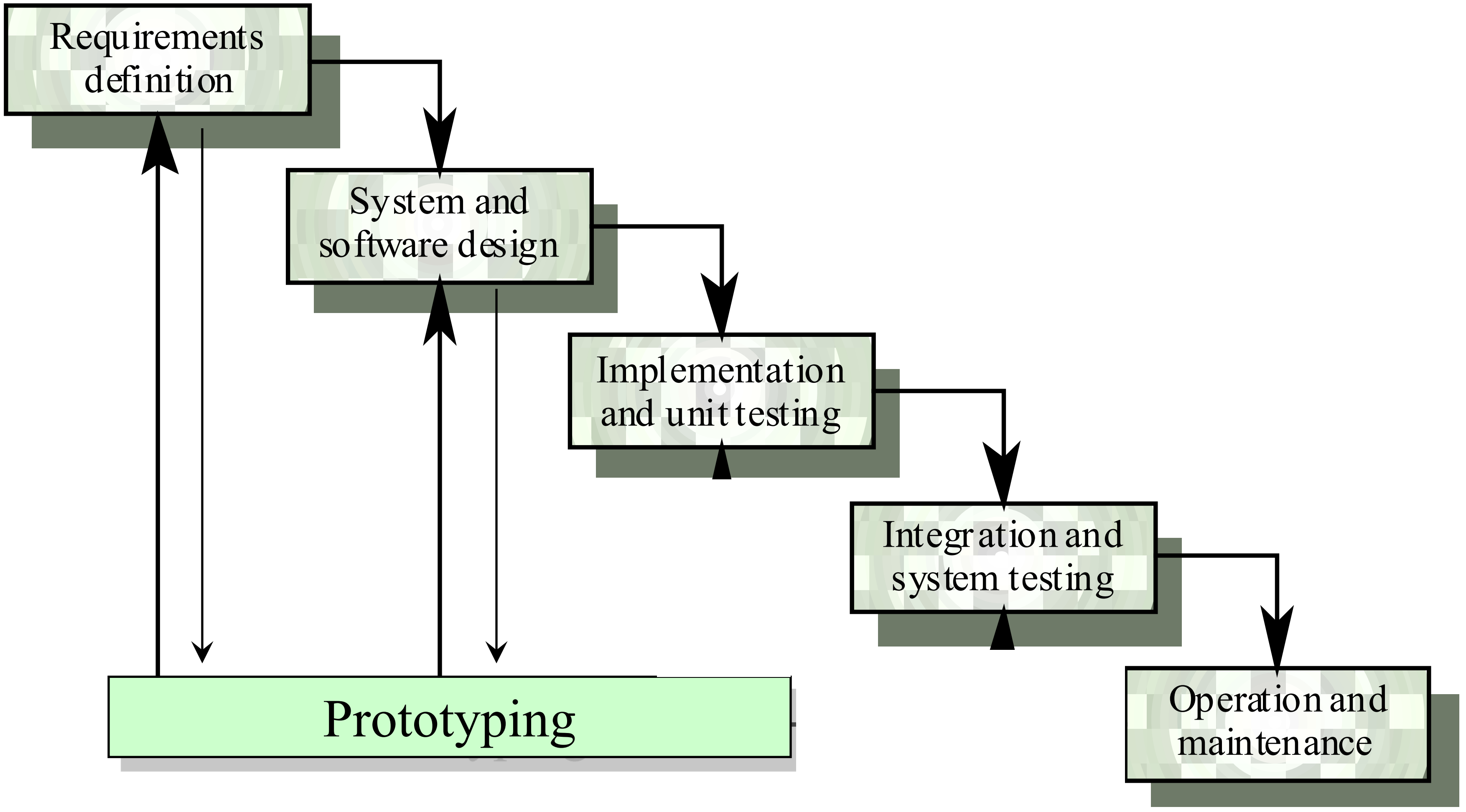
Waterfall model problems

- Inflexible partitioning of the project into distinct stages
- Difficult to accommodate changing customer requirements after the process is underway
- Appropriate only when the requirements are well-understood

V Model


Requirement Analysis
Validate Requirement
Operation & Maintenance
System Design
Acceptance Testing
Verify Design
System Testing
Program Design
Unit & Integr- ation Testing
Coding

