

**OBJECT-ORIENTED
MODELLING (OOM) - 32536**

MODULE 9

**Implementation Diagrams: Component,
Deployment and Composite
Structure**
(POOD – Chapter 8)

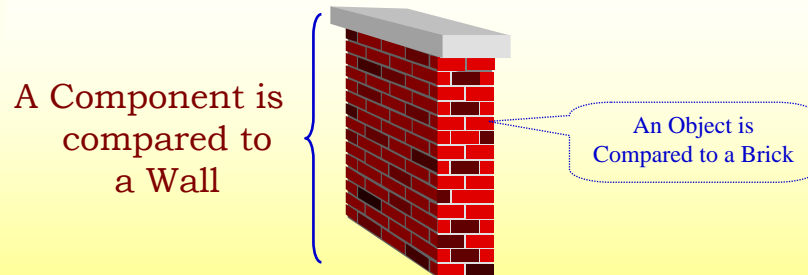
Module Outline

- Understanding Components and Component Diagrams
 - Classes are realized by Components
 - Components and OO
- Composite Structure Diagrams
 - Run-time structure of classes and components
- Deployment Diagrams
 - Hardware diagrams assisting system architecture and deployment of Components

©(c) Thomson Publishing, 2005: Practical OO Design, 1998-2005; v 3.1

2

A Component is a Large and Cohesive Collection of Objects



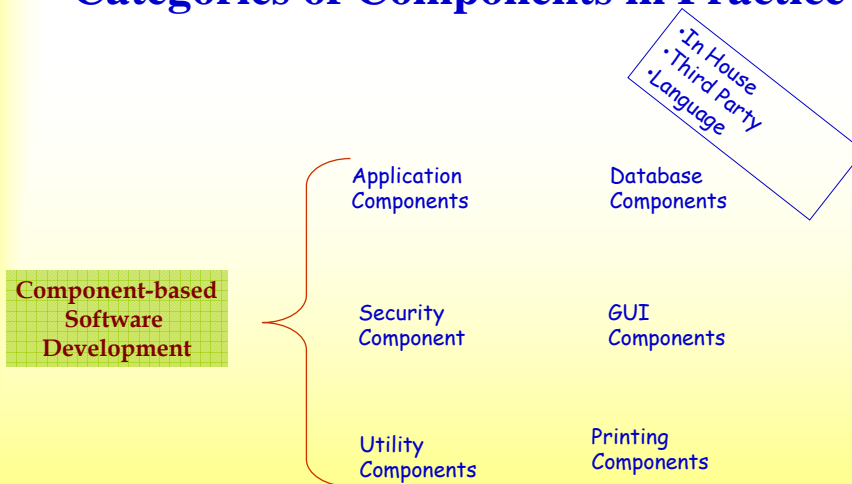
A Component

- A Component is a Physical Module of Code
- Typical Component Examples:
 - Source Code: e.g. .CPP and .H files
 - Executable: e.g. .EXE
- Component stereotypes can be
 - ActiveX, Applet, Application, Executable, DLL etc.

Component Types

- Design Time – e.g. Patterns
- Link Time – e.g. Language Class Libraries
- Run Time – e.g. Executable Components
- Distributed
 - DCOM: Distributed Component Object Model
 - CORBA: Common Object Request Broker Architecture
- GUI & Database Components
- Components Available on the Internet may be a combination of above

