



# Chapter 2 – Software Processes

# Topics covered

---



- ✧ Software process models
- ✧ Process activities
- ✧ Coping with change
- ✧ Process improvement

# The software process



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

# Software process descriptions



- ✧ When we describe and discuss processes, we usually talk about the activities in these processes such as specifying a data model, designing a user interface, etc. and the ordering of these activities.
- ✧ Process descriptions may also include:
  - Products, which are the outcomes of a process activity;
  - Roles, which reflect the responsibilities of the people involved in the process;
  - Pre- and post-conditions, which are statements that are true before and after a process activity has been enacted or a product produced.

# Plan-driven and agile processes



- ✧ Plan-driven processes are processes where all of the process activities are planned in advance and progress is measured against this plan.
- ✧ In agile processes, planning is incremental and it is easier to change the process to reflect changing customer requirements.
- ✧ 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 process models



## ✧ The waterfall model

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

## ✧ Incremental development

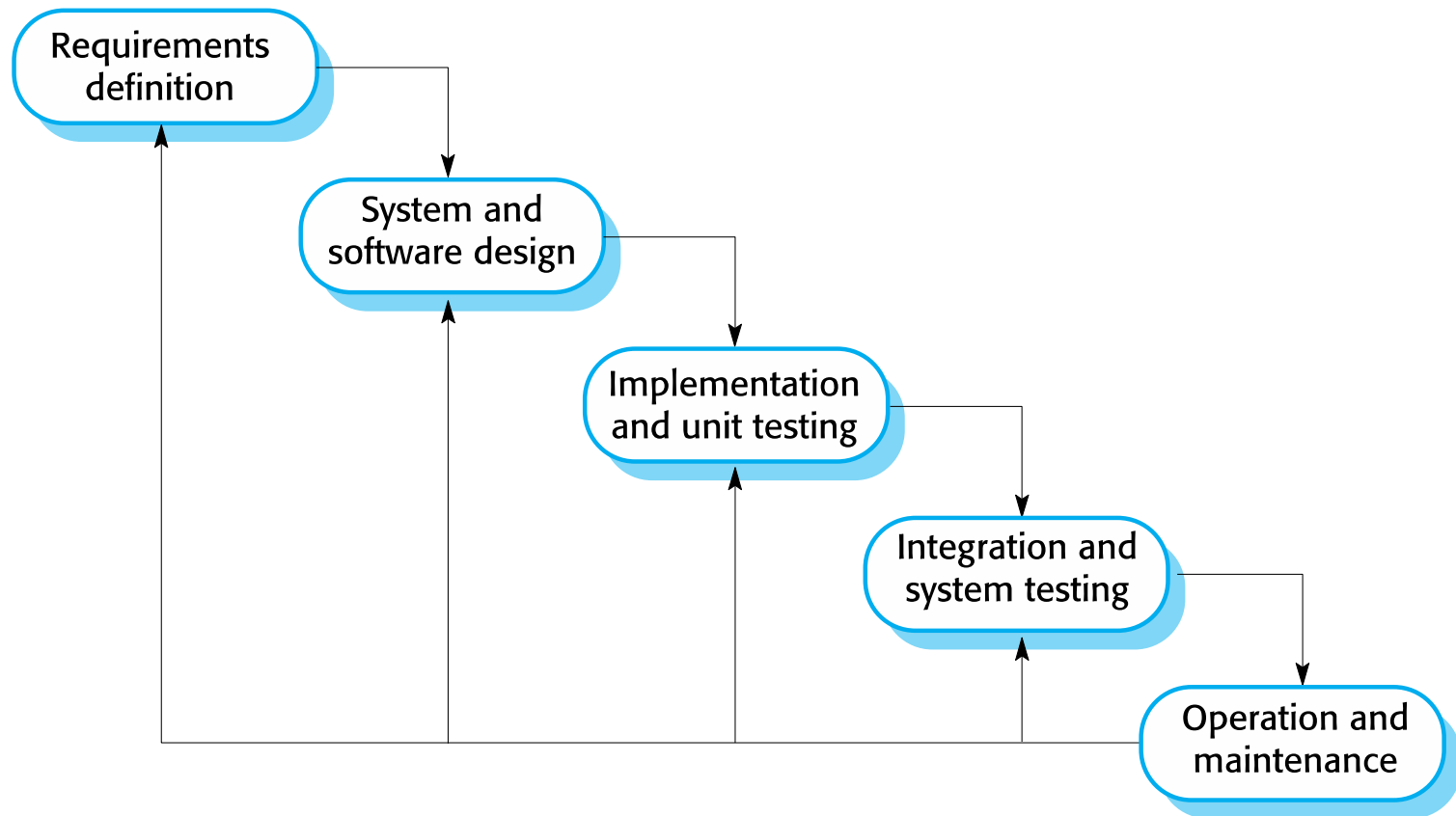
- Specification, development and validation are interleaved. May be plan-driven or agile.