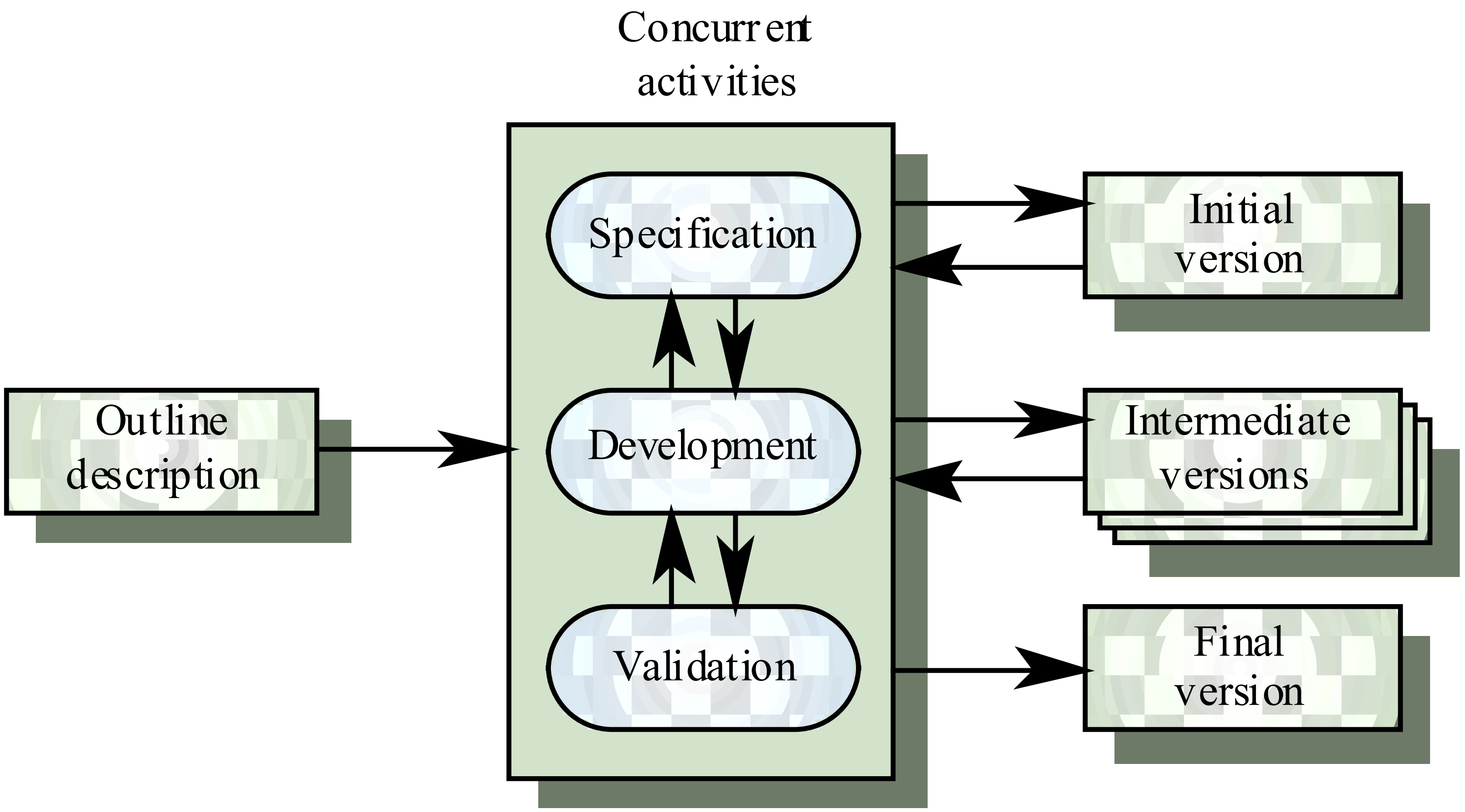
Waterfall model


Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance
Prototyping

Evolutionary development

- Exploratory development
  - Objective is to work with customers and to evolve a final system from an initial outline specification. Should start with well-understood requirements
- Throw-away prototyping
  - Objective is to understand the system requirements. Should start with poorly understood requirements

Evolutionary development



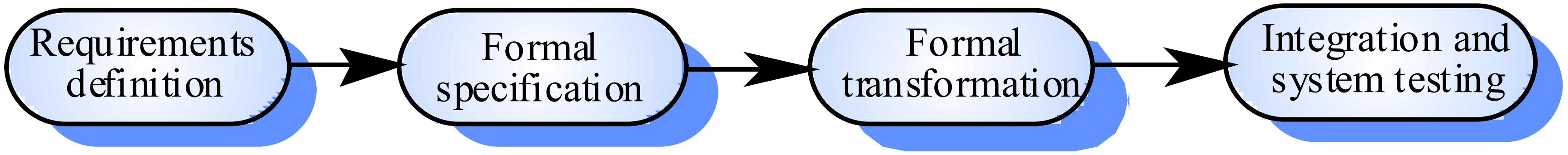
Evolutionary development

- Problems
  - Lack of process visibility
  - Systems are often poorly structured
  - Special skills (e.g. in languages for rapid prototyping) may be required
- Applicability
  - For small or medium-size interactive systems
  - For parts of large systems (e.g. the user interface)
  - For short-lifetime systems

