Software Engineering – CSC 343

Chapter 2

Software Processes

Topics covered

- □ Software process models
- □ Process iteration
- □ Process activities
- □ Computer-aided software engineering

1. Introduction

□ A structured set of activities required to develop a software system

- Specification;
- Design;
- Testing/Validation;
- Evolution.

1. Introduction

- A software process model:
  - is an abstract representation of a process
  - it presents a description of a process from some particular perspective.
- Many organization still rely on ad-hoc processes
  - no use of sw processes methods
  - no use of best practice in sw industry

2. Generic software process models

- □ The waterfall model
  - ■ Separate and distinct phases of specification and development: Requirements, design, implementation, testing, …
  - ■ No evolution process, only development
  - ■ Widely used & practical
  - ■ Recommended when requirements *are well known and stable at start*

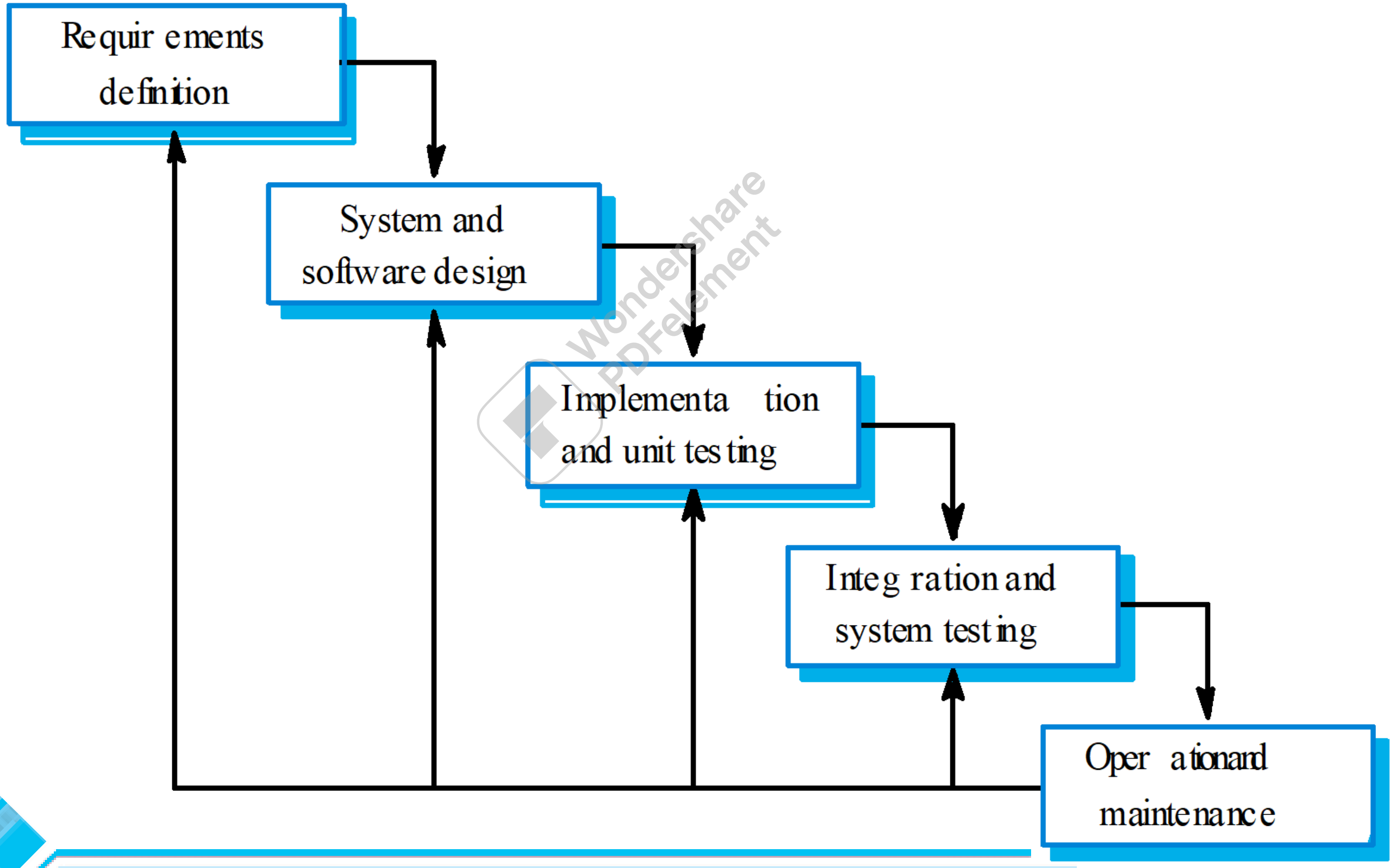
- □ Evolutionary development
  - ■ Specification and development are interleaved
  - ■ Develop rapidly & refine with client
  - ■ Widely used & practical
  - ■ Recommended when requirements *are not well known at start*

Generic software process models

□ Reuse-based (Component-based) development

- The system is *assembled* from existing components
  »Components already developed within the organization
  »COTS “Commercial of the shelf ” components
- Integrating rather than developing
- Allows rapid development
- Gaining more place
- Future trend

Waterfall model


Requir ements definition
System and software design
Implementa tion and unit tes ting
Integ ration and system test ing
Oper ationand maintenance

Waterfall model

- □ The classic way of looking at S.E. that accounts for the importance of requirements, design and quality assurance.
  - The model suggests that software engineers should work in a series of stages.
  - Before completing each stage, they should perform quality assurance (verification and validation).
  - The waterfall model also recognizes, to a limited extent, that you sometimes have to step back to earlier stages.

Limitations of the waterfall model

- The model implies that you should attempt to complete a given stage before moving on to the next stage
  - □ Does not account for the fact that requirements constantly change.
  - □ It also means that customers can not use anything until the entire system is complete.
- The model makes no allowances for prototyping.
- It implies that you can get the requirements right by simply writing them down and reviewing them.
- The model implies that once the product is finished, everything else is maintenance.

