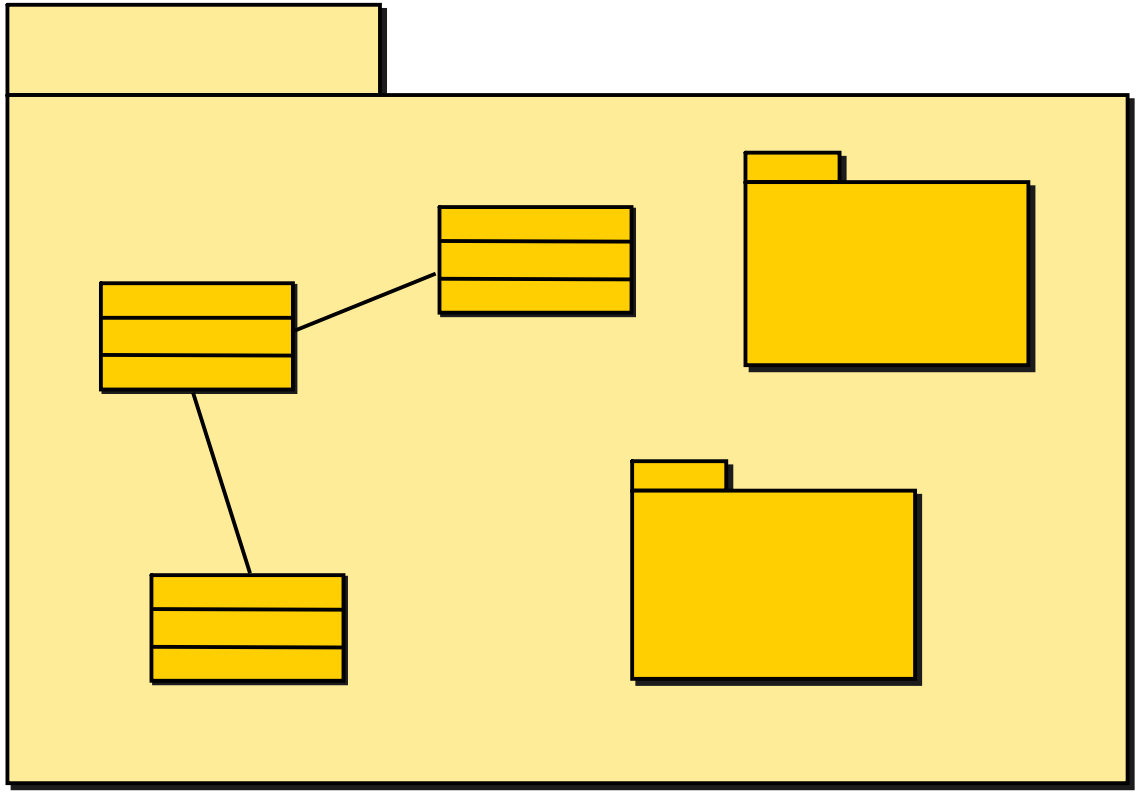
CS560

Lecture: UML Packages and Subsystems

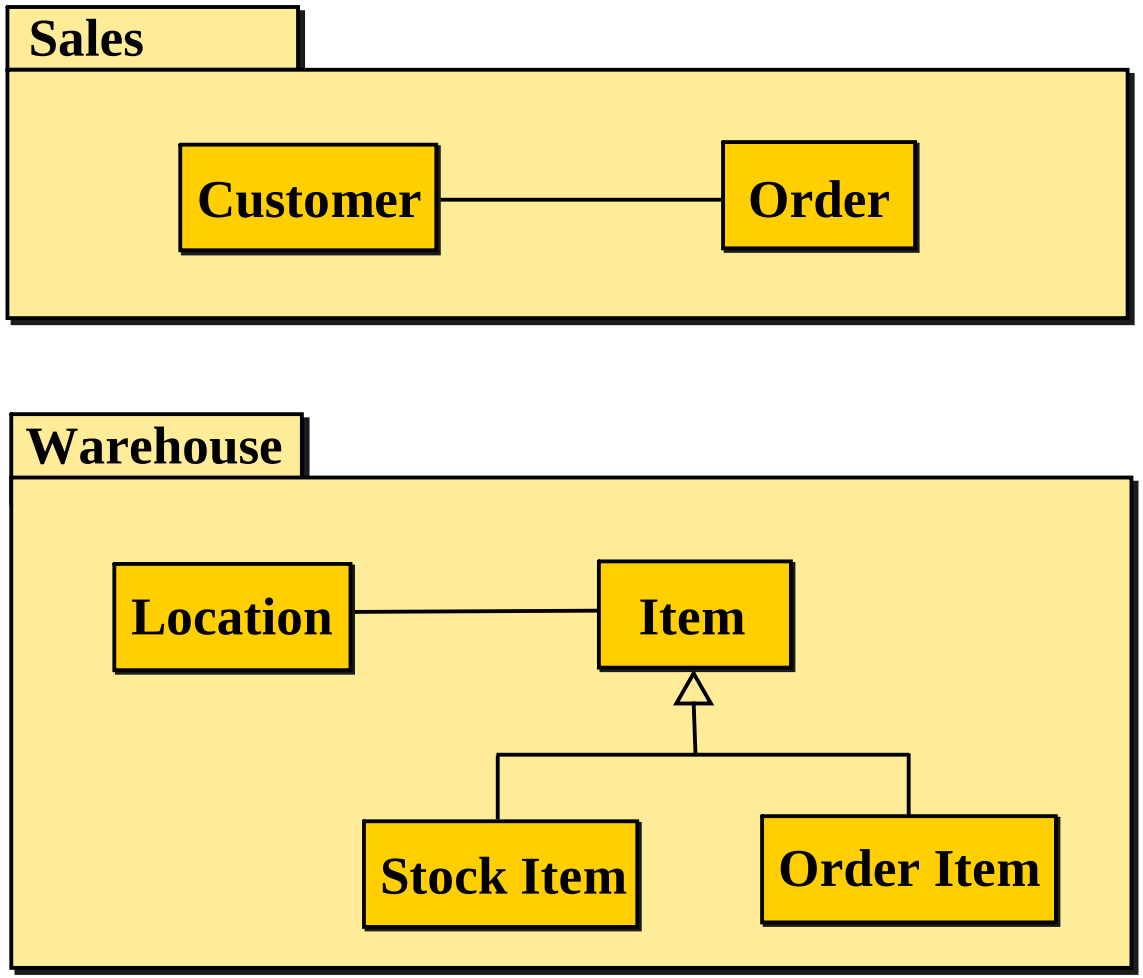
Based on slides by OMG UML Revision Task Force

Package



A package is a grouping of model elements

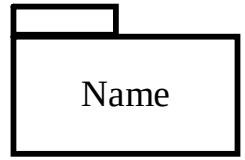
Package – Example



Package

- A package can contain model elements of different kinds
  - Including other packages to create hierarchies
- A package defines a namespace for its contents
- Packages can be used for various purposes

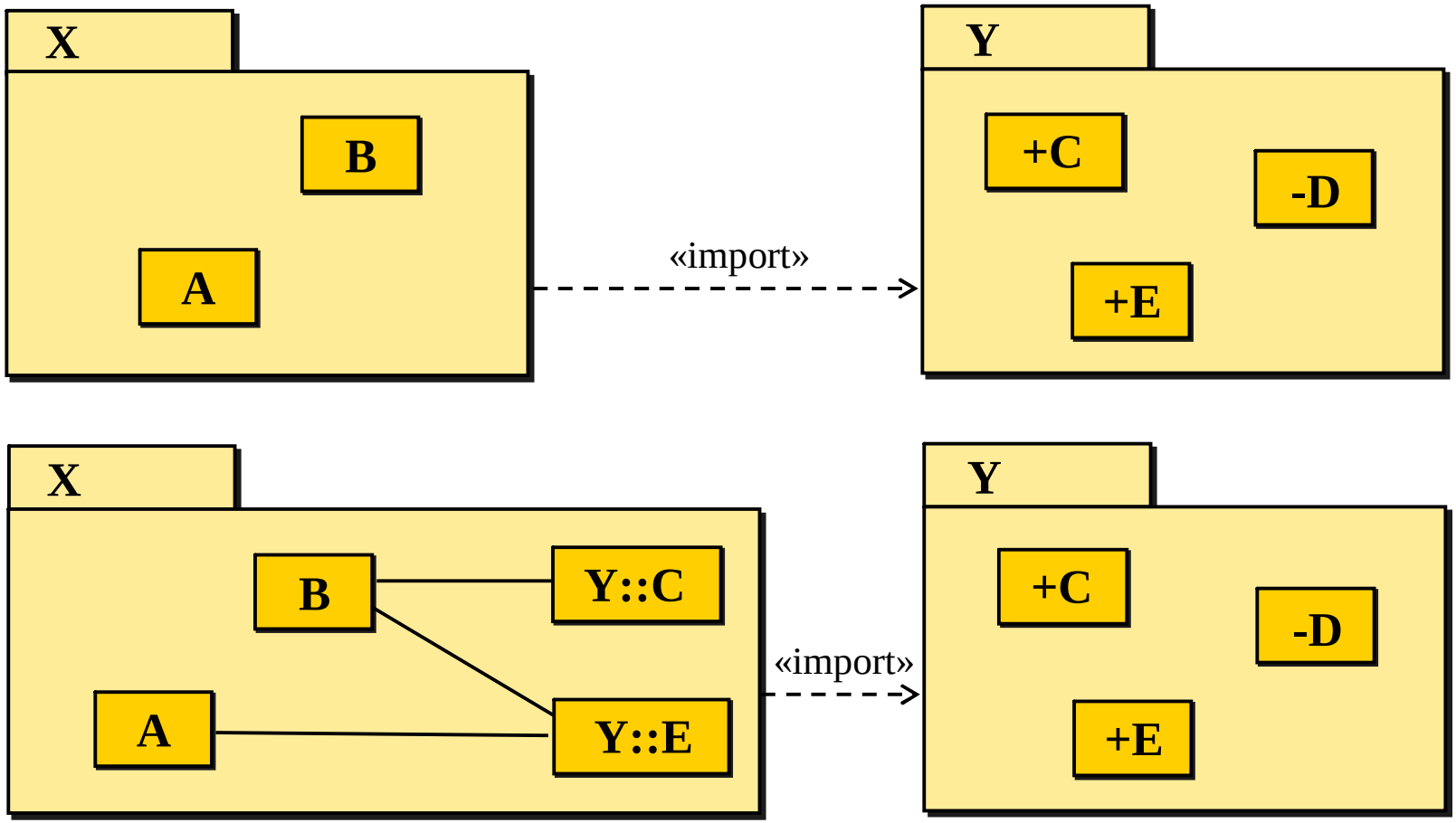
Core Concepts

Construct	Description	Syntax
Package	A grouping of model elements.	
Import	A dependency indicating that the public contents of the target package are added to the namespace of the source package.	«import» 
Access	A dependency indicating that the public contents of the target package are available in the namespace of the source package.	«access» 

