CS251
Dept. of Computer Science
Undergraduate - Level 2

Software Engineering I

Module Outline

Spring “Semester 2” 2018 / 2019

Module Info.

❑ Lectures:

Group A: Thursdays *(from 10:00 a.m. to 12:00 p.m.)*

Group B: Thursdays *(from 12:00 p.m. to 2:00 p.m.)*

❑ Instructor:

Dr. Amr S. Ghoneim

- Office hours: *Thursdays from 9 a.m. – 10 a.m.*
- Short Bio:
  - Bachelor in Computer Science from Helwan University
  - Masters in Computer Science from Helwan University
  - PhD. in Computer Science (Artificial Intelligence) from University of New South Wales, Australia

Module Info. *(Continued)*

❑ TAs:

- To be announced …

❑ Indicative Reading List (textbooks):

Textbook:

- Ian Sommerville, "Software Engineering (9th Edition)", Addison Wesley, ISBN: 978-0137035151, 2010.

For OO Design Principles, Analysis & Design, UML Modelling:

- Bernd Bruegge, Allen H. Dutoit , Object-Oriented Software Engineering: Using UML, Patterns and Java, 3rd Edition, Prentice Hall 2009.

For the Design Patterns:

- Vlissides, J., Helm, R., Johnson, R. and Gamma, E., 1995. Design patterns: Elements of reusable object-oriented software. Reading: Addison-Wesley.

And / Or

- Freeman E, Freeman E, Robson E, Bates B, Sierra K. Head first design patterns. O'Reilly Media Inc., 2004.

Assessment Scheme

- **Group Project (Weeks 4, 6, & 12):** 25%
  - Practical Project 15% and 10% for Individual Assessment
  - 10,000 words *(approx.)* report + the detailed diagrams for software modelling + code *(implementation)*

- **Midterm Test: (week 7)** 15%

- **3-hours Final Written Exam:** 60%

Group Project 25%

Register with the TA(s); Your group members, & the project idea, by the end of week 3.

To apply the concepts discussed in lectures the student will get engaged in a practical project.

Project Ideas: The students will have the freedom to present the project idea that they would like to develop (however the idea should be within one of themes that will be announced later).

Project groups: Each group will be 5 to 6 students.

The students will be required to perform a 10-15 min demo of their project during the final discussion.

Group Project 25%

Tentative/Indicative Project deliverables:

Week 4: (Part of registering the project)

A report of 200 -500 word indicating the following:

- Project idea
- Main functionalities
- Similar applications in the market
- Development platform

Group Project 25%

Week 7 or 8: (Phase I Submission - 10%)

UML design document indicating the following:

- Functional Requirements (1 mark)
- Non-Functional Requirements (1 mark)
- System Architecture (1 mark)
- Activity Diagram(s) (1 mark)
- Use-Case Diagram(s) (1 mark)
  - *including general use-cases for the system, and the detailed use-cases description*
- Sequence Diagram(s) (1 mark)
  - *including system sequence diagrams*
- Collaboration Diagram(s) (1 mark)
- Database Specification (1 mark)
- Class Diagram (2 mark)
- Snap shots of User Interface

Group Project 25%

Tentative/Indicative Project deliverables:

Week 12: (Phase II Submission - 15%)

- A final report that includes:

- Finalized UML Diagrams
- Design Patterns (if applicable)
- Snap Shots of User Interfaces
- Test plan and Test cases performed

- Working code

(details will be announced later)

What is expected from you…

- Attend the class regularly.
- Study and learn the material presented in the class *(and refer to your reading list)*.
- Do the project *(in a team of 5 to 6)*.
- Perform well in the exams.
- Don’t cheat *(Plagiarism)*.

Tentative Weekly Plan

- **Week 1: *9 February – 15 February***
  - Lecture 0 (Module Outline, Introduction)
  - Lecture 1 (Software Processes - Part I)
- **Week 2: *16 February – 22 February***
  - Lecture 2 (Requirements Engineering)
- **Week 3: *23 February – 1 March***
  - Lecture 3 (Introduction to Modelling and Object Oriented Modelling)
- **Week 4: *2 March – 8 March***
  - Lecture 4 (Class Diagrams, & Package Diagrams)
- **Week 5: *9 March – 15 March***
  - Lecture 5 (UML Functional & Dynamic Diagrams)
- **Week 6: *16 March – 22 March***
  - Lecture 6 (Intro. to Design & Architectural Patterns)

Tentative Weekly Plan

- **Week 7: *23 March – 29 March***
  - Midterm Exam
- **Week 8: *30 March – 5 April***
  - Project Discussions (Phase 1)
  - Lecture 7 (UML Dynamic Diagrams – Part II)
- **Week 9: *6 April – 12 April***
  - Lecture 8 (Software Processes - Part II)
- **Week 10: *13 April – 19 April***
  - Lecture 9 (UML for Realtime Systems, & Testing)
- **Week 11: *20 April – 26 April***
  - Public Holiday (Sinai Liberation Day)
- **Week 12: *27 April – 3 May***
  - Project Discussions (Phase 2)
  - Lecture 10 (More on Design Patterns)

Why … Software Engineering?