## ✧ Integration and configuration

- The system is assembled from existing configurable components. May be plan-driven or agile.

## ✧ In practice, most large systems are developed using a process that incorporates elements from all of these models.

# The waterfall model



# Waterfall model phases



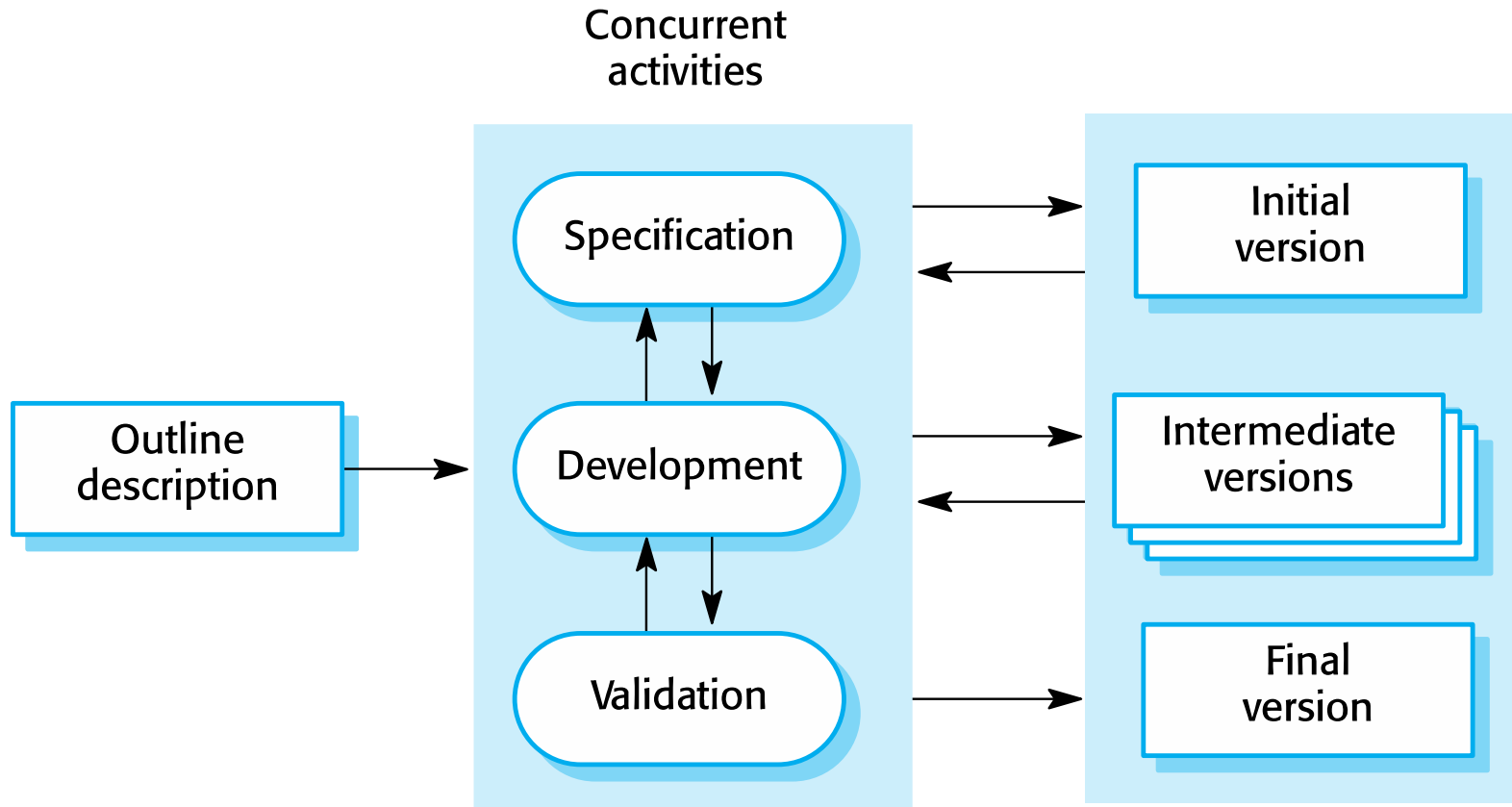
- ✧ There are separate identified phases in the waterfall model:
  - Requirements analysis and definition
  - System and software design
  - Implementation and unit testing
  - Integration and system testing
  - Operation and maintenance
- ✧ The main drawback of the waterfall model is the difficulty of accommodating change after the process is underway. In principle, a phase has to be complete before moving onto the next phase.

