National University
Of Computer and Emerging Sciences

Software Design and Architecture

Implementation Models: Component/package diagram

Deployment diagram

Sajid Anwer

Department of Software Engineering,
FAST-NUCES, CFD Campus

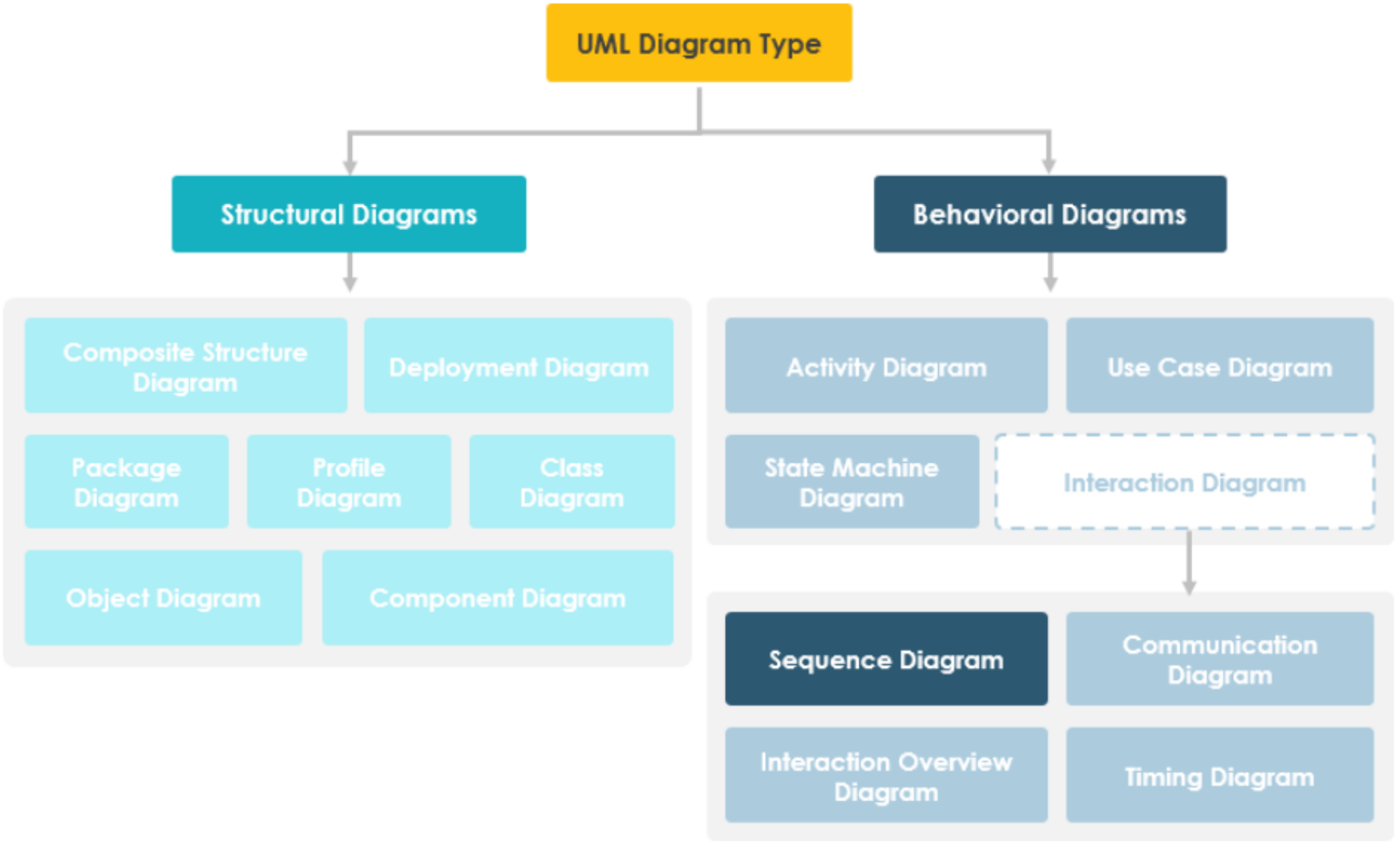
Lecture Material

- System Analysis and Design in a Changing World (Chapter 6, 13)