- The economies of ALL developed nations are dependent on software.
- More and more systems are software controlled.
- Software engineering is concerned with theories, methods and tools for professional software development.
- Expenditure on software represents a significant fraction of GNP *(Gross National Product)* in all developed countries.

Why … Software Engineering?

- The Standish Group has been publishing reports on the success of software projects since 1994. In its latest report (2012) it identified that:
  - Only 39% of all projects were being delivered on time, on budget, with the required features and functions.
  - .. 43% were late, or over budget, or not delivering all the required features.
  - .. 18% were either cancelled before delivery, or delivered but never used.
- In those reports, the major problems have been identified. Among these problems were the following:
  - Poor, or lack of, user involvement.
  - Lack of clear business objectives.
  - Under and over building – building features that are never used and not building all the required features.

Frequently asked questions about Software Engineering

Question

- **Answer**

What is software?

- Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market.

What are the attributes of good software?

- Good software should deliver the required functionality and performance to the user (useful) and should be flexible, usable, reliable, available, affordable, and maintainable (greatly influenced by how it is designed and written).

What is software engineering?

- Software engineering is an engineering discipline that is concerned with all aspects of software production.

What are the fundamental software engineering activities?

- Software specification, software development, software validation and software evolution.

Frequently asked questions about Software Engineering

What is the difference between software engineering and computer science?

- Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.

What is the difference between software engineering and system engineering?

- System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering. Software engineering is part of this more general process.

What is a System boundary?

- a conceptual line that divides the system that we wish to study from 'everything else', (scope of a system).

What is a System's environment?

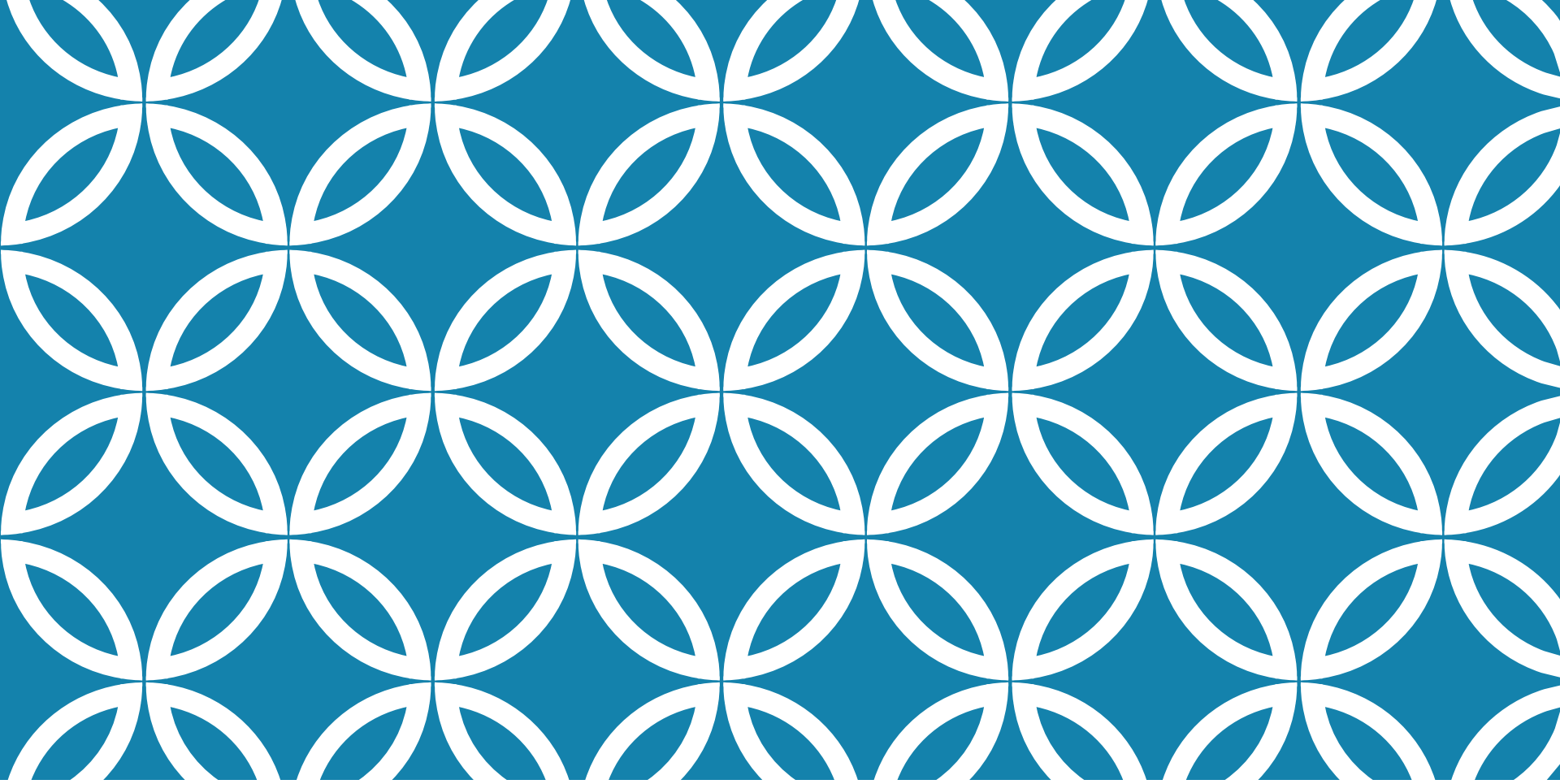
- It is made up of those things which are not part of the system, but which can either affect the system or be affected by it.