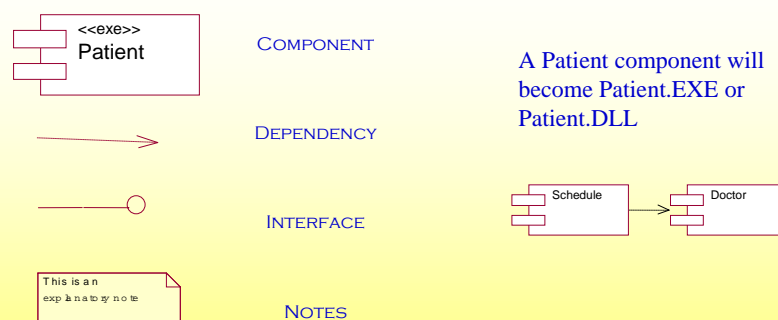
Categories of Components in Practice



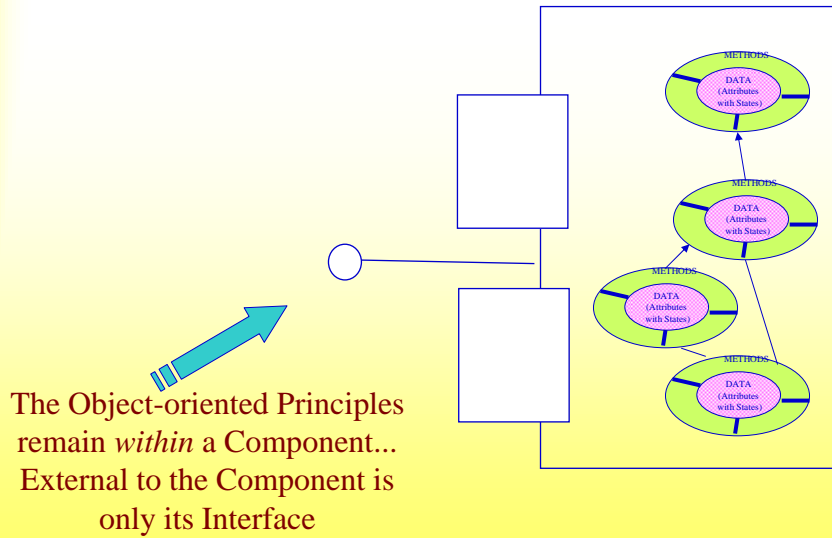
Component Types

- Application
 - Collection of Classes having business logic
- Database
 - Deals with storage and retrieval
- Utility
 - Support to the core modules
- GUI
 - Provides standard interface

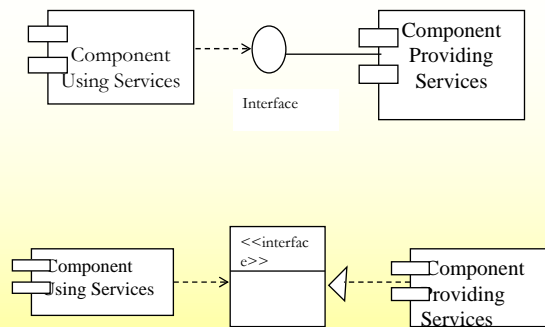
Representing a Component in UML



Encapsulation and Components



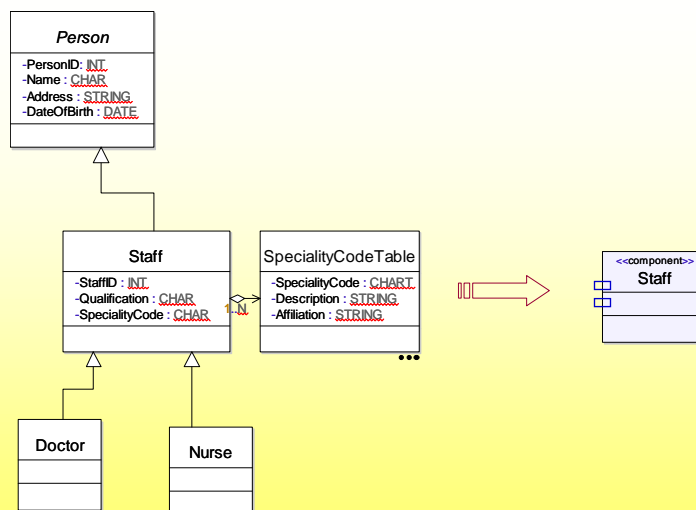
Component Relating through Interfaces



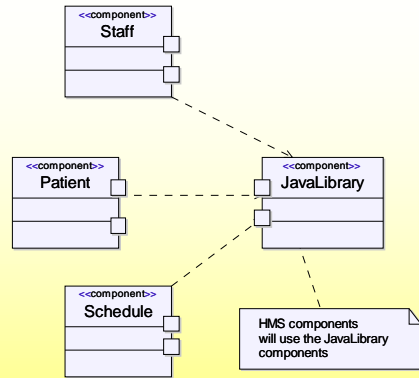
Mapping Classes to Components

- C++: Each Class in earlier designs in mapped to TWO Components
 - One Component represents the .CPP
 - Another Component represents the .H
- Java: Map each Class in earlier designs to ONE Component
 - representing the .JAVA file for the Classes

Staff Component realizes Classes for HMS



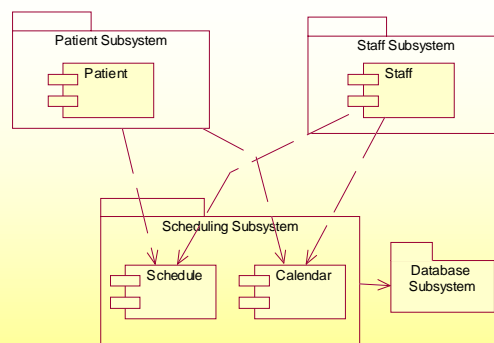
Practical Component Diagram for HMS showing Java Library Dependency