Lecture Outline

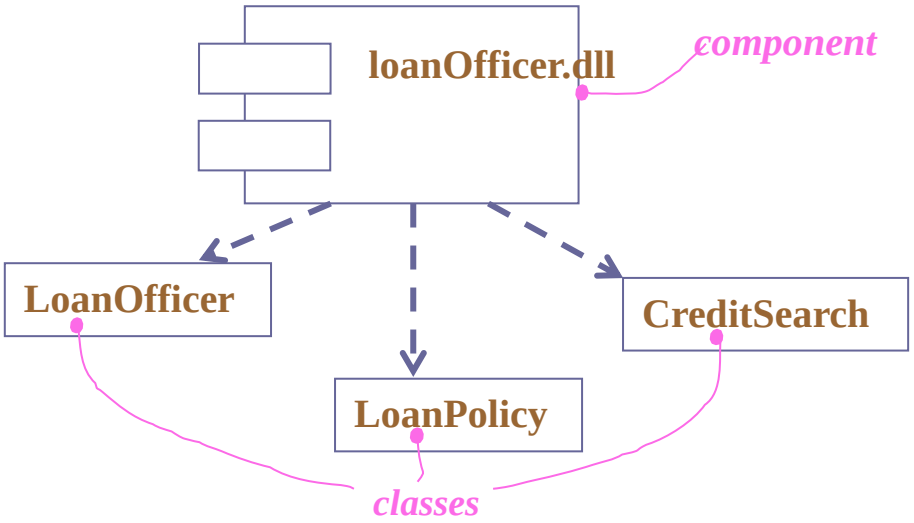
- Implementation Models
  - » Component Diagram
  - » Package Diagram
  - » Deployment Diagram

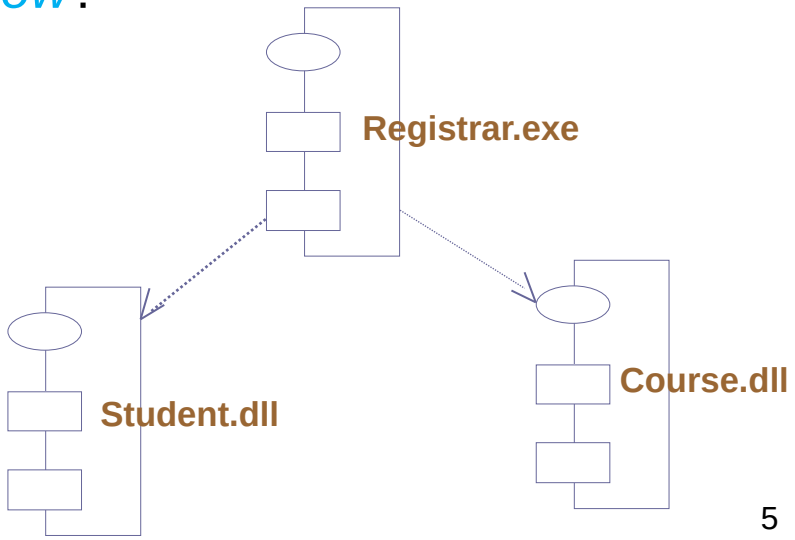
Behavioral Models



Structured Approach--Component Diagrams

- Shows a set of *components* and their *relationships*.
- Represents the *static implementation* view of a system.
  - » Serve as a transition from design to implementation.
- Components map to one or more *classes*, *interfaces*, or *collaborations*.
- Can relates to architecture diagram, *how*?





Component Diagrams

Component Diagrams

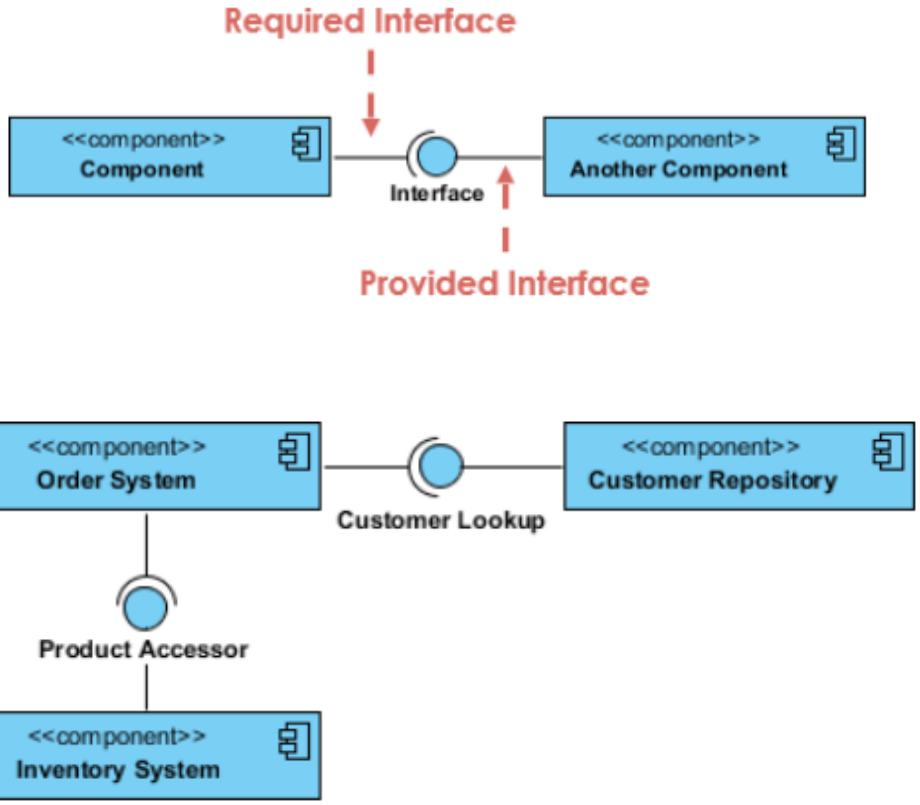
- Modular unit with well-defined interfaces that is replaceable within its environment.
- Autonomous unit within a system.
- Has one or more *provided and required* interfaces.
- Its internals are hidden and inaccessible.
- A component is encapsulated
  - » Its dependencies are designed such that it can be treated as independently as possible.