Visibility

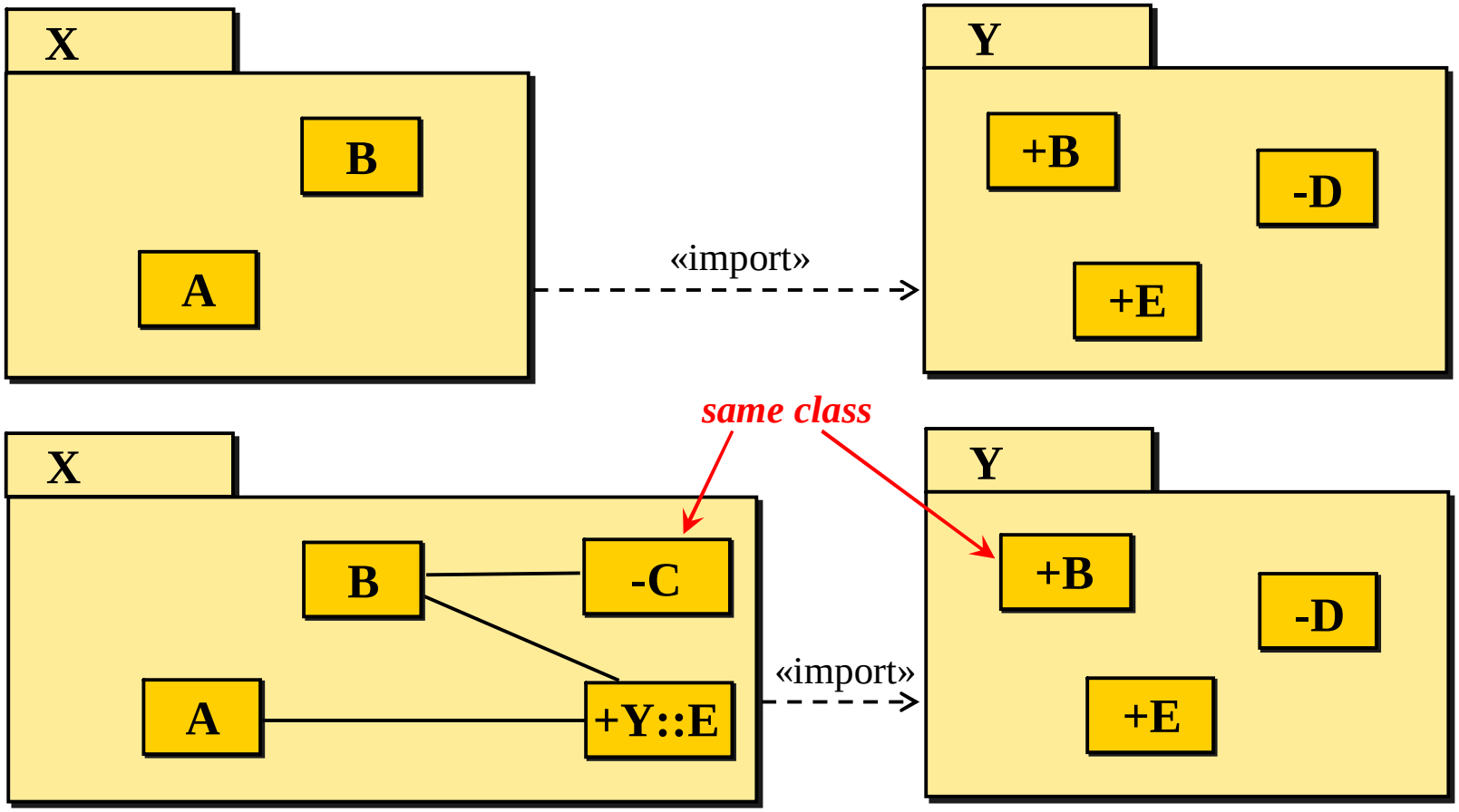
- Each contained element has a visibility relative to the containing package
  - A *public* element is visible to elements outside the package, denoted by ‘+’
  - A *protected* element is visible only to elements within inheriting packages, denoted by ‘#’
  - A *private* element is not visible at all to elements outside the package, denoted by ‘-’
- Same syntax for visibility of attributes and operations in classes