©(c) Thomson Publishing, 2005: Practical OO Design, 1998-2005; v 3.1

13

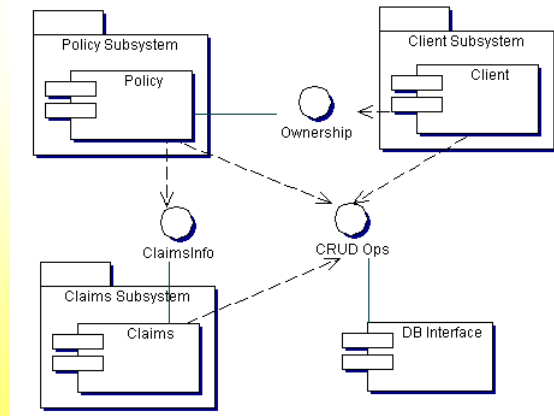
Practical Component Diagram showing Inter-Dependencies and Packages for HMS



©(c) Thomson Publishing, 2005: Practical OO Design, 1998-2005; v 3.1

14

Practical Component Diagram showing Packages and Interfaces for HMS



©(c) Thomson Publishing, 2005: Practical OO Design, 1998-2005; v 3.1

15

Strengths of Component Diagrams

- Component diagrams provide the means to create an executable model of the system.
- Component diagrams are a physical diagram that represents how the software will be physically structured and related to each other.
- Interfaces used on a component diagram are helpful in facilitate well-organised reuse of multiple classes simultaneously.
- Component diagrams help in creating a good, efficient architecture.
- Component diagrams facilitate reuse by breaking software into reusable parts, which in turn increases overall quality.

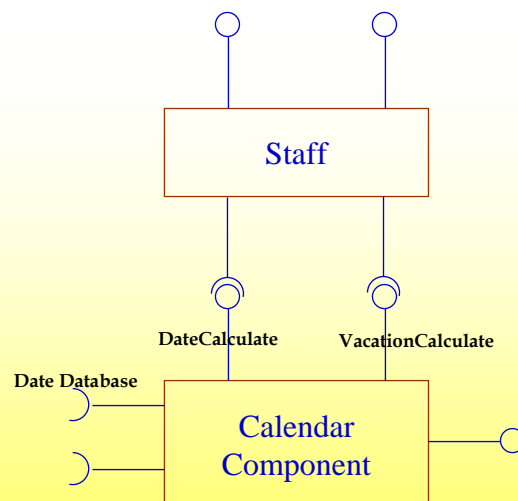
©(c) Thomson Publishing, 2005: Practical OO Design, 1998-2005; v 3.1

16

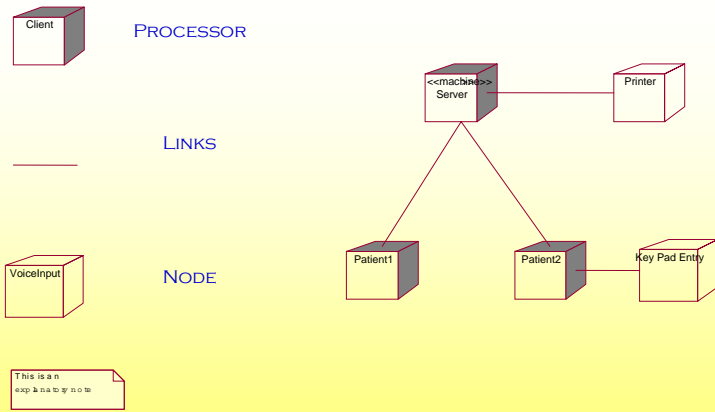
Weakness of Component Diagrams

- Component diagrams can be too coarse-grained.
- They are not OO in nature.
- Components diagrams often have circular dependencies between
- components.
- Mapping component diagrams to deployment diagrams is not always clear.
- Component diagrams have been implemented in many variations in CASE tools, and require a standardised to be more useful in UML.

