Implementation Diagrams: Component and Deployment Diagrams

Massimo Felici

Room 1402, JCMB, KB

0131 650 5899

mfelici@inf.ed.ac.uk

Implementation Diagrams

- UML **Implementation Diagrams** show aspects of implementation, including source code structure and run-time implementation structure.
- There are two kinds of Implementation Diagrams:
    - **Component Diagrams** show the structure of the code itself
    - Components Diagrams capture the relationships between software components in the system
    - **Deployment Diagrams** show the structure of the run-time system
    - Deployment Diagrams capture the hardware that will be used to implement the system and the links between different items of hardware.

Implementation Diagrams' Rationale

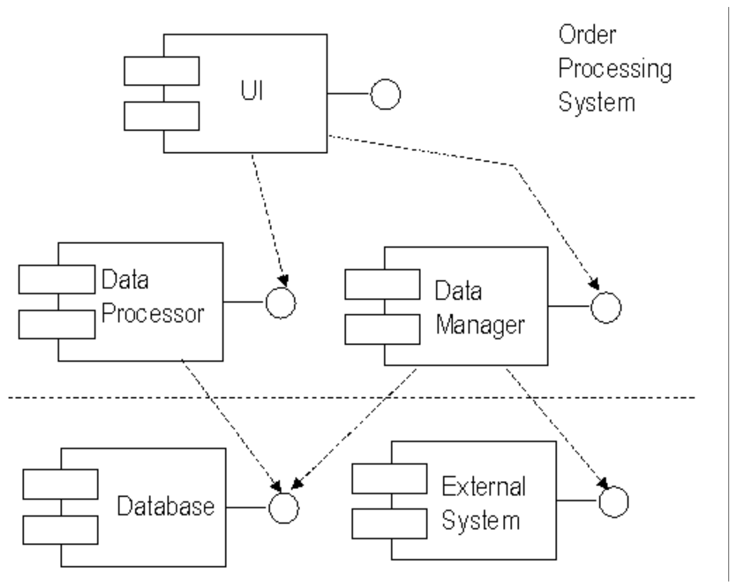
- **Components Diagrams**
  - Model physical software components and the relationships between them
  - Model source code and relationships between files
  - Model the structure of software releases
  - Specify the files that are compiled into an executable
- **Deployment Diagrams**
  - Model physical hardware elements and the communication paths between them
  - Plan the architecture of a system
  - Document the deployment of software components or nodes

Component Diagrams

Components Diagrams

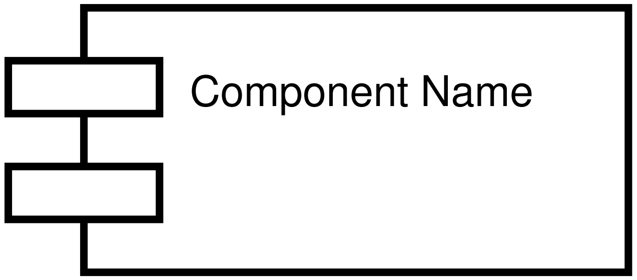
- A Component Diagram shows the dependencies among software components, including source code, binary code and executable components.
- Some components exist at compile time, some exist at link time, and some exist at run time; some exist at more that one time.

An Example of Component Diagram



Components

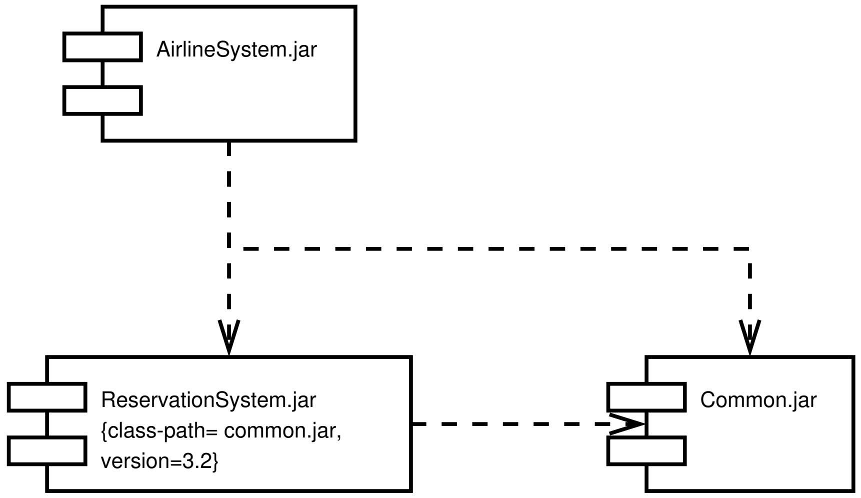
- A **Component** is a physical piece of a system, such as a compiled object file, piece of source code, shared library or Enterprise Java Bean (EJB)
- **Components** have:
  - **Interfaces**
  - Context Dependencies
    - **Implementation-specific**: shown on diagram
    - **Use-context**: may be described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



