UNIVERSITY of GUELPH

ENGINEERING

Software Processes

Fattane Zarrinkalam

Software Process

- A software process is a set of related activities that leads to the production of a software system.

- Many different software processes but all involve:
  - Specification – defining what the system should do;
  - Design and implementation – defining the organization of the system and implementing the system;
  - Validation – checking that it does what the customer wants;
  - Evolution – changing the system in response to changing customer needs.

Software Process

- Plan-driven process
  - Processes where all of the process activities are planned in advance and progress is measured against this plan.
  - Long lifetime, critical and embedded systems

- Agile process
  - In agile processes, planning is incremental, and it is easier to change the process to reflect changing customer requirements.
  - Software products and apps

In practice, most practical processes include elements of both plan-driven and agile approaches.
There are no right or wrong software processes.

Software Process Models

(Software Development Life Cycle)

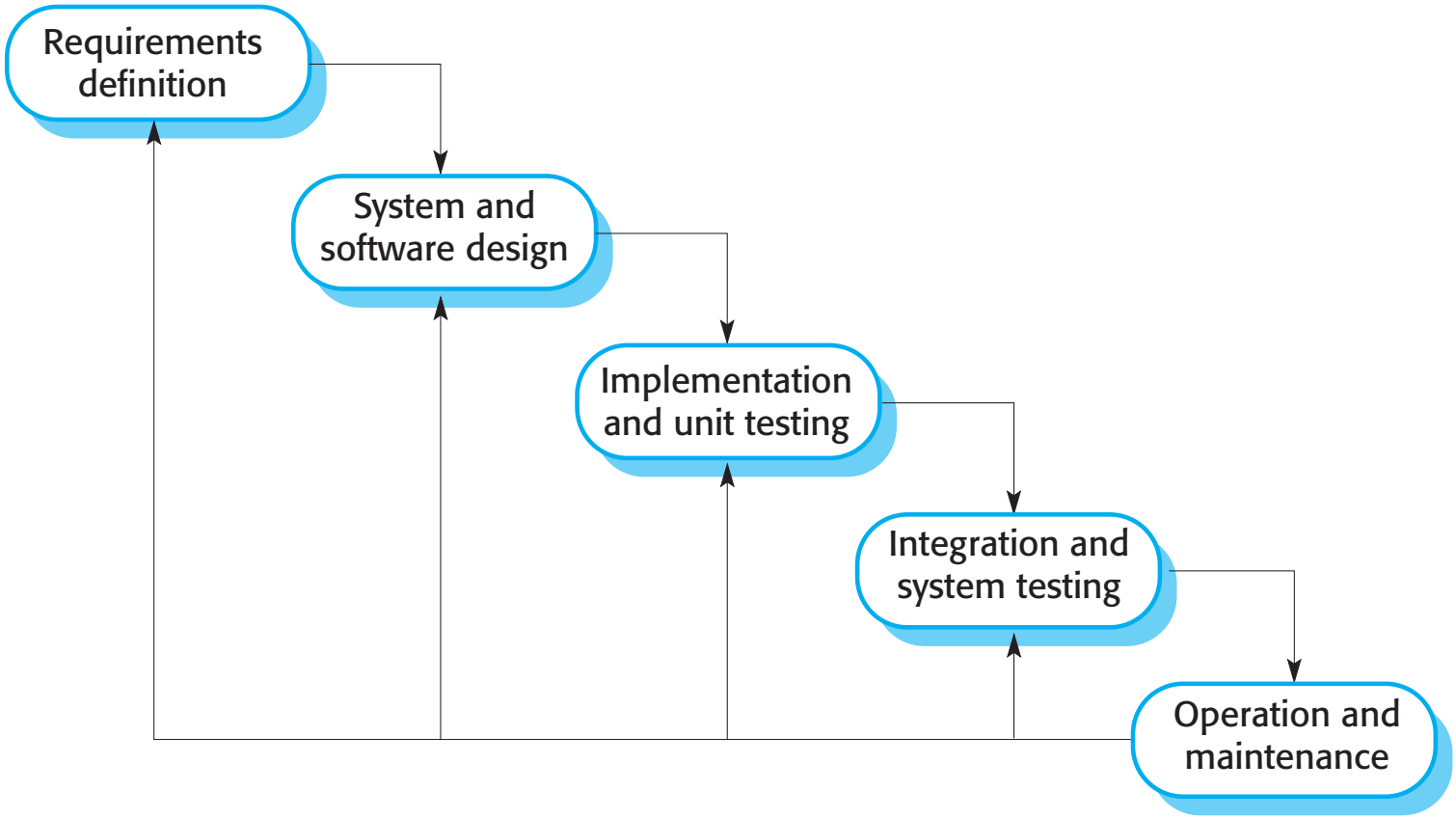
Software Process Models

- Waterfall model
- Incremental development
- Integration and configuration

Waterfall model

Waterfall Model

- Plan-driven model.
  - Separate and distinct phases of specification and development.



