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

Modeling Architecture Views

def'n: projection into organization and structure of a system that is focused on a particular aspect of that system

System Decomposition
- Design View: class level design
- H/W and S/W View: h/w and s/w physical deployment
- Use Case View: system functionality view

Structural Family: Package

def'n: general purpose mechanism for organizing modeling elements into groups
- control visibility
- present different views of system's architecture
- group elements that are semantically close, and tend to change together
- cohesive within and loosely coupled between (other packages)
- mechanism to organize things in a model
- no identity outside of system

Package

Convention

- tabbed folder (simple or path name)
- can be nested
  - package name in a package implies that package in question is nested in an enclosing package (e.g., sensors::vision::camera)
- package may own other elements
  - classes, interfaces, components, nodes, other packages
- packages imply composition relationship
  - destroying package destroys elements in the package
- all elements are owned by 0..1 package (element cannot be owned by >1 package)

Visibility

- +public: visible to contents of any package that imports element's enclosing package
  - public parts of components make up the interface
  - strict definition of "interface" relevant in component diagram
- #protected: only seen by children
  - visible only to packages that inherit from a parent package
- -private: cannot be seen outside of the package in which declared

Importing and Exporting

Importing def'n: granting one-way permission for the elements of one package to access elements in another package

- not transitive

Convention

- dependency relationship with stereotype <<import>>

Exporting def'n: public part of a package

- visible only to the contents of those packages that explicitly import the package

Generalization

- Used to specify families of packages
- Children inherit public (+) and protected (#) elements of parent
- Can replace general elements and add new ones
- Specialized package can be used anywhere a more general package can

Standard Elements

- facade: specifies a package that is only a view of some other package
- framework: specifies a package consisting mainly of patterns
- stub: specifies a package that serves as a proxy for the public contents of another package
- subsystem: specifies a package representing an independent part of the entire system being modeled
- system: specifies representing the entire system being modeled

Modeling Groups

- Look for clumps that are conceptually or semantically close
- Surround with a package
- Distinguish public elements, mark all others protected or private
- Draw explicit connections of packages via an <<import>> dependency
- If possible, find generalizations and connect families of packages

Hints and Tips

- Package represents a crisp boundary around a set of related elements
- Package is loosely coupled with other elements but highly cohesive within package
- Are not nested deeper than 3 levels
- Is balanced (one package does not own too much work)