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



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

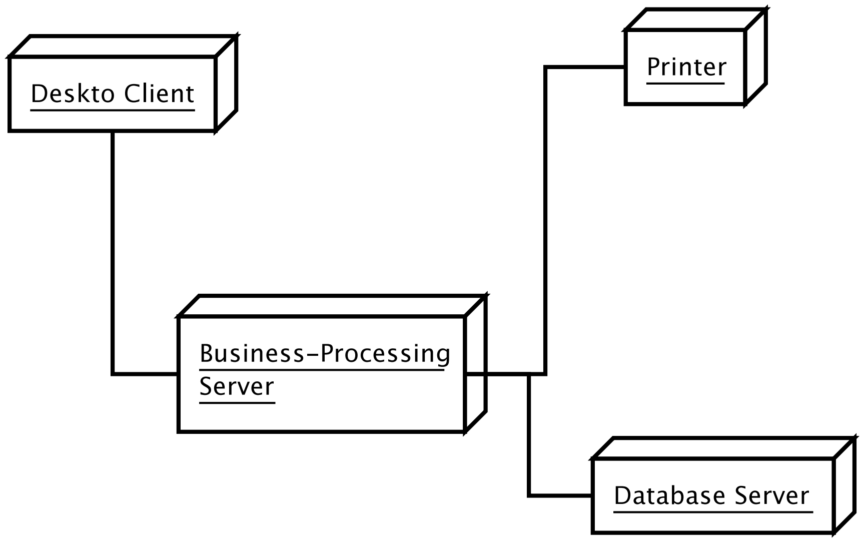
Deployment Diagrams

Deployment Diagrams

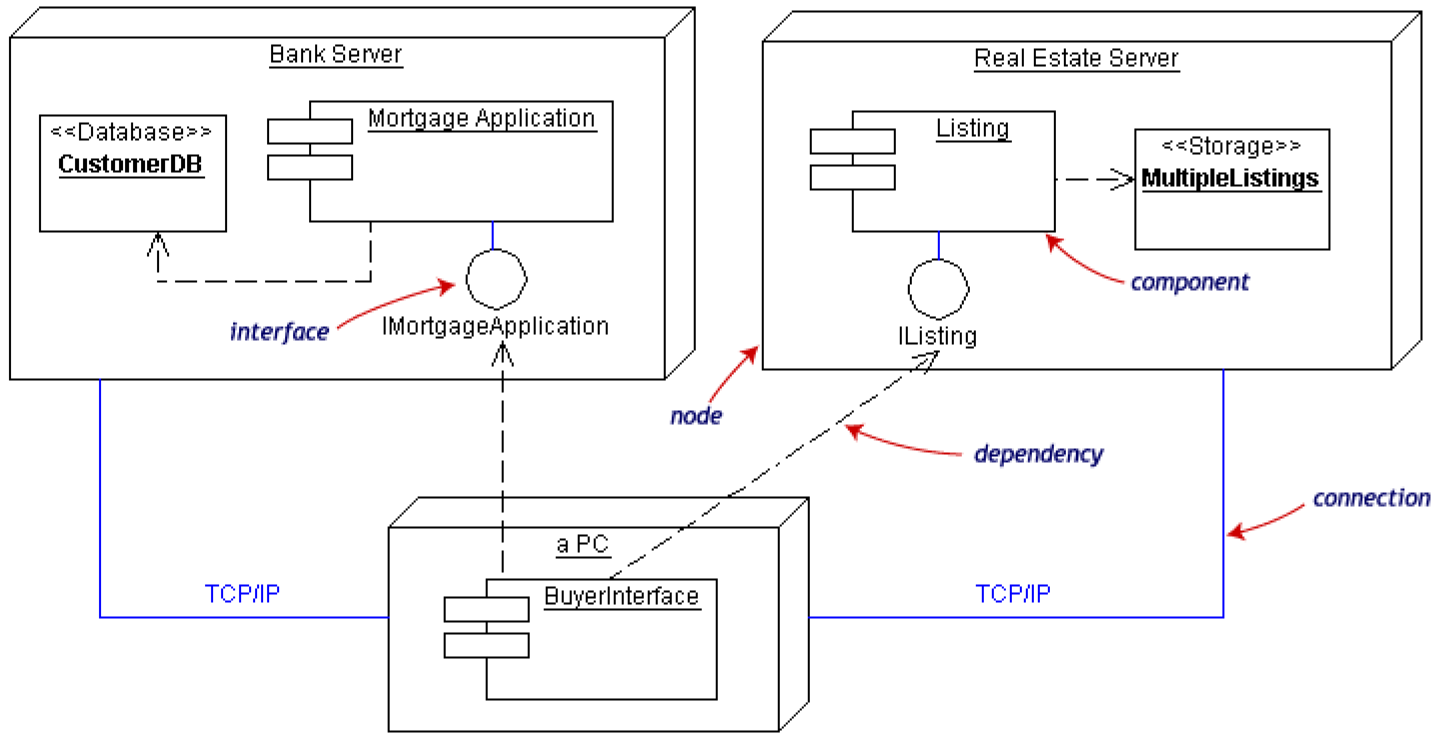
- A Deployment Diagram shows the configuration of run-time processing elements and the software components, processes, and objects
- Software component instances represent **run-time** manifestations of code units
- Deployments Diagrams capture only components that exist as run-time entities
- A deployment diagram shows the system's hardware, the software installed on that hardware, and the middleware that connects the disparate machines together
- A Deployment Diagram is a collection of one or more deployment diagrams with their associated documentation

Communication Association

- A communication associations between nodes indicates a communication path between the nodes that allows components on the nodes to communicate with one another



An Example of Deployment Diagram



Deployment diagrams show the physical configurations of software and hardware. The example shows the relationships among software and hardware components involved in real estate transactions.