Waterfall Model Problems

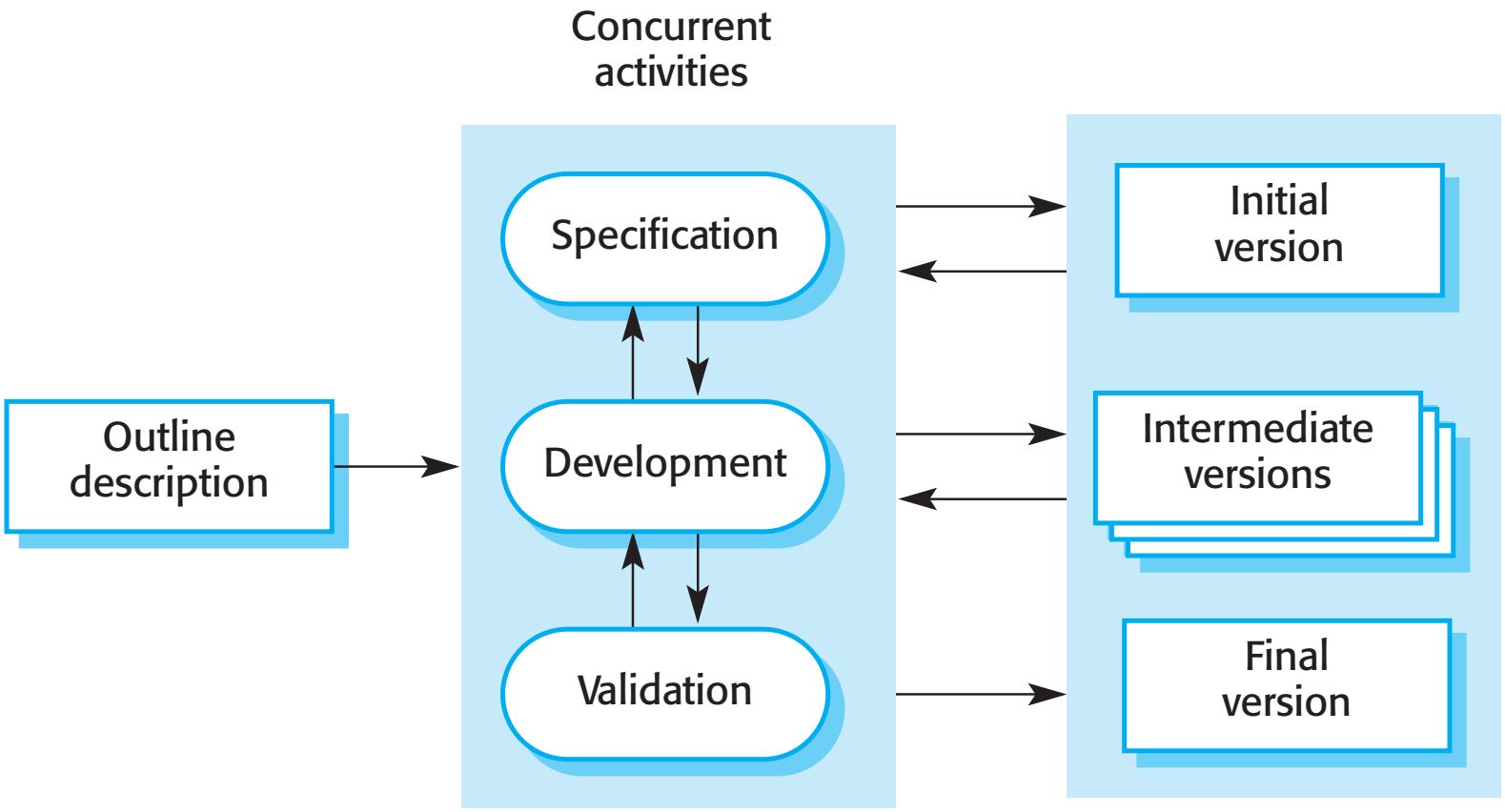
- Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.
  - Therefore, this model is only appropriate when the requirements are well-understood, and changes will be fairly limited during the design process.
  - Few business systems have stable requirements.