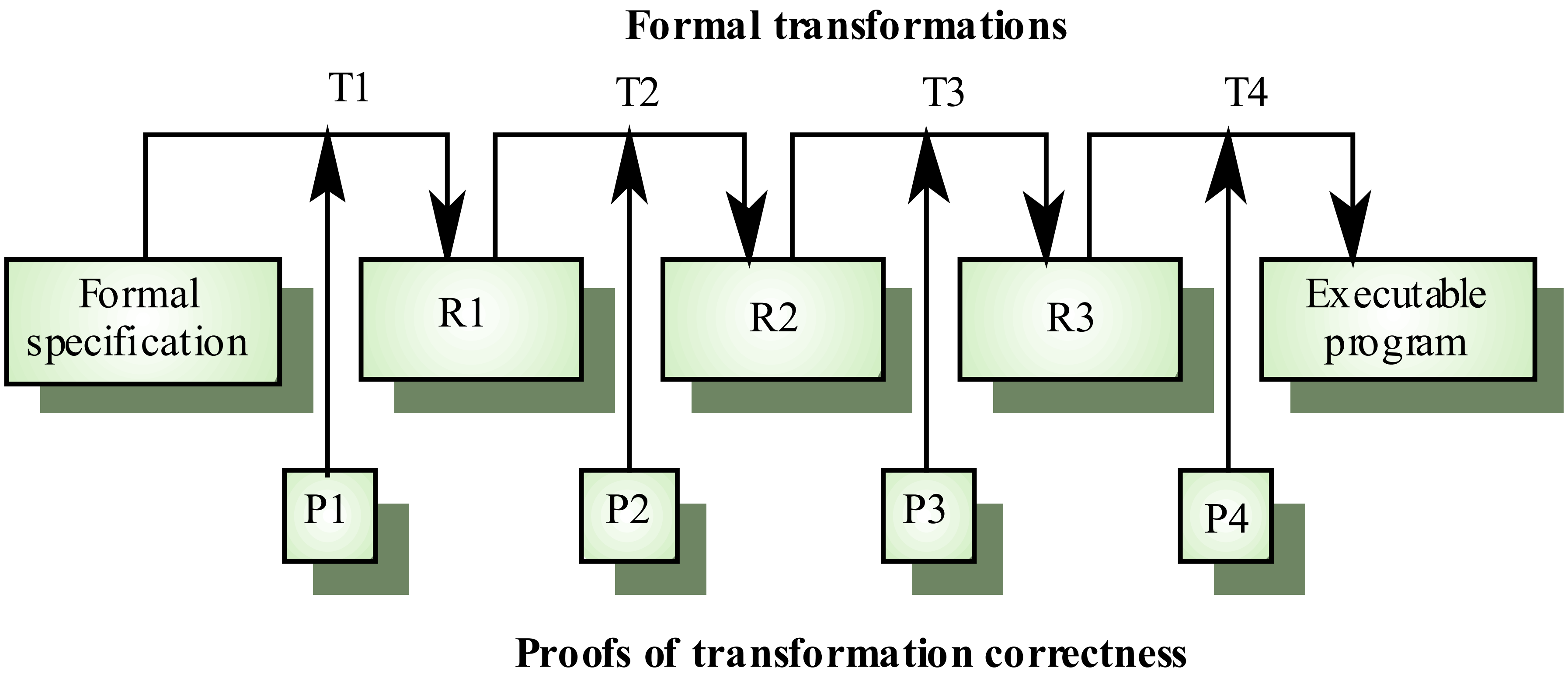
Formal systems development

- Based on the transformation of a mathematical specification through different representations to an executable program
- Transformations are ‘correctness-preserving’ so it is straightforward to show that the program conforms to its specification
- Embodied in the ‘Cleanroom’ approach to software development