Limitations of the waterfall model

- Drawback: the difficulty of accommodating change after the process is underway

- Inflexible partitioning of the project into distinct stages

- Inflexible: to respond to dynamic business environment leading to requirements changes

- Appropriate when the requirements are *well-understood and stable*

Evolutionary development

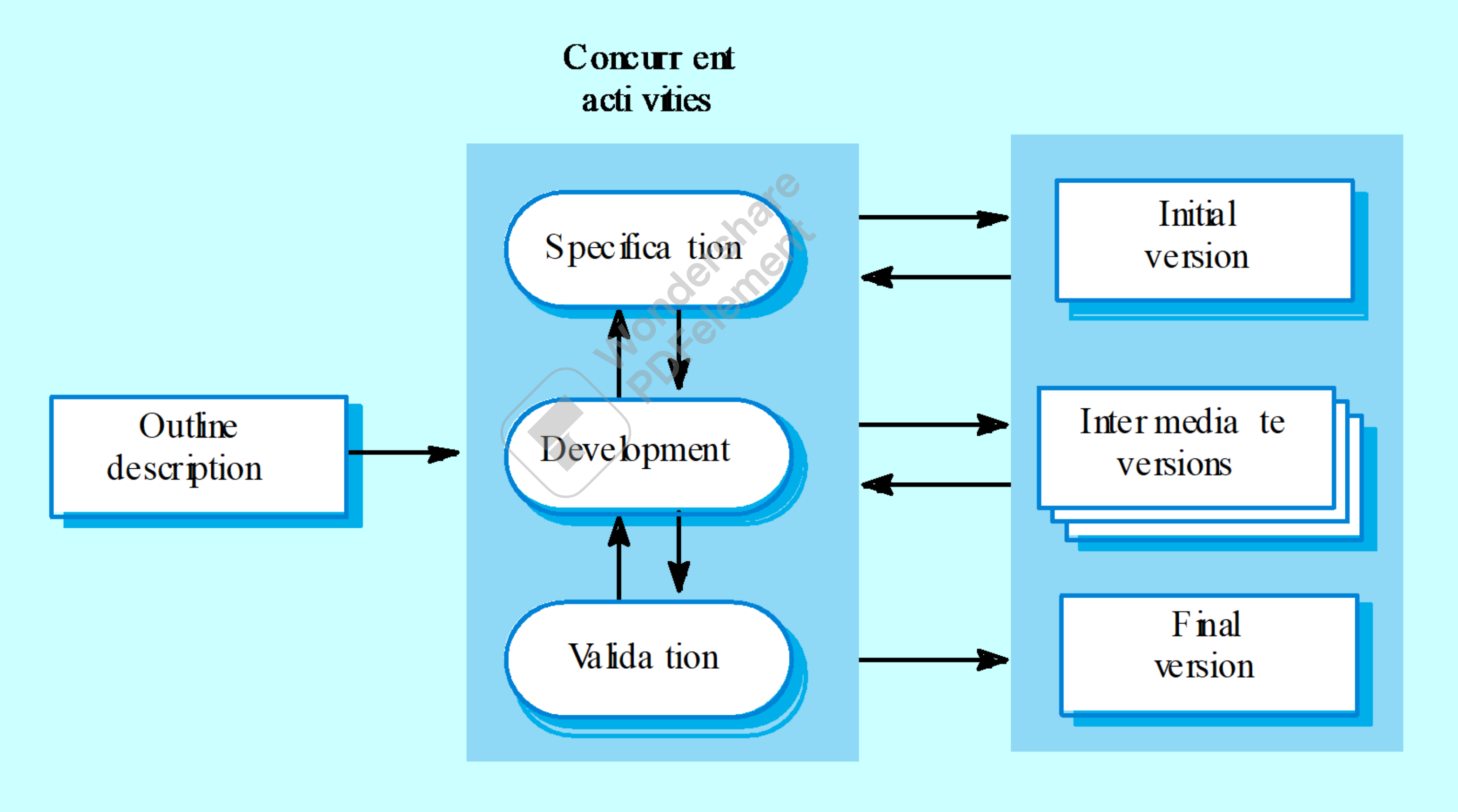
- Develop an initial implementation prototype
- Client test drive …⇨ feed back
- Refine prototype

❑ 2 types of Evolutionary development

Exploratory development

Throw-away prototyping

Evolutionary development



Evolutionary development

2 types of Evolutionary development

- □ Exploratory development
  - ■ Objective is to work with customers, explore their requirements and to evolve a final system from an initial outline specification.
  - ■ Should start with *well-understood* requirements and add new features as proposed by the customer.
- □ Throw-away prototyping
  - ■ Objective is to understand the system requirements and outline abetter definition of requirements.
  - ■ Should start with poorly understood requirements to clarify what is really needed.

Evolutionary development

Problems

- Lack of process visibility at client management level (less regular reports/documentation … the system is changing continuously)
- Systems are often poorly structured
- Special skills (e.g. in languages/tools for rapid prototyping) may be required

