

## Implementation Diagrams

CSE870: UML Component Diagrams

---

---

---


---

---

---

---

---



## Implementation Diagrams

- Both are structural diagrams
- **Component Diagrams:**
  - set of components and their relationships
  - Illustrate static implementation view
  - Component maps to one or more classes, interfaces, or collaborations
- **Deployment Diagrams:**
  - Set of nodes and their relationships
  - Illustrate static deployment view of architecture
  - Node typically encloses one or more components

CSE870: UML Component Diagrams

---

---

---


---

---

---

---

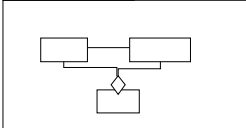
---



## Package

- General purpose mechanism for organizing elements into groups
- Can group classes or components.

Package Name



```
graph TD; subgraph PackageName; direction TB; C1[ ] --- C2[ ]; C1 --- C3[ ]; end; C3 --- C4[ ]
```

CSE870: UML Component Diagrams

---

---

---

---

---

---

---

---

### Component Diagram

- Classes
- Interfaces
- Dependency, generalization, association, and realization relationships

Special kind of class diagram focusing on system's components.

CSE870: UML Component Diagrams

---

---

---

---

---

---

---

---

### Interfaces

- Definition:
  - Collection of operation signatures and/or attribute defs
  - Defines a cohesive set of behaviors
- Realized by:
  - Implemented by classes and components
  - Implement operations/attributes defined by interface
- Relationships:
  - A class can implement 0 or more interfaces
  - An interface can be implemented by 1 or more classes
- Notation:
  - Lollipop
  - Dashed arrow

[Ambler, 2002-2005]

CSE870: UML Component Diagrams

---

---

---

---

---

---

---

---

### Sample interfaces

[Ambler, 2002-2005]

CSE870: UML Component Diagrams

---

---

---

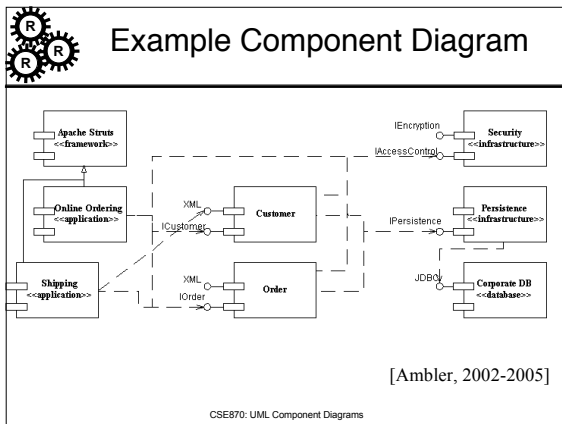
---

---

---

---

---




---

---

---

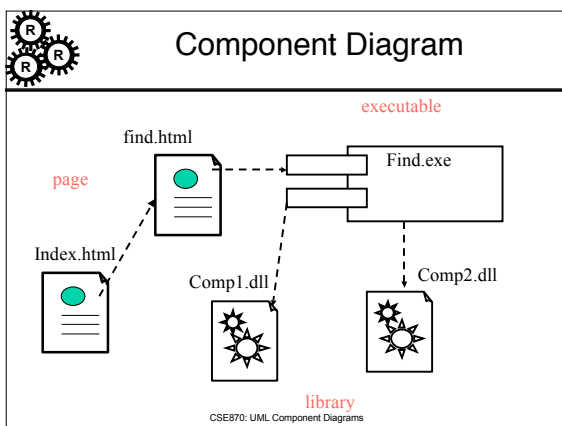
---

---

---

---

---




---

---

---

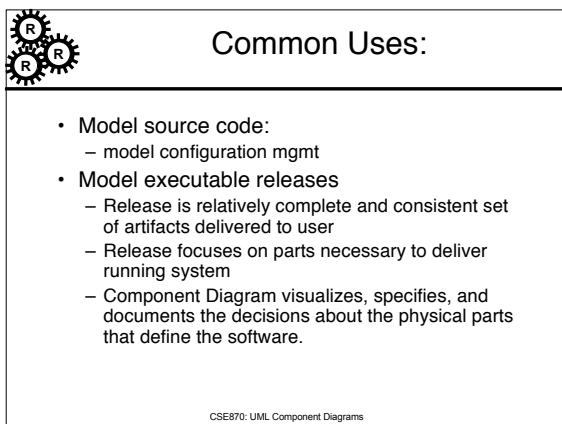
---

---

---

---

---




---

---

---

---

---


---

---

---

**Common Uses: (cont'd)**

- Model Physical databases:
  - database is concrete realization of schema
  - schemas offer an API to persistent information
  - model of physical dbases represents storage of that information in tables of a relational dbase or pages of an OO dbase.
  - Component Diagram can represent this kind of physical database
- Model Adaptable systems:
  - can model static aspects of adaptable systems
  - can model dynamic aspects (in conjunction with behavioral models)



CSE870: UML Component Diagrams

---

---

---

---

---

---

---

---

**Modeling Source Code**

- (Forward/Reverse Eng): identify set of source code files of interest
  - model as components stereotyped as files
- Larger systems: use packages to show groups of source code files
- Model compilation dependencies among files

CSE870: UML Component Diagrams

---

---

---

---

---

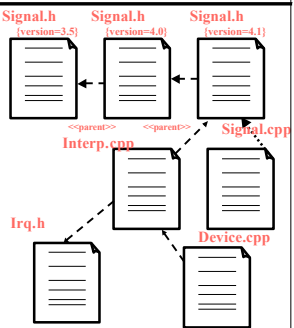
---

---

---

**Modeling Source Code Example**

- 5 source code files
  - signal.h (header)
  - used by 2 other files (signal.cpp, interp.cpp)
  - interp.cpp has compilation dependency to header file (irq.h)
  - device.cpp compilation dependency to interp.cpp