Formal systems development



Formal transformations



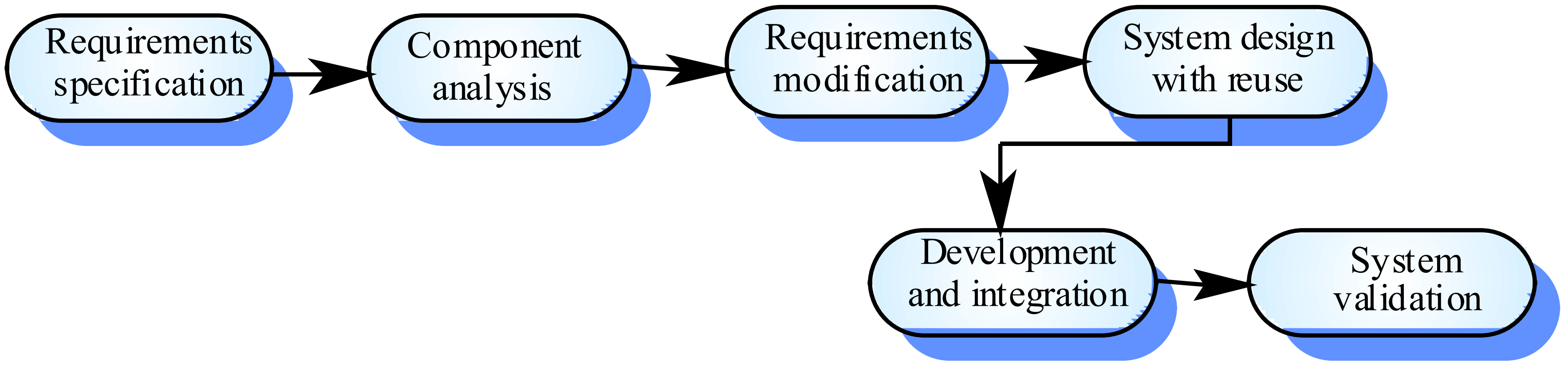
Formal systems development

- Problems
  - Need for specialised skills and training to apply the technique
  - Difficult to formally specify some aspects of the system such as the user interface
- Applicability
  - Critical systems especially those where a safety or security case must be made before the system is put into operation

Reuse-oriented development

- Based on systematic reuse where systems are integrated from existing components or COTS (Commercial-off-the-shelf) systems
- Process stages
  - Component analysis
  - Requirements modification
  - System design with reuse
  - Development and integration
- This approach is becoming more important but still limited experience with it

Reuse-oriented development


Requirements specification
Component analysis
Requirements modification
System design with reuse
Development and integration
System validation

Topics covered

- Software process models
- Process iteration
- Software specification
- Software design and implementation
- Software validation
- Software evolution
- Automated process support

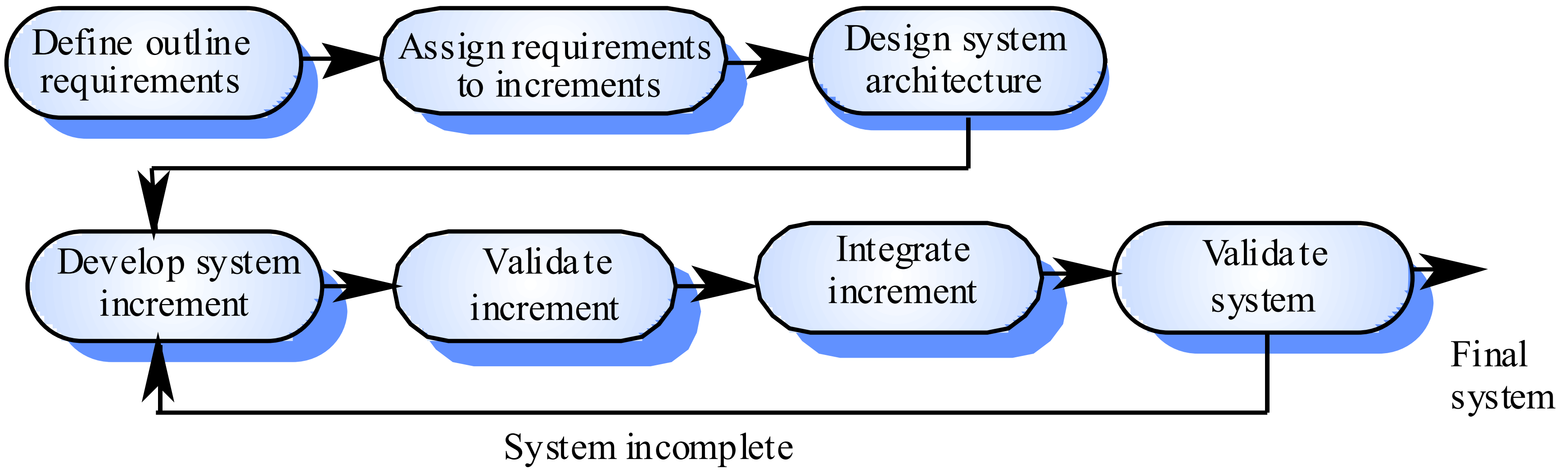
Process iteration

- Iteration means earlier stages are reworked in the process for large systems
- Iteration can be applied to any of the generic process models
- Two (related) approaches
  - Incremental development
  - Spiral development

Incremental development

- System development is decomposed into increments and each delivers a proportion of the system.
- Increments are developed based on their requirement priorities.
- When the development of an increment is started, its requirement is fixed until the development of the next increment.

