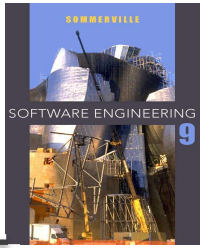
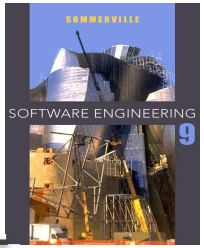


Chapter 2 – Software Processes

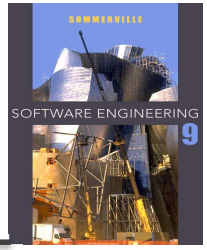
Section 2

Topics covered



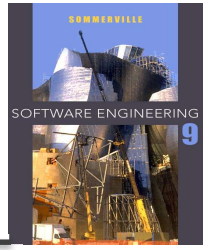
- Software process models
- Process activities

The software process



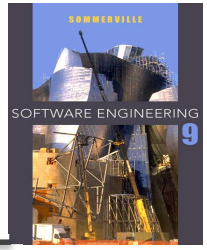
- A structured set of activities required to develop a software system.
- Many different software processes but all involve:
  - Specification – defining what the system should do;
  - Design and implementation – defining the organization of the system and implementing the system;
  - Validation – checking 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



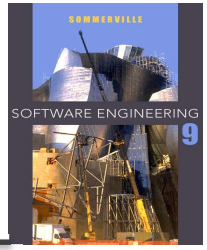
- When we describe and discuss processes, we usually talk about the activities in these processes such as specifying a data model, designing a user interface, etc. and the ordering of these activities.
- Process descriptions may also include:
  - Products, which are the outcomes of a process activity;
  - 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.
- **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



The waterfall model

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

Incremental development

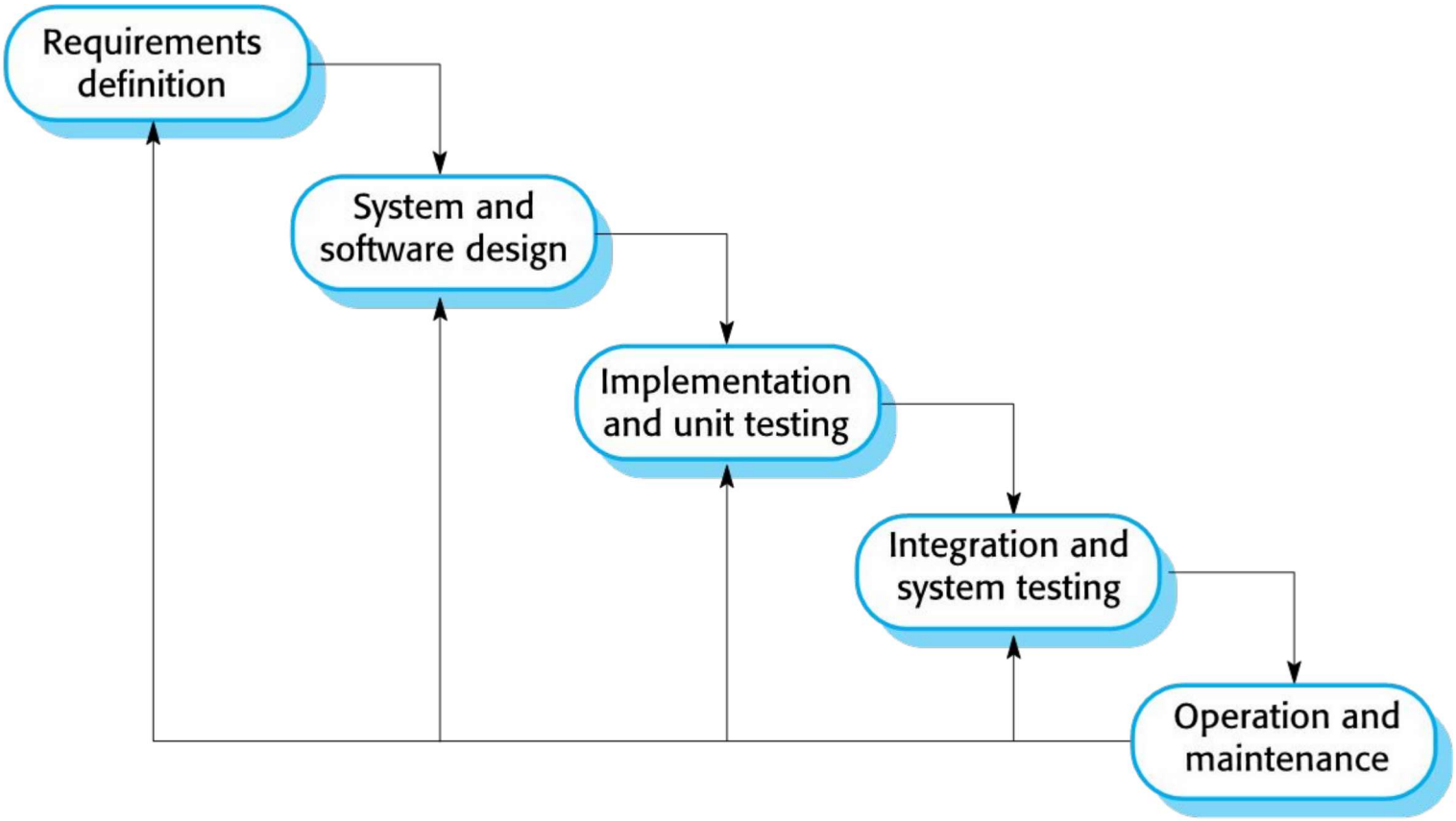
- Specification, development and validation are interleaved. May be plan-driven or agile.