Import



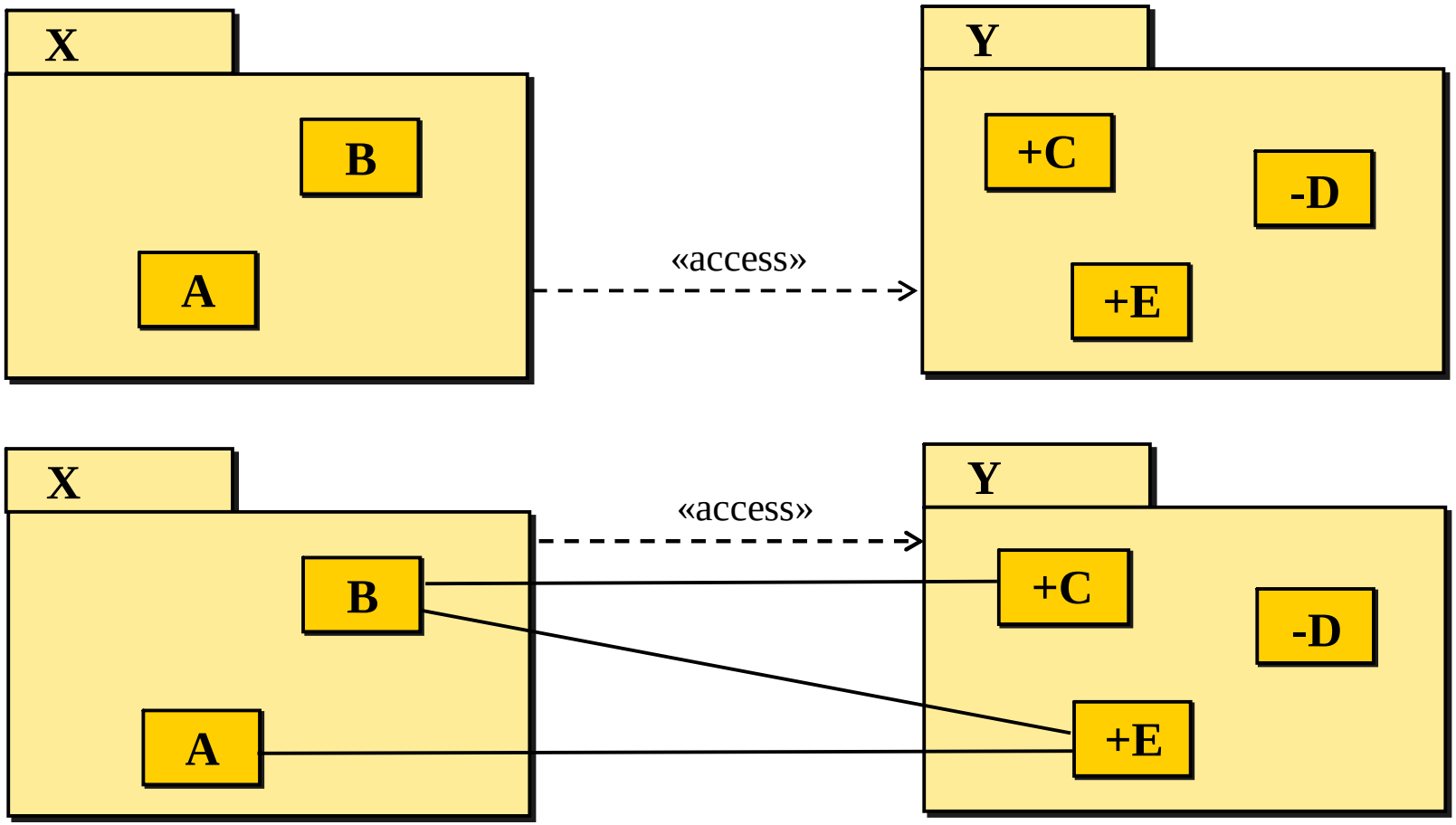
The associations are owned by package X

Import – Alias



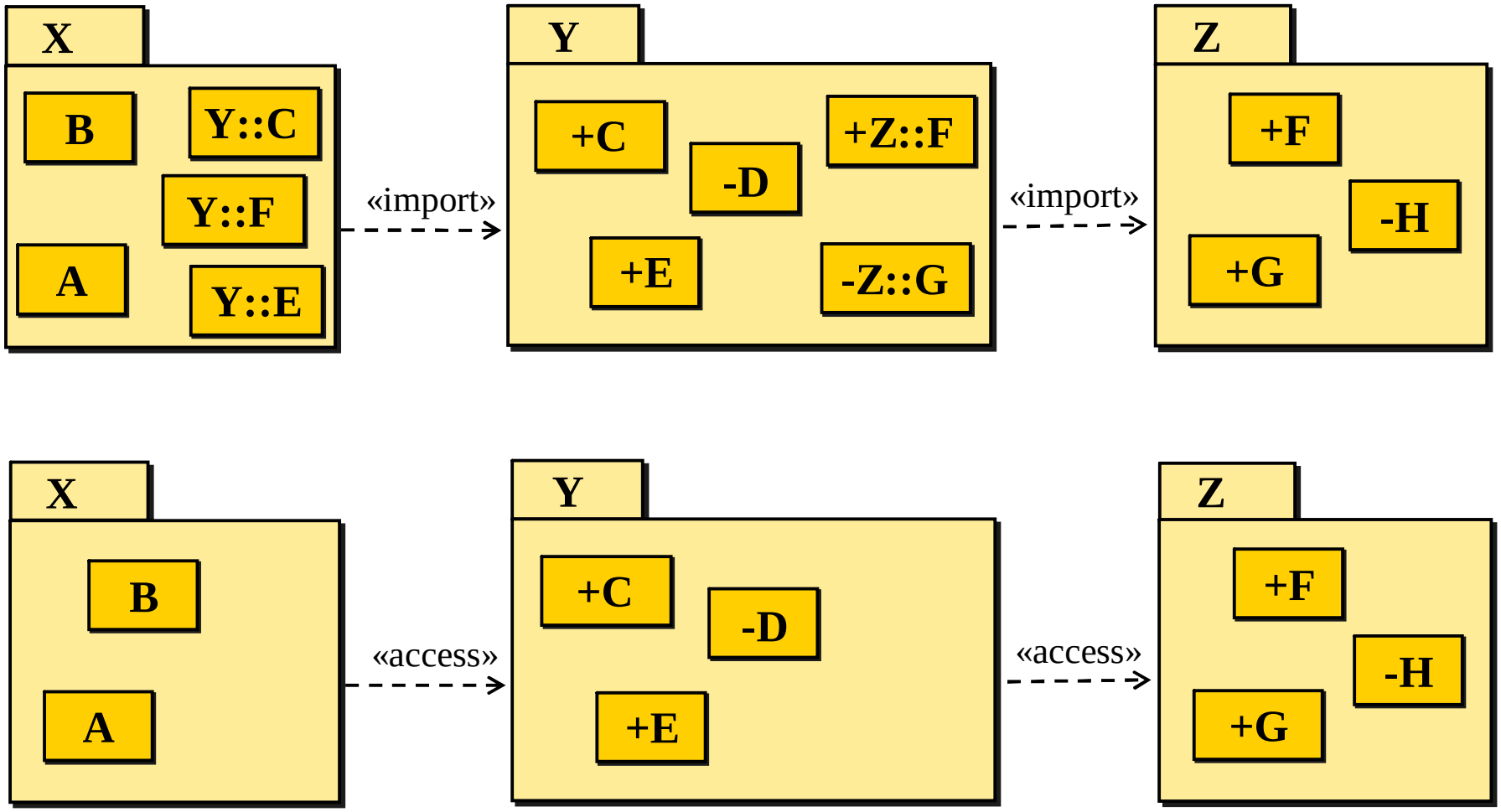
An imported element can be given a local alias and a local visibility

Access



The associations are owned by package X

Import vs. Access



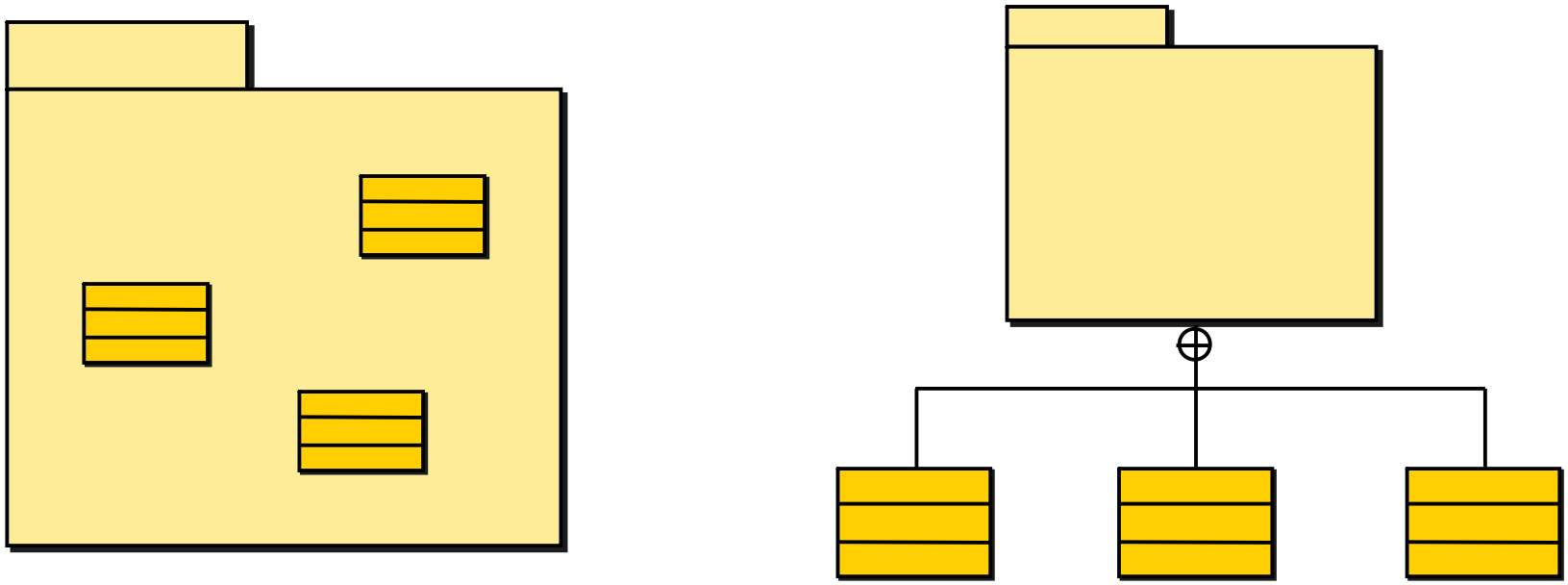
Package Inheritance

- A package with a generalization to another package inherits public and protected elements that are
  - owned or
  - imported

  by the inherited package

Diagram Tour

- Packages are shown in static diagrams
- Two equivalent ways to show containment:



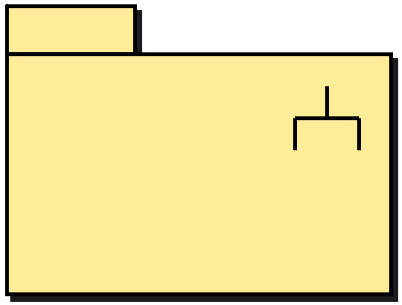
When to Use Packages

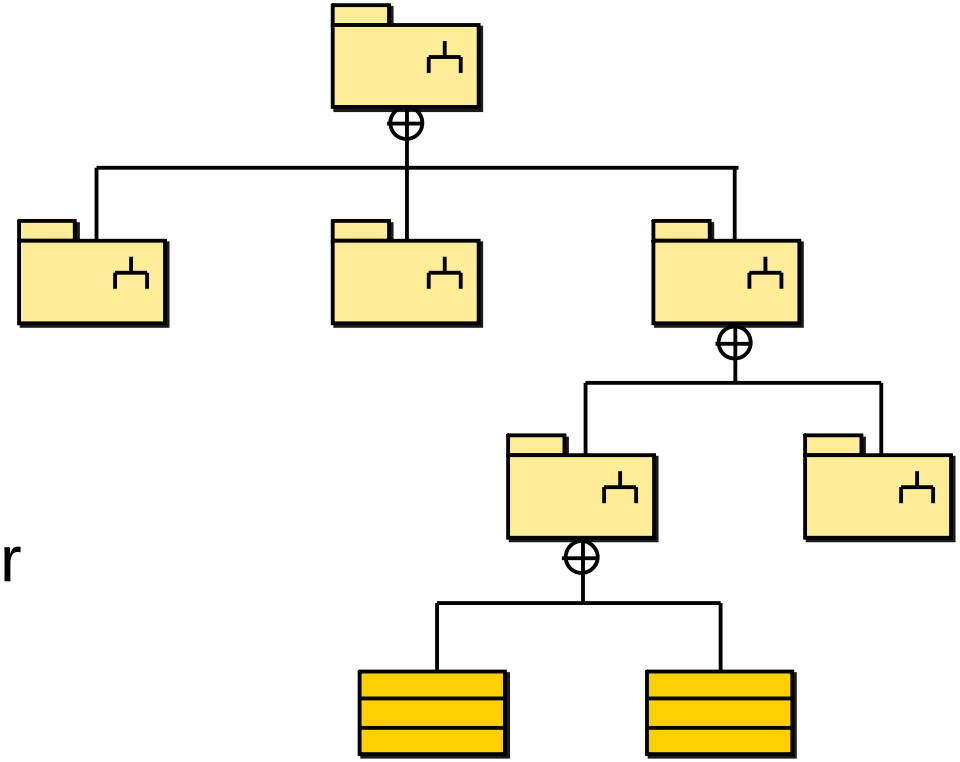
- To create an overview of a large set of model elements
- To organize a large model
- To group related elements
- To separate namespaces

Modeling Tips – Package

- Gather model elements with strong cohesion in one package
- Keep model elements with low coupling in different packages
- Minimize relationships, especially associations, between model elements in different packages
- Namespace implication: an element imported into a package does not “know” how it is used in the imported package

