UML Diagram Types

Dynamic Models
- activity diagrams
- statechart diagrams
- interaction diagrams
  - sequence diagrams
  - collaboration diagrams
- use case diagrams

Structural Models
- class diagrams
- object diagrams
- packages

Architectural Models
- component diagrams
- *deployment diagrams*

Architectural Family

- Component Diagram: shows the organization and dependencies among a set of components (i.e., software deployment)
- Deployment Diagram: shows the configuration of run-time processing nodes and the components that live on them (i.e., hardware deployment)

Deployment: Node

def'n: physical element that exists at run-time and represents a computational resource (some memory and/or some processing)

- hardware topology
- processor or device on which component may be deployed

Node

Convention

- cube with name (simple or path name)
- can use visually descriptive stereotypes
- can have adornments (tagged values)
- can have dependency, generalization, and association
- can be nested

Node vs. Component

- components represent physical packaging of logical elements
- nodes represent physical deployment of components

- logical side: classes, interfaces, state machines
- physical side: software is to components as hardware is to nodes

Connections

def'n: physical (e.g. ethernet) or indirect (satellite) connection among nodes

- can use roles, multiplicity
- can use stereotypes

Convention

- shown as solid line between nodes

Common Techniques

- Show available nodes relevant to context
- Use stereotypes to make icons intuitive. That is, render icons to give visual cues to audience
- Consider if any attributes or operations apply to any node

Deployment Diagrams

def'n: shows configuration of run-time processing nodes and components that live on them

- shown as vertices and arcs
- class diagrams that focus on system's nodes
- UML sufficient to describe hardware

Deployment Diagrams

Convention

- nodes
- dependency and association relationships
- can have other relationships (inheritance, aggregation)

Common Uses

- Embedded Systems: systems that are a part of another system and interface with the physical world
- Client/Server System: system that hold a clear distinction between the user interface (client) and persistent data (server)
- Distributed system: globally distributed systems that encompass multiple levels of servers

Embedded Systems

- model physical devices
- may have noisy, non-linear devices

To model

- identify devices
- provide visual cues
  - at minimum, separate processors from devices
- model relationships
- expand more intelligent devices

Hints and Tips

- focus on one aspect of system's static deployment
- contain only elements that are essential to context
- provide appropriate detail
- don't be too minimalist