CS5030

Software Lifecycle

Learning outcomes

- On completing this lecture and associated reading, you should be able to
  - Understand the concepts of software processes and software process models or lifecycles
  - Describe 3 general software process models, their benefits and limitations

Life cycle

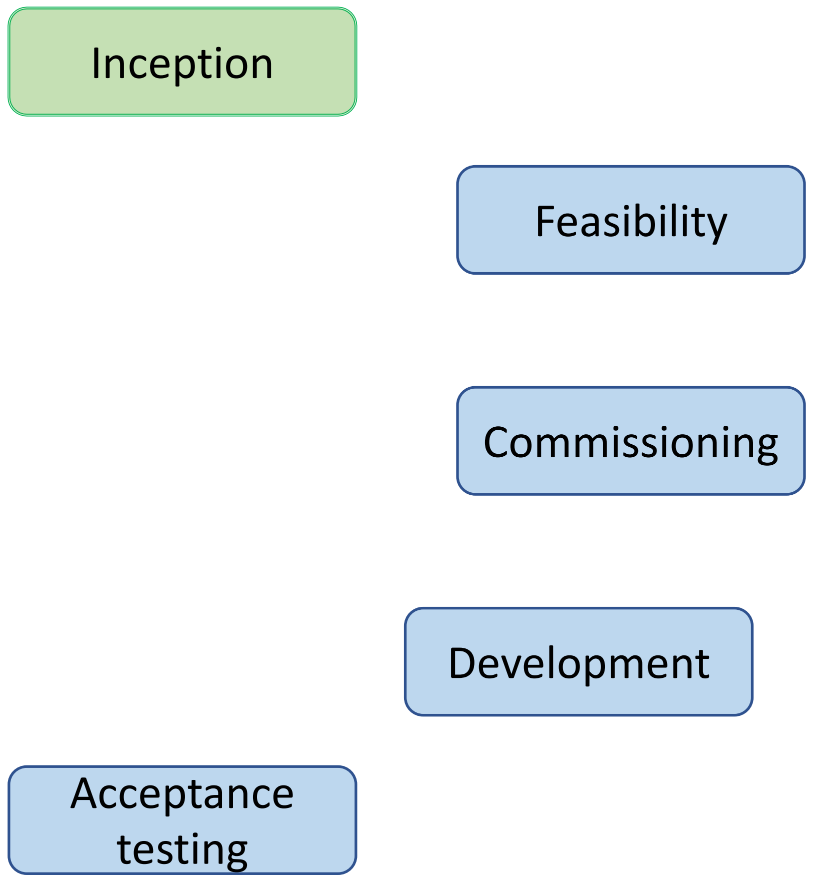
- Oxford English Dictionary

  . . .

  “In extended use:

  a course or evolution from a beginning, through development and productivity, to decay or ending.”

Software lifecycle



Software process

- A set of related activities that leads to the production of a software system [Sommerville, 2016]
- Different options
- Choice of software process
    - Characteristics of system being developed
    - Customer requirements
    - Skills and experience of development team
    - Organisational culture and practices

Software engineering activities

- Any software process will include, in some form,
  - Software specification
  - Software development
  - Software validation
  - Software evolution