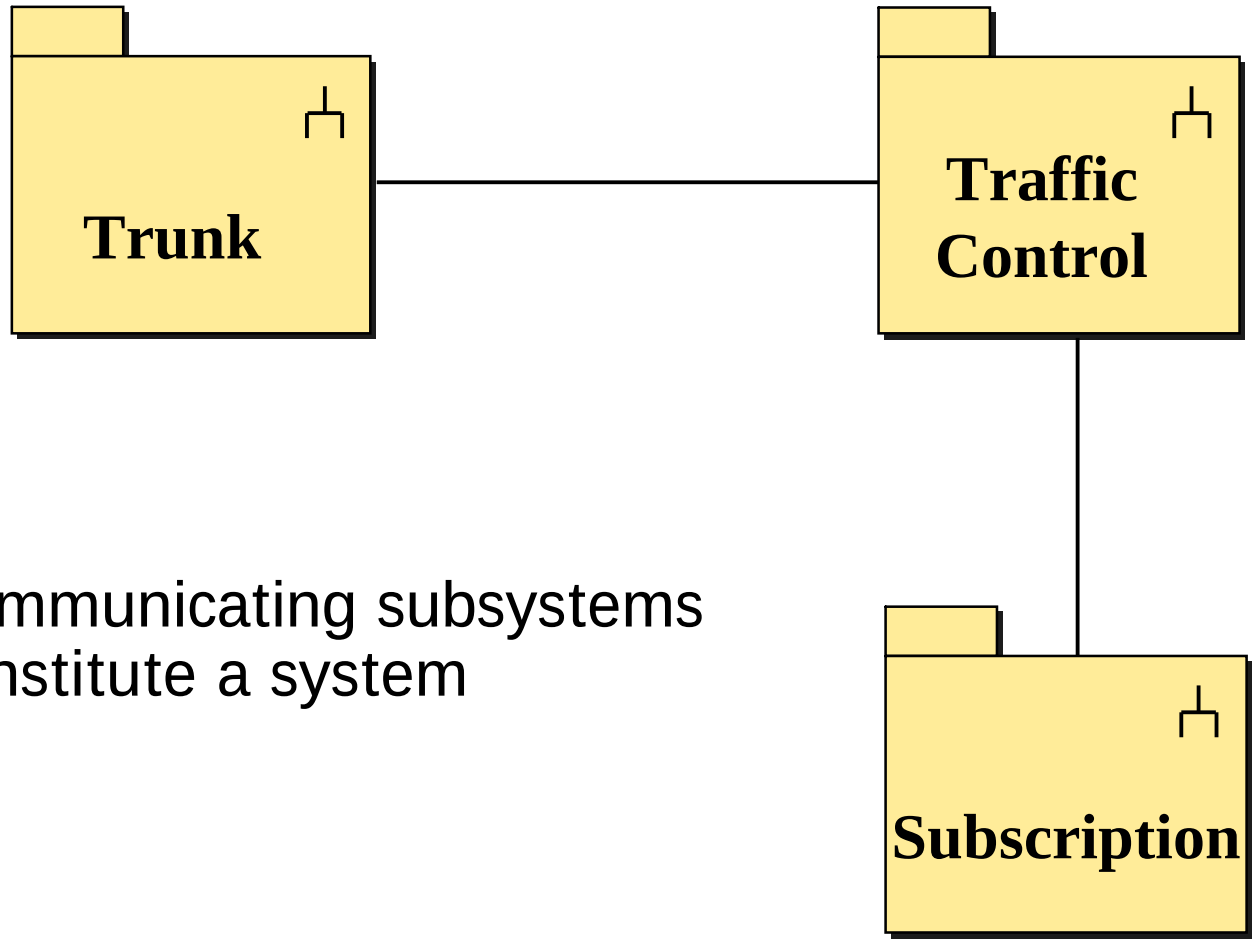
Subsystem





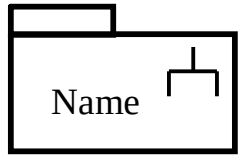
Subsystems are used for system decomposition

Subsystem – Example



Communicating subsystems constitute a system

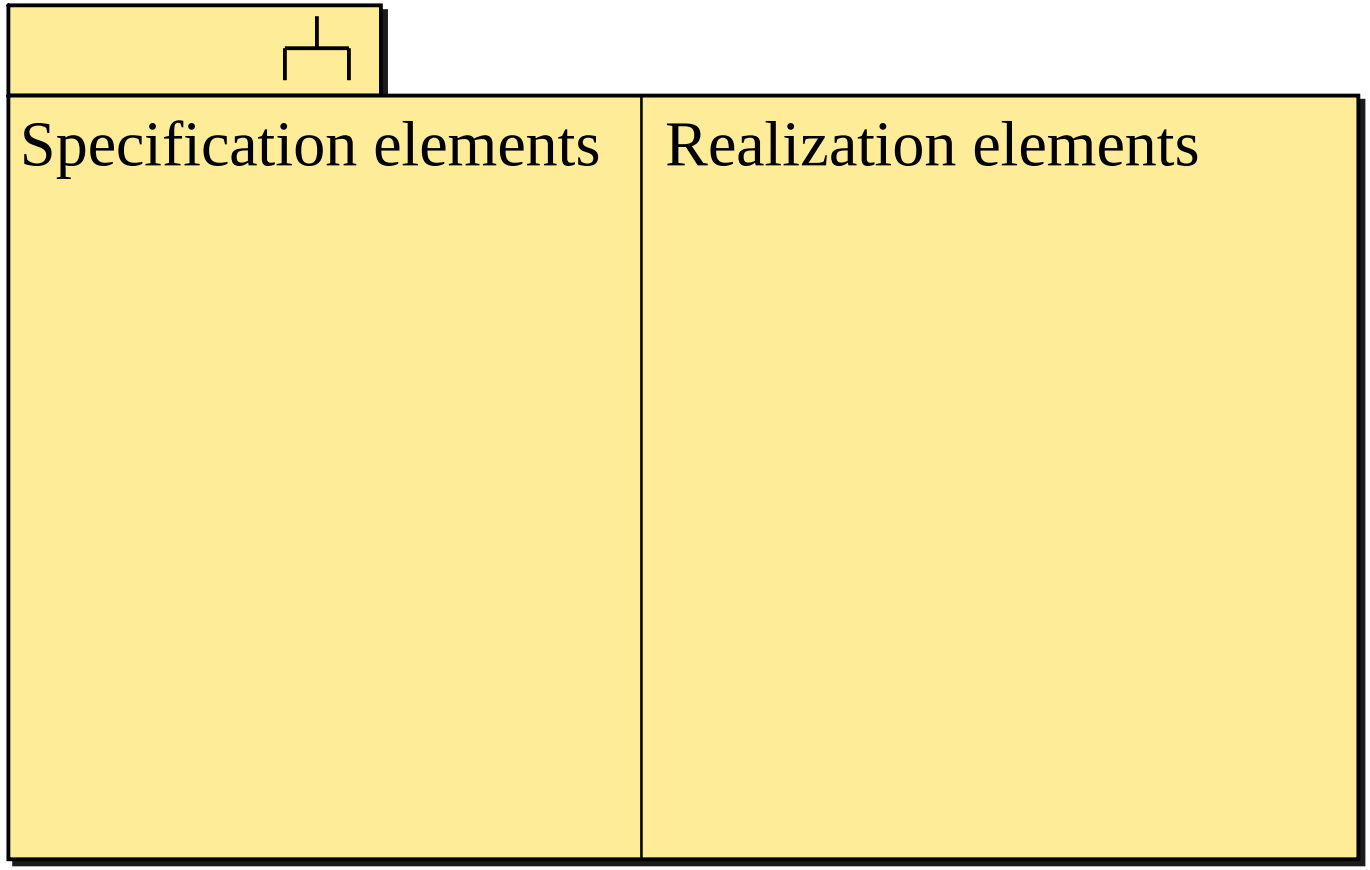
Core Concepts

Construct	Description	Syntax
Subsystem	A grouping of model elements that represents a behavioral unit in a physical system.	

Subsystem Aspects

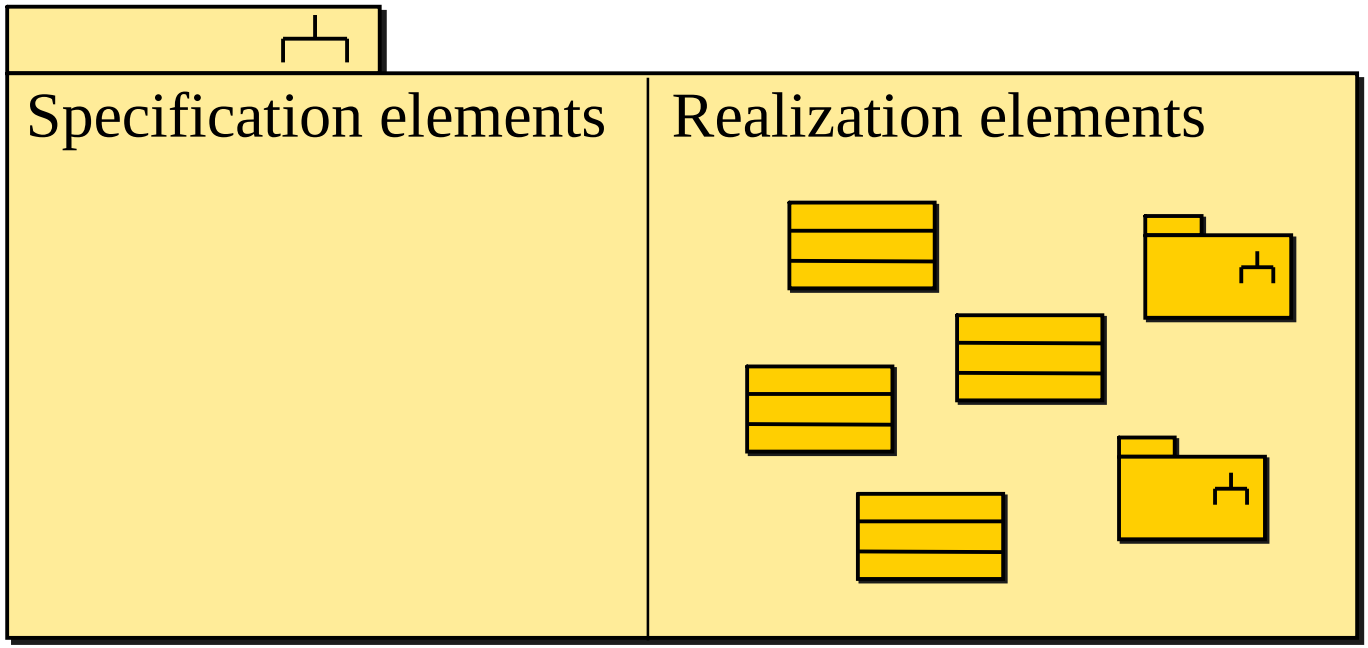
- A subsystem has two aspects:
  - An external view, showing the services provided by the subsystem
  - An internal view, showing the realization of the subsystem
- There is a mapping between the two aspects