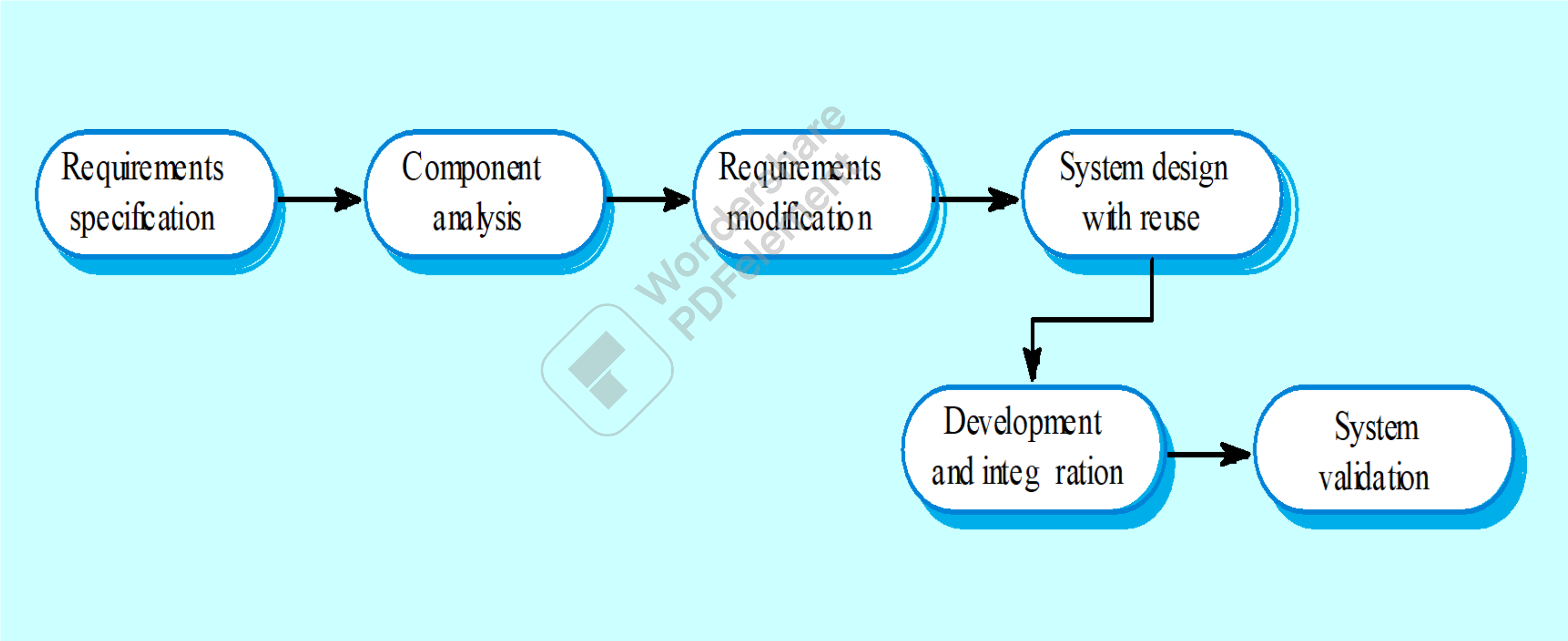
Applicability

- For small or medium-size interactive systems
- For parts of large systems (e.g. the user interface)
- For short-lifetime systems

Component-based 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.
- □ This approach is becoming increasingly used as component standards have emerged.

Reuse-oriented development


Requirements specification
Component analysis
Requirements modification
System design with reuse
Development and integ ration
System validation

3. Process iteration

- Change is inevitable in all large sw projects. As new technologies, designs and implementation change.
- The process activities are regularly repeated as the system is reworked in response to change requests.
- Iteration can be applied to any of the generic process models.
- Iterative process models present the sw as a cycle of activities.
- The advantage of this approach is that it avoids premature commitments to a specification or design.

Process iteration

- Two (related) approaches:
  - Incremental delivery: the software specification, design and implementation are broken into a series of increments that are each developed in turn.
  - Spiral development: the development of the system spirals outwards from an initial outline through to the final developed system.

Incremental delivery

Software process models - Comparison

- □ **Waterfall model**

  Requirements should be well defined **at start and committed to**

- □ **Evolutionary model**

  Requirements & design decisions may be delayed: Poor structure difficult to maintain

- □ **Incremental development**

  - Is an in-between approach that combines the advantages of these models.
  - Incremental ***prioritized delivery of modules***
  - *Hybrid* of Waterfall and Evolutionary

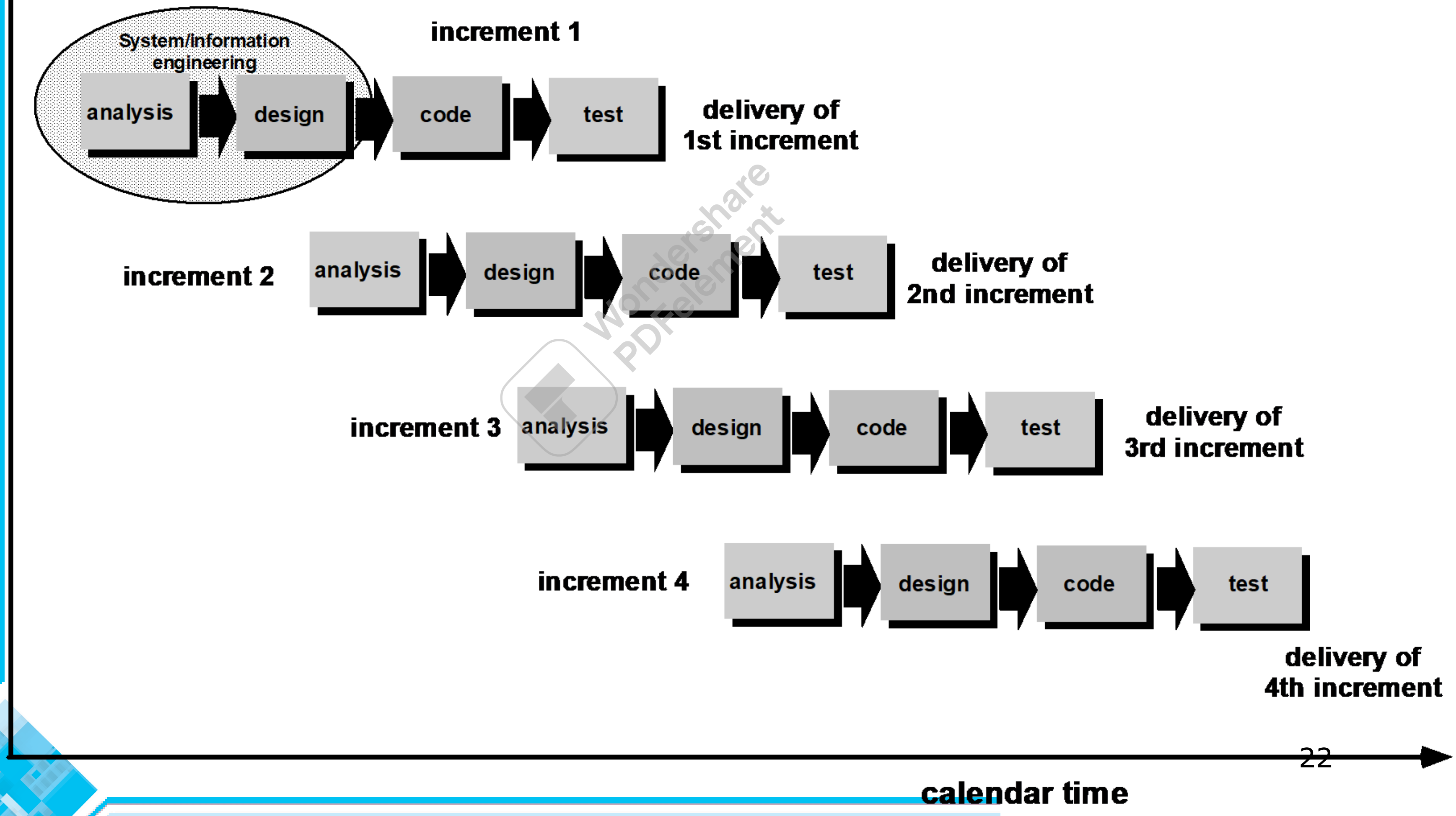
Incremental delivery

- □ Rather than deliver the system as a single delivery, the development and delivery is broken down into increments with each increment delivering part of the required functionality.

