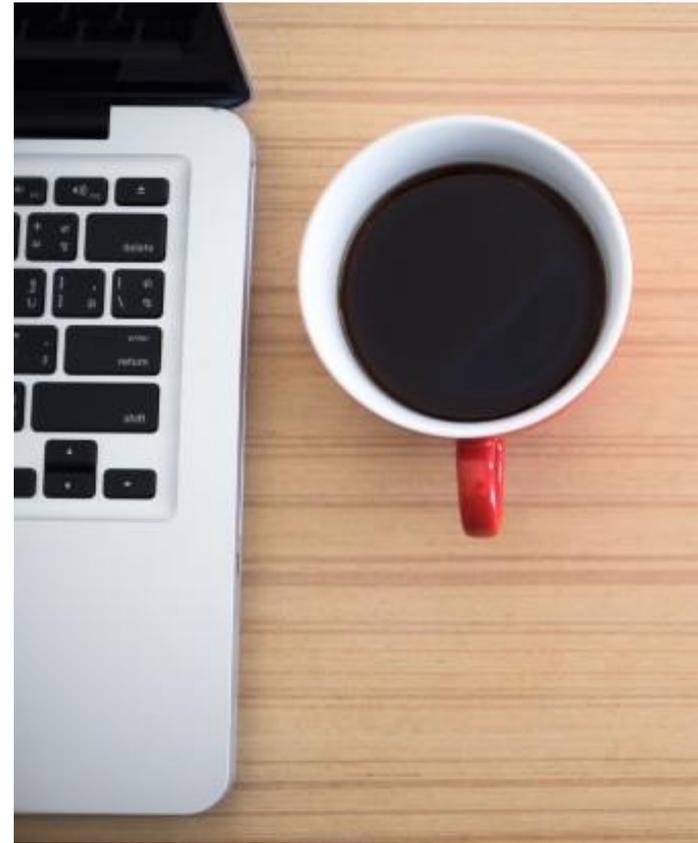


Course Topics

- Introduction
- Software Process Models
- Requirements Engineering
- Modeling
- Programming Languages
- Software Construction Techniques
- Testing
- Project Management
- Refactoring
- Ethical Issues

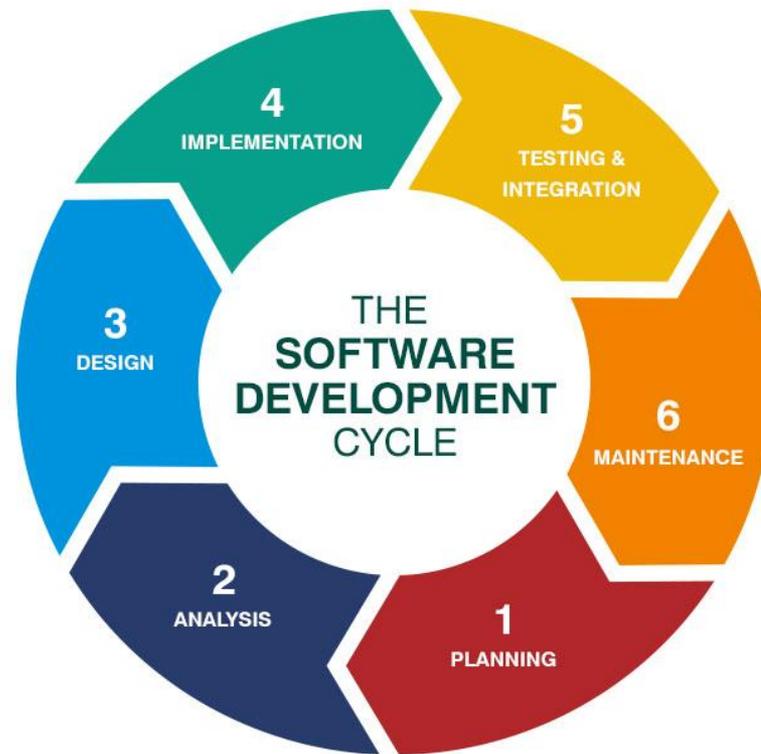
Lecture Objectives

- ✓ What is Software Lifecycle?
- ✓ Software processes
 - Principle characteristics.
- ✓ Software process models
 - Waterfall model
 - Incremental model
 - Component based software development.