# 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.
- ✧ The waterfall model is mostly used for large systems engineering projects where a system is developed at several sites.
  - In those circumstances, the plan-driven nature of the waterfall model helps coordinate the work.

# Incremental development



# Incremental development benefits



- ✧ The cost of accommodating changing customer requirements is reduced.
  - The amount of analysis and documentation that has to be redone is much less than is required with the waterfall model.
- ✧ It is easier to get customer feedback on the development work that has been done.
  - Customers can comment on demonstrations of the software and see how much has been implemented.
- ✧ More rapid delivery and deployment of useful software to the customer is possible.
  - Customers are able to use and gain value from the software earlier than is possible with a waterfall process.

# 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



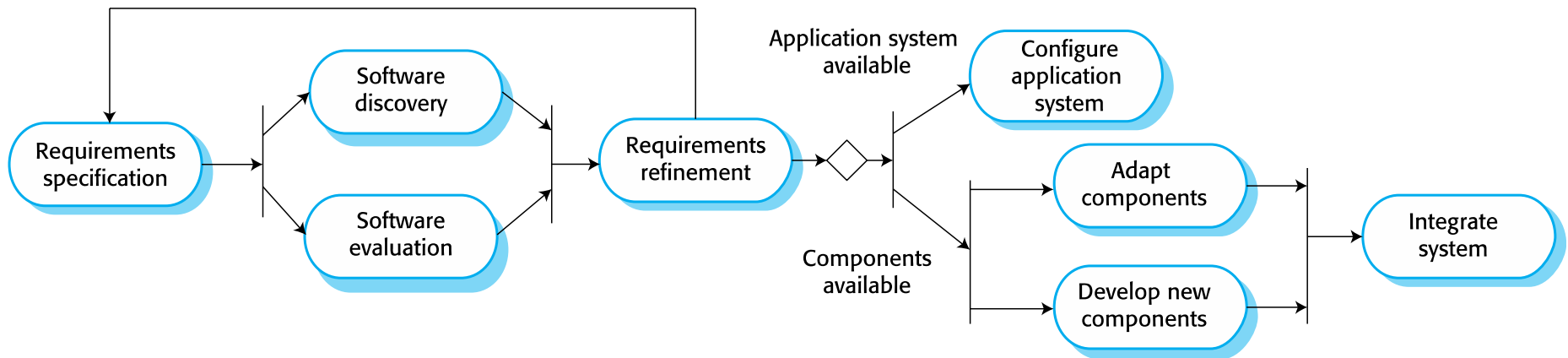
- ✧ Based on software reuse where systems are integrated from existing components or COTS (Commercial-off-the-shelf) systems.
- ✧ Reused elements may be configured to adapt their behaviour and functionality to a user's requirements
- ✧ Reuse is now the standard approach for building many types of business system

# Types of reusable software



- ✧ Stand-alone application systems (sometimes called COTS) 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 (e.g. .NET)
- ✧ Web services that are developed according to service standards and which are available for remote invocation.

# Reuse-oriented software engineering



# Key process stages

---



- ✧ Requirements specification
- ✧ Software discovery and evaluation
- ✧ Requirements refinement
- ✧ Application system configuration
- ✧ Component adaptation and integration

# Advantages and disadvantages

---



- ✧ Reduced costs and risks as less software is developed from scratch
- ✧ Faster delivery and deployment of system
- ✧ But requirements compromises are inevitable so system may not meet real needs of users
- ✧ Loss of control over evolution of reused system elements