- Each will have sub-activities

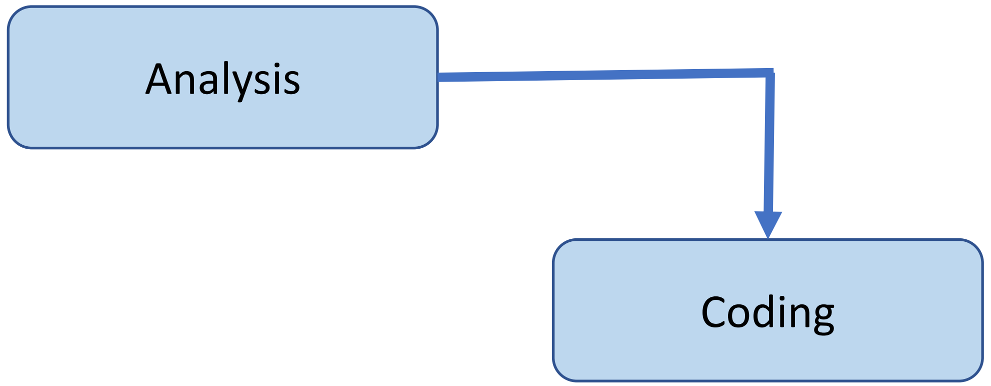
Software engineering activity

- Outcomes of activity
  - Products, deliverables – depend on chosen process
  - For eg, requirements specification, architecture model, code, test suite, documentation
- Roles of stakeholders
  - For eg, project manager, architect, programmer, tester
- Pre- and post-conditions
  - Must hold before and after an activity

Software lifecycle / process model

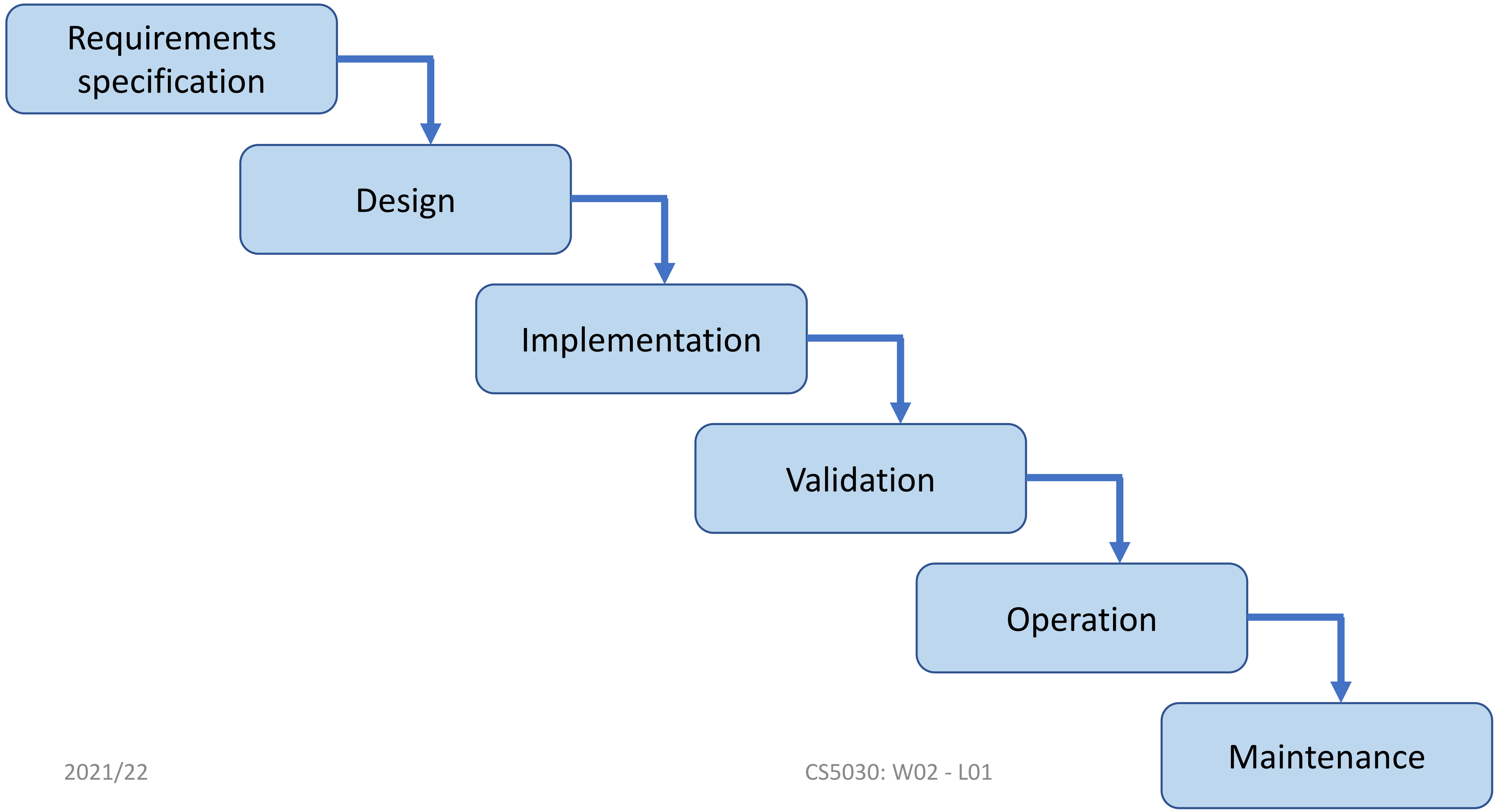
- An abstract representation of a process
  - What activity should be done next
  - How long it should last
- Examples
  - Waterfall
  - Incremental development
  - Integration and configuration

