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

Structural Family: Subsystem

def'n: part of complete system defined by an interface with hidden structure
- control visibility
- present different views of system's architecture
- domain-specific system partitioning on a coarse level
- hides implementation details while defining consistent interface

Subsystem

Convention
- tabbed folder (simple or path name)
- <<subsystem>> stereotype or 'fork' icon
- can be nested
- subsystem may own other elements
  - classes, components, and other subsystems
- subsystems imply composition relationship
  - destroying subsystem destroys elements in the subsystem
- all elements are owned by 0..1 subsystem (element cannot be owned by >1 subsystem)

Structural Family: Package

def'n: general purpose mechanism for organizing modeling elements into groups

- control visibility
- 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
  - subpackage name in a package implies that subpackage 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

Modeling Groups

- Look for clumps that are conceptually or semantically close
- Surround with a subsystem or 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 subsystems and packages

Summary

Subsystem

- Part of system that is defined to outside by an interface
- Has hidden implementation details

Package

- Collections of model elements of arbitrary types used to structure model into smaller units
- Loosely coupled with other elements but highly cohesive within