Component Diagrams -- Notation

- A component is shown as a rectangle with:
  - » A keyword <<component>>
  - » Optionally, in the *right hand corner* a component icon can be displayed
  - » A component icon is a rectangle with two smaller rectangles jutting out from the left-hand side

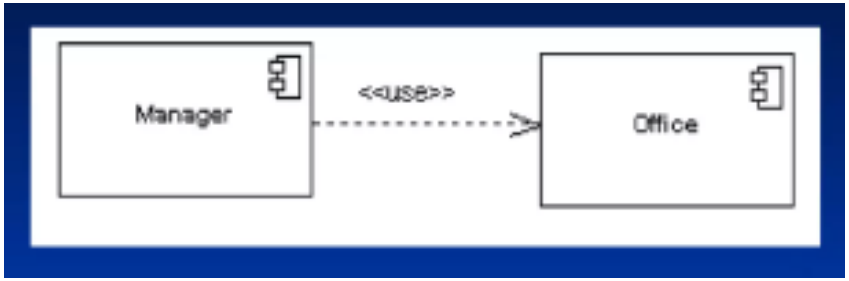
Component Diagrams -- Notation

- A component can have:
  - » Interfaces: An interface represents a declaration of a set of *operations and obligations*.

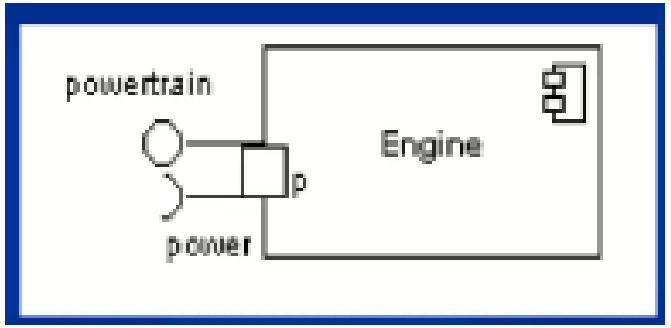


Component Diagrams -- Notation

- A component can have:
  - » Usage dependencies: A usage dependency is relationship which one element requires another element for its *full implementation*.

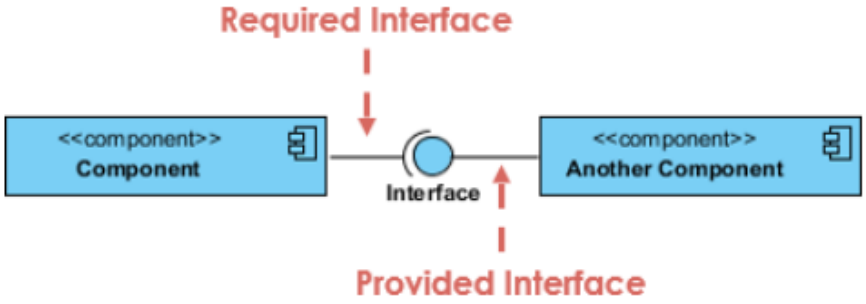


  - » Ports: Port represents an interaction point *between a component and its environment*.