A minimal software process model

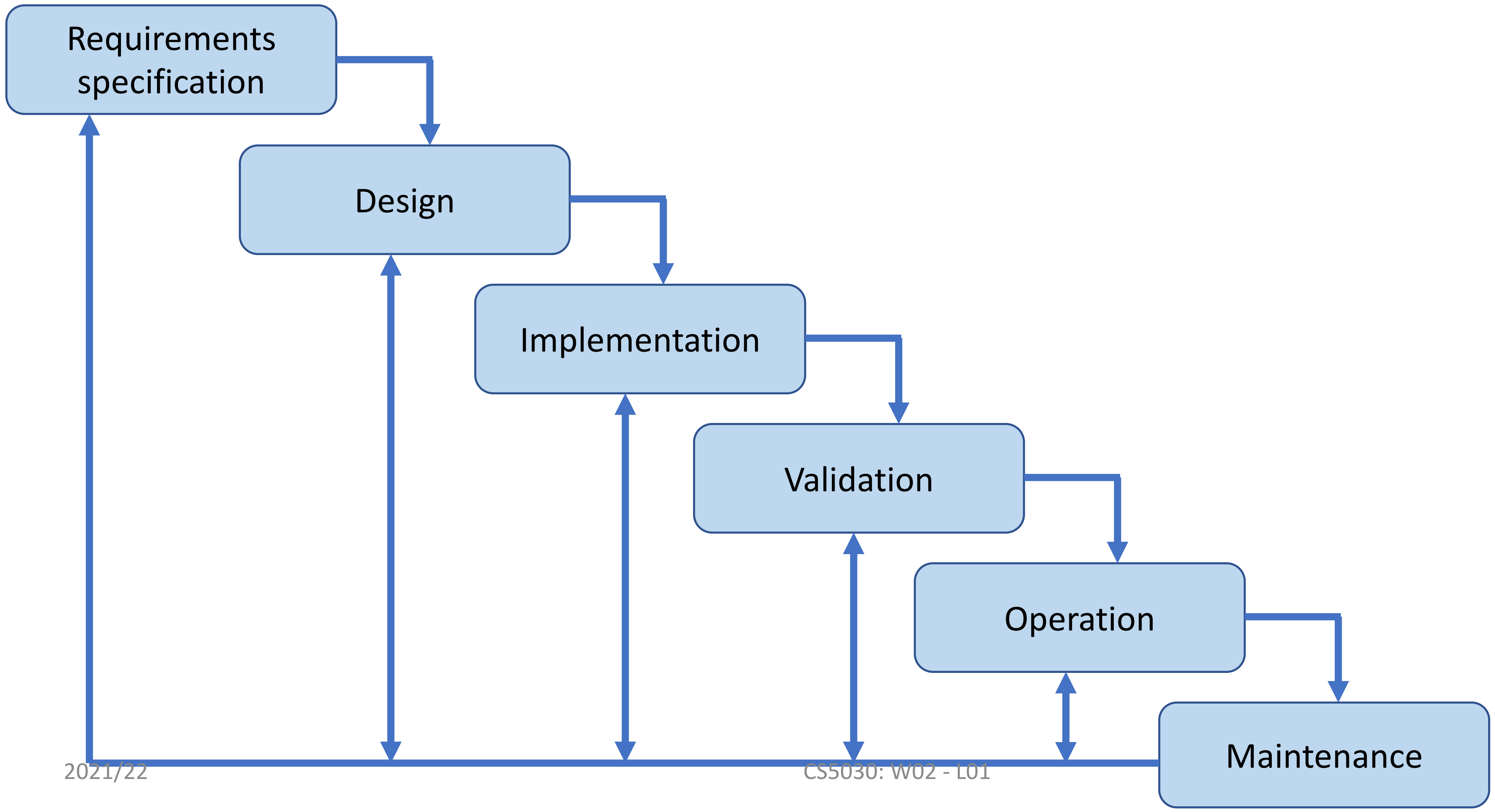


[Managing the Development of Large Software Systems, W Royce, 1970]