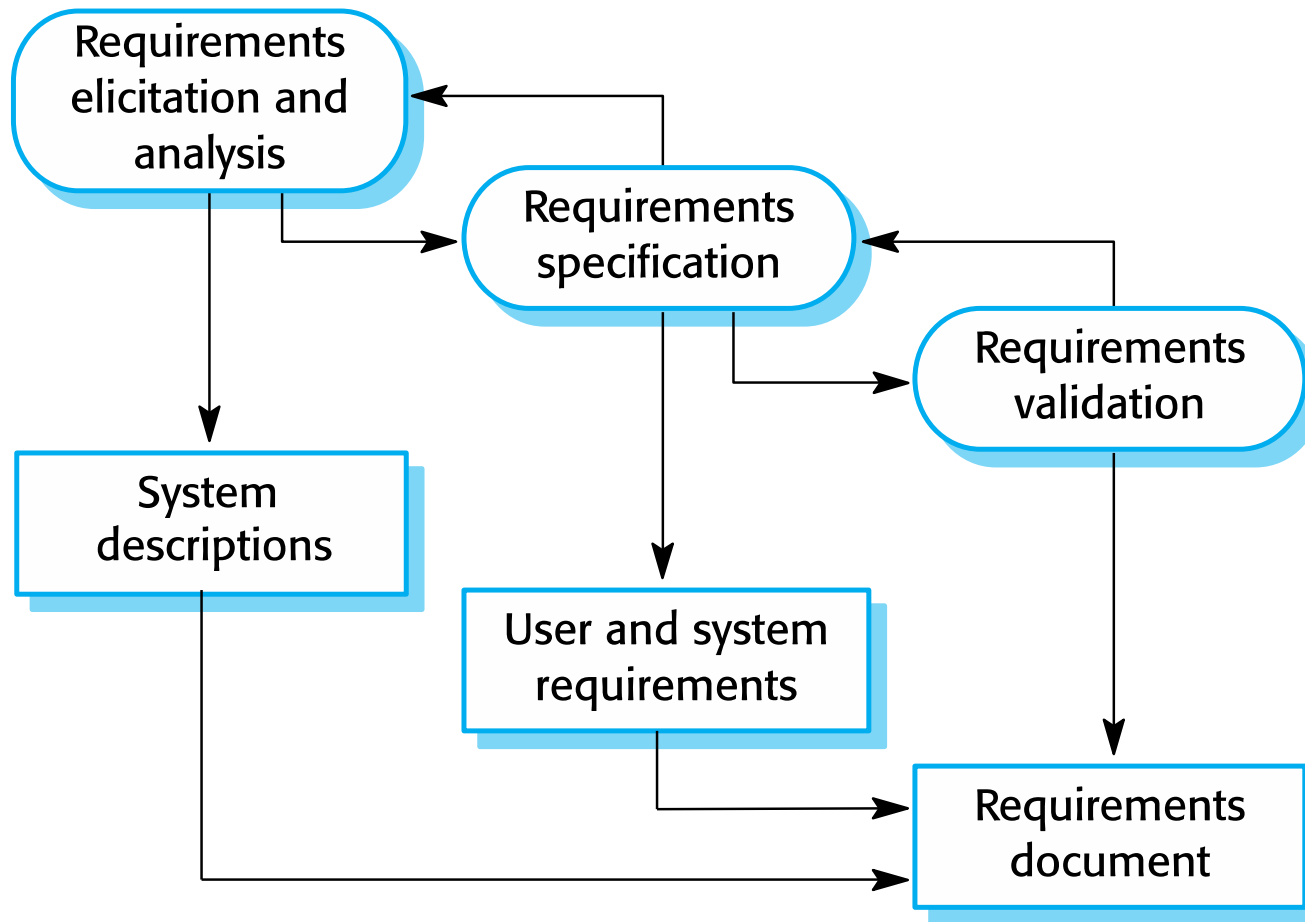
# Process activities

# Process activities



- ✧ Real software processes are inter-leaved sequences of technical, collaborative and managerial activities with the overall goal of specifying, designing, implementing and testing a software system.
- ✧ The four basic process activities of specification, development, validation and evolution are organized differently in different development processes.
- ✧ For example, in the waterfall model, they are organized in sequence, whereas in incremental development they are interleaved.