Incremental development



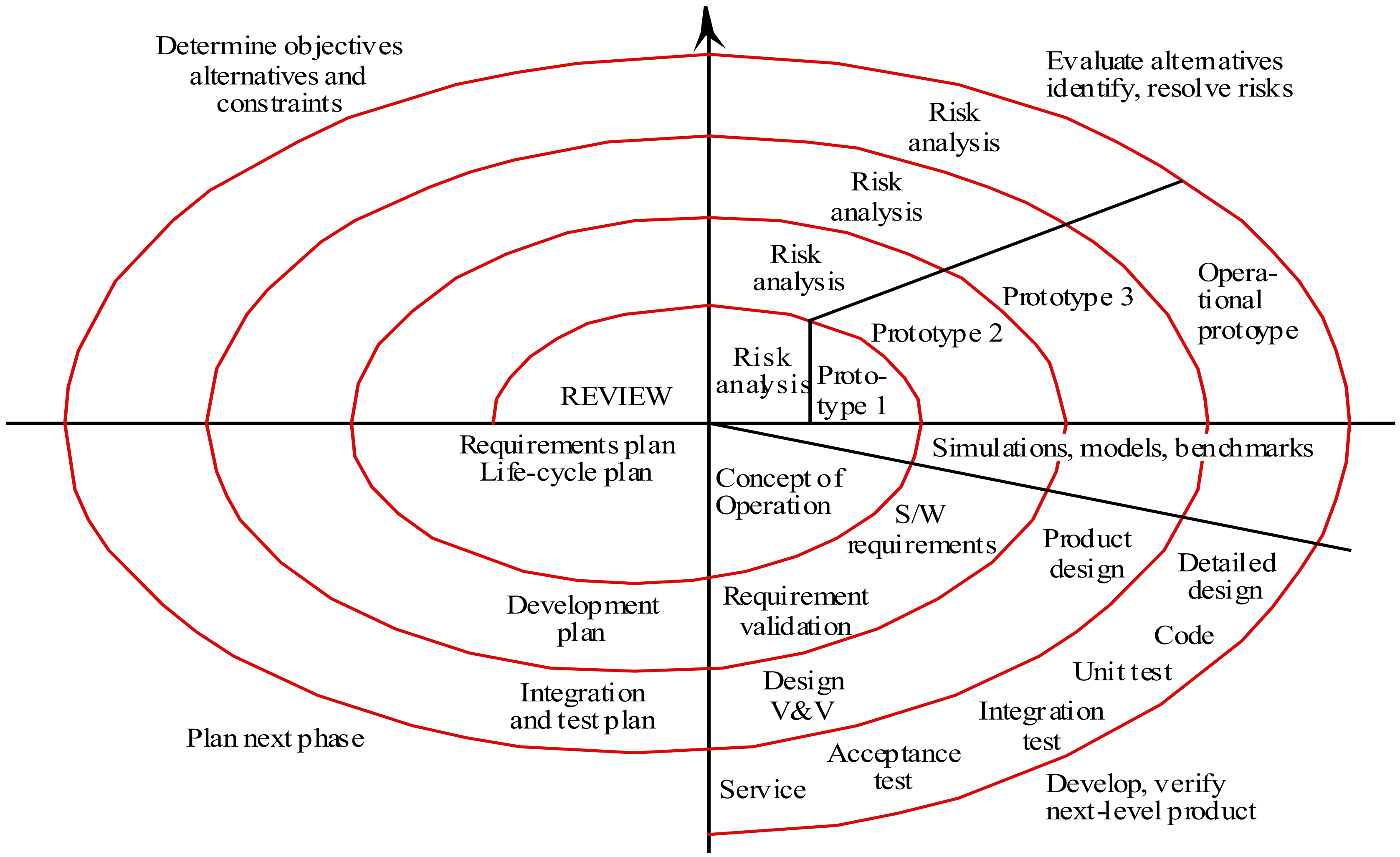
Incremental development advantages

- Some system functionalities are available earlier
- Early increments help elicit requirements for later increments
- Lower risk of overall project failure
- The high priority system services receive more testing

Spiral model sectors

- Objective setting
  - Specific objectives for the phase are identified
- Risk assessment and reduction
  - Risks are assessed and activities put in place to reduce the key risks
- Development and validation
  - A development model for the system is chosen which can be any of the generic models
- Planning
  - The project is reviewed and the next phase of the spiral is planned

Spiral model of the software process



Spiral development

- Process is represented as a spiral
- Each loop in the spiral represents a phase in the process
- No fixed phases such as specification or design - loops in the spiral are chosen depending on what is required
- Risks are explicitly assessed and resolved throughout the process

