Component Diagrams

Massimo Felici

IF-3.46 0131 650 5899

mfelici@inf.ed.ac.uk

Component Diagrams

- A component is an **encapsulated**, **reusable**, and **replaceable** part of your software

Rationale

- Reducing and defying **coupling** between software components
- Reusing existing components

Component Diagrams

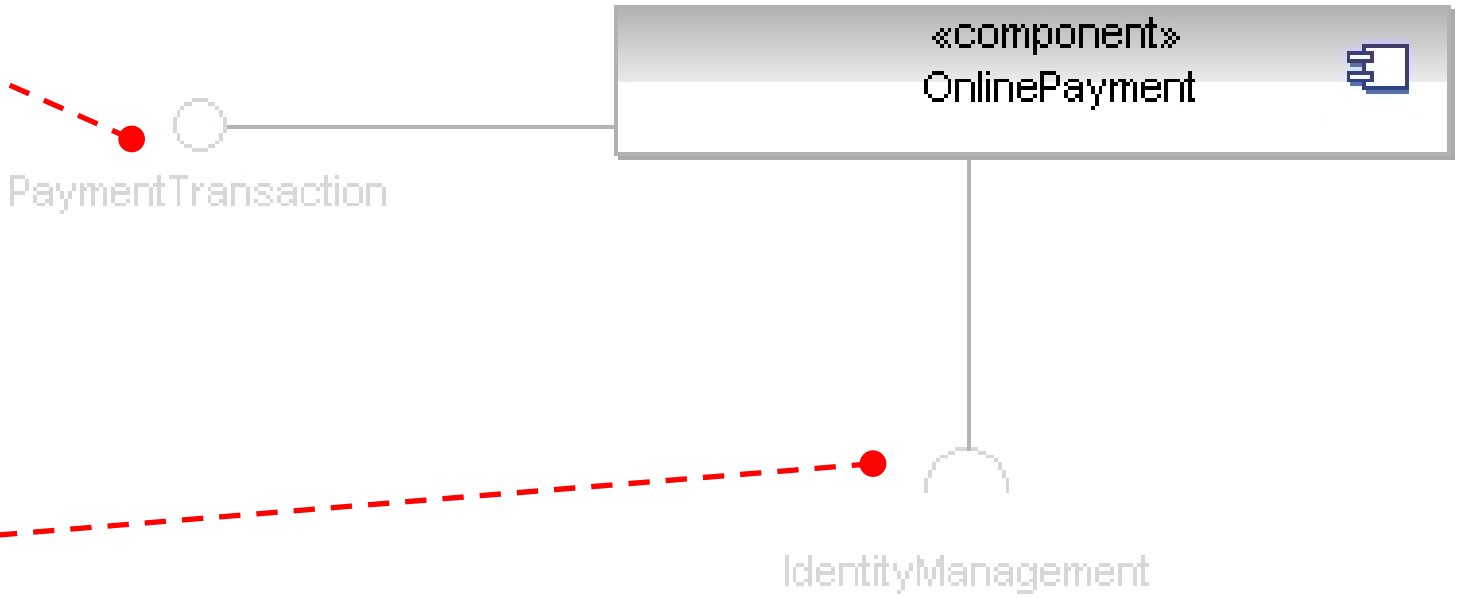
- Model physical software components and the **interfaces** between them
- Show the structure of the code itself
- Can be used to hide the specification detail (i.e., **information hiding**) and focus on the relationship between components
- Model the structure of software releases; show how components integrate with current system design
- Model source code and relationships between files
- Specify the files that are compiled into an executable

Component Notation

- 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)