Subsystem Aspects



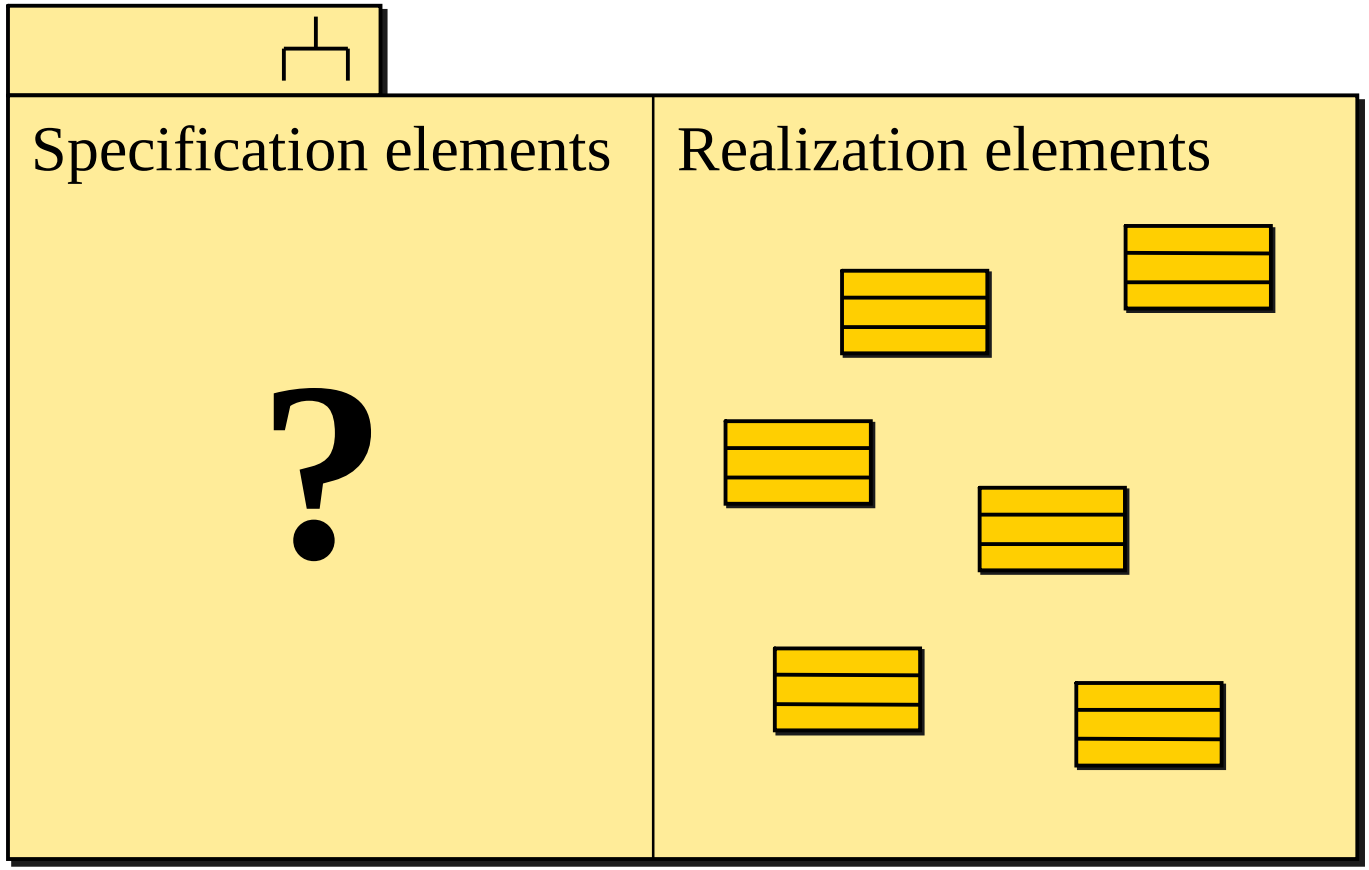
A subsystem has a specification and a realization to represent the two views

Subsystem Realization



- The subsystem realization defines the actual contents of the subsystem
- The subsystem realization typically consists of classes and their relationships, or a contained hierarchy of subsystems with classes as leaves

Subsystem Specification



The subsystem specification defines the external view of the subsystem

Subsystem Specification

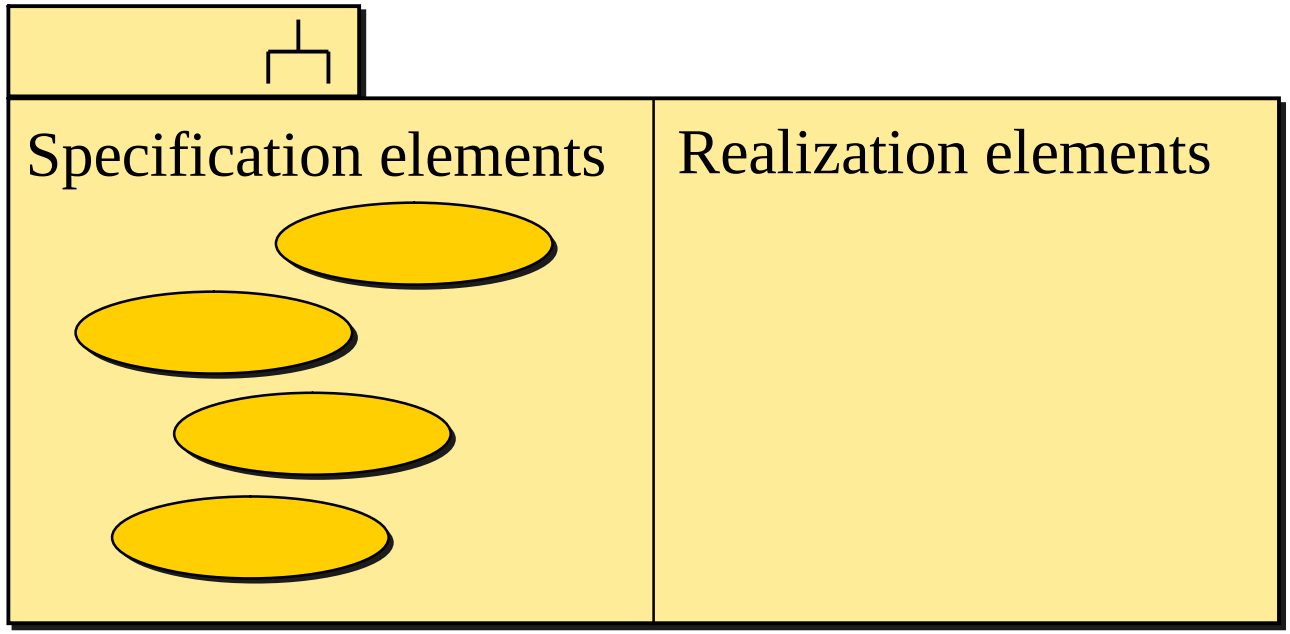
- The subsystem specification
  - describes the services offered by the subsystem
  - describes the externally experienced behavior of the subsystem
  - does not reveal the internal structure of the subsystem
  - describes the interface of the subsystem

Specification Techniques

- The Use Case approach
- The Logical Class approach
- The Operation approach

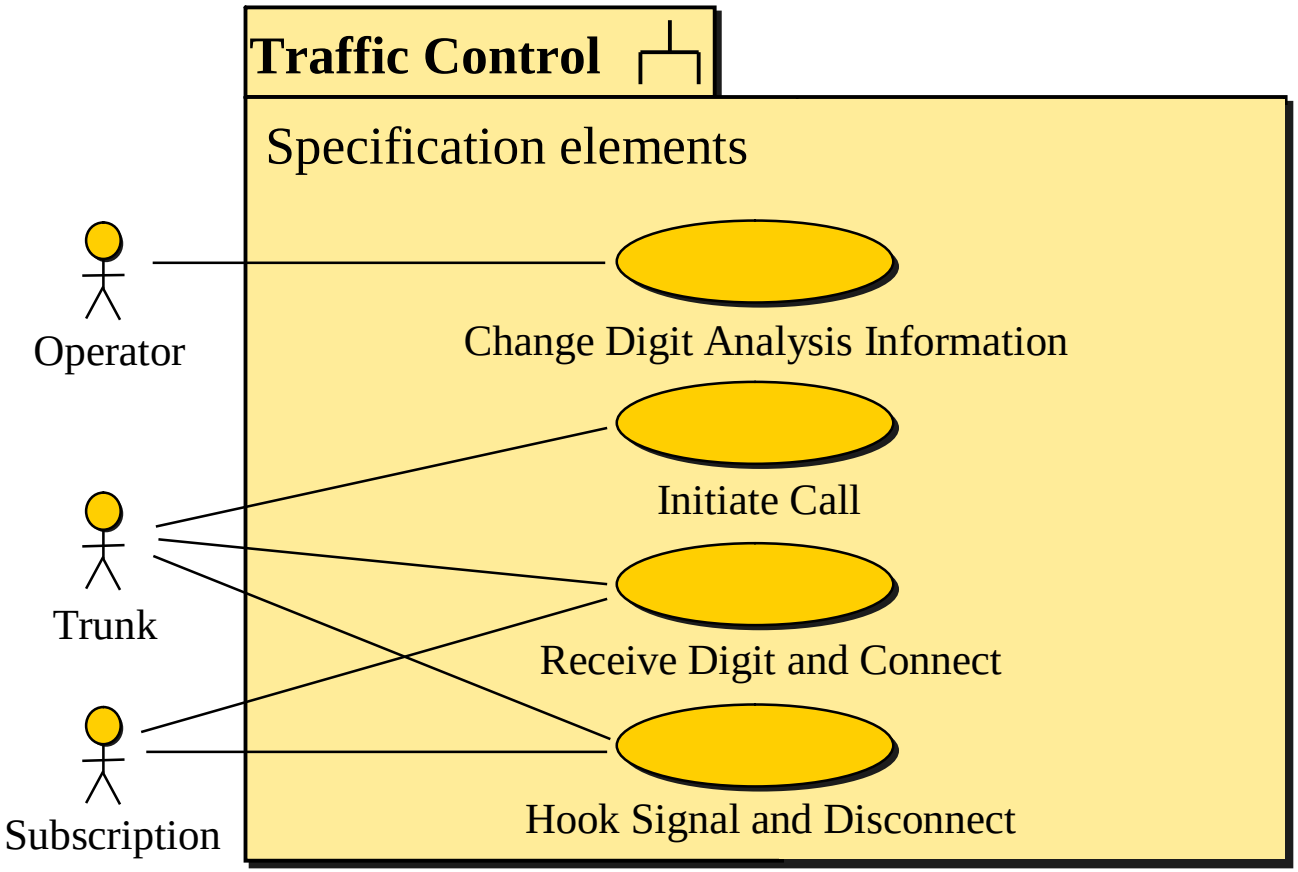
...and combinations of these.