What is a Software Development Life Cycle?

- Describes the life of the software from conception through its implementation, delivery, use and maintenance.



Software Development Process



- A set of ordered tasks to produce output of some kind
- Involving activities, constraints and resources.

A set of activities whose goal is the development and evolution of software.



Software Development Process



- Generic activities in all software processes are
 - Specification – what the system should do and its development constraints.
 - Development – production of the software system.
 - Validation – checking that the software is what the customer wants.
 - Evolution – changing the software in response to changing demands.

Why we need software process?



- Common understanding of the activities, resources and constraints involved in software development.

 - Creating processes helps
 - Find inconsistencies
 - Redundancies
 - Omissions

 - There is no universal process model that is right for all kinds of software development
 - Customer and regulatory requirements
 - Environment where the software will be used
 - The type of software being developed.
- 

Software process models

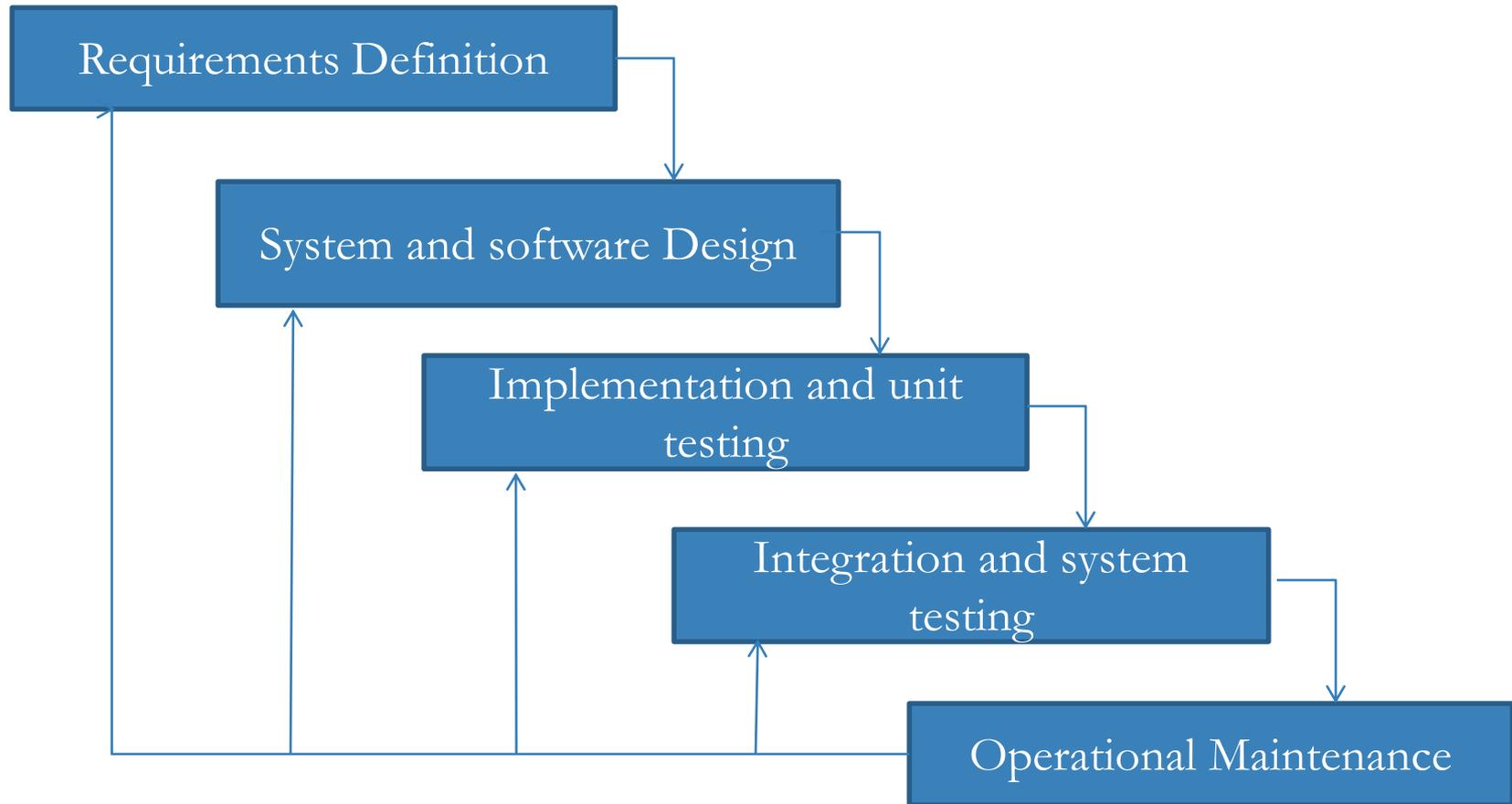


- The waterfall model
 - Separate and distinct phases of specification and development.

 - Incremental development
 - Specification, development and validation are interleaved.
 - Series of versions (increments), with each version adding functionality to the previous version.

 - Integration and configuration
 - Relies on the availability of reusable components or systems.
 - Focuses on integrating and configuring these components.
 - Also called: Component-based software development
- 

Waterfall Model



Waterfall Model



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



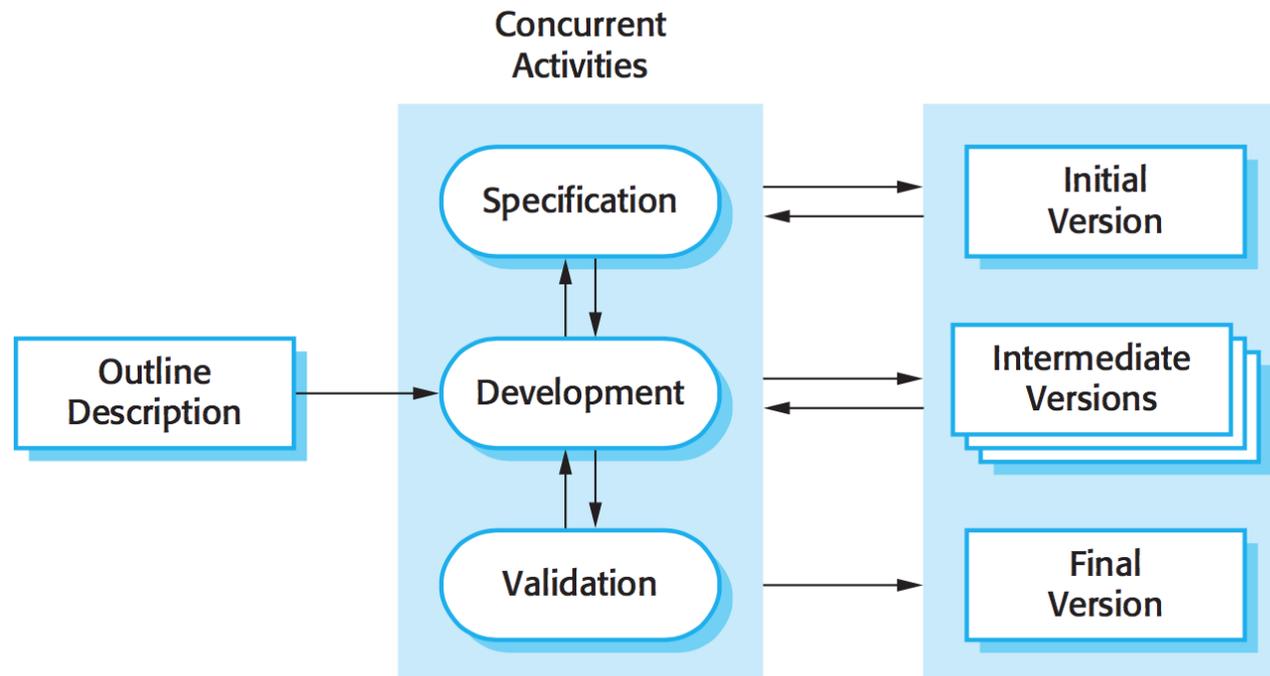
Waterfall Model is appropriate for:



