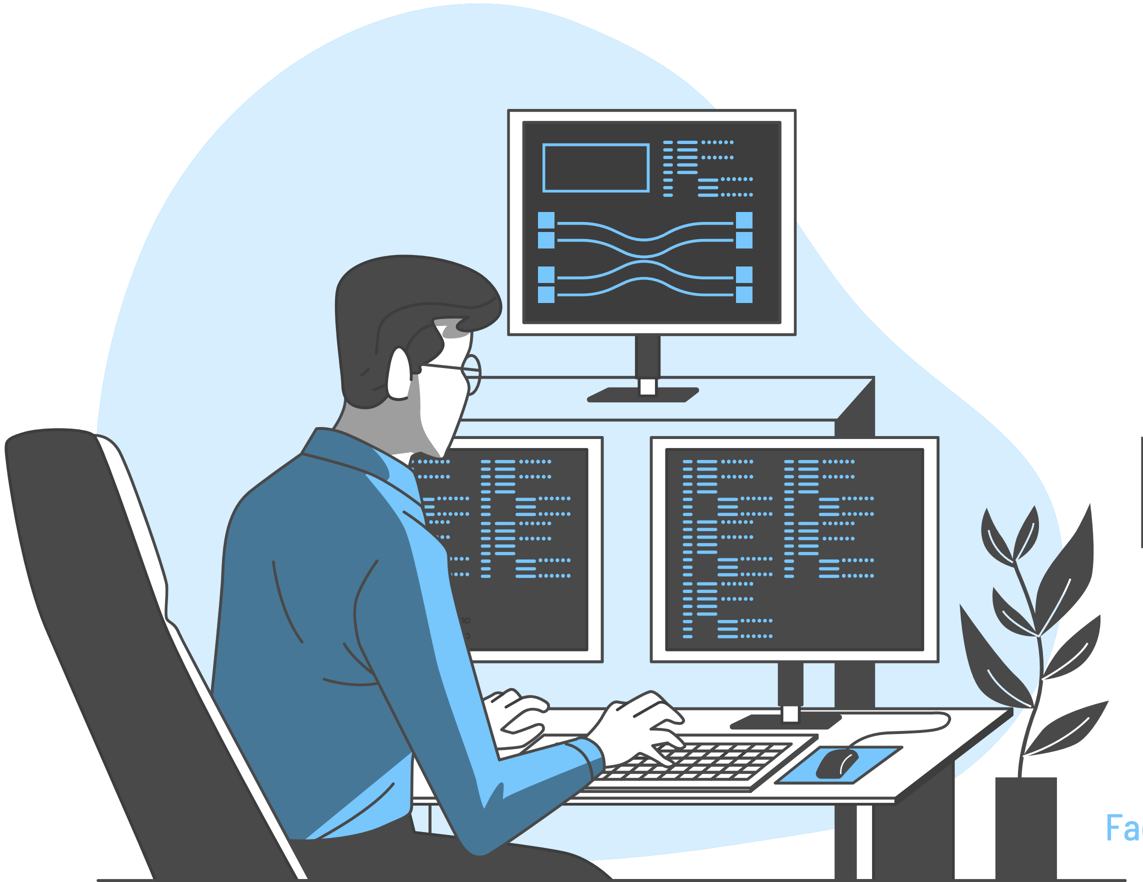


Software Engineering

Lecture 3

Dr. Sarah Ayyad
Faculty of engineering, Mansoura university

Plan-driven and agile processes

Software process models are categorized as either plan-driven or agile processes.

01

Plan driven processes

Where all of the process activities are planned in advance and progress is measured against this plan.

02

Agile processes

Planning is incremental and it is easier to change the process to reflect changing customer requirements.

There is no 'ideal' software process.
In practice, most practical processes include elements of both plan-driven and agile approaches.