Use Case Approach

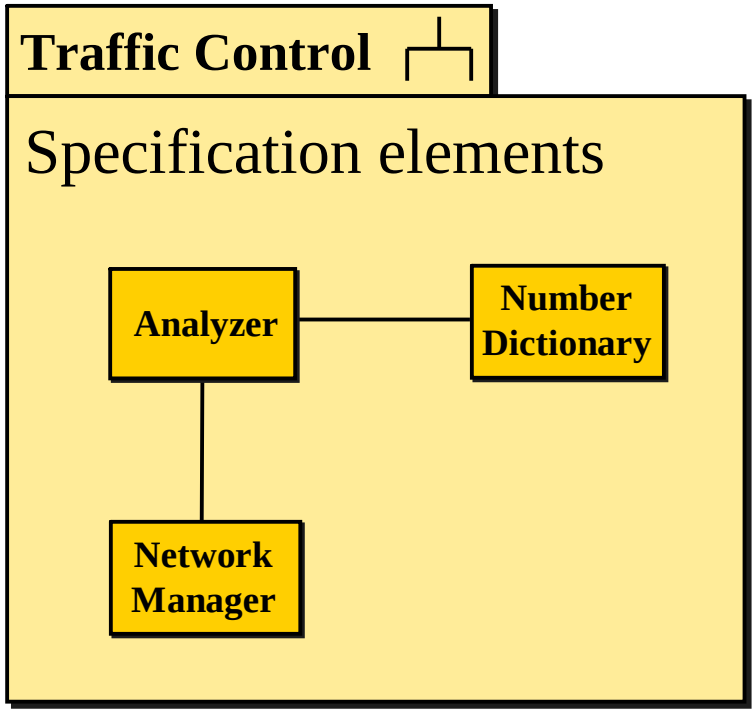


- For subsystem services used in certain sequences
- When the specification is to be understood by non-technical people

Use Case Approach – Example

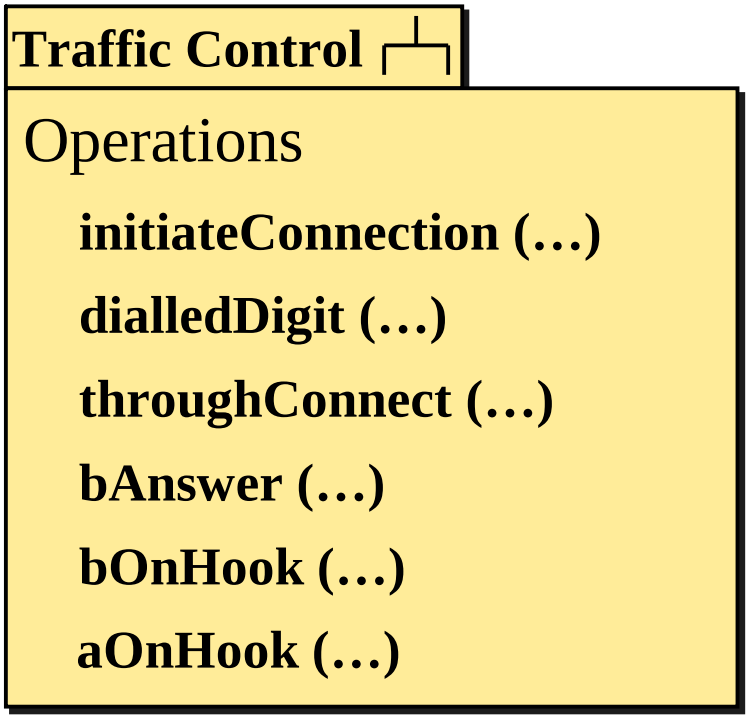


Logical Class Approach



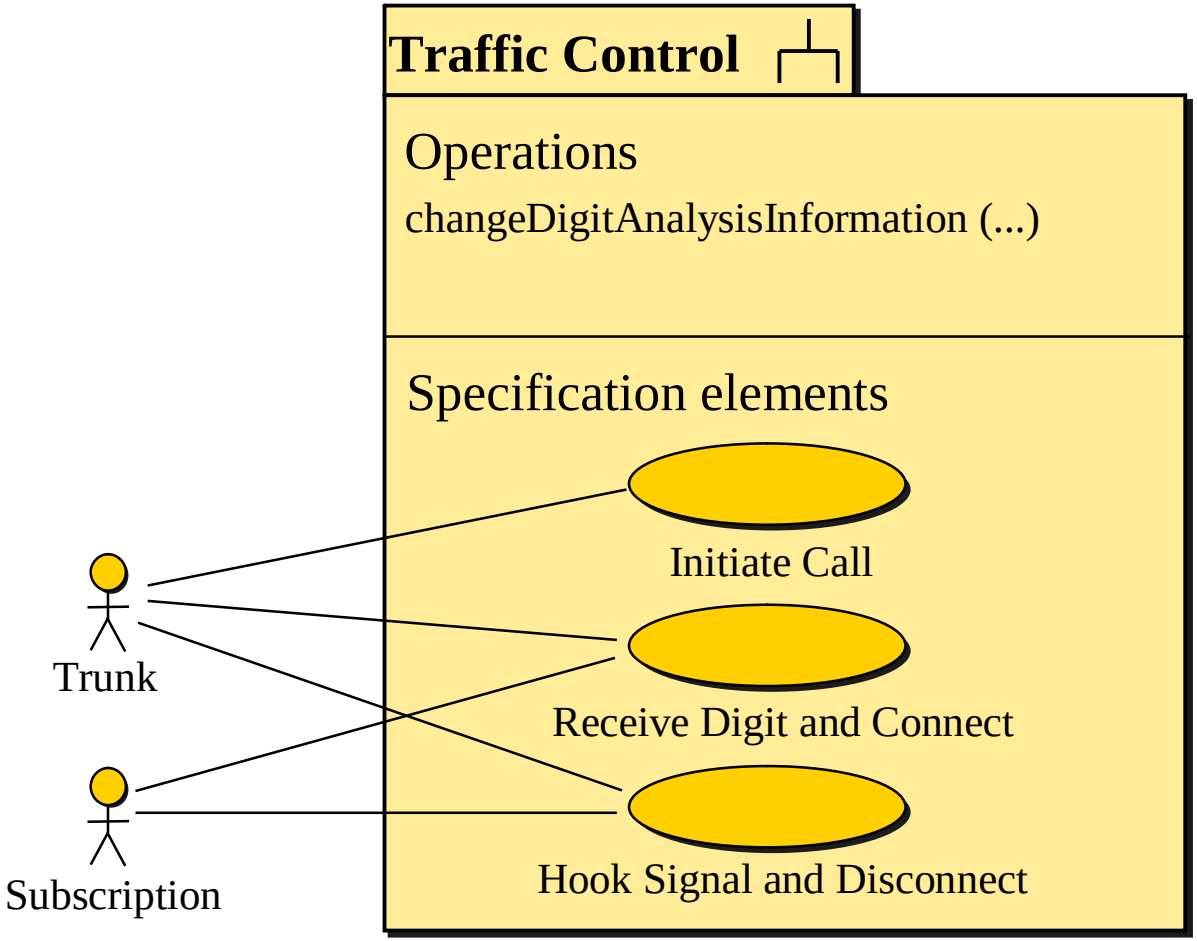
- When usage of the subsystem is perceived as manipulation of objects
- When the requirements are guided by a particular standard

Operation Approach



- For subsystems providing simple, “atomic” services
- When the operations are invoked independently