An Example Composite Structure Diagram for HMS



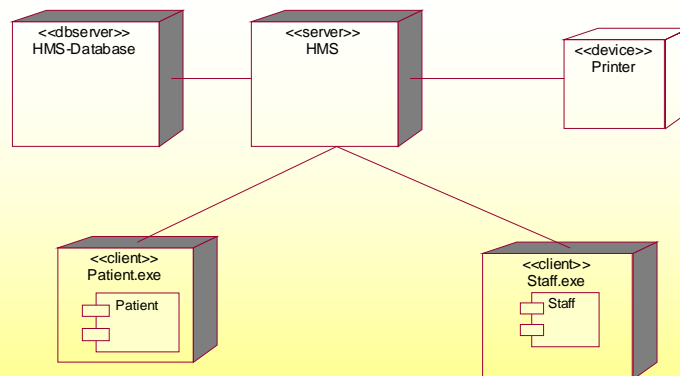
UML Elements on a Deployment Diagram



©(c) Thomson Publishing, 2005: Practical OO Design, 1998-2005; v 3.1

19

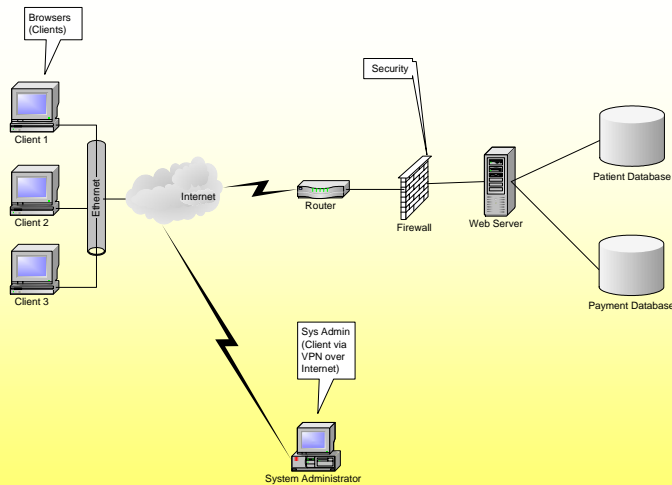
Deployment diagram with HMS Component and their Distribution



©(c) Thomson Publishing, 2005: Practical OO Design, 1998-2005; v 3.1

20

Deployment diagram for HMS Web infrastructure

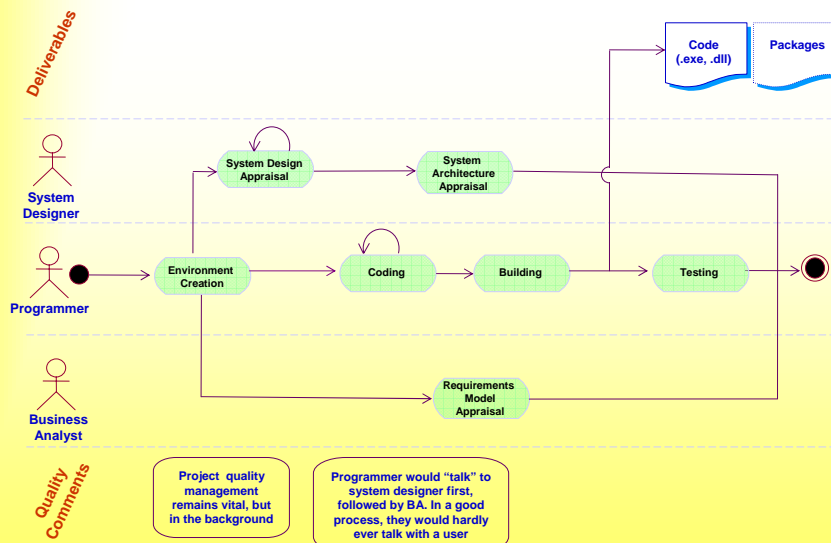


©(c) Thomson Publishing, 2005: Practical OO Design, 1998-2005; v 3.1

21

Implementation

Implementation Process-Component



©(c) Thomson Publishing, 2005: Practical OO Design, 1998-2005; v 3.1

22

Implementation Process Component Activities and Deliverables

- Code
- Packages
- Environment Creation
- Requirements Model
- System Design Appraisal
- Coding
- Building
- Testing

Conclusions

- Components realize classes
- Component diagrams show dependencies between components
- Composite structure diagrams shown run-time structure of objects and components
- Deployment diagrams show the hardware on which the system (components) will be deployed

WORKBOOK



EXERCISES

For The Class Room!!

Workbook Exercises

1. Identify ONE component and show how it realises a few <<entity>> classes
2. Draw a simple component diagram with two or three components
3. Draw one deployment diagram with two client machines, one server and one printer



PROJECT WORK :
(DURING TUTORIAL;
***ATTENDANCE IS MANDATORY*)**

This is a friendly Reminder:

Follow the Project Work Requirements;
Discuss Your Project Work with your Tutor;
Ensure you Minute your Group Tutorial
Activities & Attendance