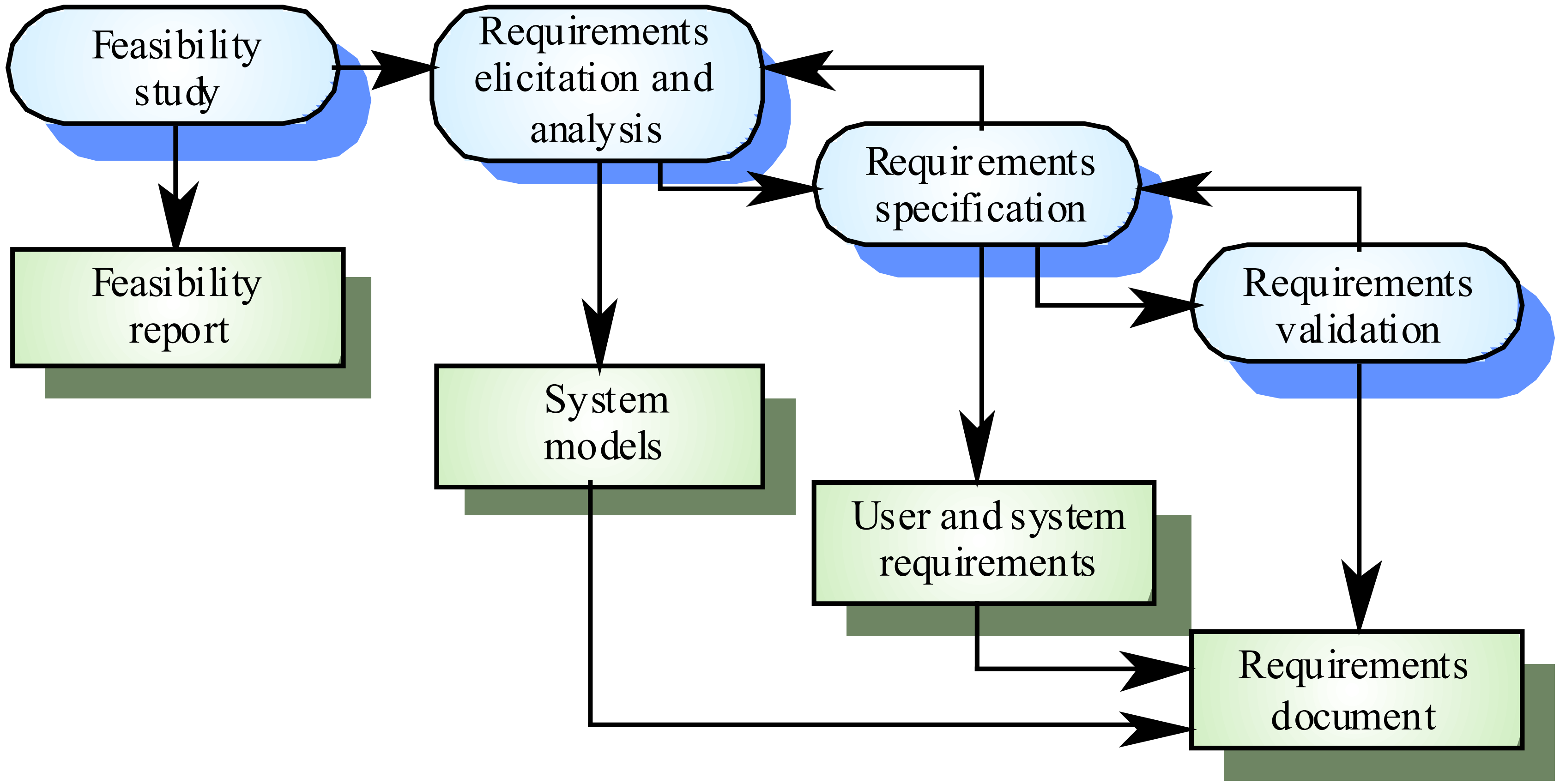
Topics covered

- Software process models
- Process iteration
- Software specification
- Software design and implementation
- Software validation
- Software evolution
- Automated process support

Software specification

- Define required services and constraints for system development
- Requirements engineering process (Ch. 6)
    - Feasibility study
    - Requirements elicitation and analysis
    - Requirements specification
    - Requirements validation