Waterfall Model Applications

- Embedded systems
  - where the software has to interface with hardware systems.
  - Due of the inflexibility of hardware, it is not usually possible to delay decisions on the software's functionality until it is being implemented.
- Critical systems
  - where there is a need for extensive safety and security analysis of the software specification and design.
  - In these systems, the specification and design documents must be complete so that this analysis is possible.
- Large software systems
  - where several companies are involved, complete specifications may be needed to allow for the independent development of subsystems.

Incremental Development

Incremental Development



Incremental Development

- Specification, development and validation are interleaved. May be plan-driven or agile.
  - Plan-driven: the system increments are identified in advance
  - Agile: the early increments are identified, but the development of later increments depends on progress and customer priorities.

- Incremental software development, which is a fundamental part of agile development methods, is better than a waterfall approach for systems whose requirements are likely to change during the development process.

Incremental Development Benefits

- The cost of accommodating changing customer requirements is reduced.
- It is easier to get customer feedback on the development work that has been done.
- More rapid delivery and deployment of useful software to the customer is possible.

Incremental Development Problems

- The process is not visible
  - Managers need regular deliverables to measure progress.
  - If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.

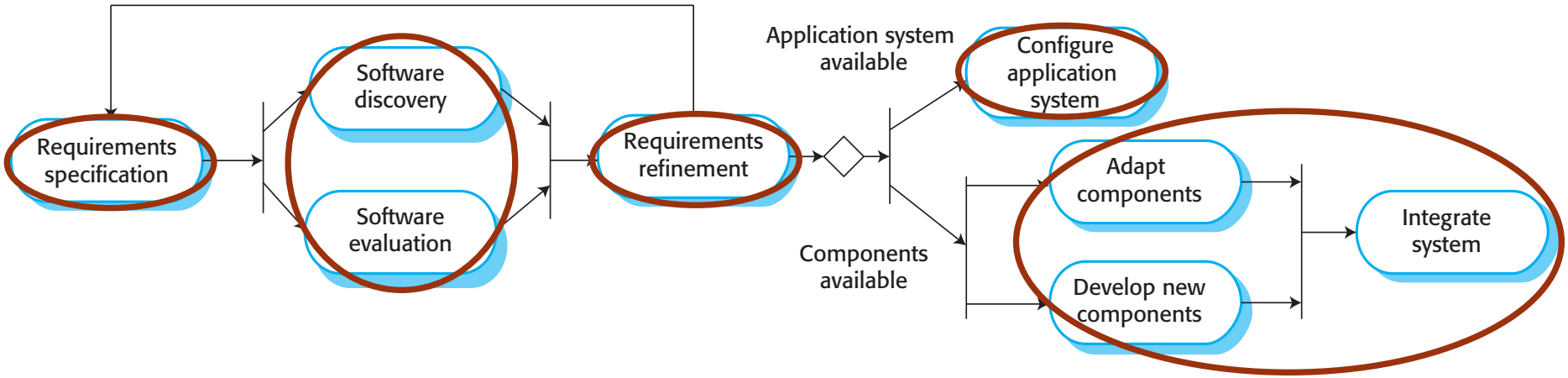
- System structure tends to degrade as new increments are added
  - Unless time and money is spent on refactoring to improve the software, regular change tends to corrupt its structure.
  - Incorporating further software changes becomes increasingly difficult and costly.

Integration and Configuration

Integration and Configuration

- The system is integrated from existing configurable components (reuse-oriented approach).
- Reuse is now the standard approach for building many types of business system
- Reused elements may be configured to adapt their behaviour and functionality to a user's requirements

Reuse-oriented Software Engineering



Types of Reusable Software

- Stand-alone application systems
  - that are configured for use in a particular environment.
- Collections of objects
  - that are developed as a package to be integrated with a component framework such as .NET or J2EE.
- Web services
  - that are developed according to service standards, and which are available for remote invocation.