- *Embedded systems*: Software has to interface with hardware systems. Because of the inflexibility of hardware, decisions about software functionalities cannot be delayed.
 - *Critical systems*: Need for extensive safety and security analysis of the software specification and design. Safety related problems in the specification and design are usually very expensive to correct at the implementation stage.
 - *Large software systems* where several companies are involved, complete specifications may be needed to allow for the independent development of different subsystems.
- 

Incremental Development

- Develop and initial implementation, get feedback from stakeholders and evolve the software through several versions.
 - Specification, development and validation are interleaved with rapid feedback across activities.



Incremental Development



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

Incremental development problems



From a management perspective:

- 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.
- 

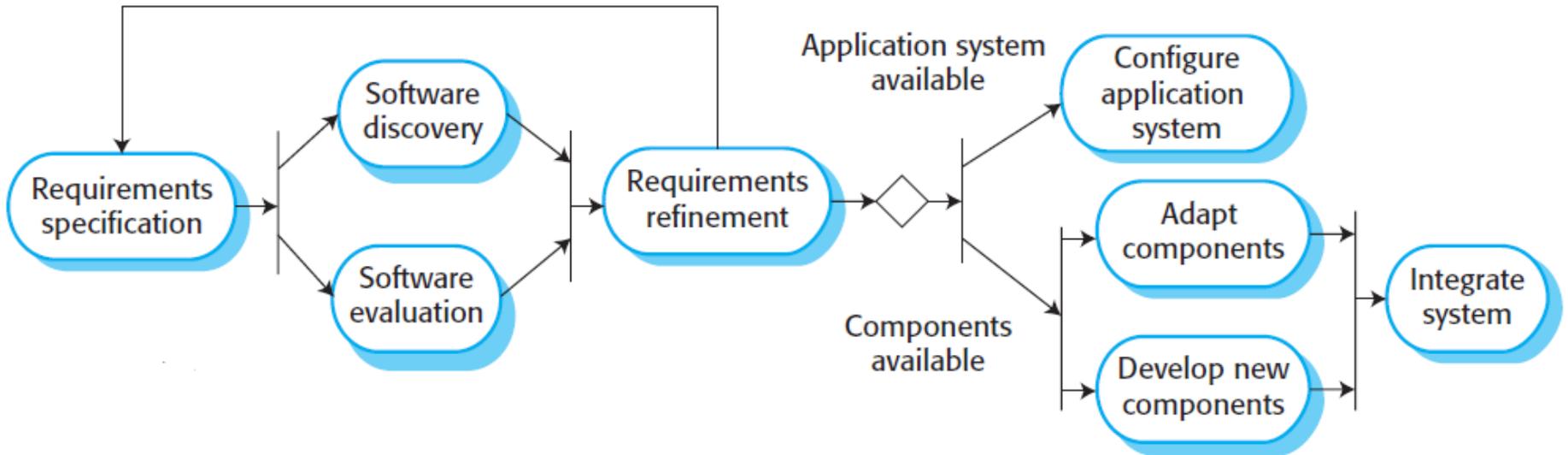
Integration and Configuration

- Also known as:
 - Component Based Software Development

- Systematic reuse where systems are integrated from
 - In-house developed components; OR
 - COTS (Commercial-off-the-shelf) components.