The requirements engineering process



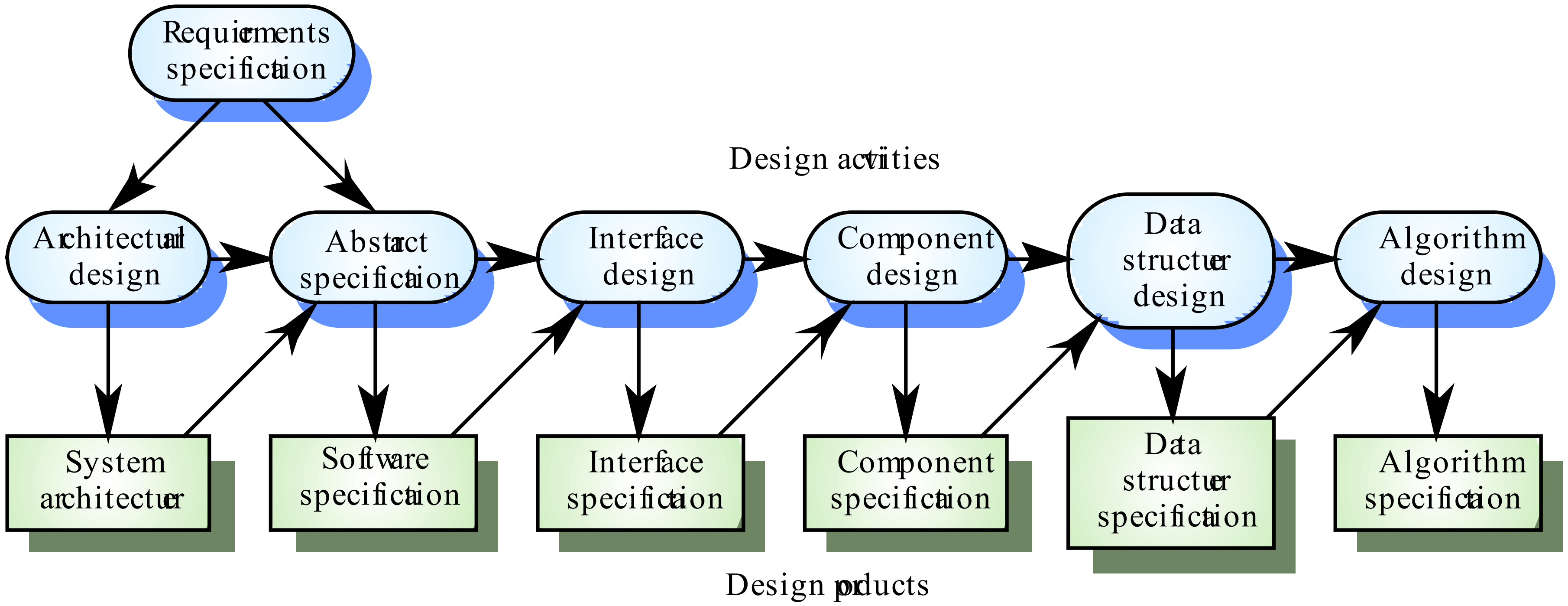
Topics covered

- Software process models
- Process iteration
- Software specification
- Software design and implementation
- Software validation
- Software evolution
- Automated process support

Software design and implementation

- The process of converting the system specification into an executable system
- Software design - design software structure
- Implementation - translate structure into an executable program
- The activities of design and implementation can be interleaved

Design process activities



Design methods

- Systematic approaches to developing a software design (Ch. 7)
  - Data-flow model
  - Entity-relation-attribute model
  - Structural model
  - Object models