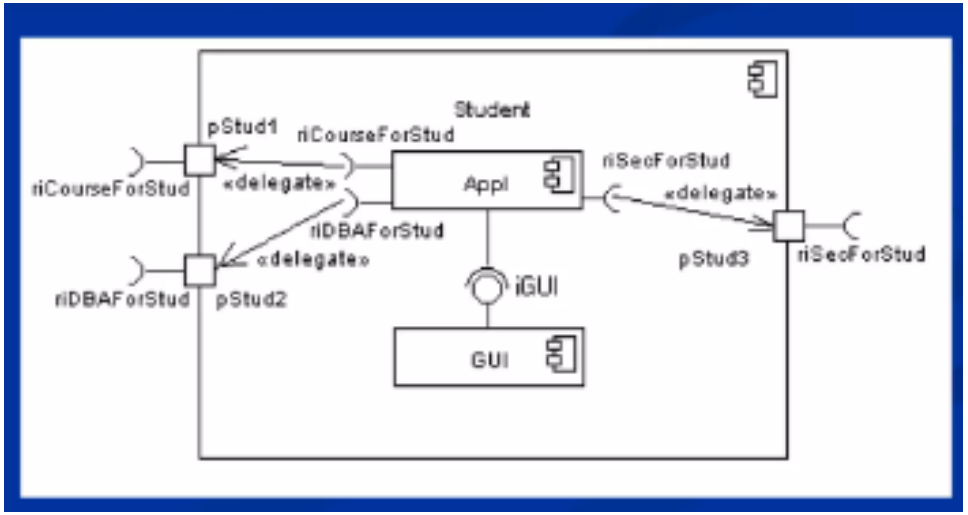


Component Diagrams -- Notation

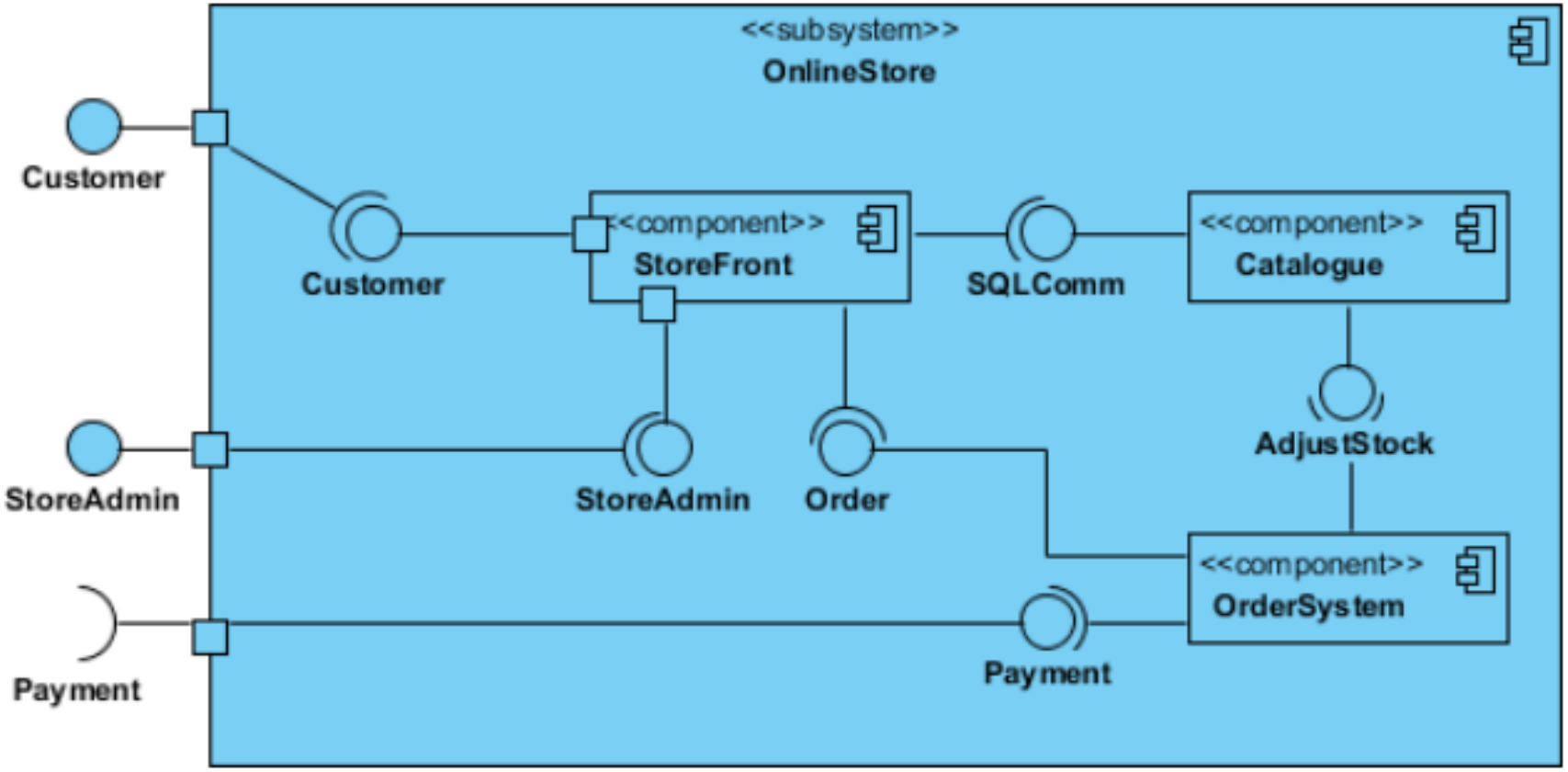
- A component can have:
  - » Connectors: Connect *two components*
  - » Assembly connectors