Integration and configuration

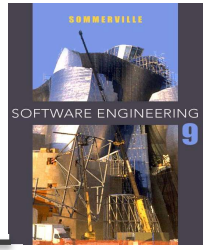
- The system is assembled from existing components. May be plan-driven or agile.

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

The waterfall model

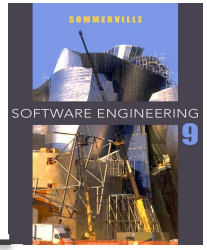


Waterfall model phases



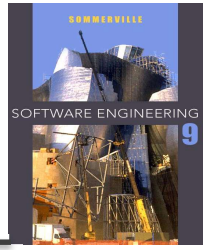
- There are separate identified phases in the waterfall model:
  - Requirements analysis and definition
    - The system's services, constraints, and goals are established by consultation with system users. They are then defined in detail and serve as a system specification.
  - System and software design
    - **Systems design** process allocates the requirements to either hardware or software systems. It establishes an overall system architecture.
    - **Software design** involves identifying and describing the fundamental software system abstractions and their relationships.

Waterfall model phases



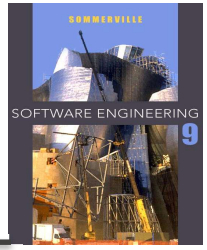
- There are separate identified phases in the waterfall model:
  - Implementation and unit testing
    - During this stage, the software design is realized as a set of programs or program units. Unit testing involves verifying that each unit meets its specification.
  - Integration and system testing
    - The individual program units or programs are integrated and tested as a complete system to ensure that the software requirements have been met. After testing, the software system is delivered to the customer.

Waterfall model phases



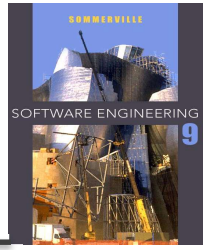
- There are separate identified phases in the waterfall model:
  - Operation and maintenance
    - This is the longest life-cycle phase. The system is installed and put into practical use.
    - **Maintenance** involves:
      - Correcting errors that were not discovered in earlier stages of the life cycle.
      - Improving the implementation of system units.
      - Enhancing the system's services as new requirements are discovered.

Waterfall model problems



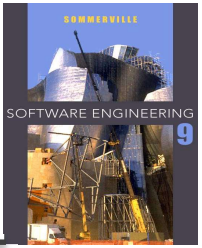
- The main drawback of the waterfall model is the difficulty of accommodating change after the process is underway. In principle, a phase has to be complete before moving onto the next phase.
- Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer 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.