What is a domain?

- It is a particular area of interest.

The Important characteristics of Software that affect its development

- **Malleability:** Software is easy to change. This malleability creates a constant pressure for software to be changed rather than replaced.
- **Complexity:** Software is often complex. Complexity can usually be recognized, but it is less easy to define. One item of software can be considered more complex than another if the description of the first requires more explanation than that of the second. If complexity is increased, errors will be increased.
- **Size:** It is likely that there will be more errors in a large piece of software than there will be in a small one..



(CS251) SOFTWARE ENGINEERING I

Lecture 1
Software Processes
(Chapter 2)

TOPICS COVERED

- Software Process Models
  - Plan-Driven and Agile Processes
  - The Waterfall Model
  - Incremental Development
  - Reuse-Oriented Software Engineering
- Process Activities
  - Software Specification
  - Software Design and Implementation
    - Design activities
  - Software Validation
    - Testing stages
  - Software Evolution

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

The waterfall model

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

Incremental development

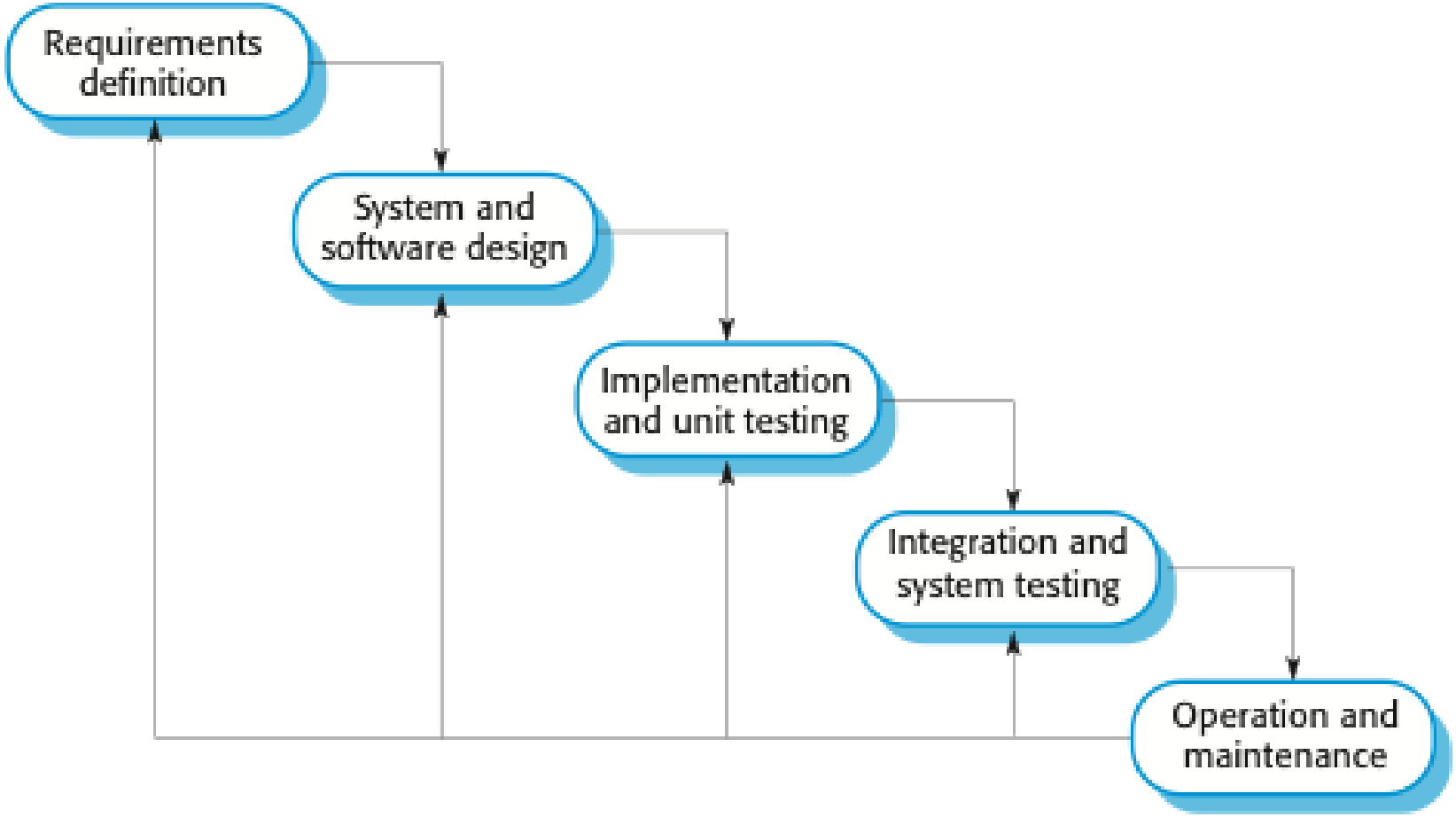
- Specification, development and validation are interleaved. May be plan-driven or agile.