Component Interfaces

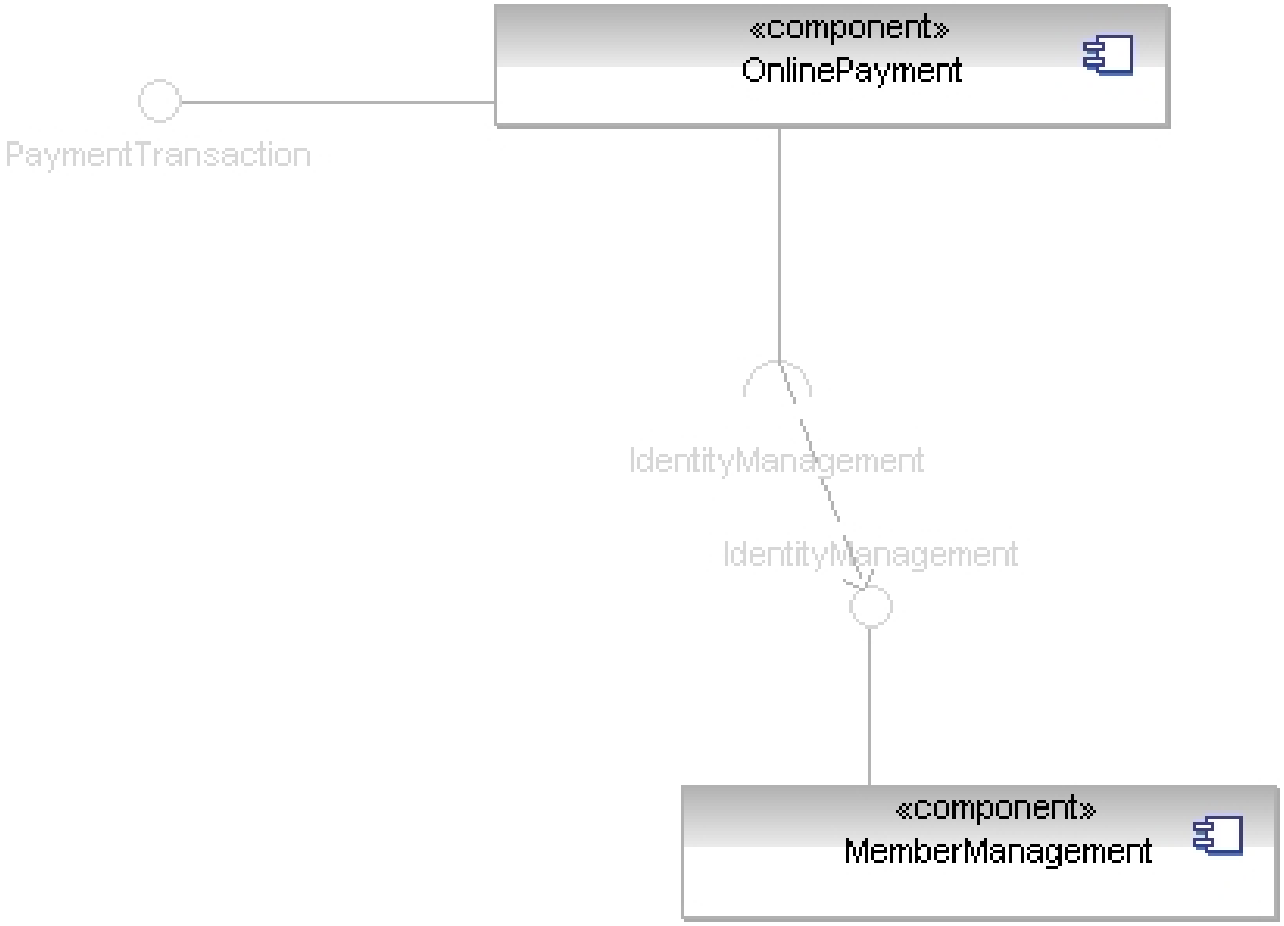
- A **provided interface** of a component is an interface that the component realizes