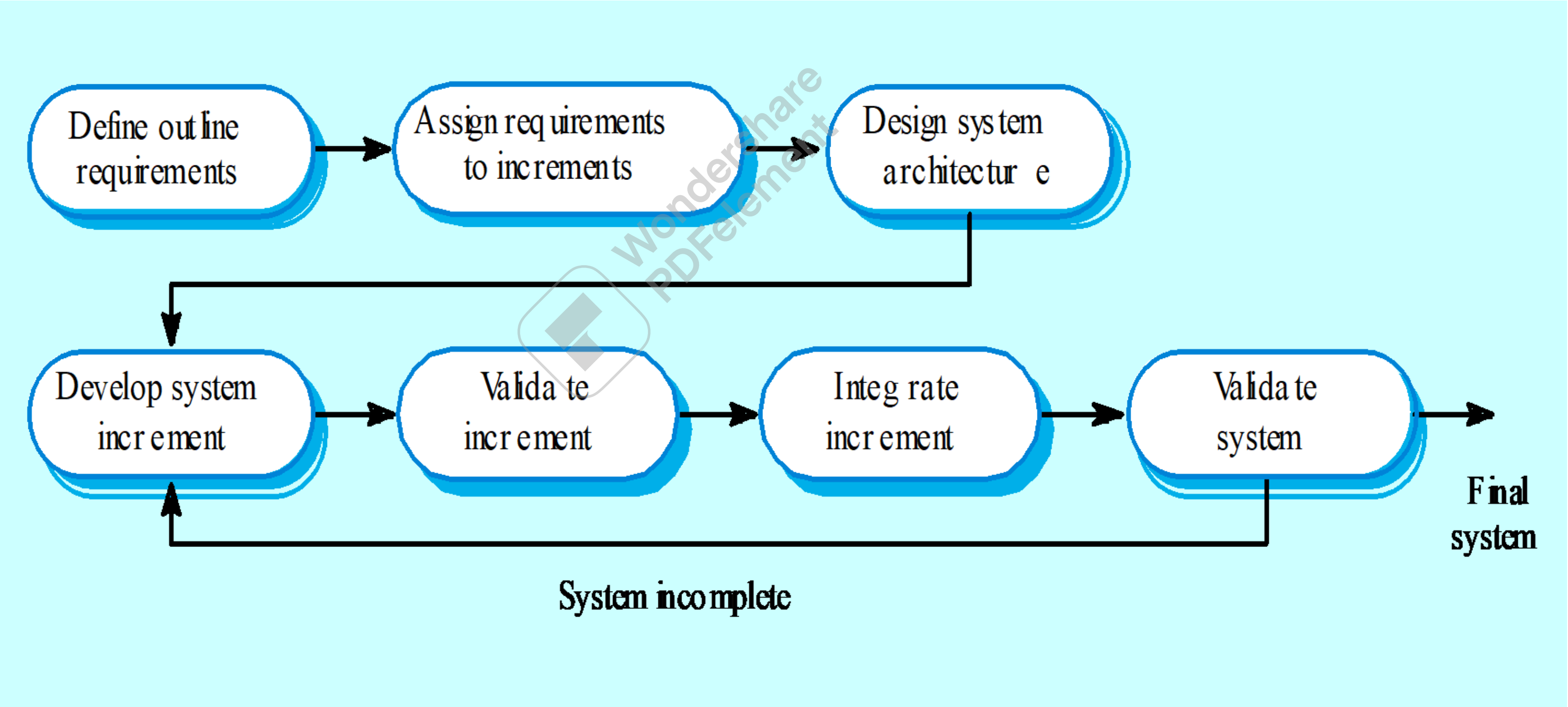
- □ User requirements are prioritised and the highest priority requirements are included in early increments.

- □ Once the development of an increment is started, the requirements are frozen though requirements for later increments can continue to evolve.

Incremental delivery



Incremental development



Incremental development advantages

□ Customer value can be delivered with each increment so system functionality is available earlier.

□ Early increments act as a prototype to help elicit requirements for later increments.

□ Lower risk of overall project failure.

□ The highest priority system services tend to receive the most testing.