Software Process Models

Traditional software process models/frameworks

- Waterfall Model
- Incremental development
- Reuse-oriented software engineering

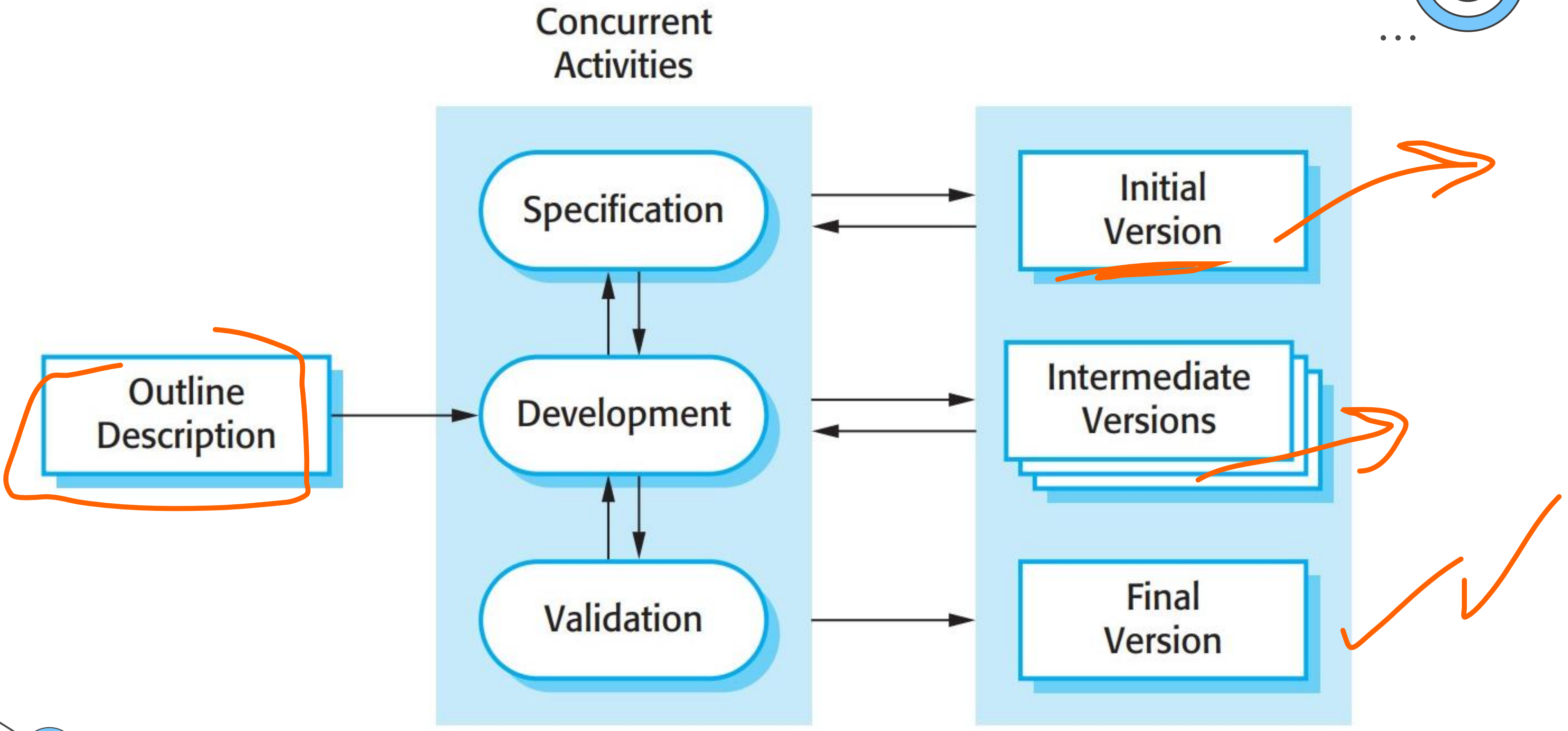
➢ All models have the same goals: Manage risks and produce high quality software.

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

02

Incremental development

Incremental development



This approach interleaves the activities of specification, development, and validation.