Reuse-oriented software engineering

- 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
- 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

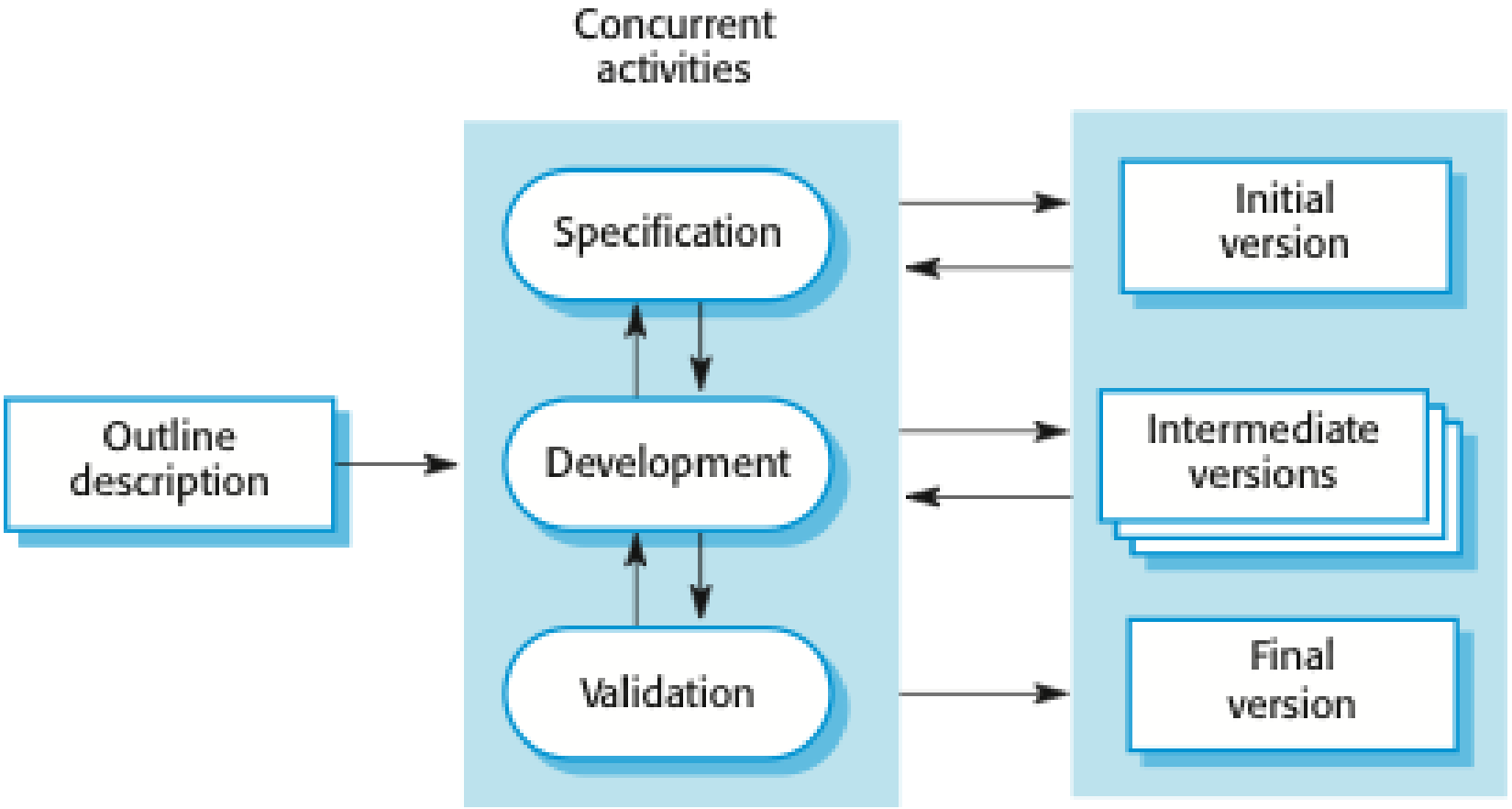
Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.