Spiral development

- □ Best features of waterfall & prototyping models

  + **Risk Analysis (missed in other models)**

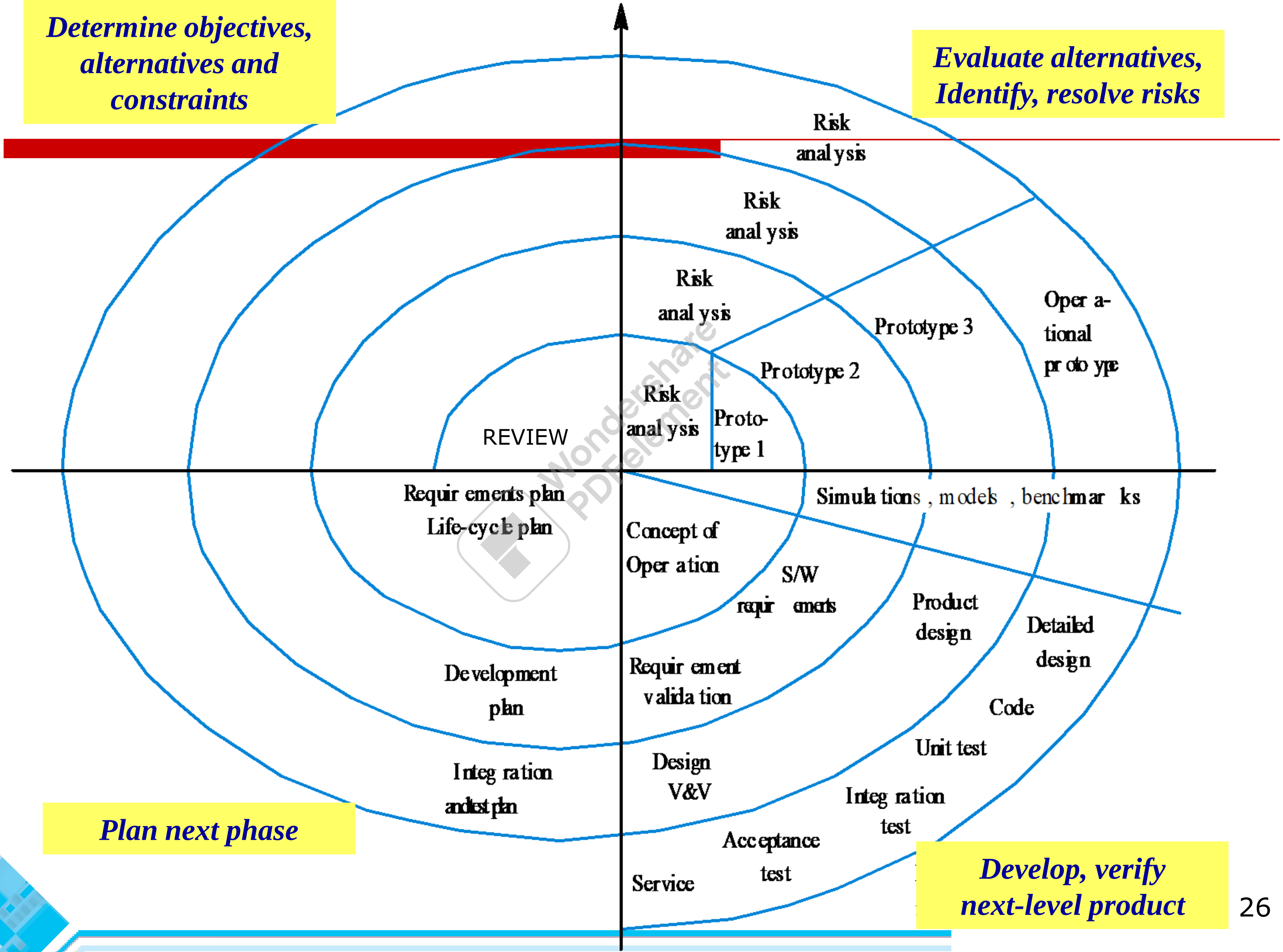
- □ Process is represented as a spiral rather than as a sequence of activities with backtracking.

- □ Each loop in the spiral represents a phase in the process.

- □ Risks are explicitly assessed and resolved throughout the process.

Informally, risk simply means something that can go wrong.

Spiral model of the software process



Spiral development

- □ It explicitly embraces prototyping and an *iterative* approach to software development.
  - ■ Start by developing a small prototype.
  - ■ Followed by a mini-waterfall process, primarily to gather requirements.
  - ■ Then, the first prototype is reviewed.
  - ■ In subsequent loops, the project team performs further requirements, design, implementation and review.
  - ■ The first thing to do before embarking on each new loop is risk analysis.
  - ■ Maintenance is simply a type of on-going development.

Spiral model: 4 sectors

Each loop in the spiral is split into four sectors:

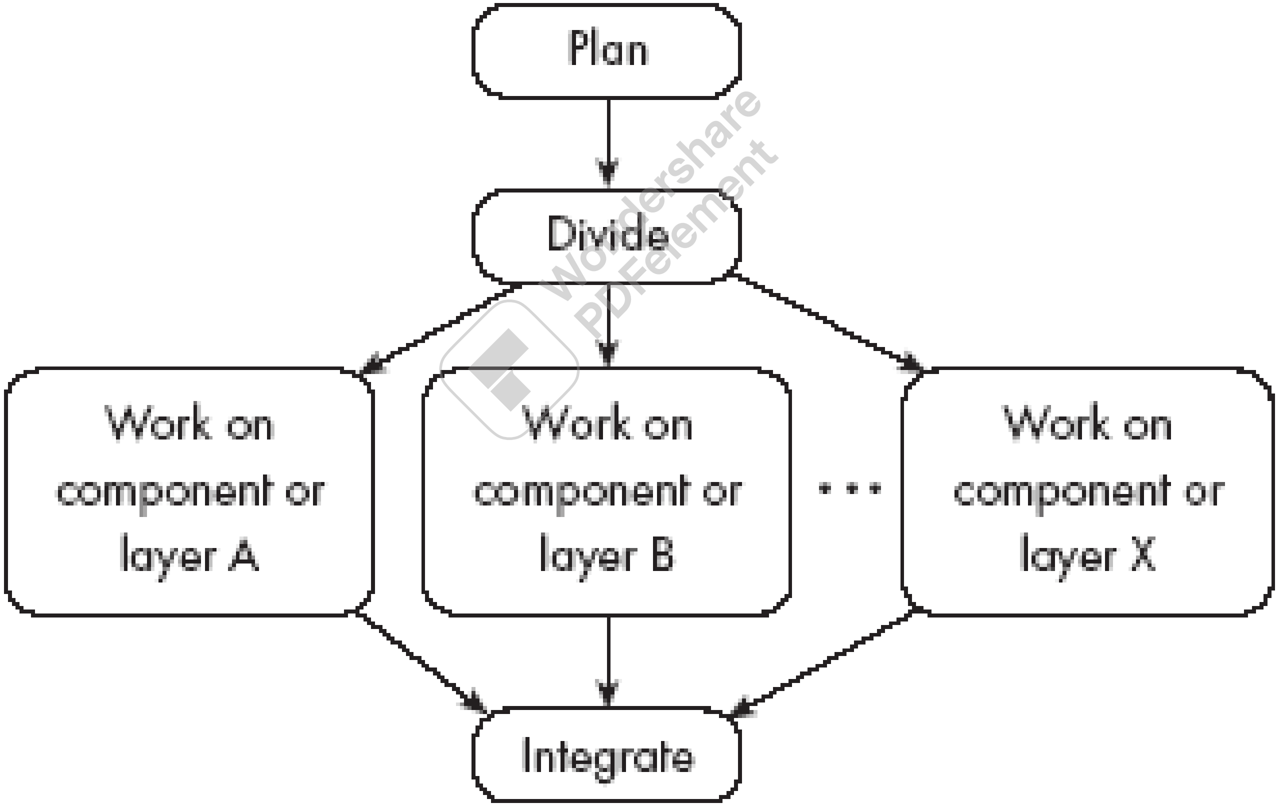
- □ Objective setting
  - Specific objectives for the phase are identified.
- □ Risk assessment and reduction
  - Risks are assessed and activities put in place to reduce the key risks. For example if there is a risk that the requirement. are inappropriate, a prototype system may be developed.
- □ Development and validation
  - A development model for the system is chosen which can be any of the generic models.
- □ Planning
  - Review with client
  - Plan next phase of the spiral if further loop is needed

Spiral model usage

- □ Spiral model has been very influential in helping people think about iteration in software processes and introducing the risk-driven approach to development.

