

Component and Deployment Diagram

Chapter 7

Systems Analysis and Design
in a Changing World 7th Ed

Satzinger, Jackson & Burd

Overview

- UML includes two kinds of views for representing implementation units: the **implementation view** and **the deployment view**.

Implementation View

- The **implementation view** shows the physical packaging of the reusable pieces of the system into substitutable units, called **components**.
- An implementation view shows the implementation of design elements (such as classes) by components, as well as interfaces of and dependencies among components.
- Components are the high-level reusable pieces out of which systems can be constructed.

Deployment View

- The **deployment view** shows the physical arrangement of run-time computational resources, such as computers and their interconnections. They are called **nodes**.
- At run time, nodes can contain components and objects. The assignment of components and objects to nodes can be static, or they can migrate among nodes.
- The deployment view may show performance bottlenecks if component instances with dependencies are placed on different nodes.

Component

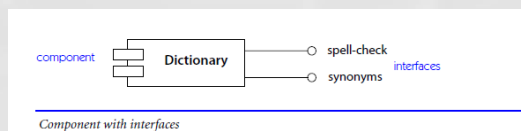
- A **component** is a physical unit of implementation with well-defined **interfaces** that is intended to be used as a replaceable part of a system.
- Components have **interfaces they support** and **interfaces they require** from other components.
- An interface is a list of operations supported by a piece of software or hardware.

UML Component Diagram



- Used to model the top-level view of the system design in terms of components and dependencies among the components. Components can be
 - source code, linkable libraries, executables
- The dependencies (edges in the graph) are shown as dashed lines with arrows from the client component to the supplier component:
 - The lines are often also called connectors
 - The types of dependencies are implementation language specific
- Informally also called “software wiring diagram” because it shows how the software components are wired together in the overall application.

Component

- A component is drawn as a rectangle with two small rectangles on its side. It may be attached by solid lines to circles that represent its interfaces.

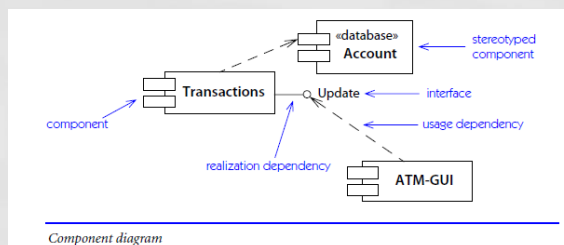


UML Interfaces: Lollipops and Sockets

- A UML interface describes a group of operations used or created by UML components.
 - There are two types of interfaces: provided and required interfaces.
 - A **provided interface** is modeled using the lollipop notation 
 - A **required interface** is modeled using the socket notation. 

Component

- Also as we described before, a component diagram shows dependencies among components

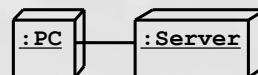


Node

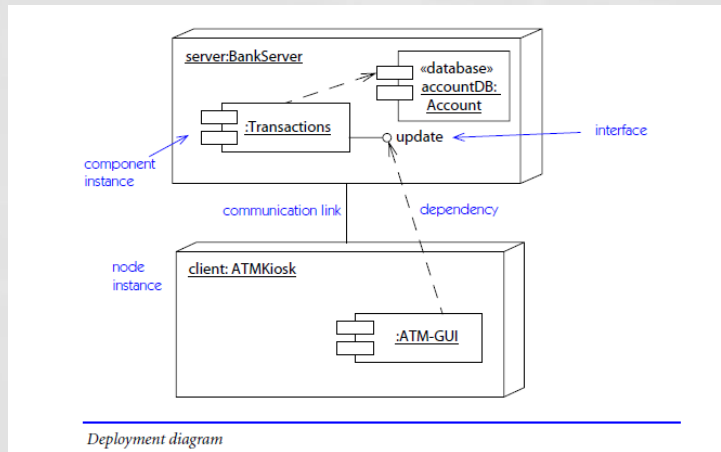
- A node is a **run-time** physical object that represents a computational resource, generally having at least a memory and often processing capability as well.
- Nodes may have **stereotypes** to distinguish different kinds of resources, such as CPUs, devices, and memories.
- A node is shown as a **stylized cube** with the name of the node and, optionally, its classification
- Associations between nodes represent communication paths.

Deployment Diagram

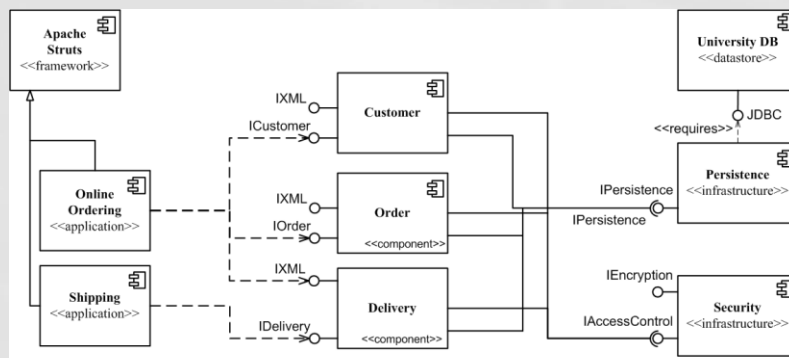
- Deployment diagrams are useful for showing a system design after these system design decisions have been made:
 - Subsystem decomposition
 - Concurrency
 - Hardware/Software Mapping
- A **deployment diagram** is a graph of nodes and connections (“communication associations”)
 - Nodes are shown as 3-D boxes
 - Connections between nodes are shown as solid lines
 - Nodes may contain components
 - Components can be connected by “lollipops” and “grabbers”
 - Components may contain objects (indicating that the object is part of the component).



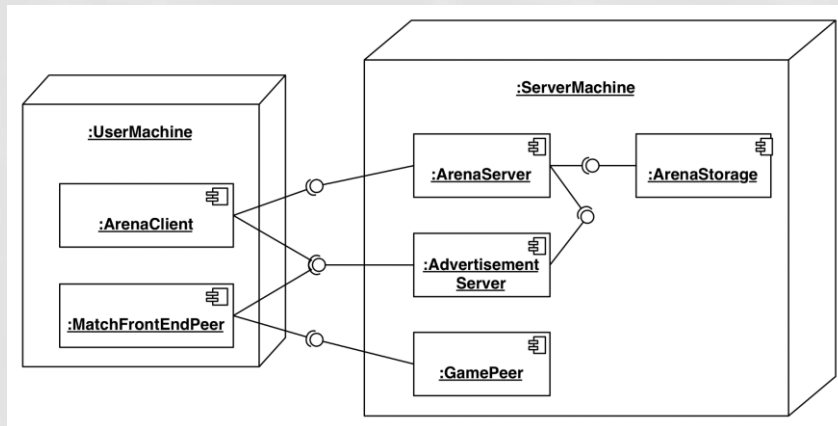
Deployment



Example: Deployment Diagram



Another Example : Deployment Diagram



Example:

- Draw a component and a deployment diagram for your project.