Reuse-oriented Software Engineering

- Advantages:
  - Reduced costs and risks as less software is developed from scratch
  - Faster delivery of the system

- Disadvantages:
  - Requirements compromises are inevitable so system may not meet real needs of users
  - Loss of control over evolution of reused system elements

Process Activities

Process Activities

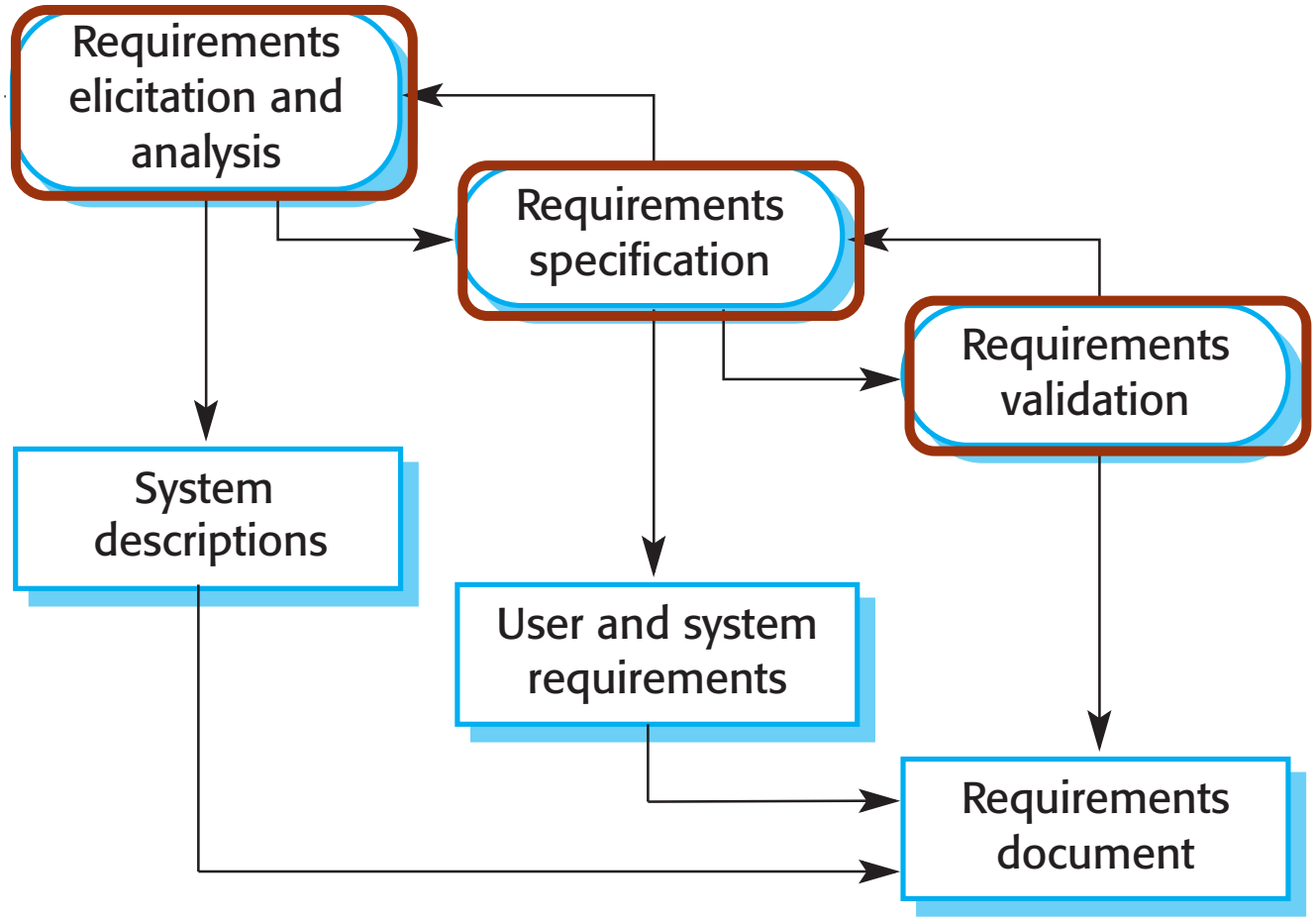
- The four fundamental software engineering activities:
  - Software specification
    - The functionality of the software and constraints on its operation must be defined.
  - Software design and implementation
    - The software to meet the specification must be produced.
  - Software validation
    - The software must be validated to ensure that it does what the customer wants.
  - Software evolution
    - The software must evolve to meet changing customer needs.