- □ In practice, however, the model is rarely used as published for practical software development.

The concurrent engineering model


Plan
Divide
Work on component or layer A
Work on component or layer B
• • •
Work on component or layer X
Integrate

The concurrent engineering model

- □ It explicitly accounts for the divide and conquer principle.
  - Each team works on its own component, typically following a spiral or evolutionary approach.
  - There has to be some initial planning, and periodic integration.

4. Process Activities

- □ Software specification
- □ Software design and implementation
- □ Software validation
- □ Software evolution

Software specification

Requirements engineering process

□ **The process of establishing**

- ▪ What services are required (Functional requirements) for the system
- ▪ Identifying the constraints on the system's operation and development (Non-functional requirements)

□ **Requirements engineering process**

1. Feasibility study: ***An estimate is made of whether the identified user needs may be satisfied using current software and hardware technologies.***

- Alternatives & Quick cost/benefit analysis
- Feasibility: Technical, Financial, Human, Time schedule
- Deliverables: Feasibility report

2. Requirements elicitation and analysis: Facts finding

- Interviews, JAD "Joint Application Development", Questionnaires, Document inspection, Observation
- Deliverables: System models (Diagrams)

Software specification

Requirements engineering process

□ **Requirements engineering process**

3. Requirements specification: ***the activity of translating the information gathered during the analysis activity into a document that defines a set of requirements.***

- User level: abstract specification
- System level: detailed specification
- Deliverables: User and system requirements

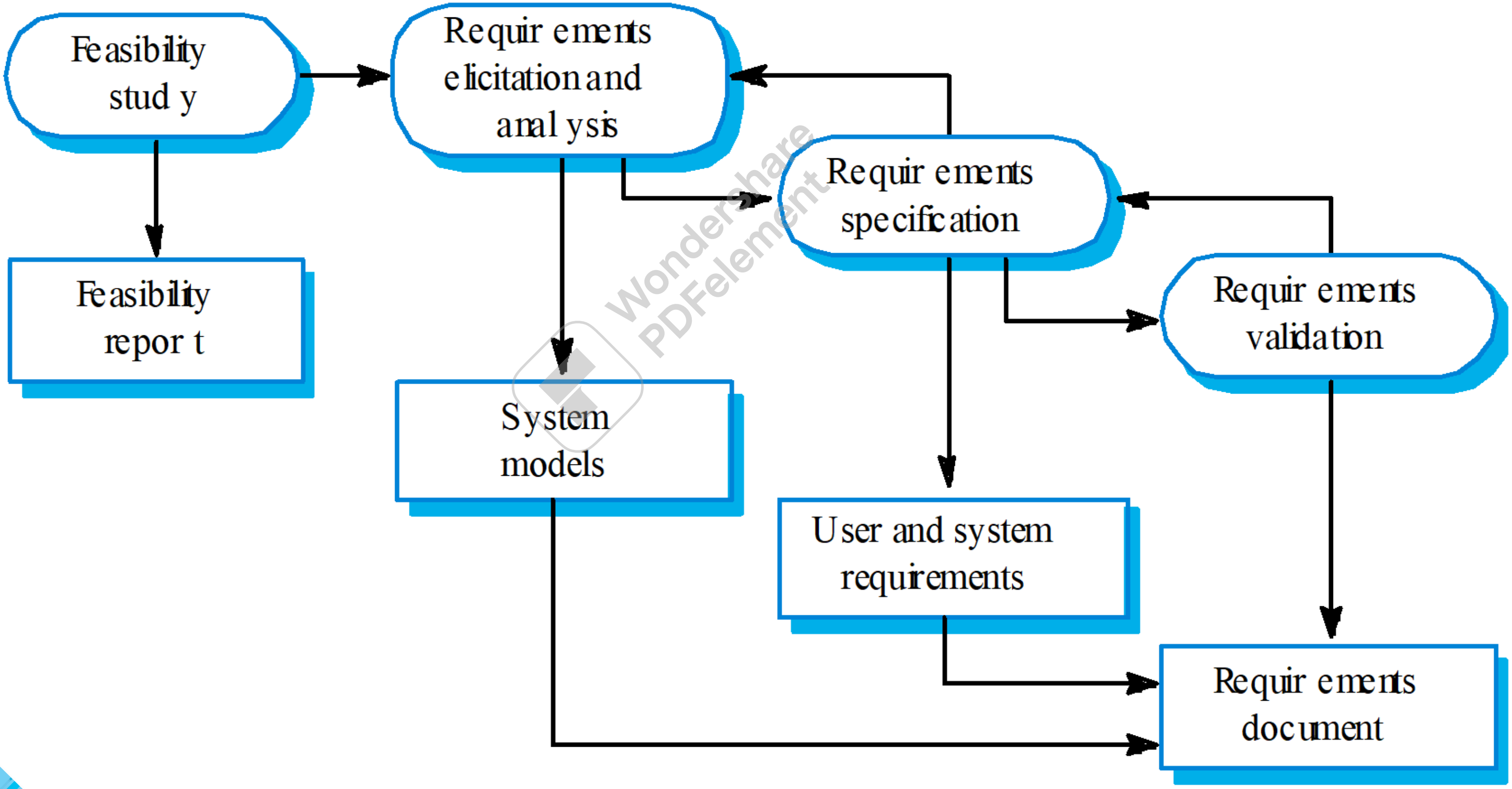
4. Requirements validation: ***this activity checks the requirements for:.***