Waterfall model - original



Waterfall model - modified



Waterfall model – pros and cons

- Plan driven - lacking flexibility

- Can be used when
  - Requirements are well understood
  - Requirements are unlikely to change radically during development

- Possibly suitable for
  - Embedded systems
  - Critical systems
  - Collaborative development scenarios for large engineering projects

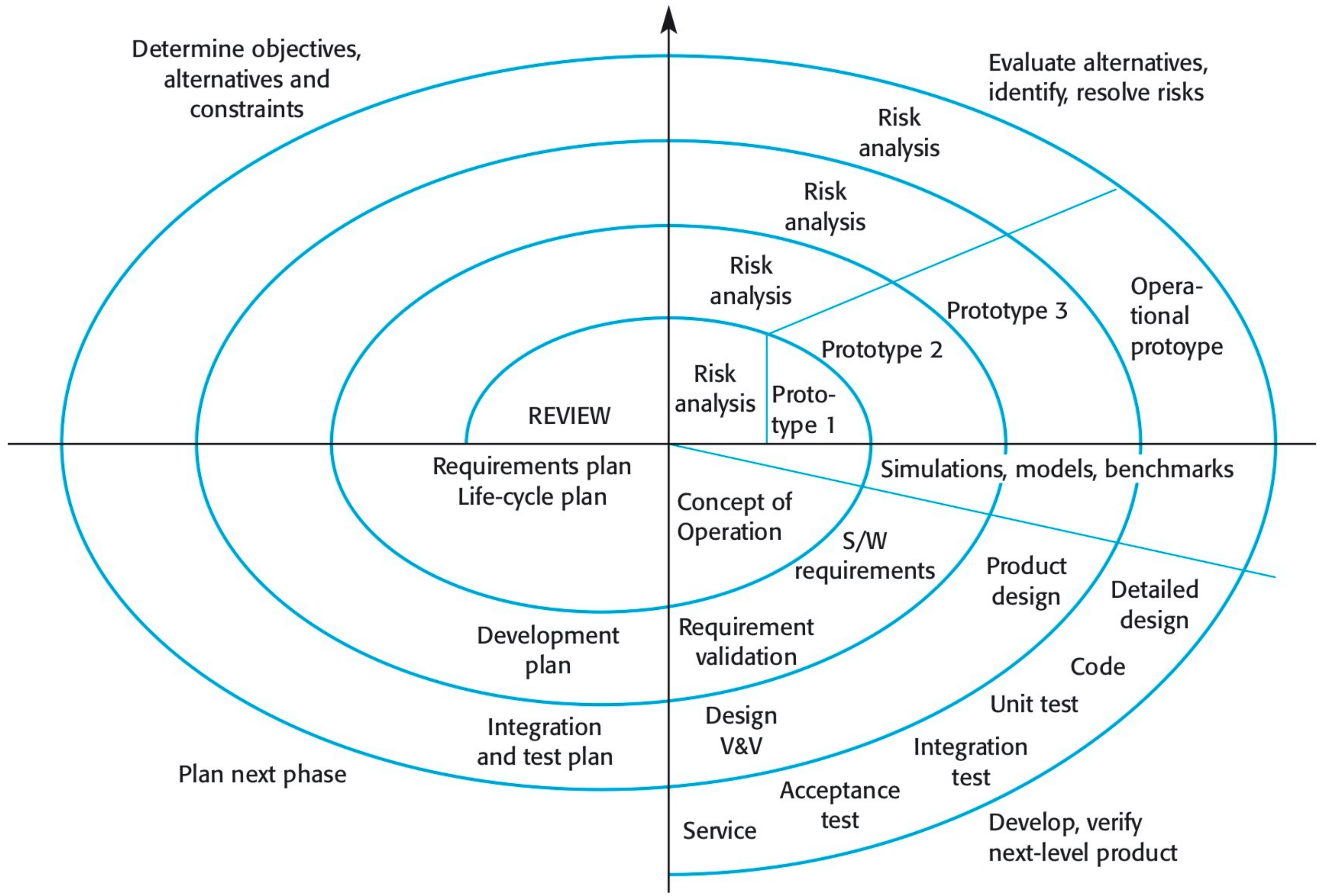
Spiral model

- Barry Boehm, 1988
- Incremental model