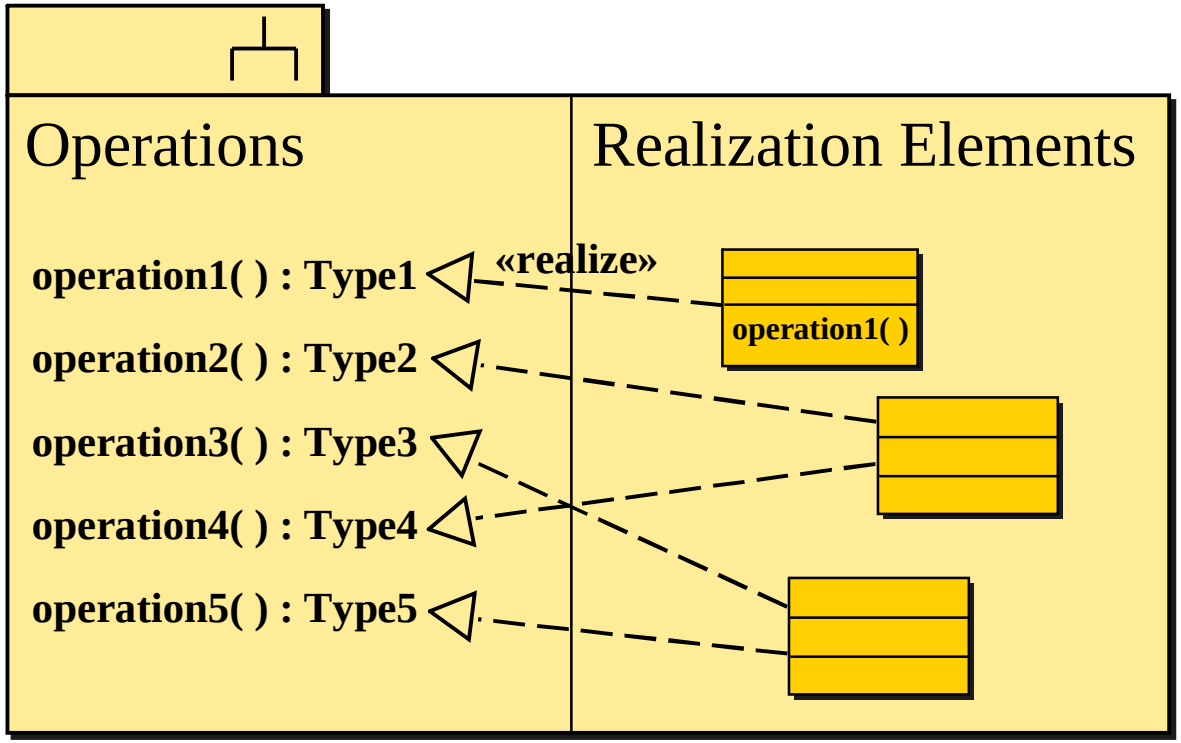
Mixing Techniques



Specification – Realization

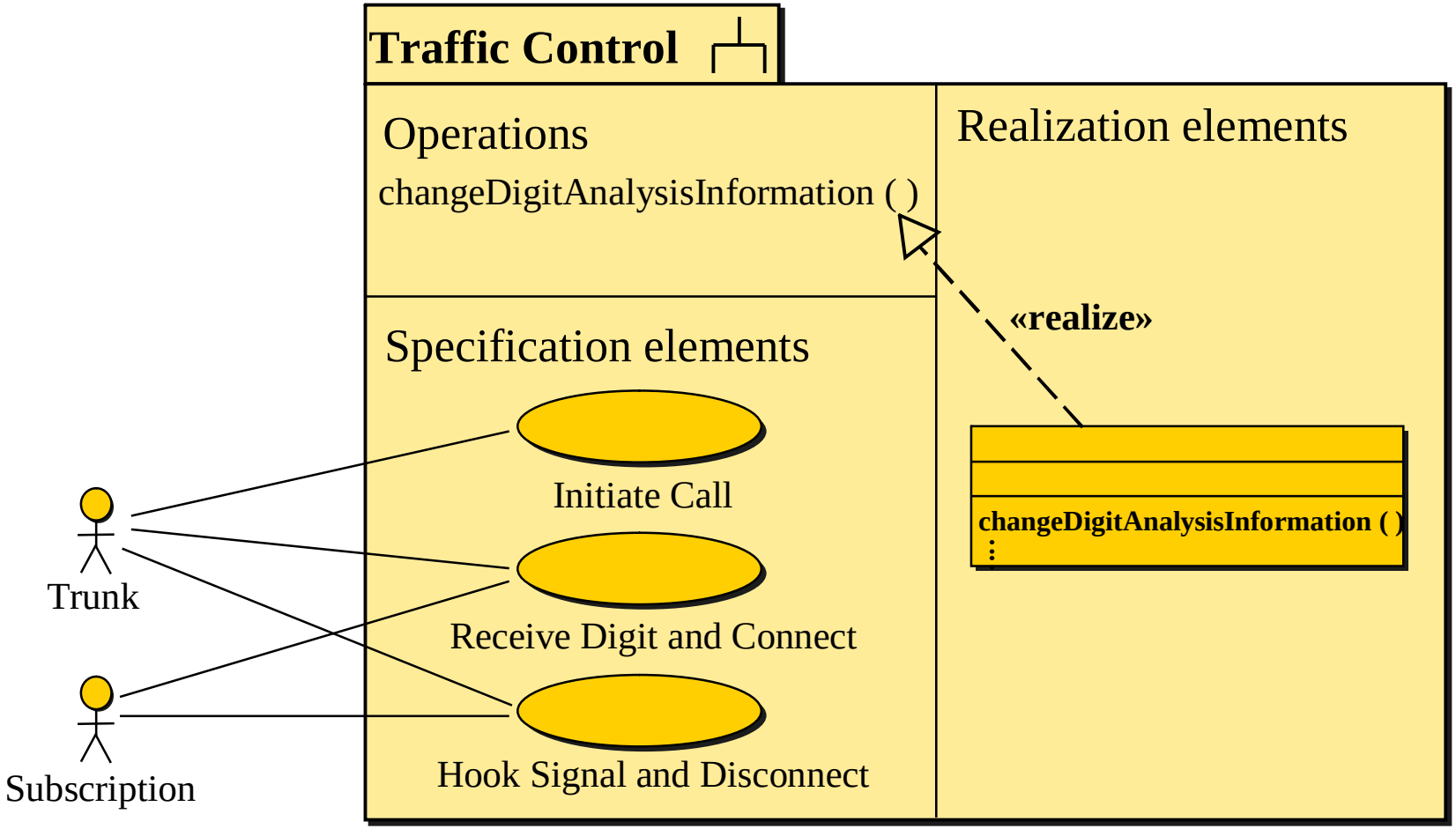
- The specification and the realization must be consistent
- The mapping between the specification and the realization can be expressed by:
  - realization relationships
  - collaborations

Realize Relationship



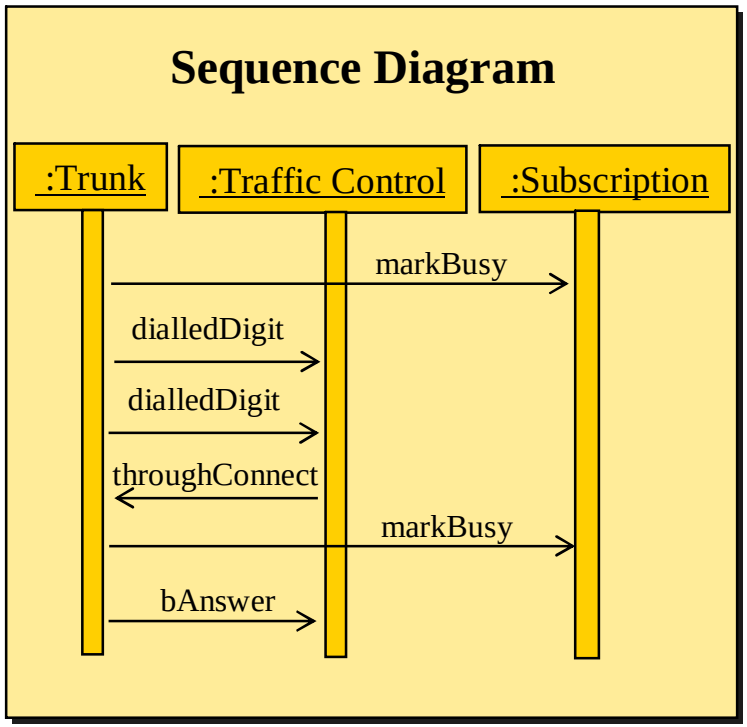
Realization is particularly useful in simple mappings

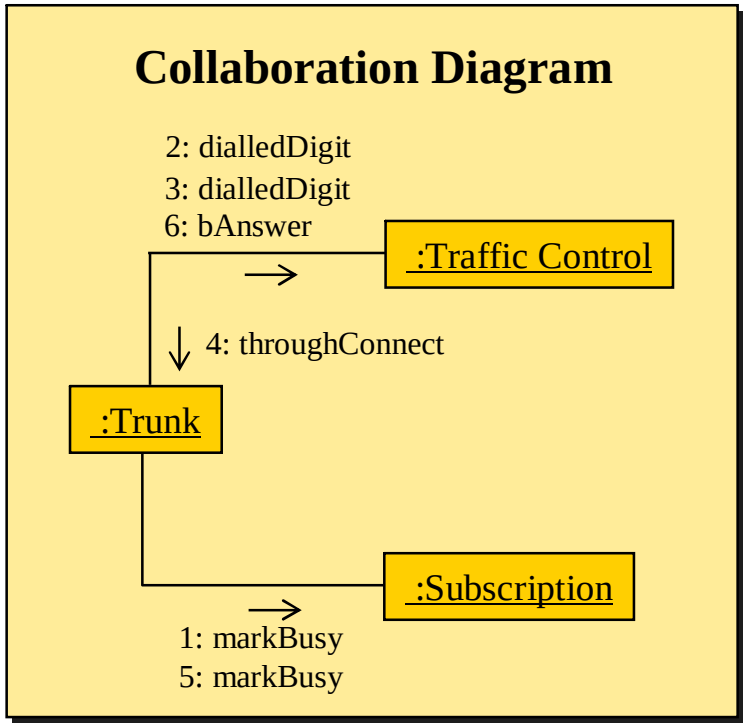
Realize – Example



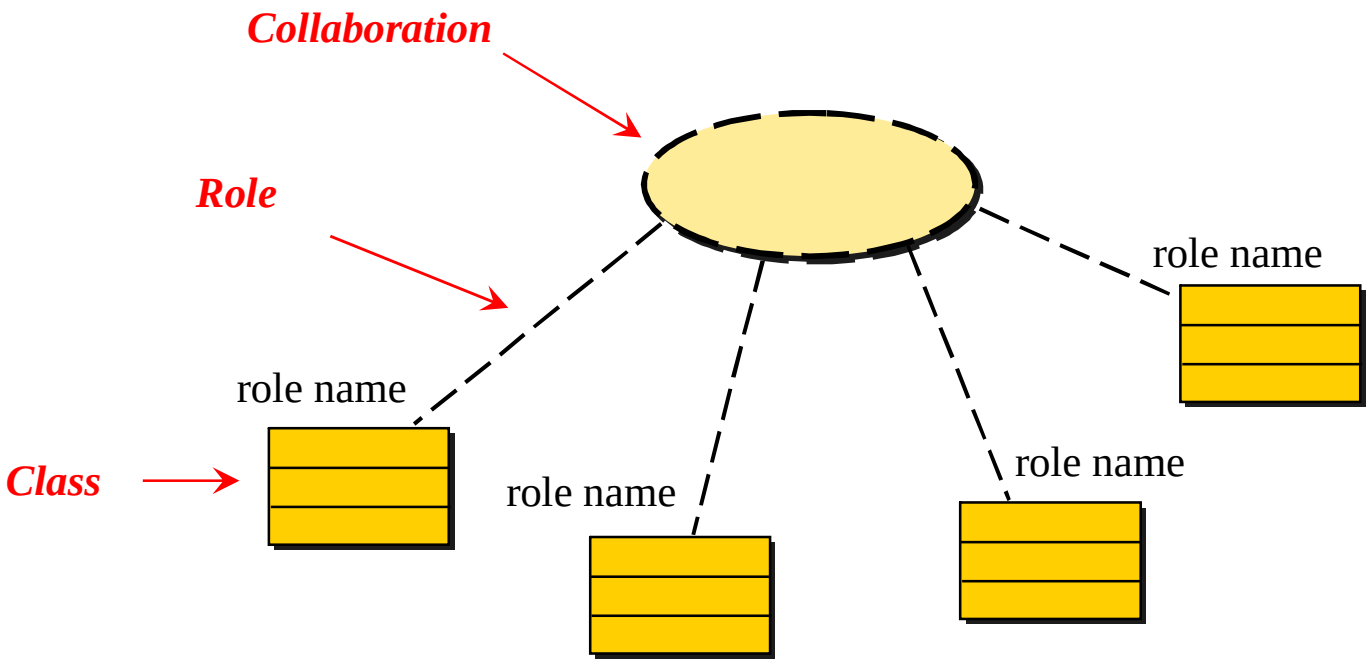
Collaboration

- A collaboration defines the roles to be played when a task is performed
- The roles are played by interacting instances