- Completeness
- Consistency
- Realism
- Deliverables: Updated requirements

Global Deliverables of the Requirements Engineering Process :
System Requirements Specification document

Software specification

Requirements engineering process



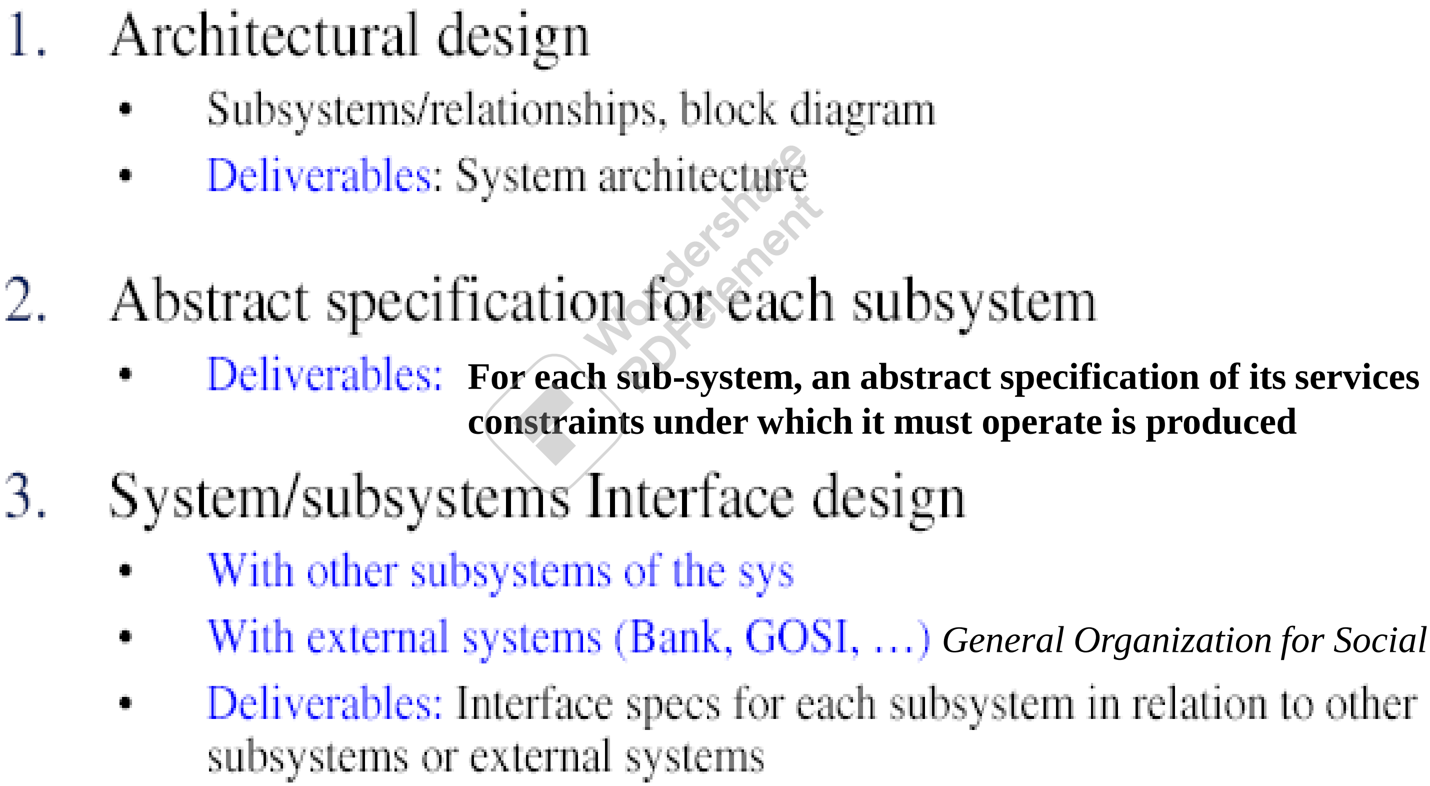
Software design and implementation

- □ **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.**

Design process activities

- □ Architectural design
- □ Abstract specification
- □ Interface design
- □ Component design
- □ Data structure design
- □ Algorithm design

Design Process Activities



1. Architectural design
   - Subsystems/relationships, block diagram
   - Deliverables: System architecture

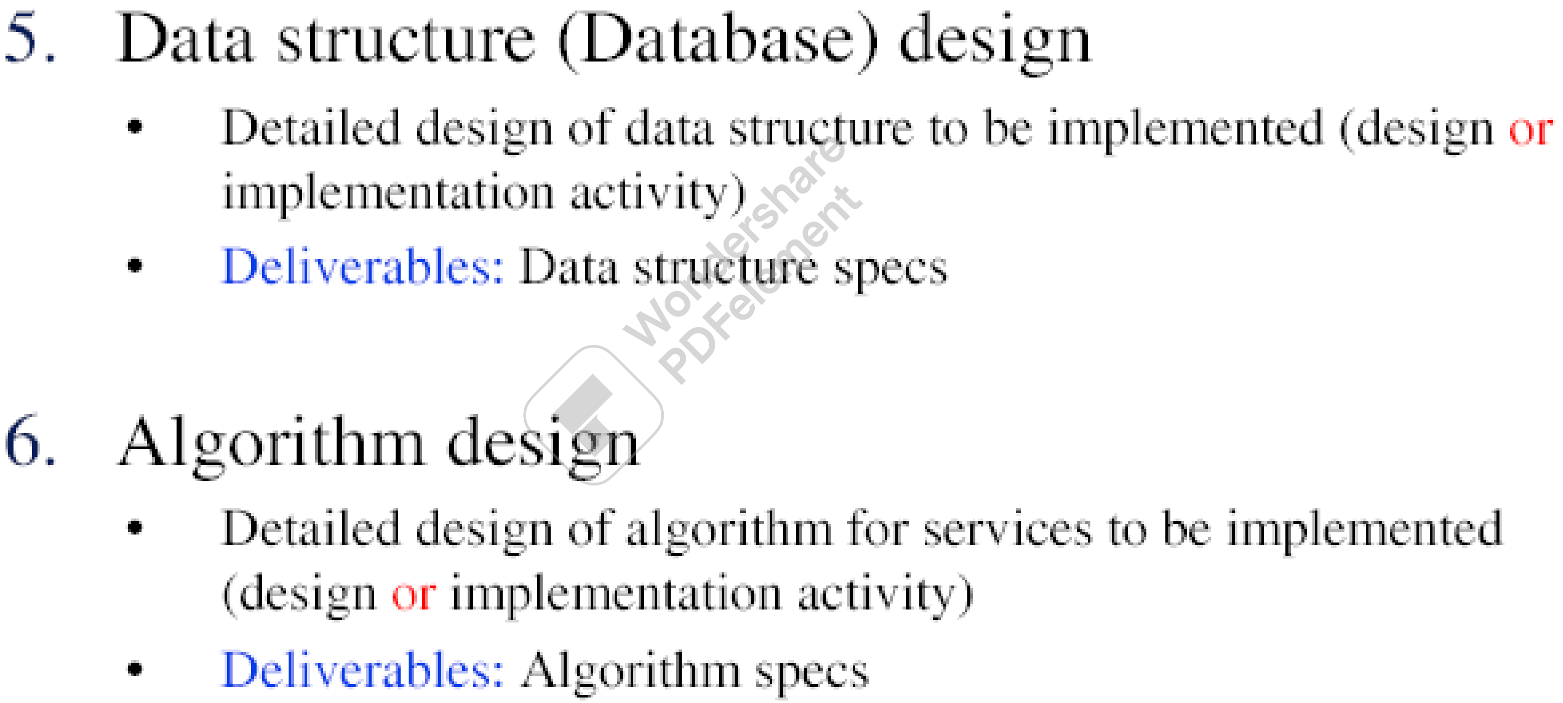
2. Abstract specification for each subsystem
   - Deliverables: **For each sub-system, an abstract specification of its services and constraints under which it must operate is produced**