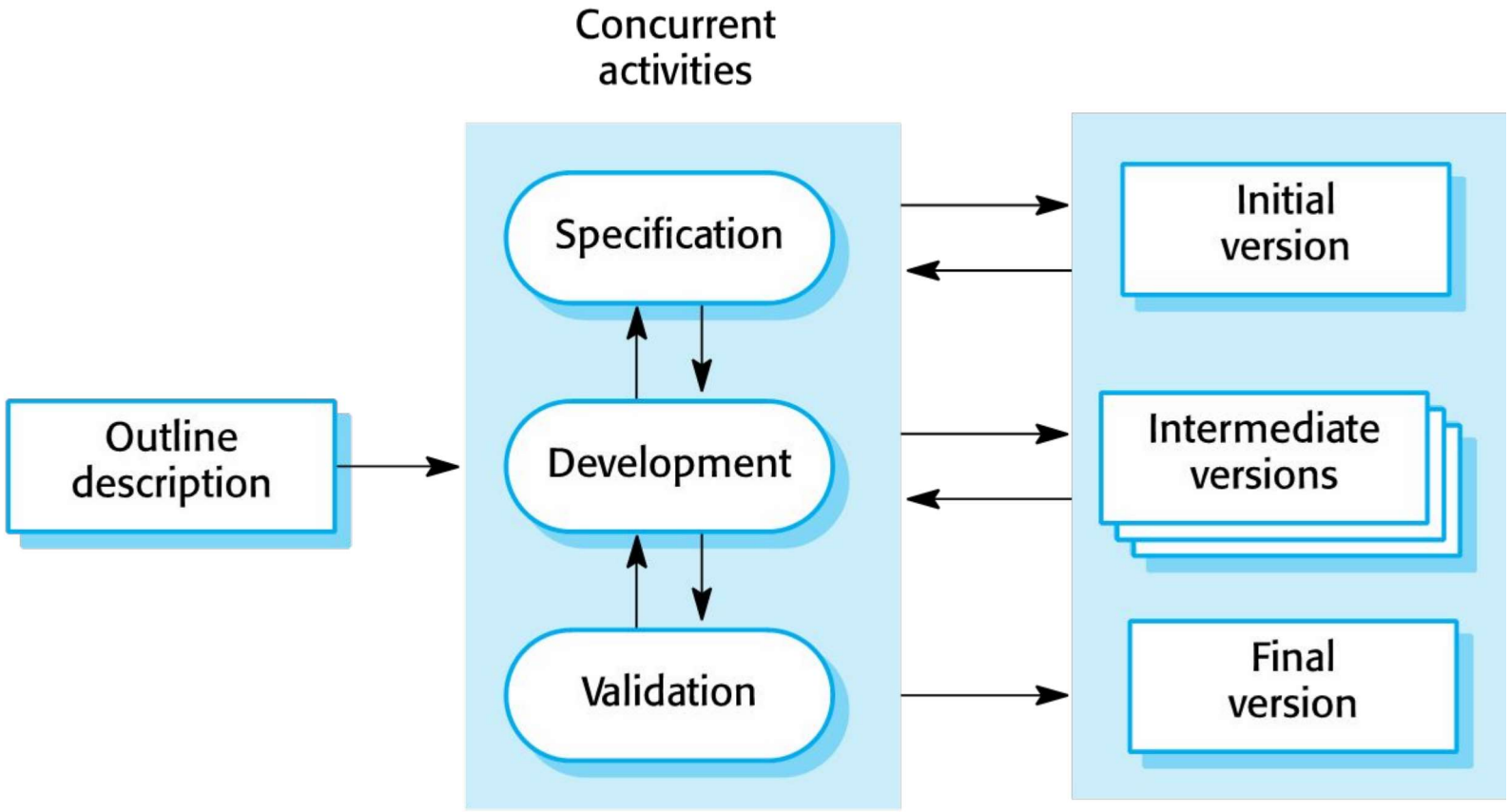
Waterfall model problems



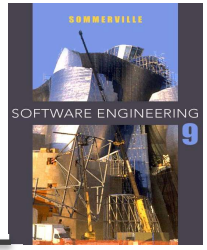
- The waterfall model is only appropriate for some types of system:
  - Embedded systems
  - Critical systems
  - Large software systems