- Risk-driven software process framework
  - Changes are results of project risks
  - Process and product are determined by risks
- Aims to lower development costs by early elimination of alternatives that are not viable

Spiral model

- Each loop in the spiral
  - A phase of the process
  - Split into 4 sectors
    - Objective setting
    - Risk assessment and reduction
    - Development and validation
    - Planning for next loop

Spiral model

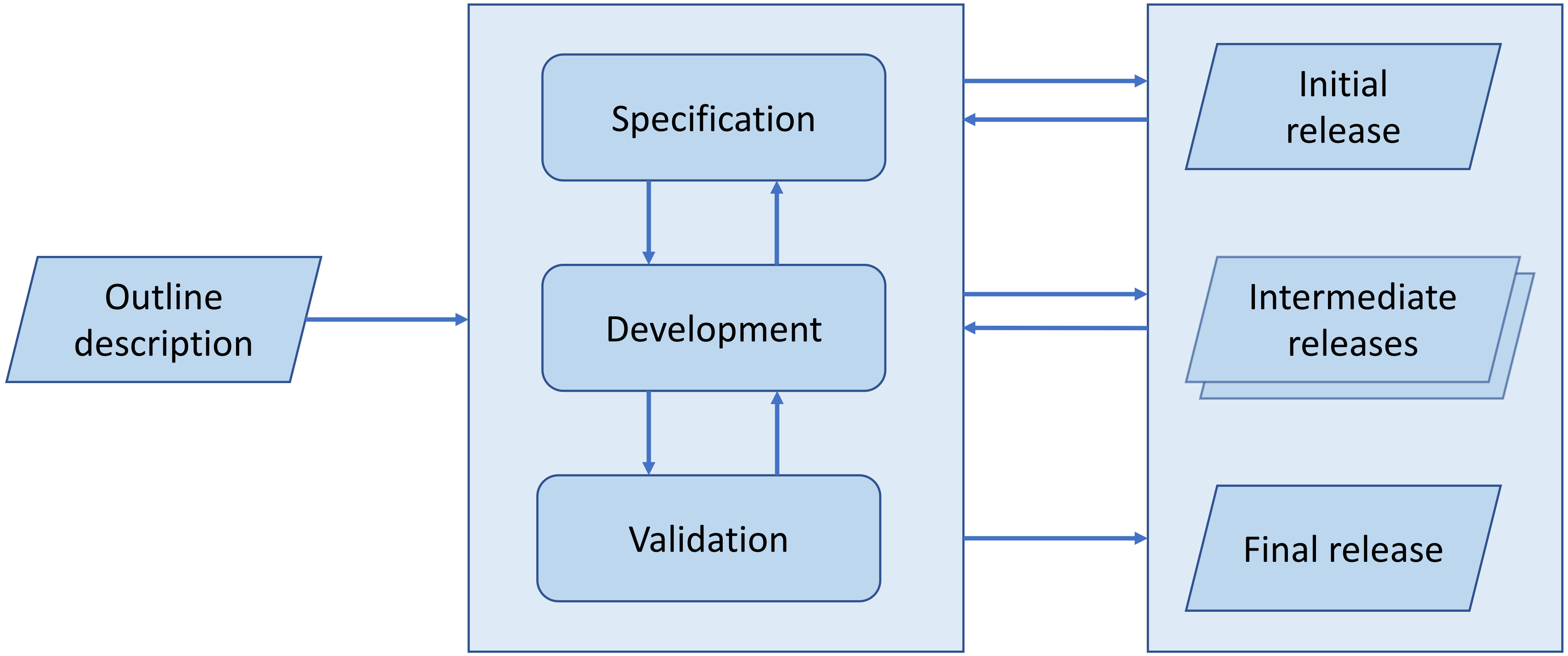


[Sommerville, 2016]