- These four basic process activities are organized differently in different development processes.

1

Software Specification

Requirements Engineering Process



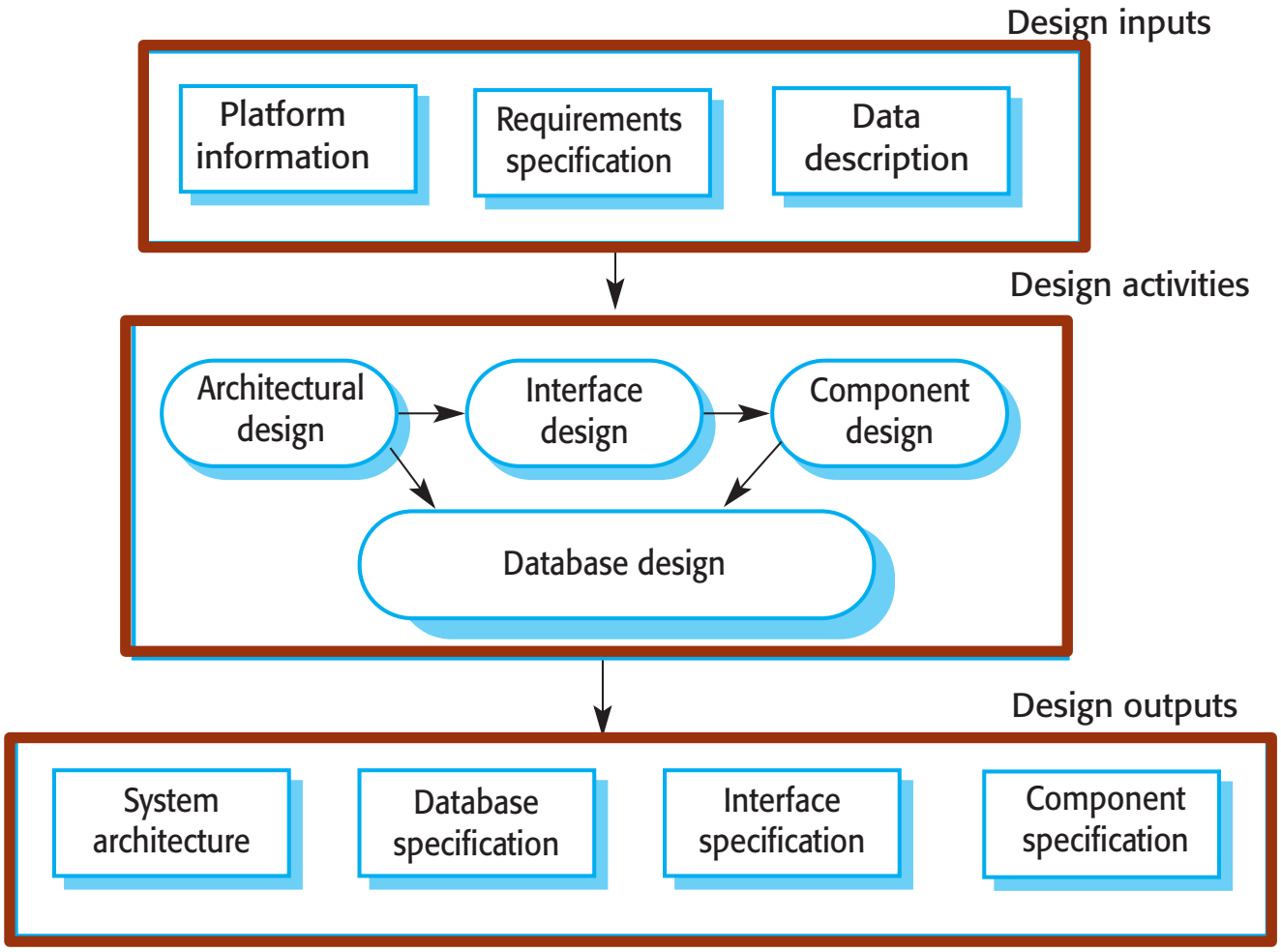
2

Software Design and Implementation

Software Design and Implementation

- The process of converting the system specification into an executable system.
- Software design
  - Design a software structure that realises the specification;
- Implementation
  - Translate this structure into an executable program;
- The activities of design and implementation are closely related and may be inter-leaved.