- 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.

REUSE-ORIENTED SOFTWARE ENGINEERING

Based on systematic reuse where systems are integrated from existing components or COTS (Commercial-off-the-shelf) systems.

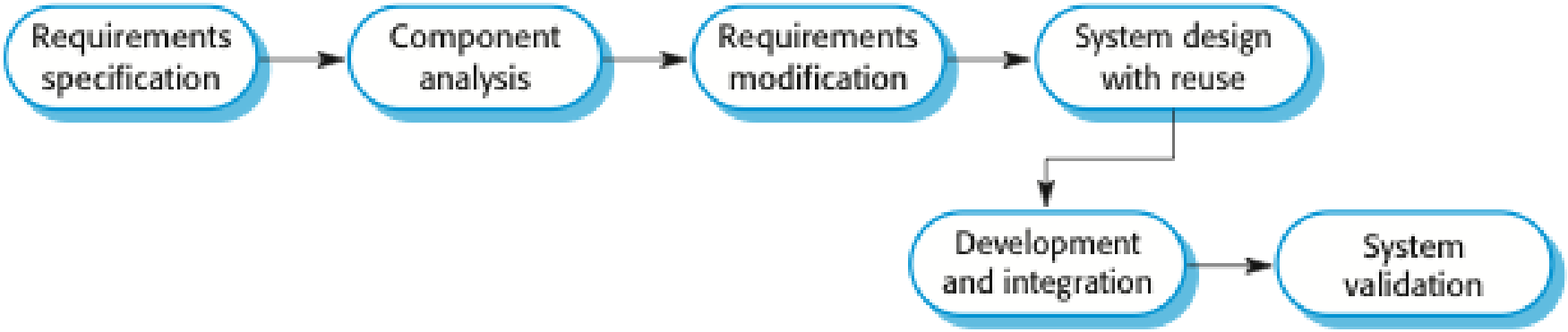
Process stages

- Component analysis;
- Requirements modification;
- System design with reuse;
- Development and integration.

Reuse is now the standard approach for building many types of business system

- Reuse covered in more depth in Chapter 16.

REUSE-ORIENTED SOFTWARE ENGINEERING



TYPES OF SOFTWARE COMPONENT

Web services that are developed according to service standards and which are available for remote invocation.

Collections of objects that are developed as a package to be integrated with a component framework such as .NET or J2EE.

Stand-alone software systems (COTS) that are configured for use in a particular environment.

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. In the waterfall model, they are organized in sequence, whereas in incremental development they are inter-leaved.