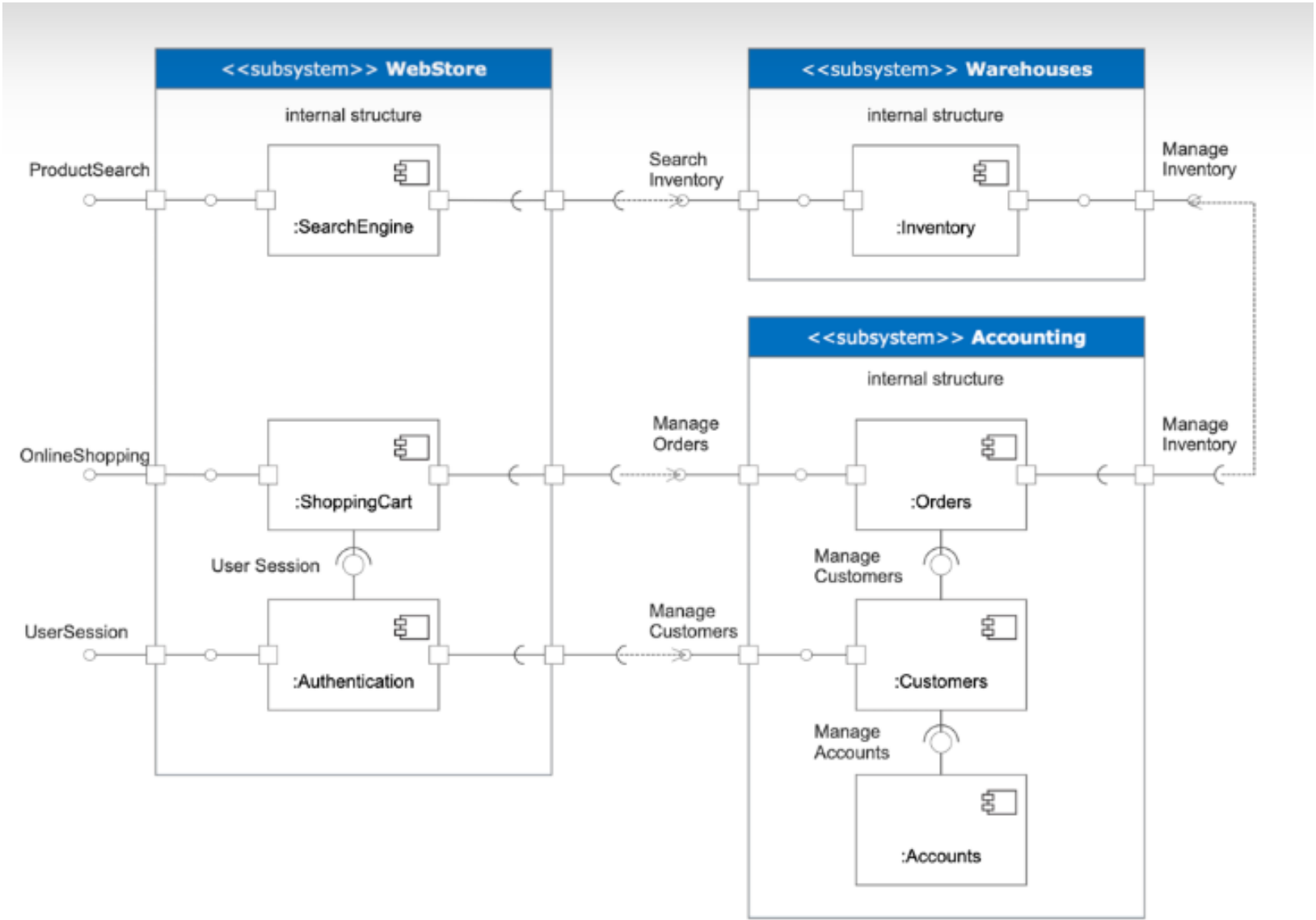
  - » Delegate connectors
  - » Links the *external contract* of the component to the internal realization