Incremental development

- Incremental development is based on the idea of developing an initial implementation, exposing this to user comment and evolving it through several versions (increments) until an adequate system has been developed.
- Each increment or version of the system incorporates some of the functionality that is needed by the customer. Generally, the early increments of the system include the most important or most urgently required functionality. This means that the customer can evaluate the system at an early stage in the development to see if it delivers what is required.
- Incremental software development, which is a fundamental part of agile approaches, is better than a waterfall approach for most e-commerce and personal systems.

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. Customers find it difficult to judge progress from software design documents.

- More rapid delivery and deployment of useful software to the customer is possible, even if all the functionality has not been included. Customers can 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.

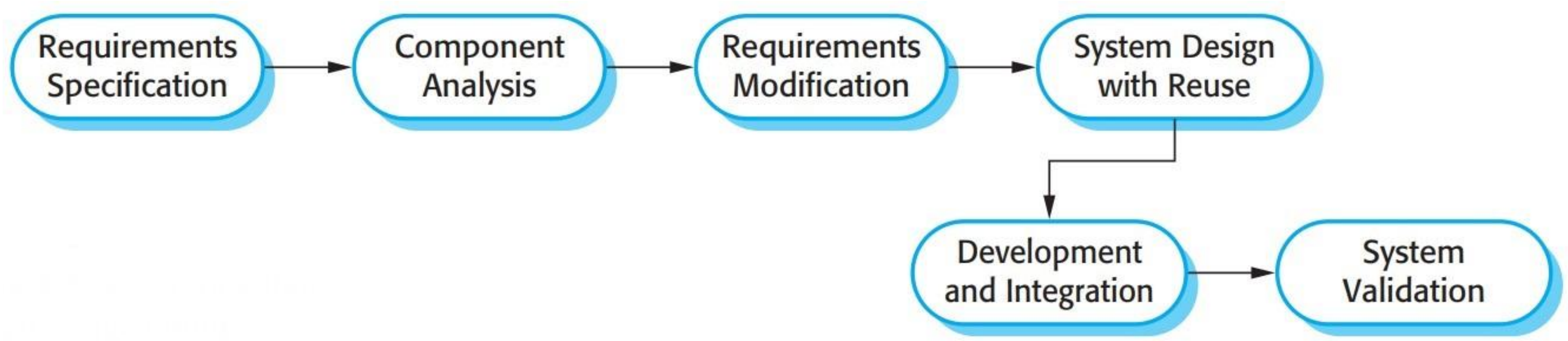
Reuse-oriented software engineering

03

Reuse-oriented software engineering

- Reuse-oriented approaches rely on a large base of reusable software components and an integrating framework for the composition of these components.
- May be plan-driven or agile.
- Sometimes, these components are systems in their own right (COTS or commercial off-the-shelf systems) that may provide specific functionality such as word processing or a spreadsheet.
- Reused elements may be configured to adapt their behavior and functionality to a user's requirements.
- The intermediate stages in a reuse-oriented process are different. These stages are; component analysis, requirements modification, system design with reuse, and development and integration.

Reuse-oriented software engineering


Requirements Specification
Component Analysis
Requirements Modification
System Design with Reuse
Development and Integration
System Validation

Reuse-oriented software engineering intermediate stages

➢ Component analysis:

- A search is made for components to implement that specification.
- Usually, there is no exact match and the components that may be used only provide some of the functionality required.

➢ Requirements modification:

- During this stage, the requirements are analyzed using information about the components that have been discovered.
- They are then modified to reflect the available components.
- Where modifications are impossible, the component analysis activity may be re-entered to search for alternative solutions.

Reuse-oriented software engineering intermediate stages

- System design with reuse:
  - During this phase, the framework of the system is designed or an existing framework is reused.
  - The designers consider the components that are reused and organize the framework to cater for this.
  - Some new software may have to be designed if reusable components are not available.
- Development and integration.
  - Software that cannot be externally procured is developed, and the components and COTS systems are integrated to create the new system.
  - System integration, in this model, may be part of the development process rather than a separate activity.

Types of software component that may be used in a reuse-oriented process

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

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

03 **Stand-alone software systems**
that are configured for use in a particular environment

Advantages

Reducing the amount of software to be developed and so reducing cost and risks.