This approach is becoming increasingly used as component standards have emerged.

Integration and Configuration Process



Integration and configuration



- Types of components
 - 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.

Example: www.componentsource.com

24/5 Customer Service Live Help +44 118 958 1111

Help Contact Us My Account Logon English ▾

All Products ▾

USD (US\$)

COMPONENTS ▾ APPLICATIONS ▾ ADD-INS ▾ CLOUD SERVICES ▾ BRANDS ▾

Trusted for 20 Years Over 580,000 licenses delivered to Developers, SysAdmins, Corporations, Governments & Resellers, worldwide. [Read more about us.](#)

The Software Superstore for Developers & IT Pros

Evaluate 1,000s of Products for Application Development & Lifecycle Management.

[Product A-Z](#)

[Brand A-Z](#)

Better tools make better apps.

Choose from our comprehensive range of products dedicated to Software Development and Operations.

[Shop Applications](#)

Filter by: Windows, Mac OS X, Linux, Unix

Components (1993)

Presentation Layer, Business Layer, Data Layer, Cross Cutting

Applications (754)

Development & Deployment, General Purpose, Systems & Operations

Add-ins (305)

Development & Deployment, General Purpose, Systems & Operations

Key Points



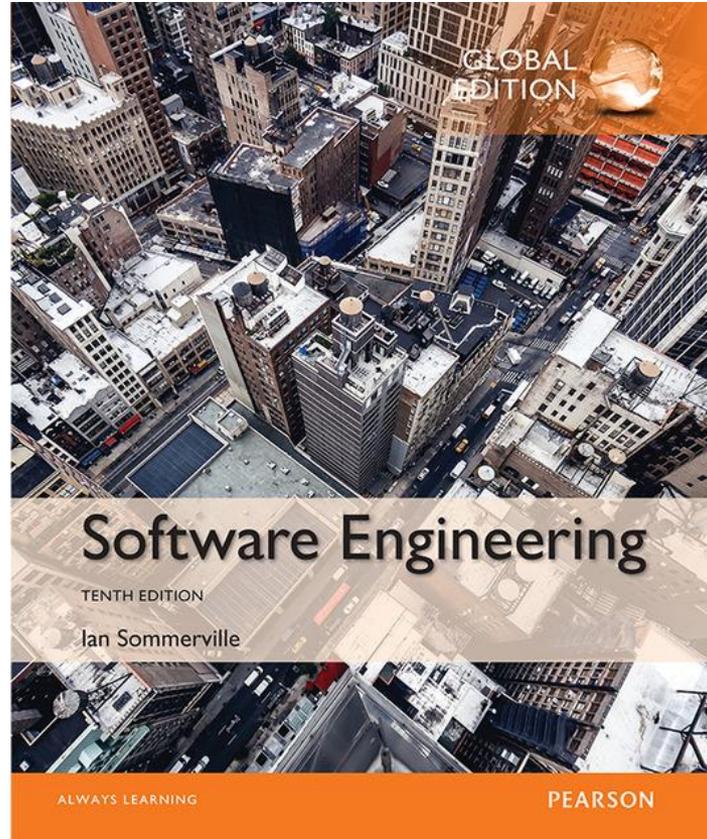
- Software Development Process
 - A set of activities whose goal is the development or evolution of software.

 - Generic activities in all software processes are
 - Specification, development, validation and evolution.

 - Generic software process models
 - Waterfall
 - Incremental
 - Component based
- 

Read

Chapter 2



References



- Ian Sommerville, “Software Engineering”, 10th Edition, Addison-Wesley, 2015.
 - Timothy C. Lethbridge and Robert Laganière, “Object-Oriented Software Engineering: Practical Software Development using UML and Java”, 2nd Edition, McGraw Hill, 2001.
 - R. S. Pressman, Software Engineering: A Practitioner’s Approach, 10th Edition, McGraw-Hill, 2005.
- 