Component Diagrams -- Notation



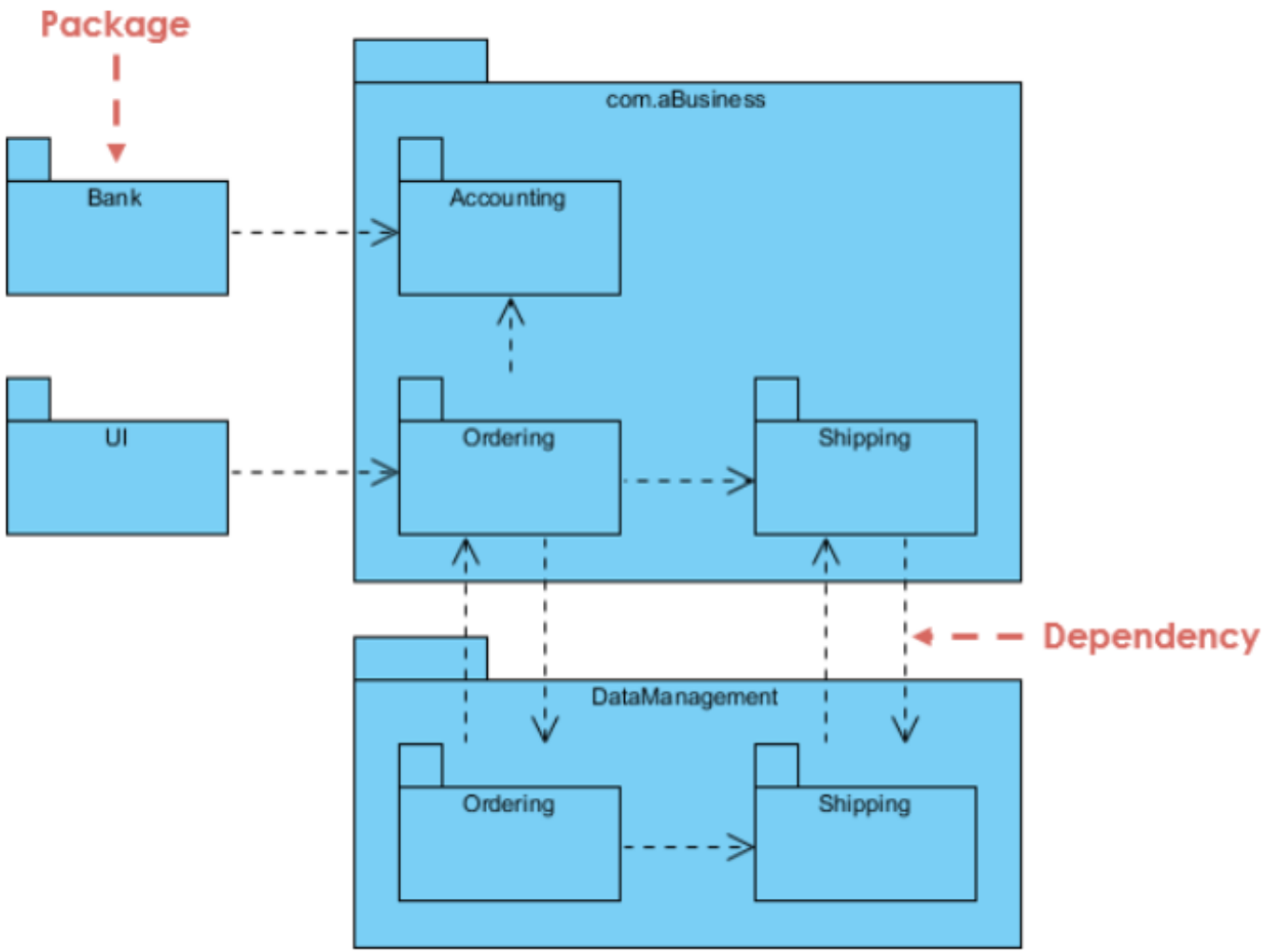
Component Diagrams -- Example



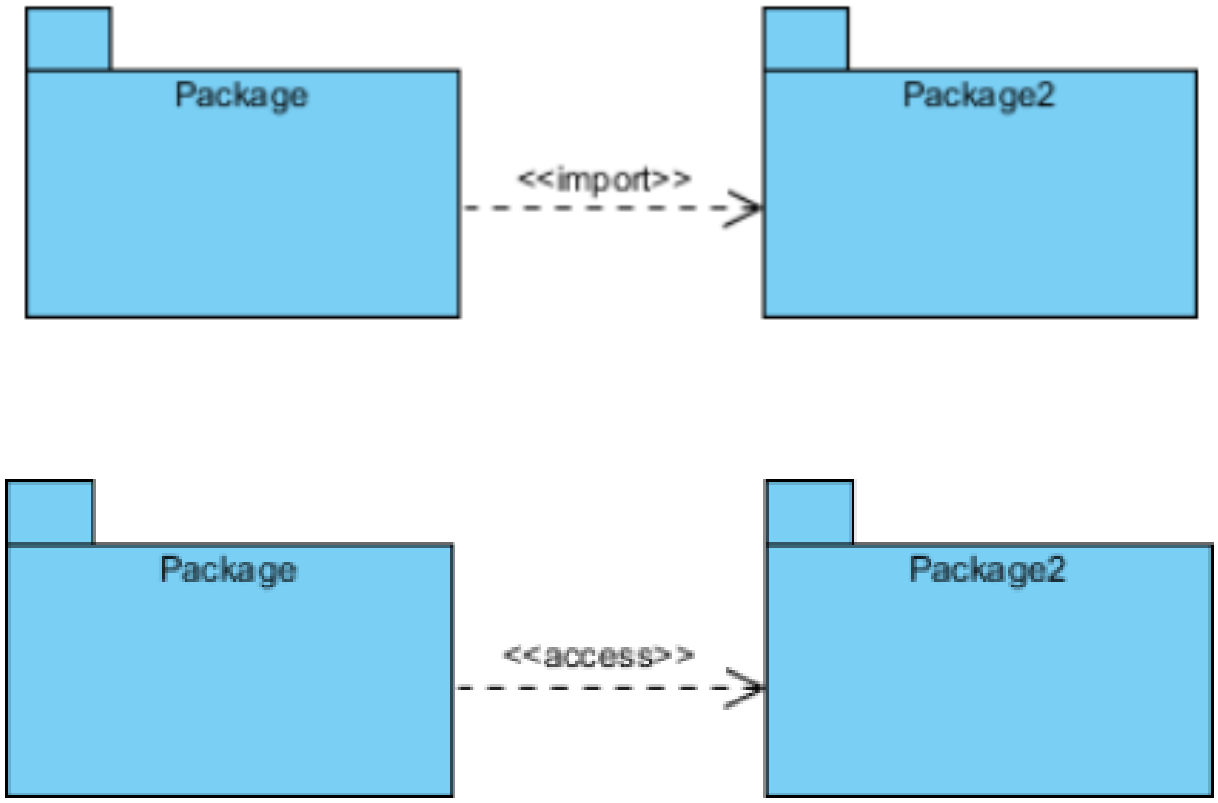
Package Diagram

- Package Diagram can be used to simplify complex class diagrams, it can *group classes* into packages.
- A package is a collection of *logically related* UML elements.
- It usually designed as *bottom-up* approach.