It usually also leads to faster delivery of the software.

Disadvantages

Requirements compromises are inevitable, and this may lead to a system that does not meet the real needs of users.

Some control over the system evolution is lost as new versions of the reusable components are not under the control of the organization using them.

Process Activities

- Real software processes are interleaved 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 1 specification, 2 development, 3 validation, and 4 evolution are organized differently in different development processes.
- In the waterfall model, they are organized in sequence, whereas in incremental development they are interleaved.

1. Software specification

- Software specification or requirements engineering is the process of understanding and defining what services are required from the system and identifying the constraints on the system's operation and development.

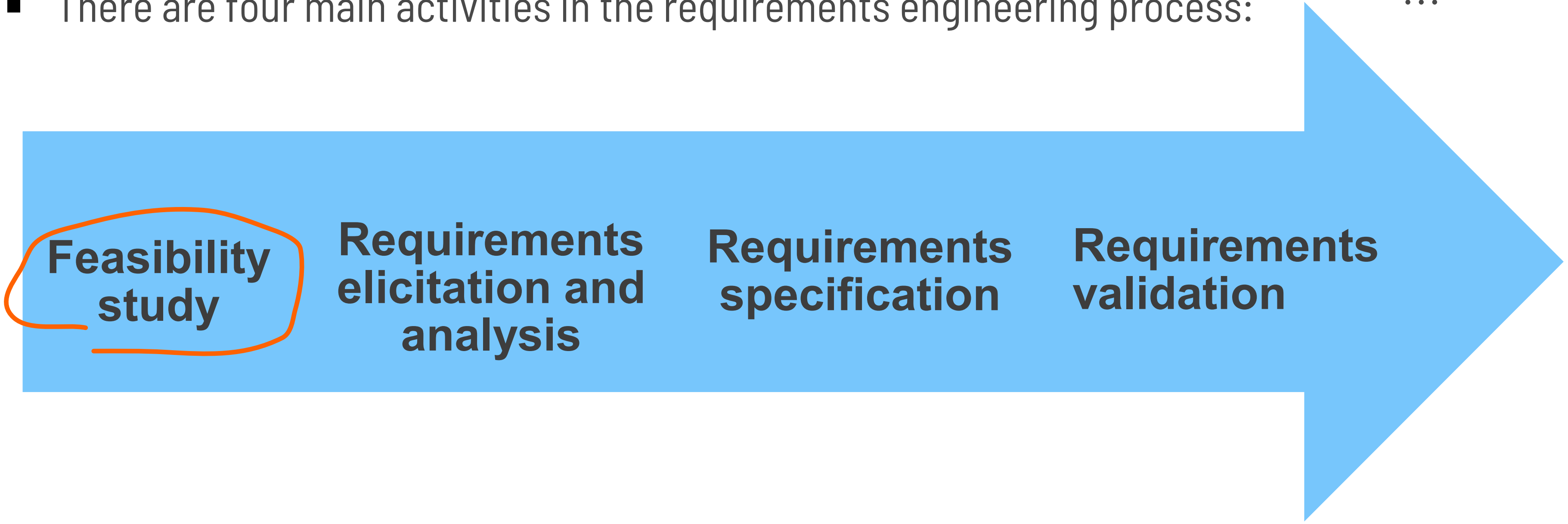
- Requirements engineering is a particularly critical stage of the software process as errors at this stage inevitably lead to later problems in the system design and implementation.

- The requirements engineering process aims to produce an agreed requirements document that specifies a system satisfying stakeholder requirements.

- Requirements are usually presented at two levels of detail. End-users and customers need a high-level statement of the requirements; system developers need a more detailed system specification.

1. Software specification

- There are four main activities in the requirements engineering process:


Feasibility study
Requirements elicitation and analysis
Requirements specification
Requirements validation

1. Software specification

- Feasibility study

Is it technically and financially feasible to build the system?

- Requirements elicitation and analysis

This is the process of deriving the system requirements through observation of existing systems, discussions with potential users and procurers, task analysis, and so on.