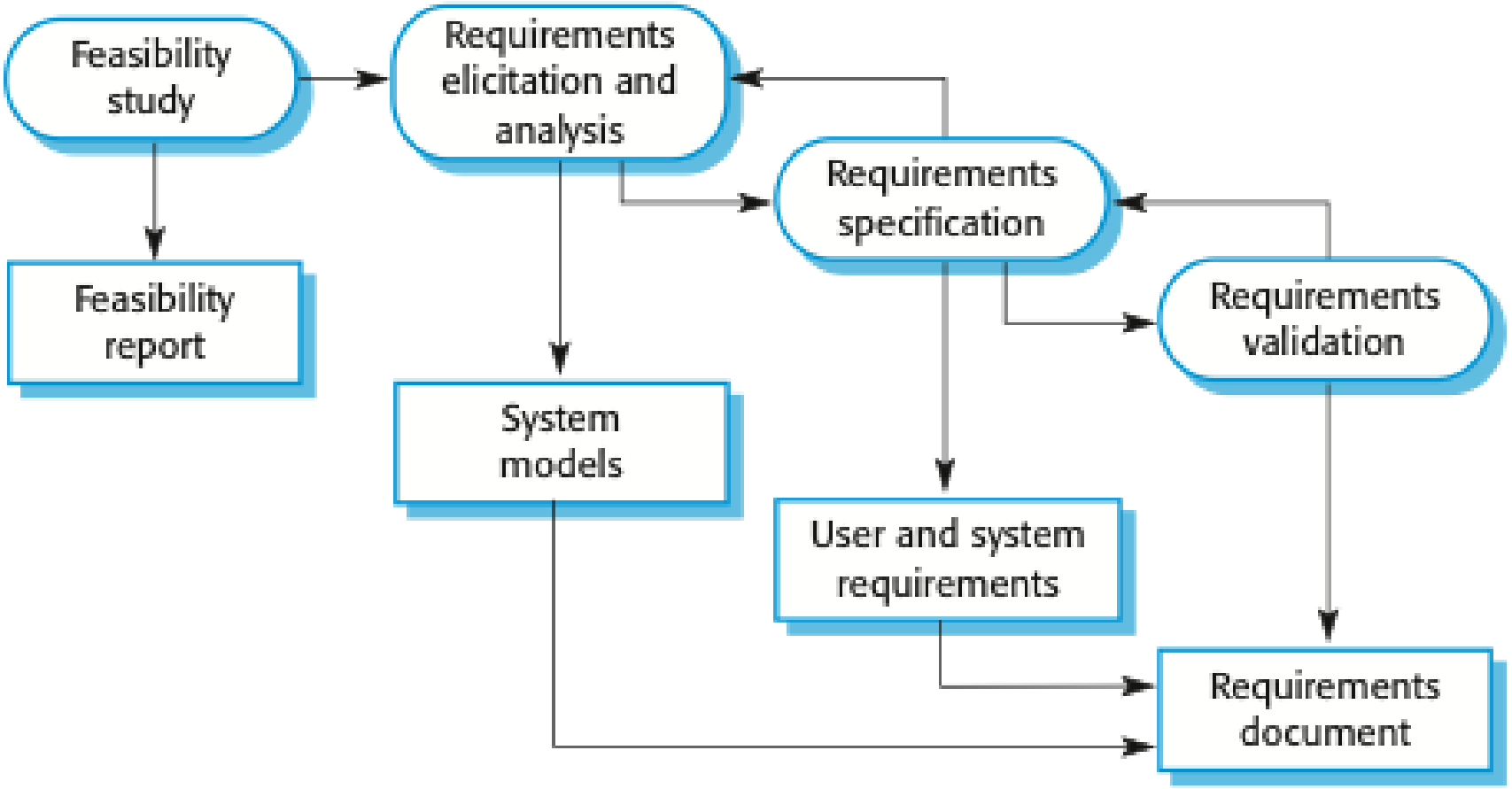
SOFTWARE SPECIFICATION

The process of establishing what services are required and the constraints on the system's operation and development.

Requirements engineering process

- Feasibility study
  - Is it technically and financially feasible to build the system?
- Requirements elicitation and analysis
  - What do the system stakeholders require or expect from the system?
- Requirements specification
  - Defining the requirements in detail
- Requirements validation
  - Checking the validity of the requirements

THE REQUIREMENTS ENGINEERING PROCESS



SOFTWARE DESIGN AND IMPLEMENTATION

The process of converting the system specification into an executable system.

Software design

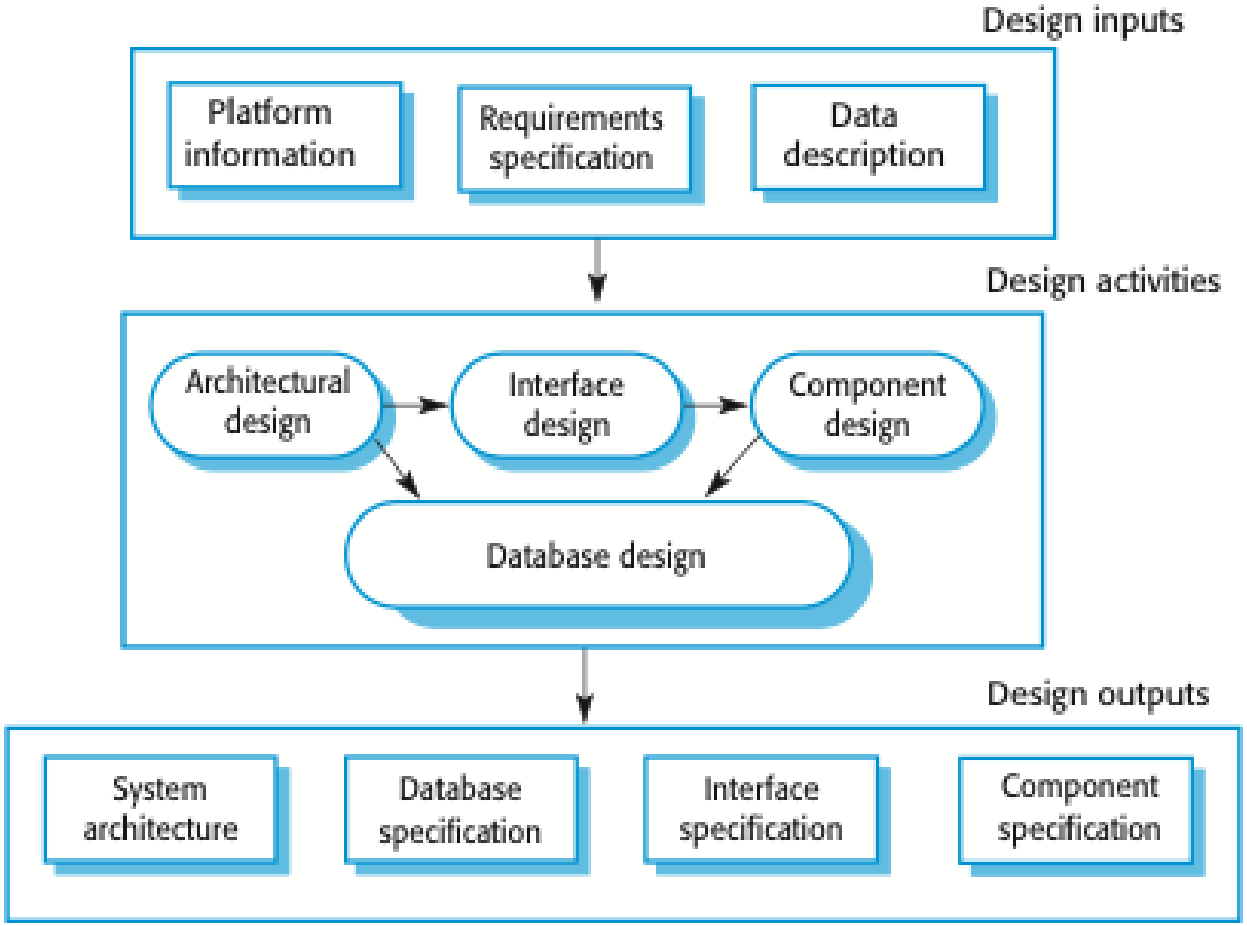
- Design a software structure that realises the specification;