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

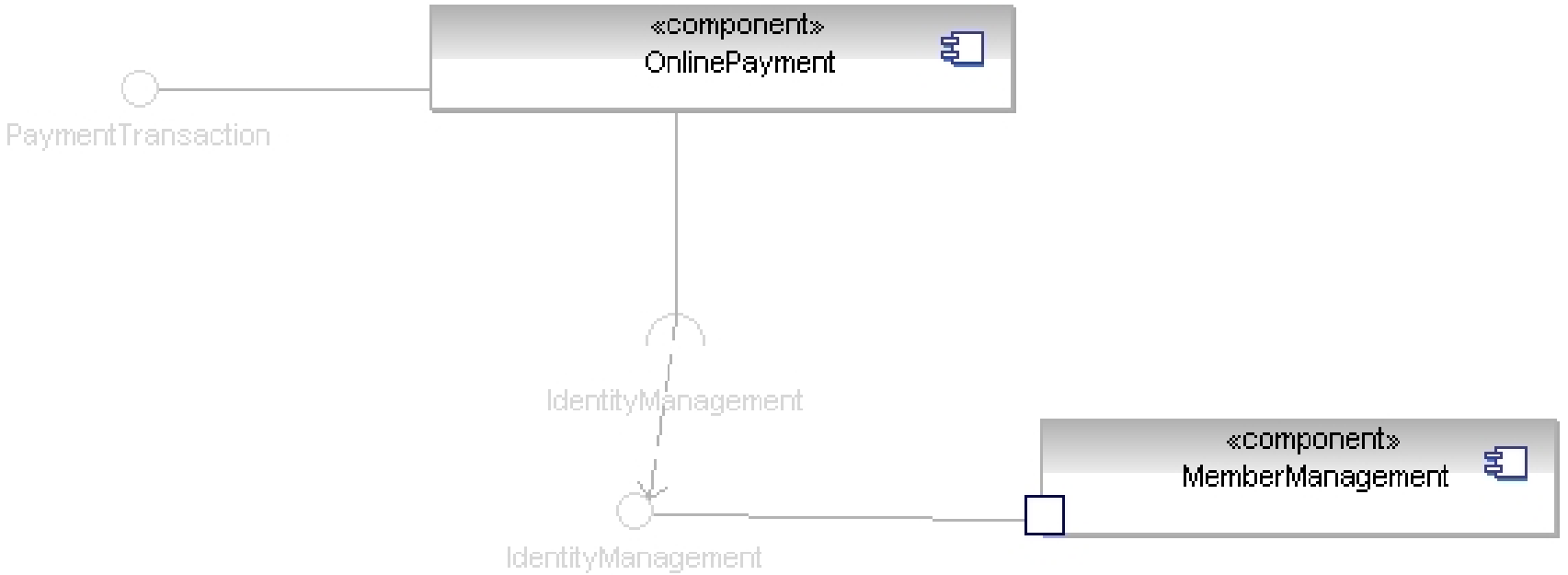
Component Assemblies

- Components can be "wired" together to form subsystems



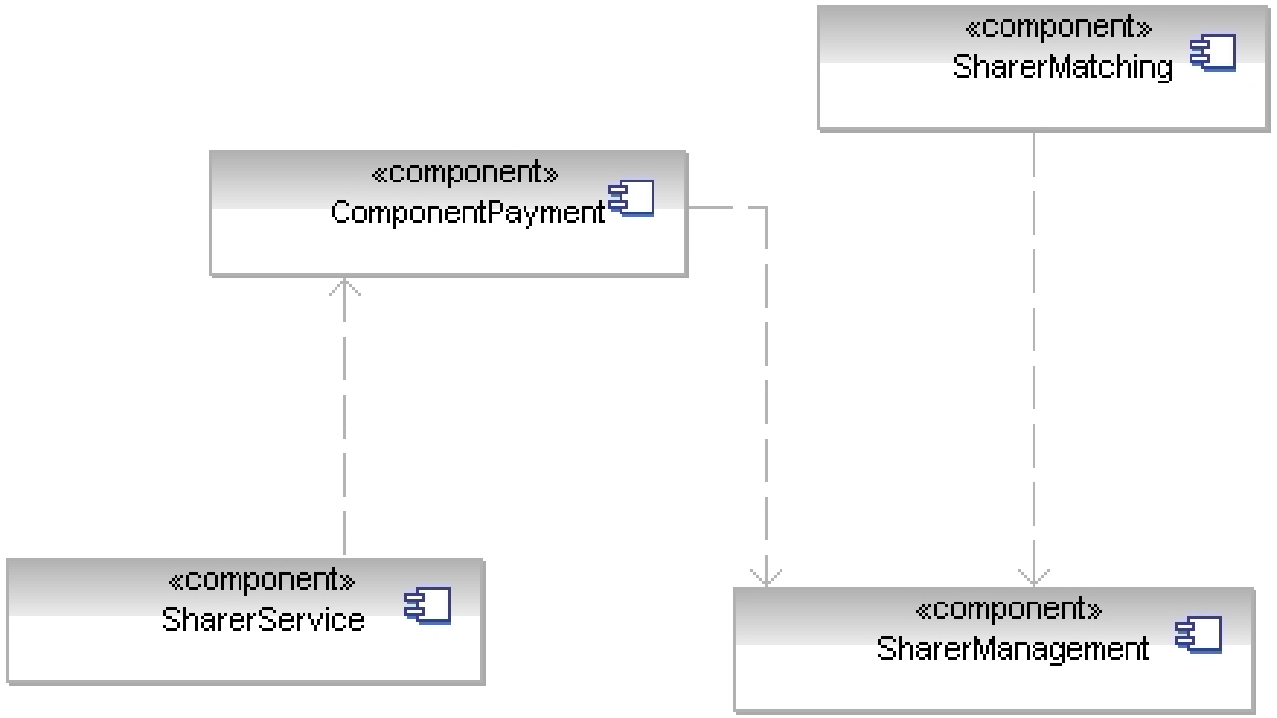
Ports

- A port (definition) indicates that the component itself does not provide the required interfaces (e.g., required or provided). Instead, the component delegates the interface(s) to an internal class.



Component Modelling

1. Find **components** and **dependencies**
2. Identify and level **subcomponents**
3. Clarify and make explicit the **interfaces** between components



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.

How to produce 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

Readings

- **UML course textbook**
  - Chapter 7 on Class Diagram: Other Notations.
  - Chapter 8 on Components Diagrams.

Summary

- Component Rationale
- Notation
- Component Diagrams
- Modelling