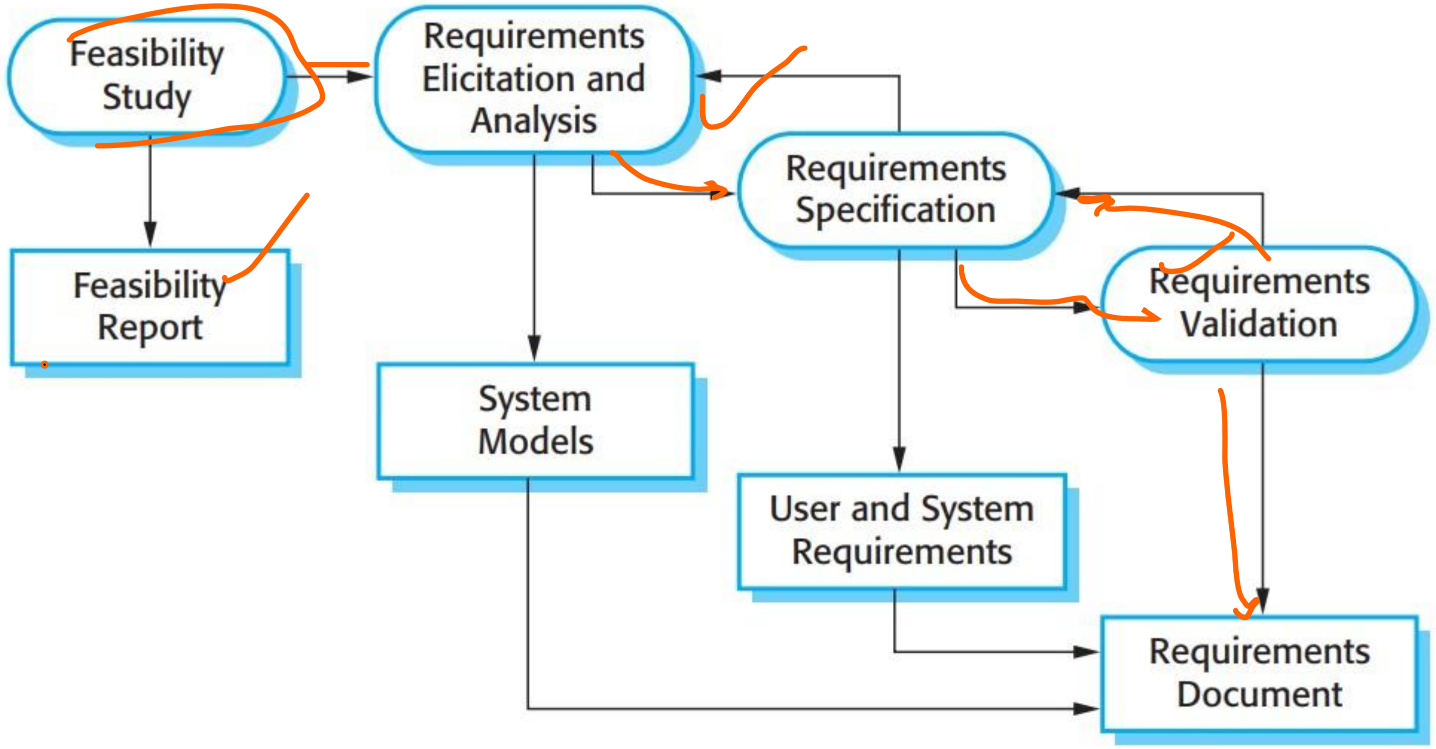
- Requirements specification

The activity of translating the information gathered during the analysis activity into a document that defines a set of requirements. Two types of requirements may be included in this document. User requirements are abstract statements of the system requirements for the customer and end-user; system requirements are a more detailed description of the functionality to be provided.

- Requirements validation

Checking the requirements for realism, consistency, and completeness. During this process, errors in the requirements document are inevitably discovered. It must then be modified to correct these problems.

1. Software specification



Thanks!

Do you have any questions?

Sarah_Ayyad@mans.edu.eg