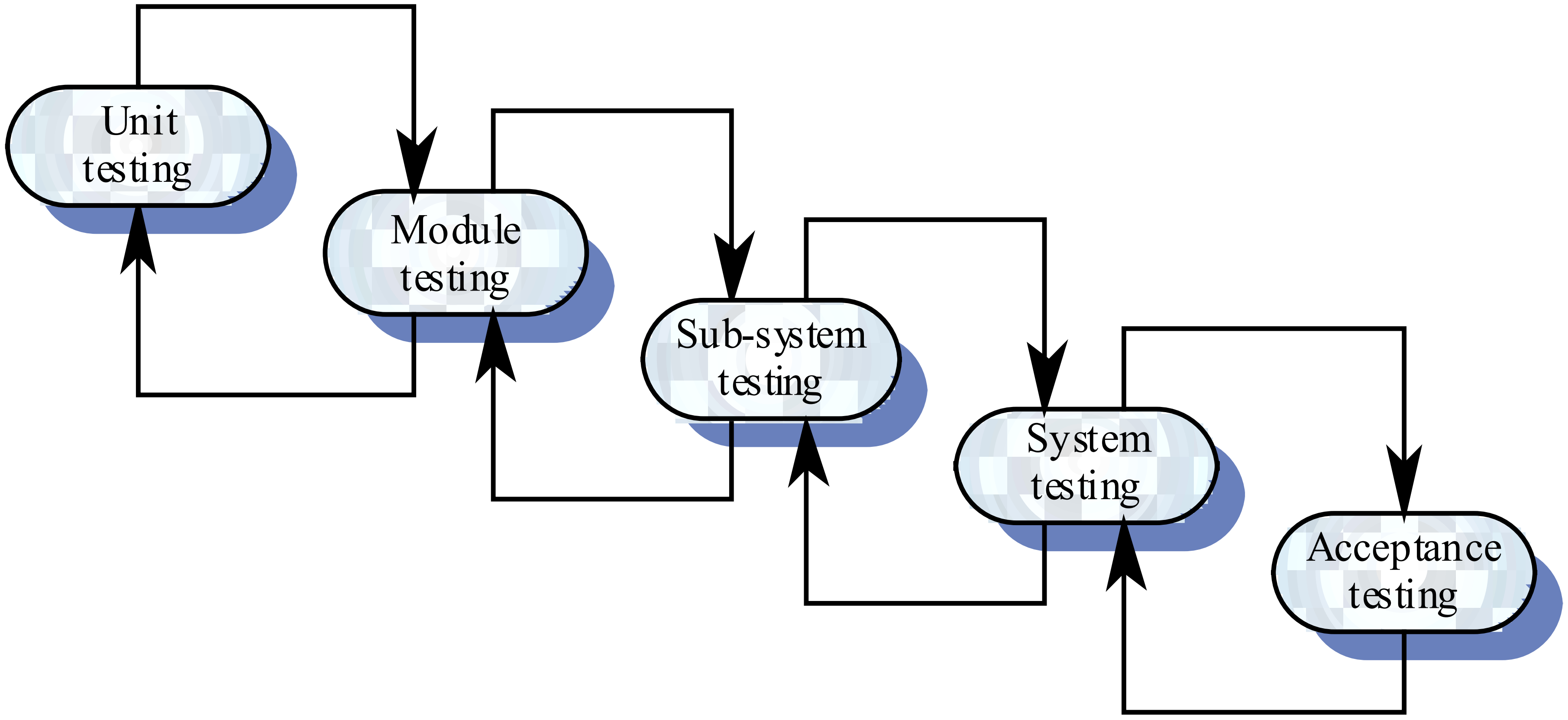
Topics covered

- Software process models
- Process iteration
- Software specification
- Software design and implementation
- Software validation
- Software evolution
- Automated process support

Software validation

- Validate user requirements and verify designs (Ch. 8, 20)
- Review processes and test system
  - Testing is to execute system with test cases that are derived from the specification, or real user data

The testing process



Component testing

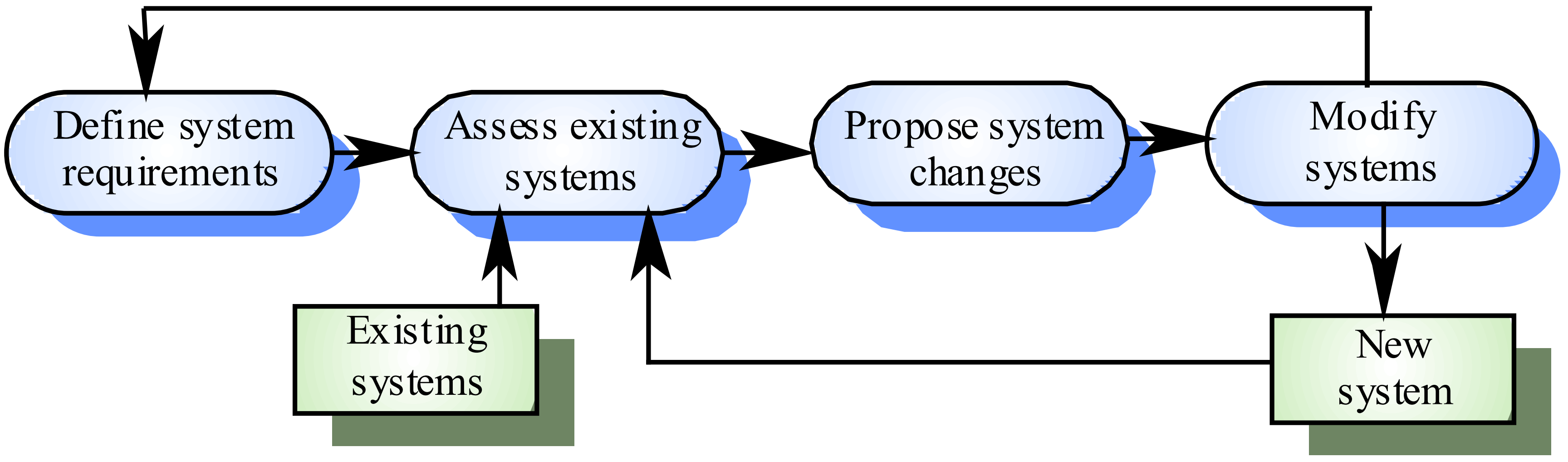
Integration testing

User testing

Topics covered

- Software process models
- Process iteration
- Software specification
- Software design and implementation
- Software validation
- Software evolution
- Automated process support

System evolution



Topics covered

- Software process models
- Process iteration
- Software specification
- Software design and implementation
- Software validation
- Software evolution
- Automated process support

Automated process support (CASE)

- CASE is software to support software development and evolution processes
- Activity automation
  - Graphical editors for system model development
  - Graphical UI builder for user interface construction
  - Debuggers to support program fault finding
  - :
  - :

CASE classification

- Functional perspective
  - Tools are classified according to their specific function (*Editing, Planning, etc.*)
- Process perspective
  - Tools are classified according to process activities that are supported (*Design, Prototyping, Testing, etc.*)
- Integration perspective
  - Tools are classified according to their organization into integrated units (*Version management, system building tools*)

Key points

- Software processes are the activities involved in producing and evolving a software system. They are represented in a software process model
- General activities are specification, design and implementation, validation and evolution
- Generic process models describe the organisation of software processes
- Iterative process models describe the software process as a cycle of activities

Key points

- Requirements engineering is the process of developing a software specification
- Design and implementation processes transform the specification to an executable program
- Validation involves checking that the system meets to its specification and user needs
- Evolution is concerned with modifying the system after it is in use
- CASE technology supports software process activities