Deployment Diagrams

- What existing systems will system need to interact or integrate with?
- How robust does system need to be (e.g., redundant hardware in case of a system failure)?
- What and who will connect to or interact with system, and how will they do it
- What middleware, including the operating system and communications approaches and protocols, will system use?
- What hardware and software will users directly interact with (PCs, network computers, browsers, etc.)?
- How will you monitor the system once deployed?
- How secure does the system need to be (needs a firewall, physically secure hardware, etc.)?

Deployment Planning

- How will your system be installed?
  - Who will install it? How long should it take to install?
  - Where the installation possibly fail? How do you back out if the installation fails? How long does it take to back out?
  - What is your installation window (during what time period can you install your system)?
  - What backups do you need before installation? Do you need to do a data conversion?
  - How do you know that the installation was successful?
- If different versions of the system will be in production at the same time, how will you resolve differences?
- What physical sites do you need to deploy to and in what order?
  - How will you train your support and operations staff?
  - Do you need to deploy a production support system so that the support staff uses their own environment to simulate problems?
- How will you train your users?
  - What documentation, and in what formats and languages, do your users, and support and operation staff need?
  - How will updates to documentation be deployed?

How to produce Implementation Diagrams

Component Diagrams

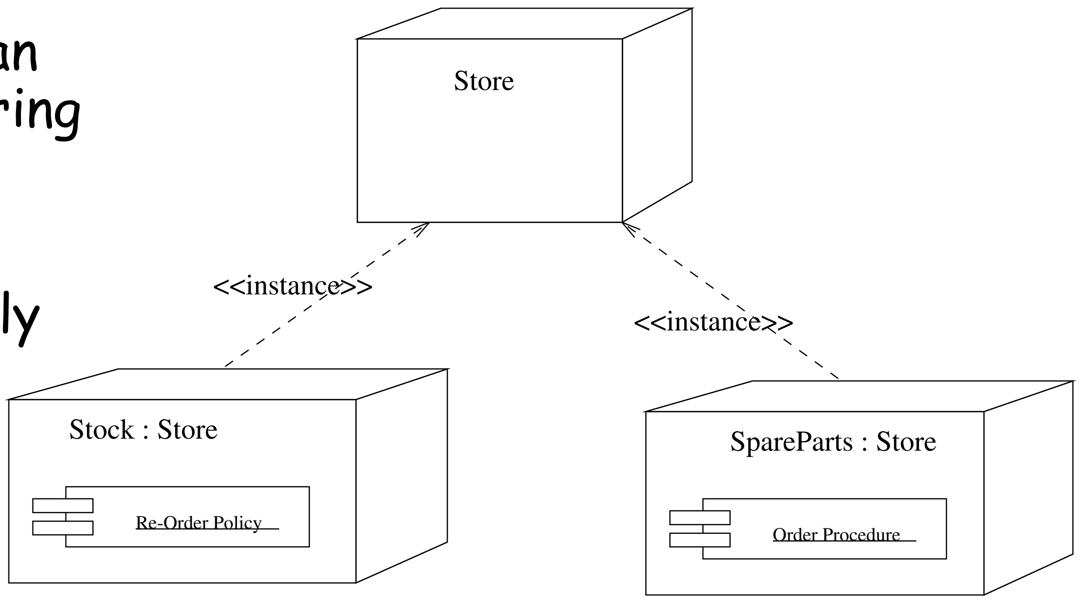
1. Decide on the **purpose** of the diagram
2. Add **Components** to the diagram, grouping them within other components if appropriate
3. Add other **elements** to the diagram, such as classes, objects and interfaces
4. Add the **dependencies** between the elements of the diagram

Deployment Diagrams

1. Decide on the **purpose** of the diagram
2. Add **nodes** to the diagram
3. Add **communication associations** to the diagram
4. Add other **elements** to the diagram, such as components or active objects, if required
5. Add **dependencies** between components and objects, if required

Modeling Business Process

- Business modeling using nodes and components is an effective means of capturing non-computer based processes and entities
- This can be done very early in development, to complement the use case model and other business modeling
- Components are the business procedures and documents; the nodes ("run-time structure") are the organization units and resources (human and other) of the business



Reading/Activity

- Read Chapter 13, Implementation Diagrams, of the Schaum's Outlines UML book

Summary

- Implementation Diagrams
- Component Diagrams
  - Modelling, Dependencies
- Deployment Diagrams
  - Deployment Planning
- How to produce Implementation Diagramsd
- Modelling Business Process