3. System/subsystems Interface design
   - With other subsystems of the sys
   - With external systems (Bank, GOSI, …) *General Organization for Social Insurance*
   - Deliverables: Interface specs for each subsystem in relation to other subsystems or external systems

Design Process Activities

4. Component design
   - Services are allocated to components
   - Components interfaces are designed
     - Interfaces with other components of the system
     - Interfaces with external systems
     - GUI
     - Input
     - Output
   - Deliverables: Component specs

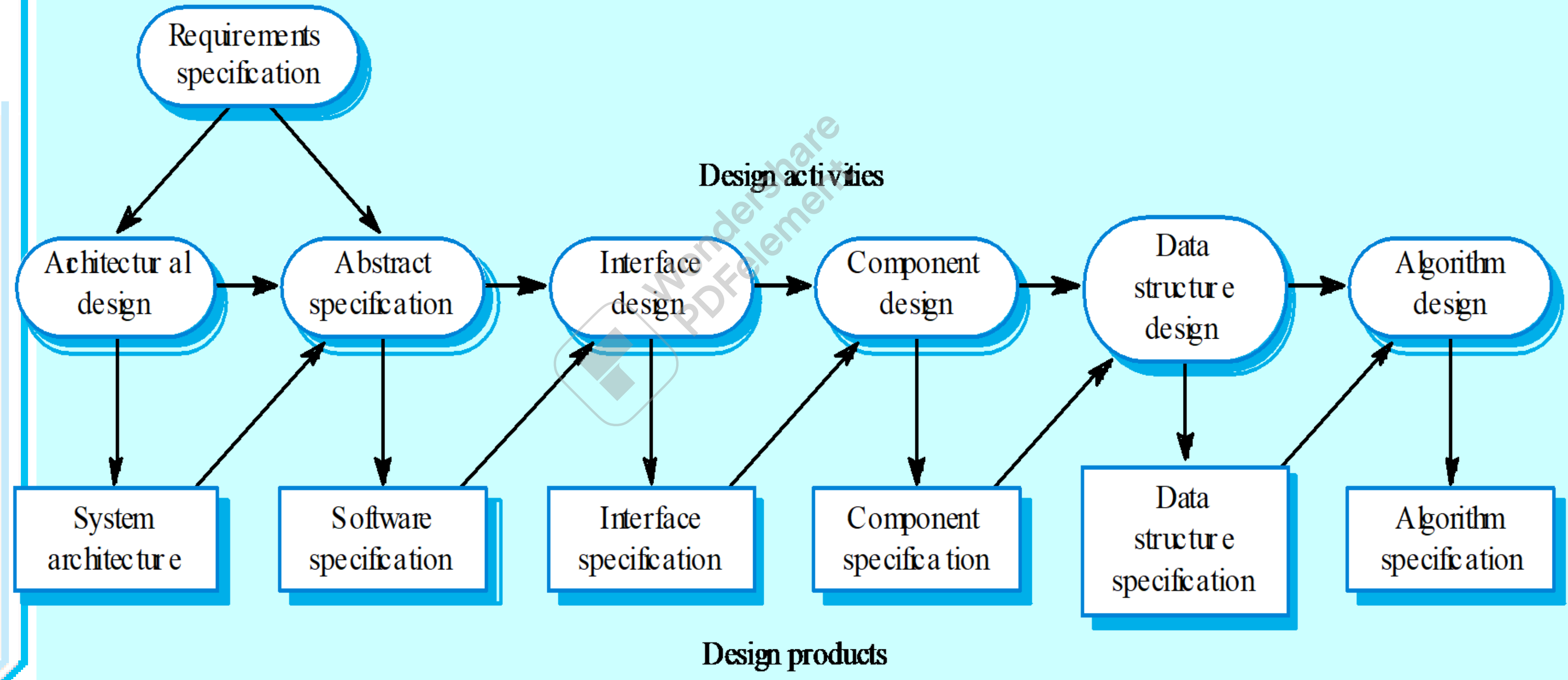
Design Process Activities



5. Data structure (Database) design
   - Detailed design of data structure to be implemented (design or implementation activity)
   - Deliverables: Data structure specs

6. Algorithm design
   - Detailed design of algorithm for services to be implemented (design or implementation activity)
   - Deliverables: Algorithm specs

The software design process


Requirements specification
Design activities
Architectural design
Abstract specification
Interface design
Component design
Data structure design
Algorithm design
System architecture
Software specification
Interface specification
Component specification
Data structure specification
Algorithm specification
Design products