Component Diagrams

Massimo Felici

Room 1402, JCMB, KB

0131 650 5899

mfelici@inf.ed.ac.uk

Component Diagrams

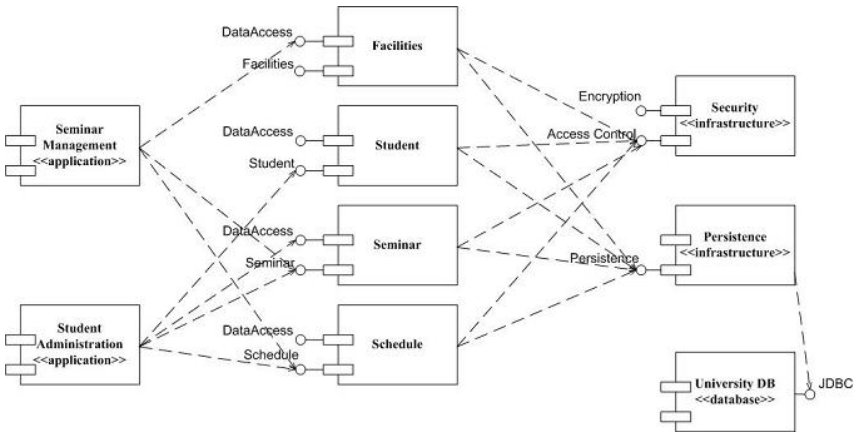
- A **component** is an **encapsulated**, **reusable**, and **replaceable** part of your software
- **Component Diagrams**
  - Model **physical software components** and the relationships between them
  - show the structure of the code itself
  - Model **source code** and relationships between files
  - Model the structure of **software releases**
  - Specify the files that are compiled into an executable

Components

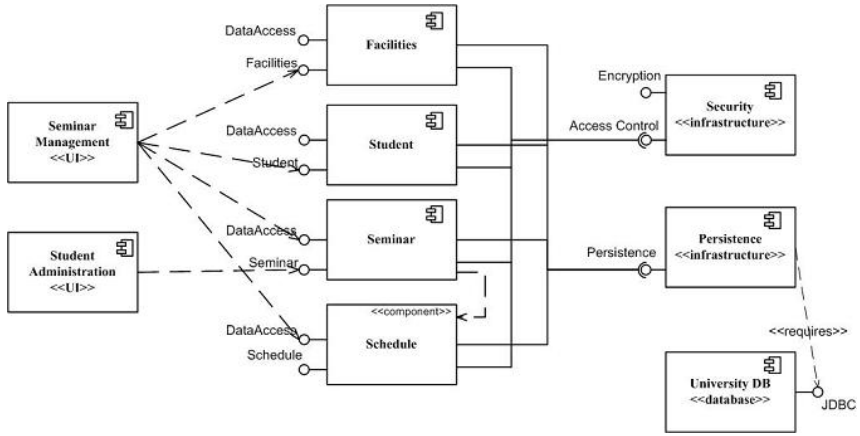
- A **Component** is a physical piece of a system, such as a compiled object file, piece of source code, shared library or Enterprise Java Bean (EJB)
- **Components** have:
  - **Interfaces**
  - Context Dependencies
    - **Implementation-specific**: shown on diagram
    - **Use-context**: may be described elsewhere – for example, documentation, use-cases, interaction diagrams, etc.
- **Note that notation has changed in UML 2.0**

New Component Notation

UML 1.x



UML 2.0

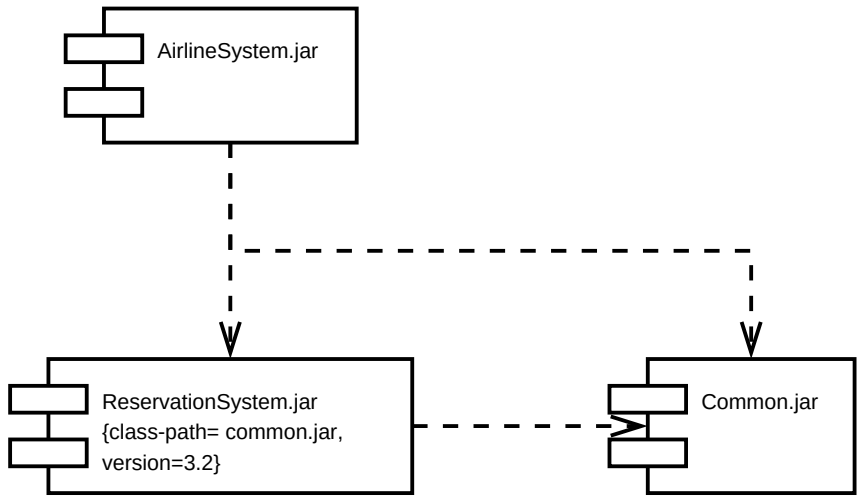


[Agile Modeling, Introduction to the Diagrams of UML 2.0]

Component Modelling

- **Component Diagrams** can show how subsystems relate and which **interfaces** are implemented by which component
- A Component Diagram shows one or more **interfaces** and their relationships to other components

An example of Component Diagram



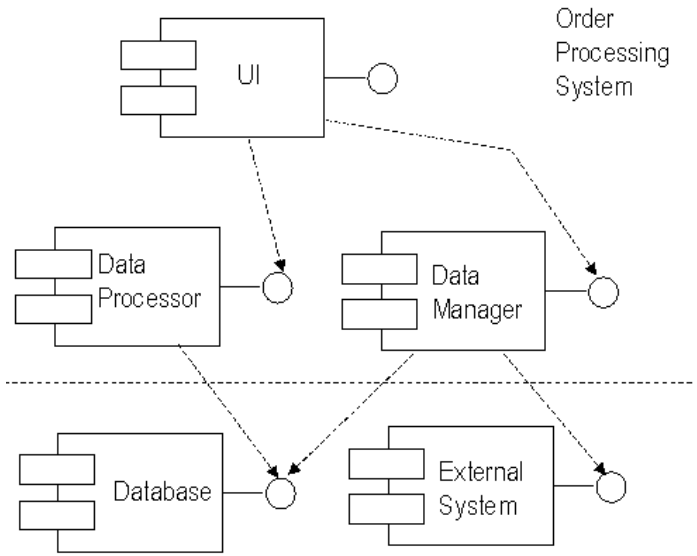
Provided and Required Interfaces

- A **provided interface** of a component is an interface that the component realizes
- A **required interface** of a component is an interface that the component needs to function

Components Diagrams

- A Component Diagram shows the **dependencies** among software components, including source code, binary code and executable components.
- Some components exist at compile time, some exist at link time, and some exist at run time; some exist at more that one time.

An Example of Component Diagram



Dependencies

- **Reside Dependencies**: A reside dependency from a component to any UML element indicates that the component is a client of the element, which is considered itself a supplier, and that the element resides in the component.
- **Use Dependencies**: A use dependency from a client component to a supplier component indicates that the client component uses or depends on the supplier component. A use dependency from a client component to a supplier component's interface indicates that the client component uses or depends on the interface provided by the supplier component.
- **Deploy Dependency**: A deploy component from a client component to a supplier node indicates that the client components is deployed on the supplier node

Hot to produce component diagrams

Component Diagrams

1. Decide on the **purpose** of the diagram
2. Add **Components** to the diagram, grouping them within other components if appropriate
3. Add other **elements** to the diagram, such as classes, objects and interfaces
4. Add the **dependencies** between the elements of the diagram