# The requirements engineering process



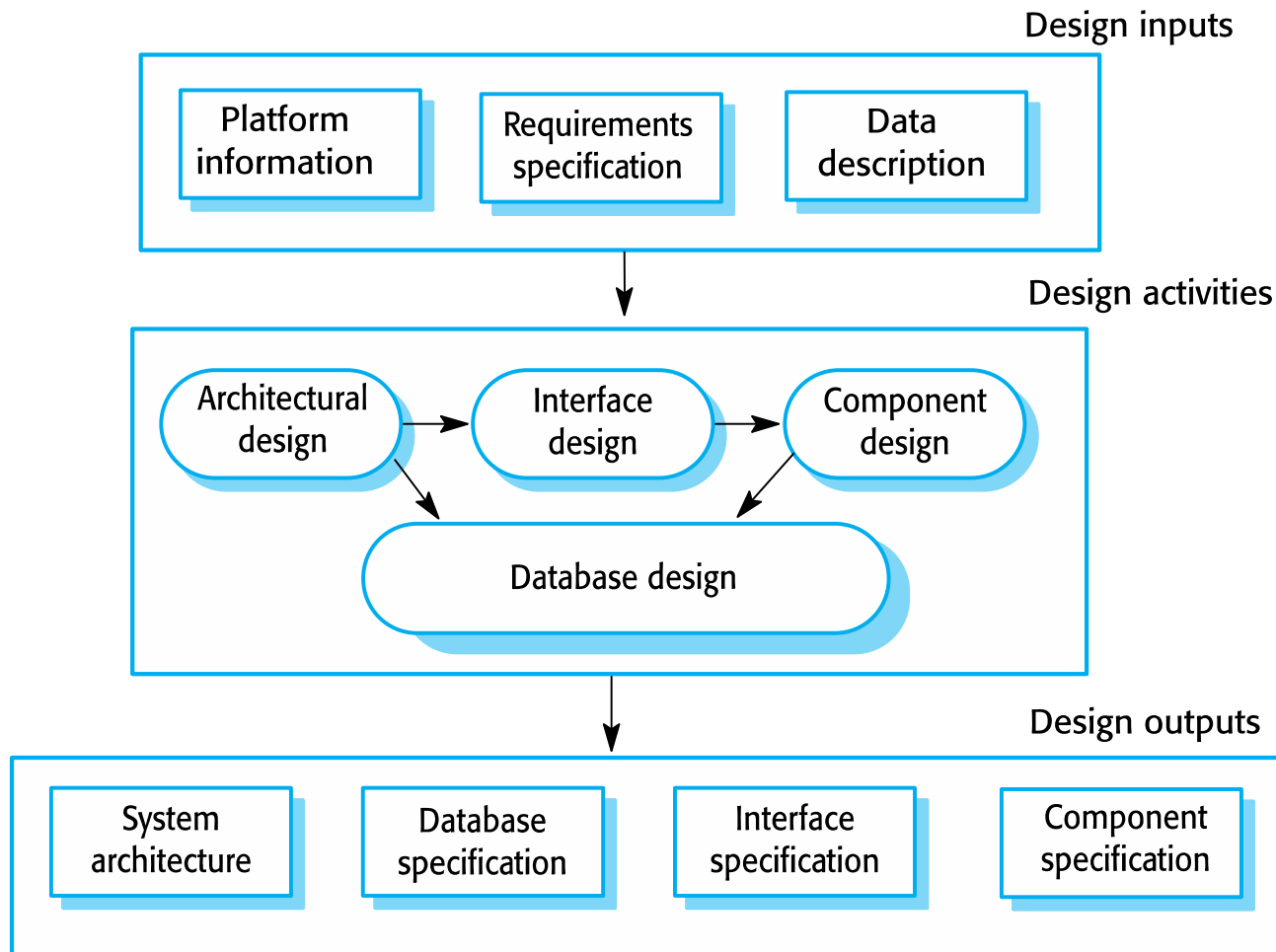
# Software design and implementation

---



- ✧ 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



# Design activities



- ✧ *Architectural design*, where you identify the overall structure of the system, the principal components (subsystems or modules), their relationships and how they are distributed.
- ✧ *Database design*, where you design the system data structures and how these are to be represented in a database.
- ✧ *Interface design*, where you define the interfaces between system components.
- ✧ *Component selection and design*, where you search for reusable components. If unavailable, you design how it will operate.