Incremental development



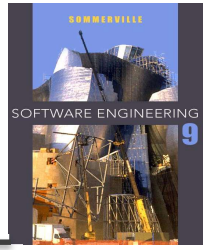


Incremental development



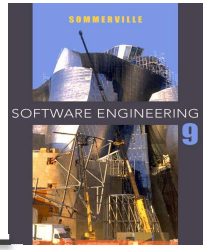
- Specification, development, and validation activities are interleaved rather than separate, with rapid feedback across activities.
- This approach can be either *plan-driven*, *agile* or, more usually, a mixture of these approaches.
- In a *plan-driven approach*, the system increments are identified in advance; if an *agile approach* is adopted, the early increments are identified, but the development of later increments depends on progress and customer priorities.

Incremental development benefits



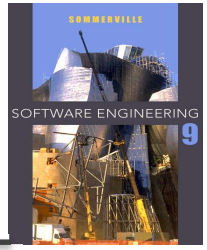
- The cost of accommodating changing customer requirements is reduced.
  - The amount of analysis and documentation that has to be redone is much less than is required with the waterfall model.
- It is easier to get customer feedback on the development work that has been done.
  - Customers can comment 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



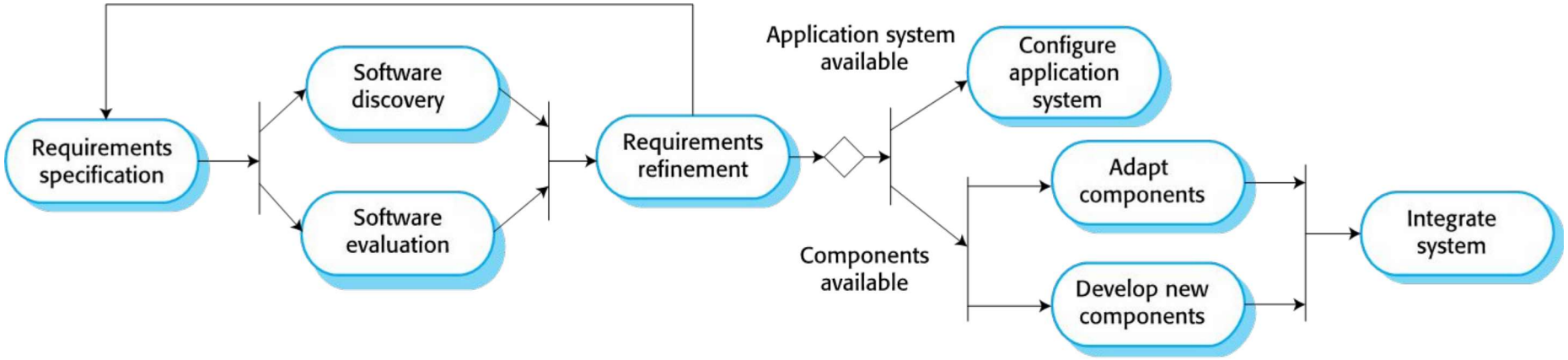
- The process is not visible.
  - Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.
- System structure tends to degrade as new increments are added.
  - Unless time and money 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

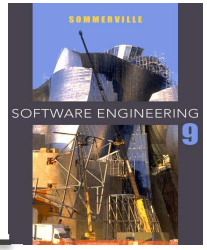


- Based on software reuse where systems are integrated from existing components or application systems (sometimes called COTS -Commercial-off-the-shelf) systems).
- Reused elements may be configured to adapt their behaviour and functionality to a user's requirements
- Reuse is now the standard approach for building many types of business system
  - Reuse covered in more depth in Chapter 15.

Integration and configuration

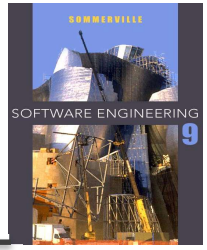


Types of software component



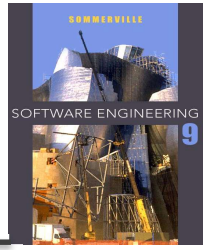
- Stand-alone application systems (sometimes called COTS) that are configured for use in a particular environment.
- Collections of objects that are developed as a package to be integrated with a component framework.
- Web services that are developed according to service standards and which are available for remote invocation.

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