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 (sometimes called sub-systems or modules), their relationships and how they are distributed.

Interface design, where you define the interfaces between system components.

Component design, where you take each system component and design how it will operate.

Database design, where you design the system data structures and how these are to be represented in a database.

SOFTWARE VALIDATION

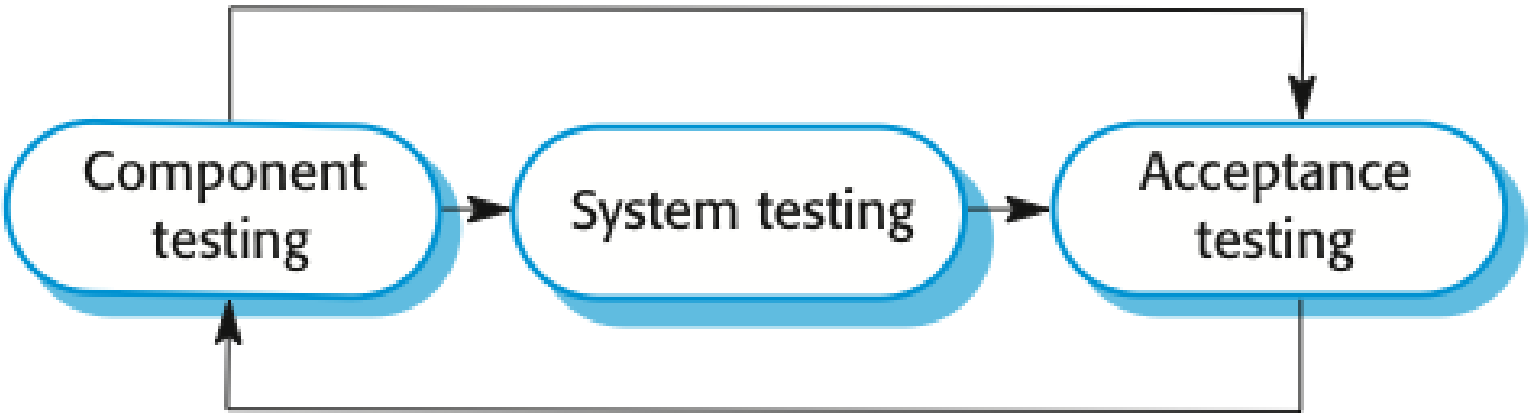
Verification and validation (V & V) is intended to show that a system conforms to its specification and meets the requirements of the system customer.