# System implementation

---



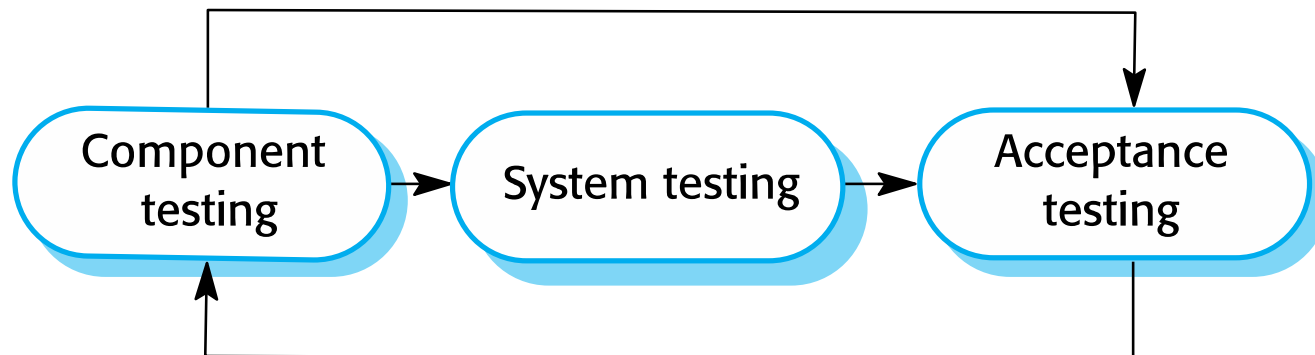
- ✧ The software is implemented either by developing a program or programs or by configuring an application system.
- ✧ Design and implementation are interleaved activities for most types of software system.
- ✧ Programming is an individual activity with no standard process.
- ✧ Debugging is the activity of finding program faults and correcting these faults.

# 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 is the most commonly used V & V activity.

# Stages of testing



# Testing stages



## ✧ Component testing

- Individual components are tested independently;
- Components may be functions or objects or coherent groupings of these entities.

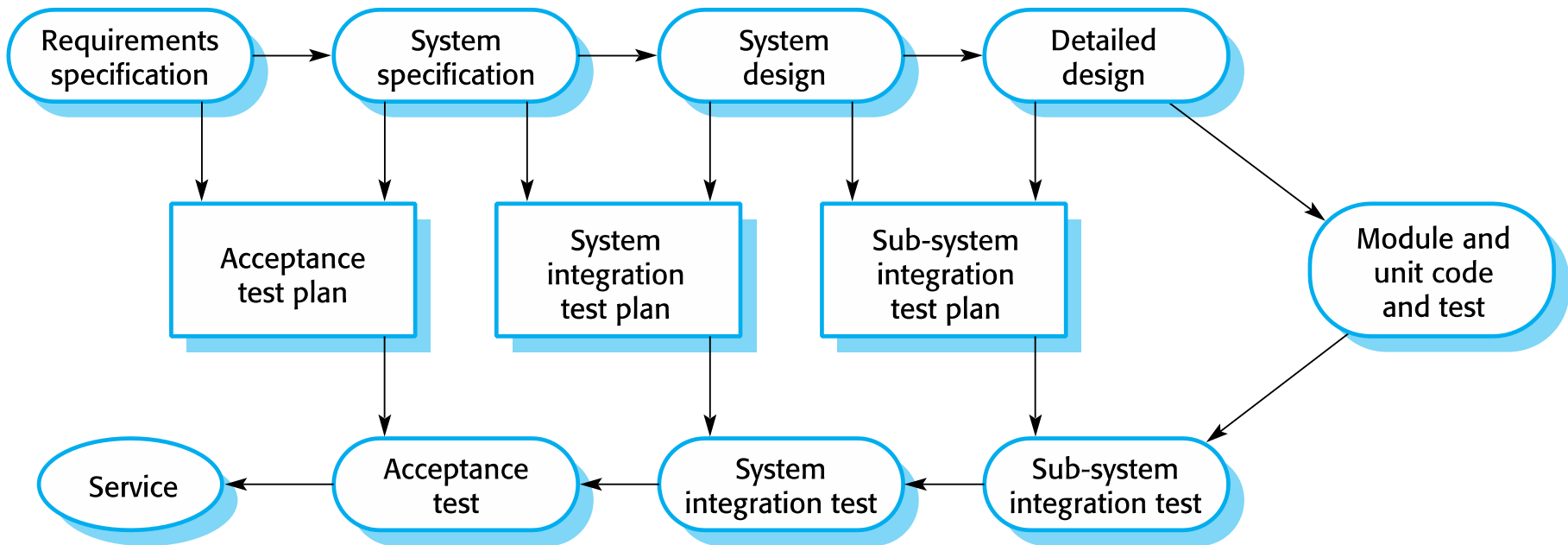
## ✧ System testing

- Testing of the system as a whole. Testing of emergent properties is particularly important.

## ✧ Customer testing

- Testing with customer data to check that the system meets the customer's needs.

# Testing phases in a plan-driven software process (V-model)



# 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