Iterative and incremental development (IID)

- Initial release of software, followed by refinements
    - Increments incorporating feedback from stakeholders
- Plan-driven or agile or a mix of both
- Basis for current development methods

Iterative and incremental development



[based on Sommerville, 2016]

IID – pros

- Quicker delivery and deployment of useful software to customers
- Earlier feedback from customers on work completed
- Reduced cost of accommodating changes to customer requirements
- Earlier identification of risks and mitigations

IID – cons

- Lack of visible process and complete documentation
- Degrading system structure as more increments are added
- Challenges in identifying common concerns and utilities that apply across the system
- Conflicts with procurement models of organisations without complete specification at the start

Integration and configuration

- Software reuse in common in projects

- Reuse-oriented approaches
  - Reusable software components
    - Examples
      - General-purpose stand-alone systems that can be configured
      - Web services satisfying service standards and can be remotely invoked
  - An integrating framework for composing components

Reuse-oriented software engineering

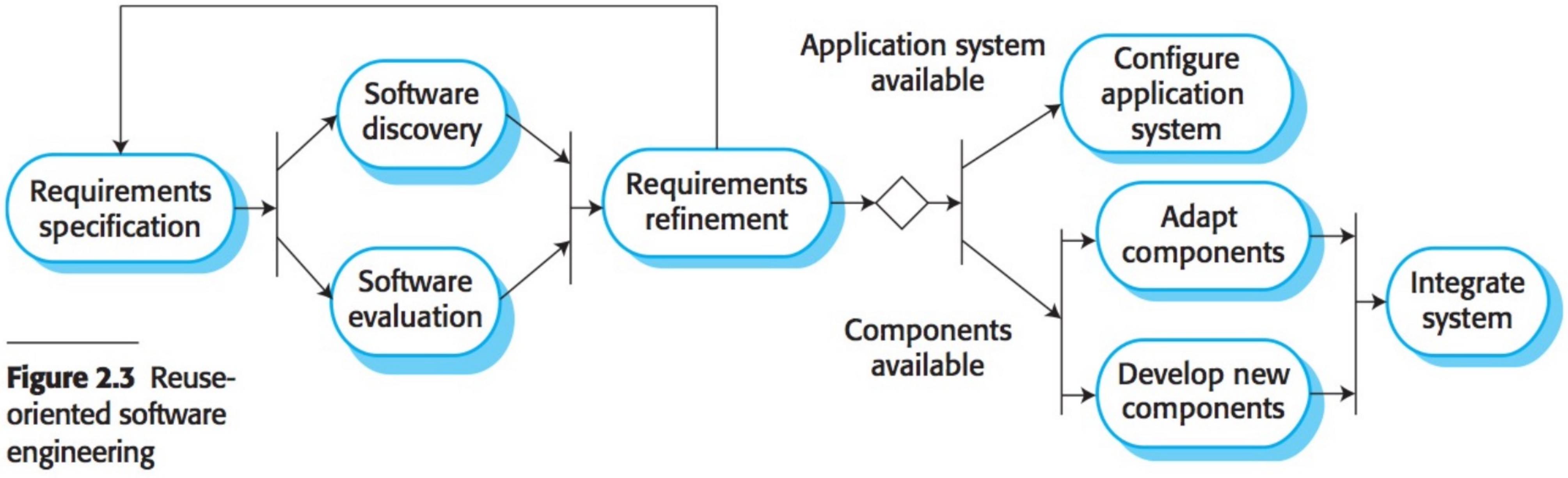


Figure 2.3 Reuse-oriented software engineering

[Sommerville, 2016]

Key points

- Software processes consist of activities that produce software
- A software process model or lifecycle is an abstract representation of software processes
- Processes can be plan-driven or iterative and incremental
- In practice, processes contain elements of both
- An appropriate software process should be chosen for a system based on the project and its context