A General Model of the Design Process



System Implementation

- The software is implemented either by developing a program or programs or by configuring an application system.
- Programming is an individual activity with no standard process.
- Debugging is the activity of finding program faults and correcting these faults.

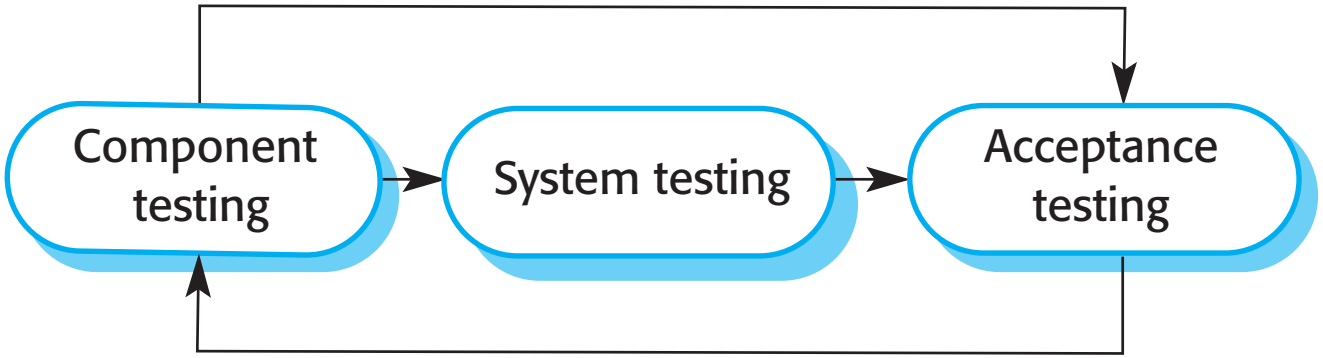
3

Software Validation

Software Validation

- Verification and validation (V & V) is intended to show that a system conforms to its specification and meets the requirements of the system customer.
- Involves checking and review processes and system testing.
- System testing involves executing the system with test cases that are derived from the specification of the real data to be processed by the system.