Package Diagram



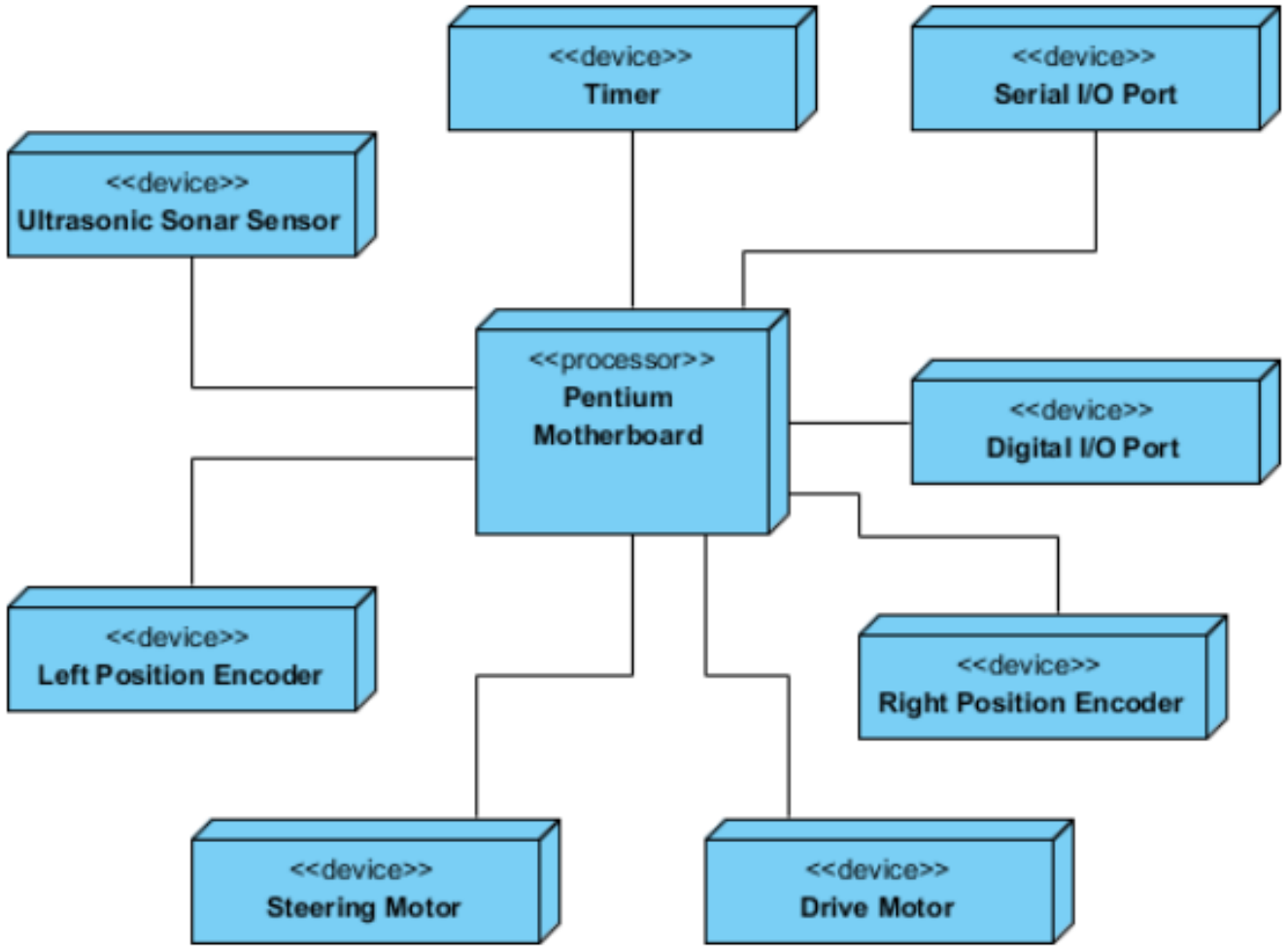
Package Diagram



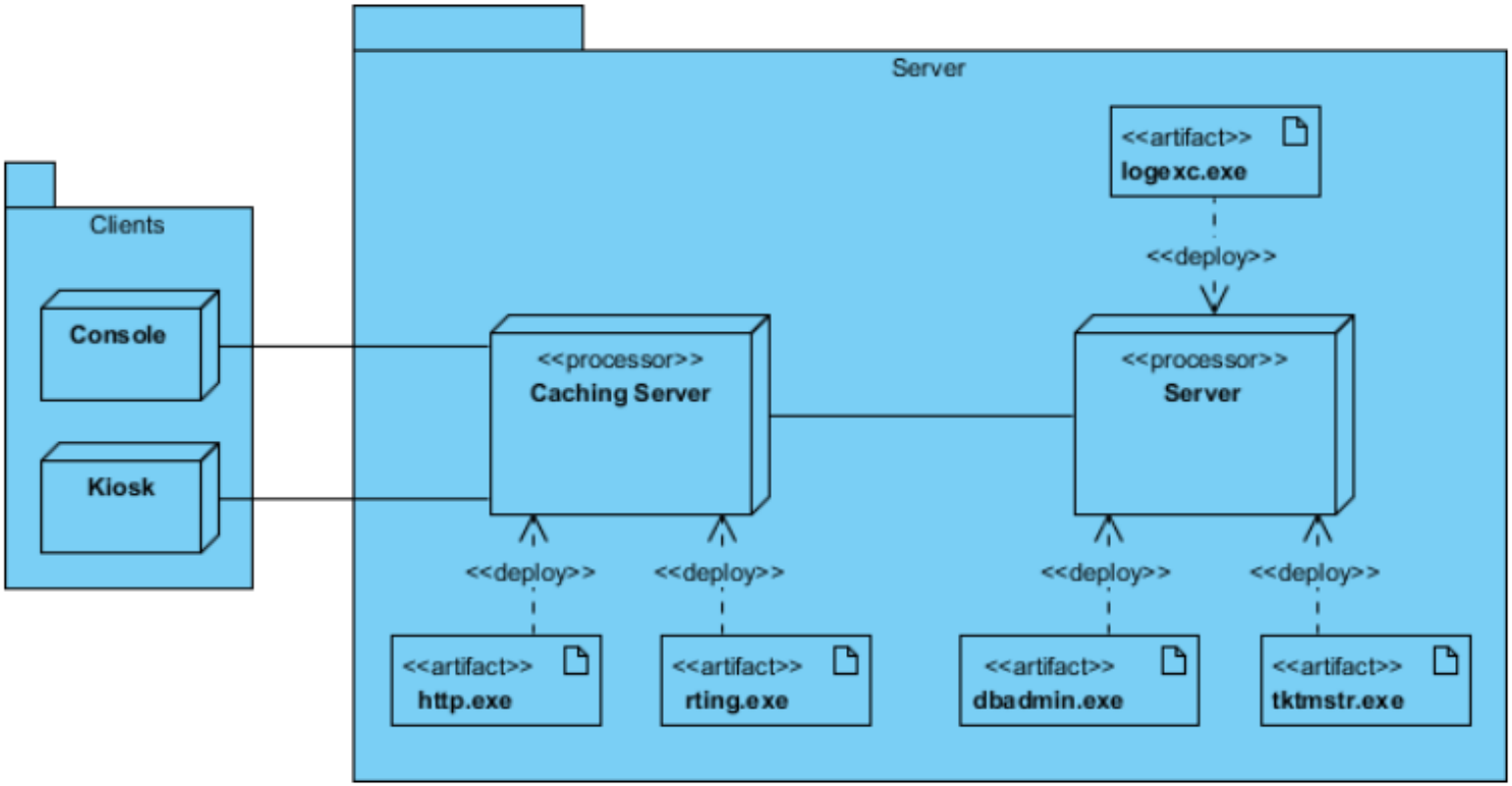
Deployment Diagram

- Shows a set of *processing* nodes and their relationships.
- Represents the static deployment view of an *architecture*.
- Nodes typically enclose one or more *components*.
- They capture the *hardware* that will be used to implement the system and the *links* between different items of hardware.
- They model physical hardware elements and the *communication* paths between them
- They are also useful for Document the deployment of software components or nodes.

Deployment Diagram



Deployment Diagram



Deployment Diagram