CSE870: UML Component Diagrams

---

---

---

---

---

---

---

---



## Component Diagram Guidelines

- **Use Descriptive Names for Architectural Components**
  - Use Environment-Specific Naming Conventions for Detailed Design Components
  - Apply Textual Stereotypes to Components Consistently
  - Avoid Modeling Data and User Interface Components
- **Interfaces**
  - Prefer Lollipop Notation To Indicate Realization of Interfaces By Components
  - Prefer the Left-Hand Side of A Component for Interface Lollipops
  - Show Only Relevant Interfaces
- **Dependencies and Inheritance**
  - Model Dependencies From Left To Right
  - Place Child Components Below Parent Components
  - Components Should Only Depend on Interfaces
  - Avoid Modeling Compilation Dependencies

CSE870: UML Component Diagrams

---

---

---

---

---

---

---

---



## Deployment Diagrams

CSE870: UML Component Diagrams

---

---

---

---

---

---

---

---



## Deployment Diagram

- Shows the configuration of:
  - run time processing nodes and
  - the components that live on them
- Graphically: collection of vertices and arcs

CSE870: UML Component Diagrams

---

---

---


---

---

---

---

---

 **Contents**

- Deployment diagrams contain:
  - Nodes
  - Dependency and association relationships
  - may also contain components, each of which must live on some node.

CSE870: UML Component Diagrams

---

---

---

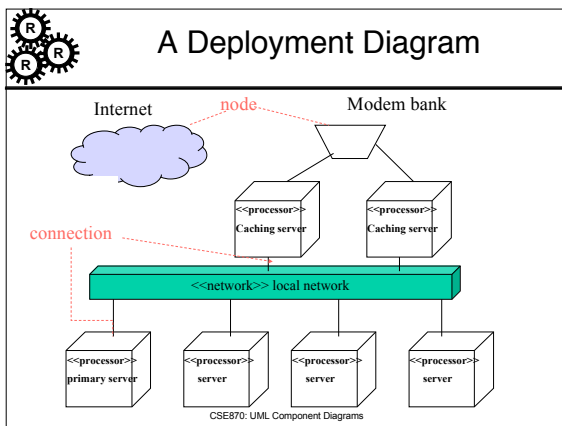
---

---

---

---

---



---

---

---


---

---

---

---

---

 **Modeling Client-Server Architecture**

- Identify nodes that represent system's client and server processors
- Highlight those devices that are essential to the behavior
  - E.g.: special devices (credit card readers, badge readers, special display devices)
- Use stereotyping to visually distinguish

CSE870: UML Component Diagrams

---

---

---

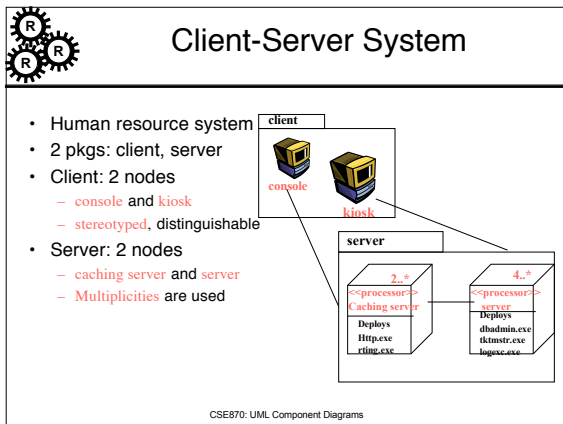
---

---

---

---

---




---

---

---

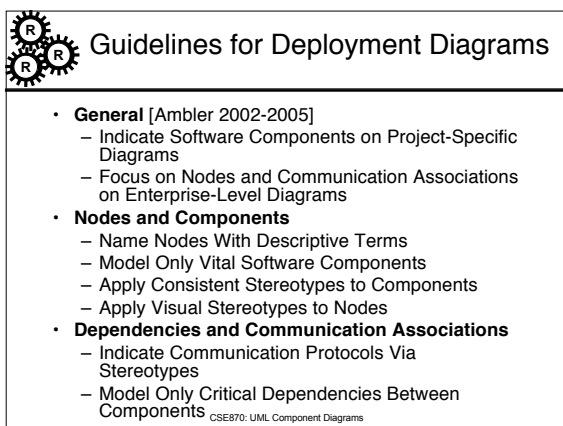
---

---

---

---

---




---

---

---

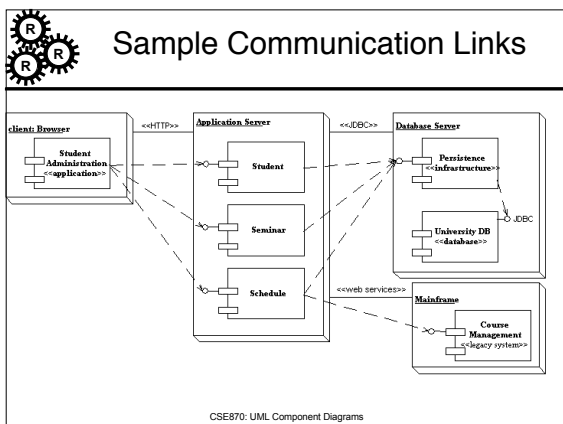
---

---

---

---

---




---

---

---

---

---

---

---

---