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

Development or component testing

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

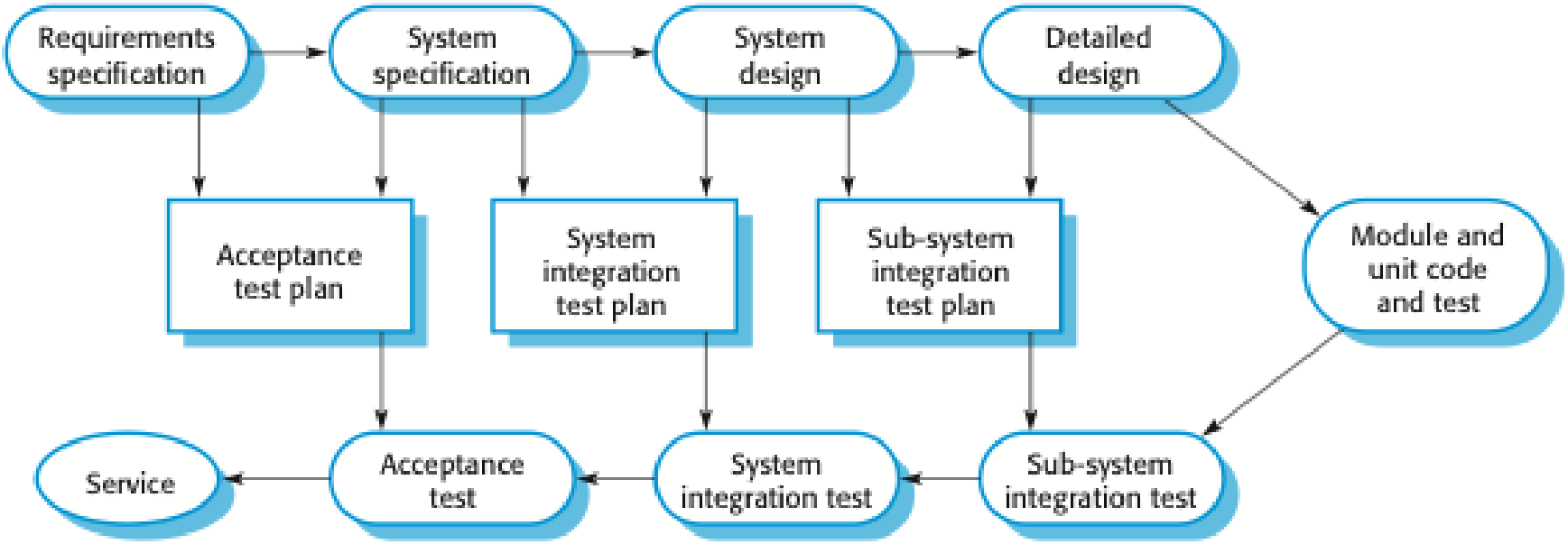
System testing

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

Acceptance testing

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

TESTING PHASES IN A PLAN-DRIVEN SOFTWARE PROCESS



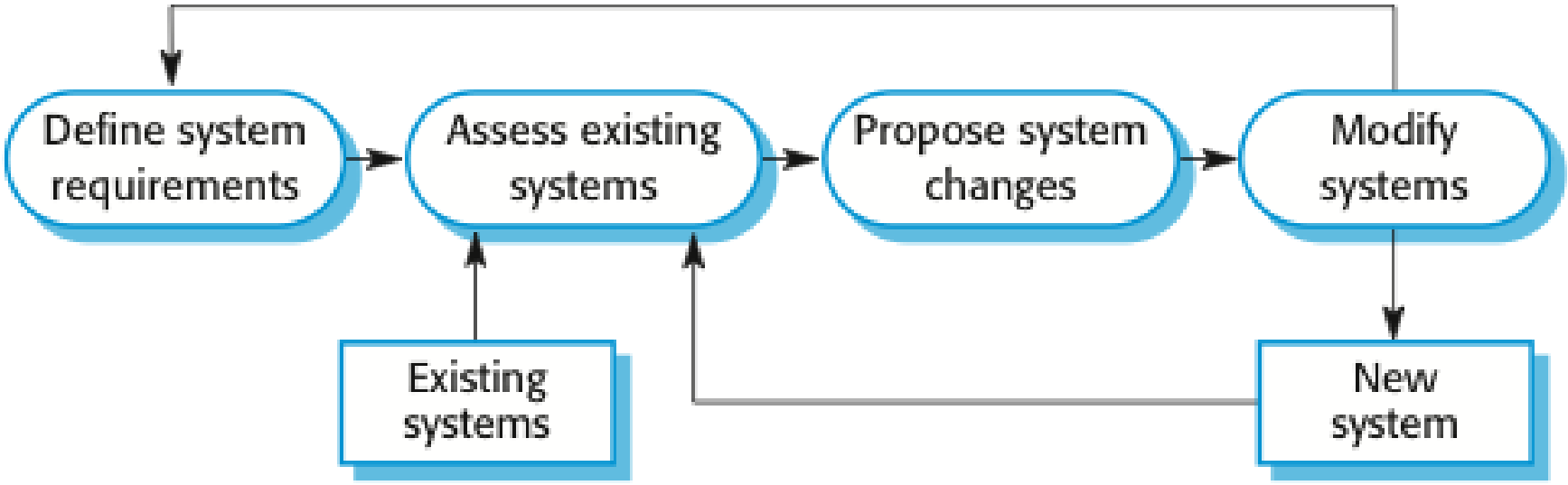
SOFTWARE EVOLUTION

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



KEY POINTS

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 reuse-oriented development.

KEY POINTS

Requirements engineering is the process of developing a software specification.

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. The software must evolve to remain useful.