Testing Stages

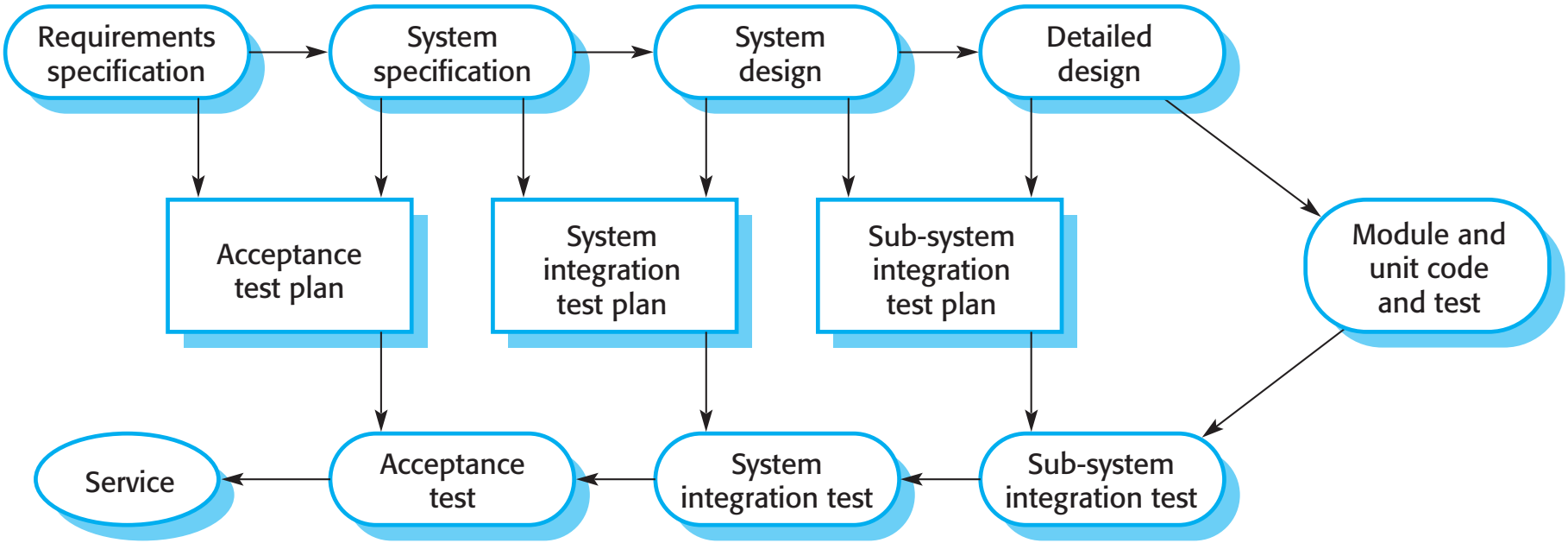


Testing Process

- Agile software process
  - Each increment should be tested as it is developed, with these tests based on the requirements for that increment.
  - In test-driven development, which is a normal part of agile processes, tests are developed along with the requirements before development starts.

- Plan-driven software process
  - Testing is driven by a set of test plans.
  - An independent team of testers works from these test plans, which have been developed from the system specification and design.

Testing Phases in Plan-driven Software Process (V-model)



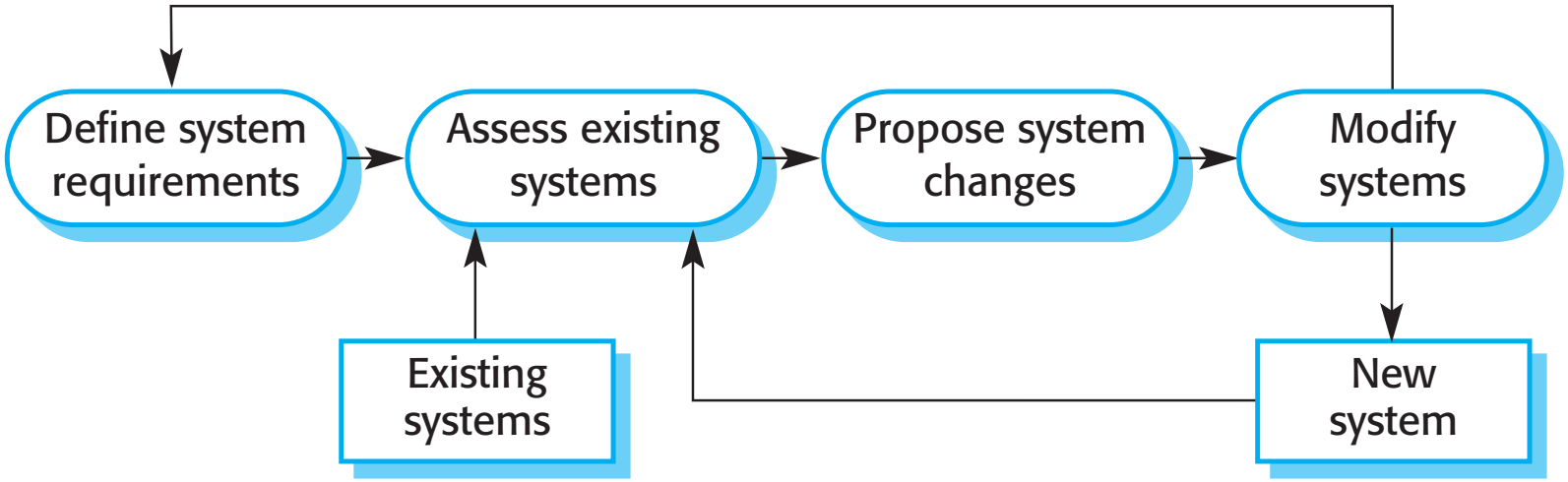
4

Software Evolution

Software Evolution

- Software is inherently flexible and can change.
- As requirements change through changing business circumstances, the software that supports the business must also evolve and change.
- Although there has been a demarcation between development and evolution (maintenance) this is increasingly irrelevant as fewer and fewer systems are completely new.

System Evolution



Summary

Summary

- Software processes are the activities involved in producing a software system. Software process models are abstract representations of these processes.

- General process models describe the organization of software processes.
  - Examples of these general models include:
    - the waterfall model,
    - incremental development, and
    - integration and configuration.

Summary

- Requirement engineering is the process of developing a software specification. Specifications are intended to communicate the system needs of the customer to the system developers.
- Design and implementation processes are concerned with transforming a requirements specification into an executable software system.
- Software validation is the process of checking that the system conforms to its specification and that it meets the real needs of the users of the system.
- Software evolution takes place when you change existing software systems to meet new requirements. Changes are continuous, and the software must evolve to remain useful.