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)

Component

def'n: physical and replaceable part of a system that conforms to and provides the realization of a set of interfaces

- physical: bits
- replaceable: substitutable, conforming to same interfaces
- part of a system: software partition of a system
- interfaces: collection of operations to specify service of a class or component

Component

Convention

- name
  - simple name: (e.g. agent)
  - path name: (e.g. system::dialog)
- adornments
  - tagged values
- symbol
  - default: rectangle, with two smaller rectangles
  - iconic representation

Components vs. Classes

Similarities

- names
- realize set of interfaces
- relationships
- nesting
- instances

Differences

- physical implementation vs. logical abstraction
- operations reachable only through interfaces vs. attributes and operations directly

Interface

def'n: collection of operations to specify service of a class or component

- represents seams in systems
- components realize the interfaces
- other components access services (dependency) through interfaces

Convention

- short form (dependency)
- elided form (realization and dependency)
- fully specified form (expanded interface)

Separation of Interface and Component

- separate what class does from how it interfaces
- break direct dependency between two components
- component will function properly as long as it uses given interface
- permits assembly of systems from binary replaceable parts
- extension through new services and new interfaces

Types of Components

Deployment

- necessary and sufficient to form an executable system
- e.g. executables, libraries

Work Product

- residue of development process
- e.g. source code files, data files

Stereotypes

- executables: component that may be executed on a node
- library: static or dynamic library
- table: represent a database table
- file: represents a document containing source code or data
- document: represents a document

Common Uses

- Model deployment components of implementation
- Configuration management of partitions of system as it evolves

Hints and Tips

- crisp abstraction of physical aspect of system
- realization of small, well-defined set of interfaces
- directly implements set of classes to carry out semantics of interfaces
- loosely coupled with other components

Component Diagram

def'n: shows organization and depend's among a set of components

- physical aspects of OO system
- static, implementation view of system
- physical things that reside on node
- whereas class diagram is abstract overview, component diagram is system software architecture

Component Diagram: Source Code

- identify files
- use packages for grouping
- used tagged values when appropriate (version, author, date)
- show compilation dependencies