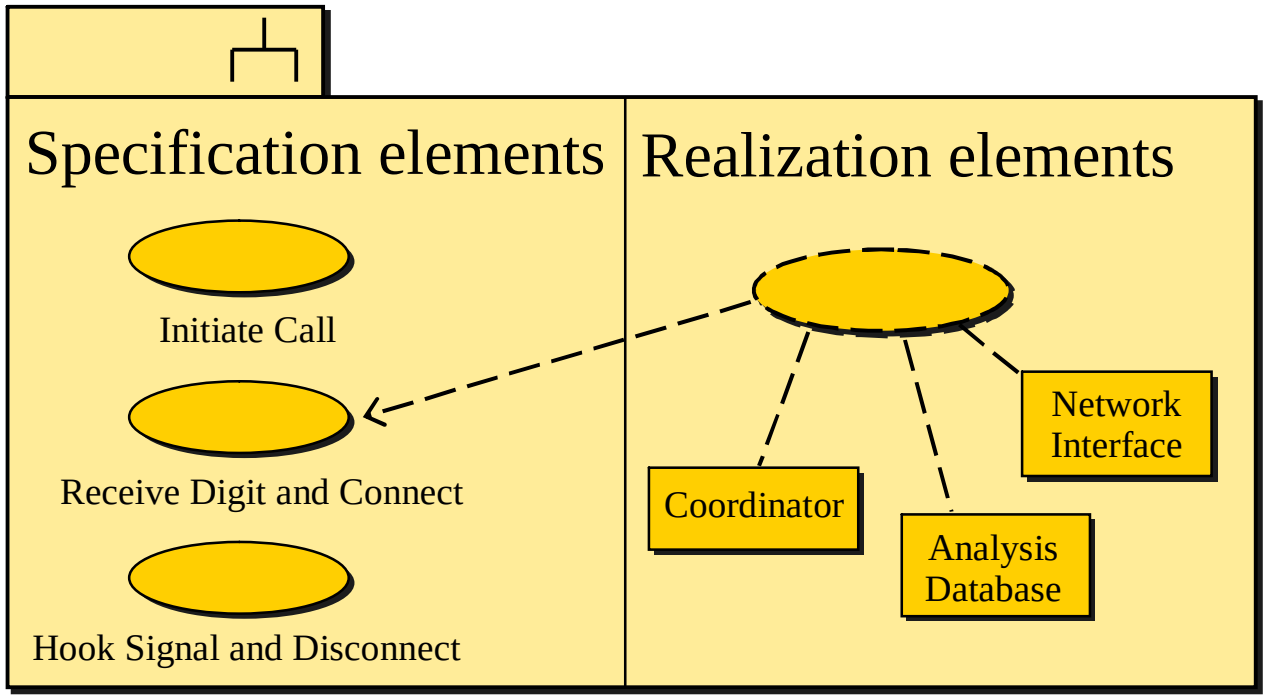


Collaboration – Notation



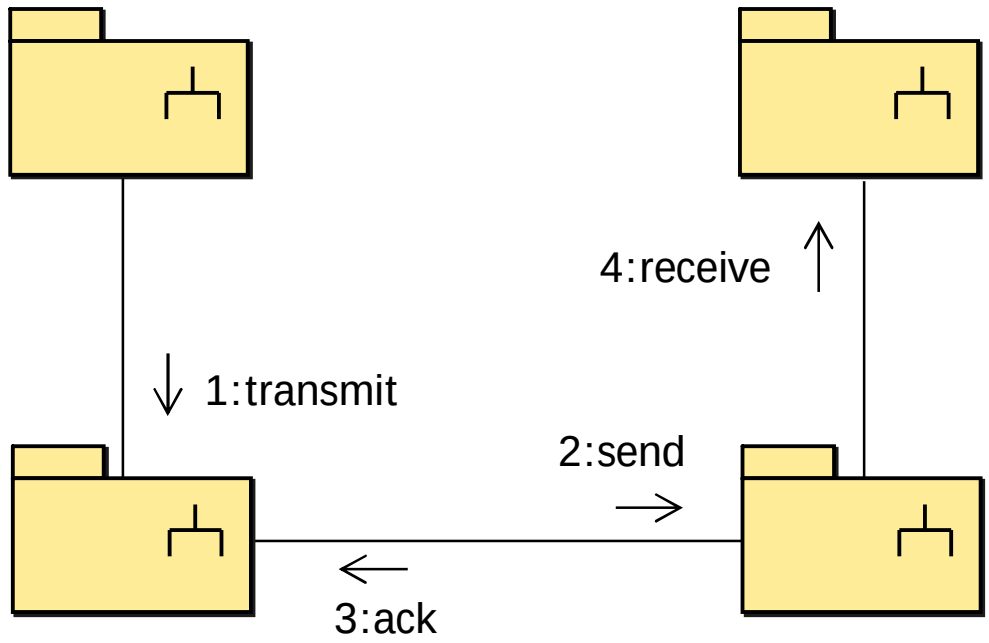
A collaboration and its participants

Collaboration – Example



Collaborations are useful in more complex situations

Subsystem Interaction



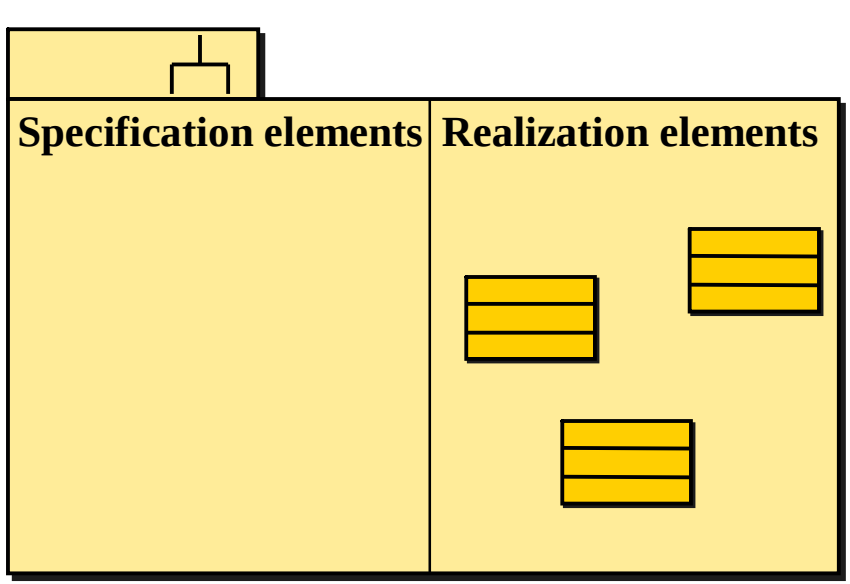
Subsystem Inheritance

- A subsystem with a generalization to another subsystem inherits public and protected elements that are
  - owned or
  - imported

  by the inherited subsystem
- Both specification elements and realization elements are inherited
- Operations are also inherited

Diagram Tour

- Subsystems can be shown in static diagrams and interaction diagrams
- “Fork” notation alternative for showing contents:



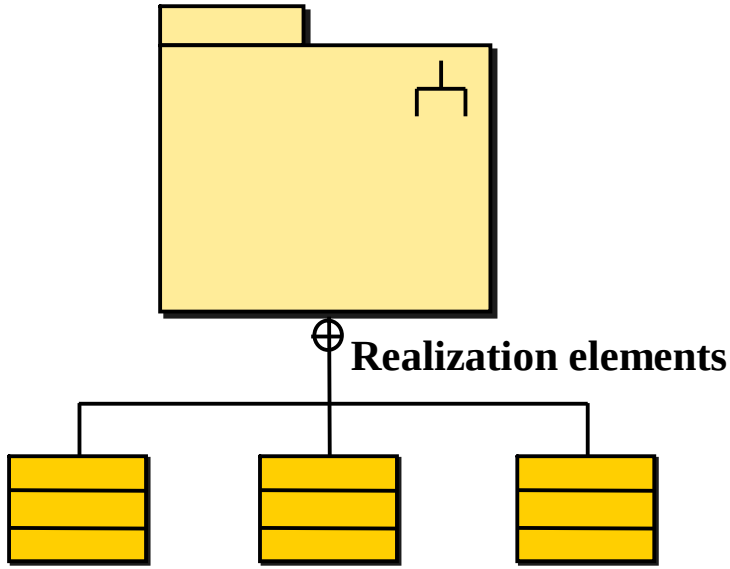
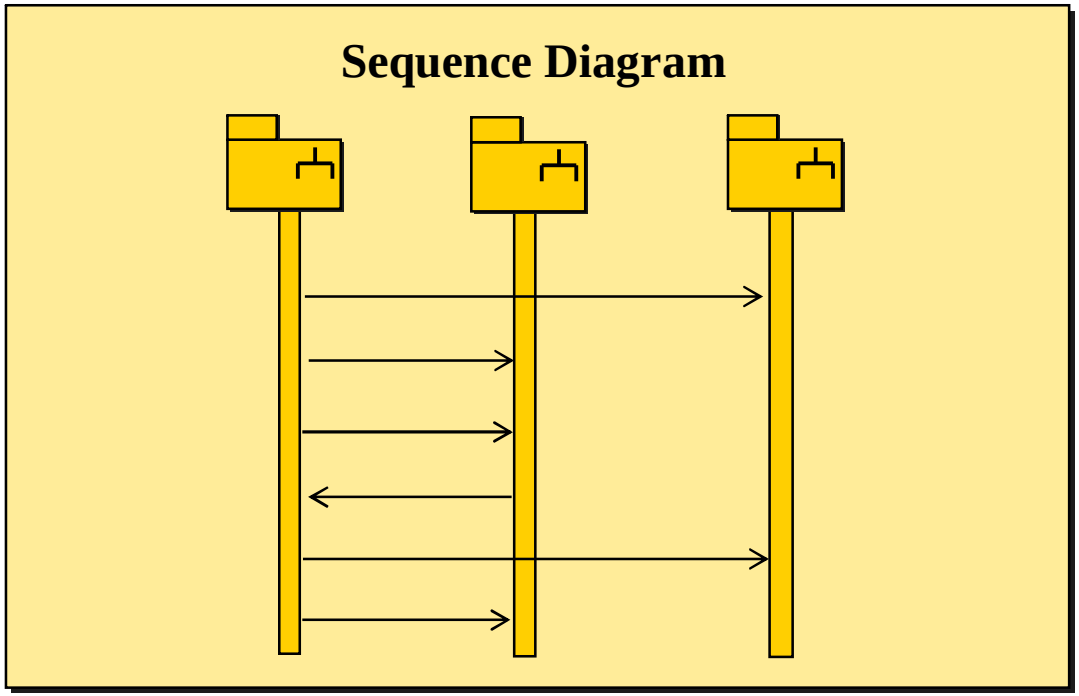


Diagram Tour – continued

- Subsystems can be shown in interaction diagrams
  - collaboration diagrams
  - sequence diagrams



When to Use Subsystems

- To express how a large system is decomposed into smaller parts
- Distributed development
- To express how a set of modules are composed into a large system
- For component based development

Modeling Tips – Subsystem

- Define a subsystem for each separate part of a large system
- Choose specification technique depending on factors like kind of system and kind of subsystem
- Realize each subsystem independently, using the specification as a requirements specification

Wrap Up Model Management

- Packages are used to organize a large set of model elements
  - Visibility
  - Import
  - Access
- Subsystems are used to represent parts of a system
  - Specification
  - Realization