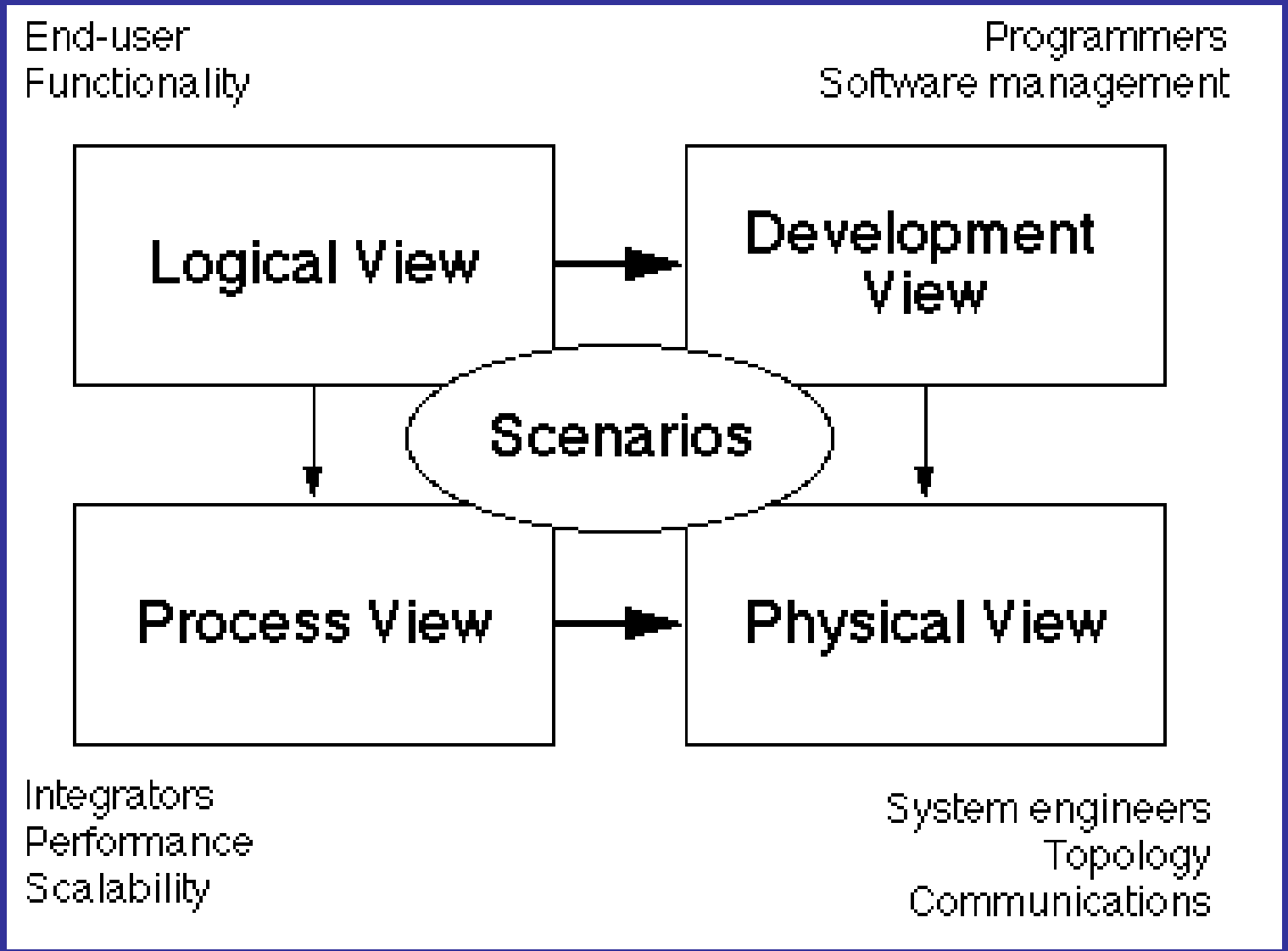
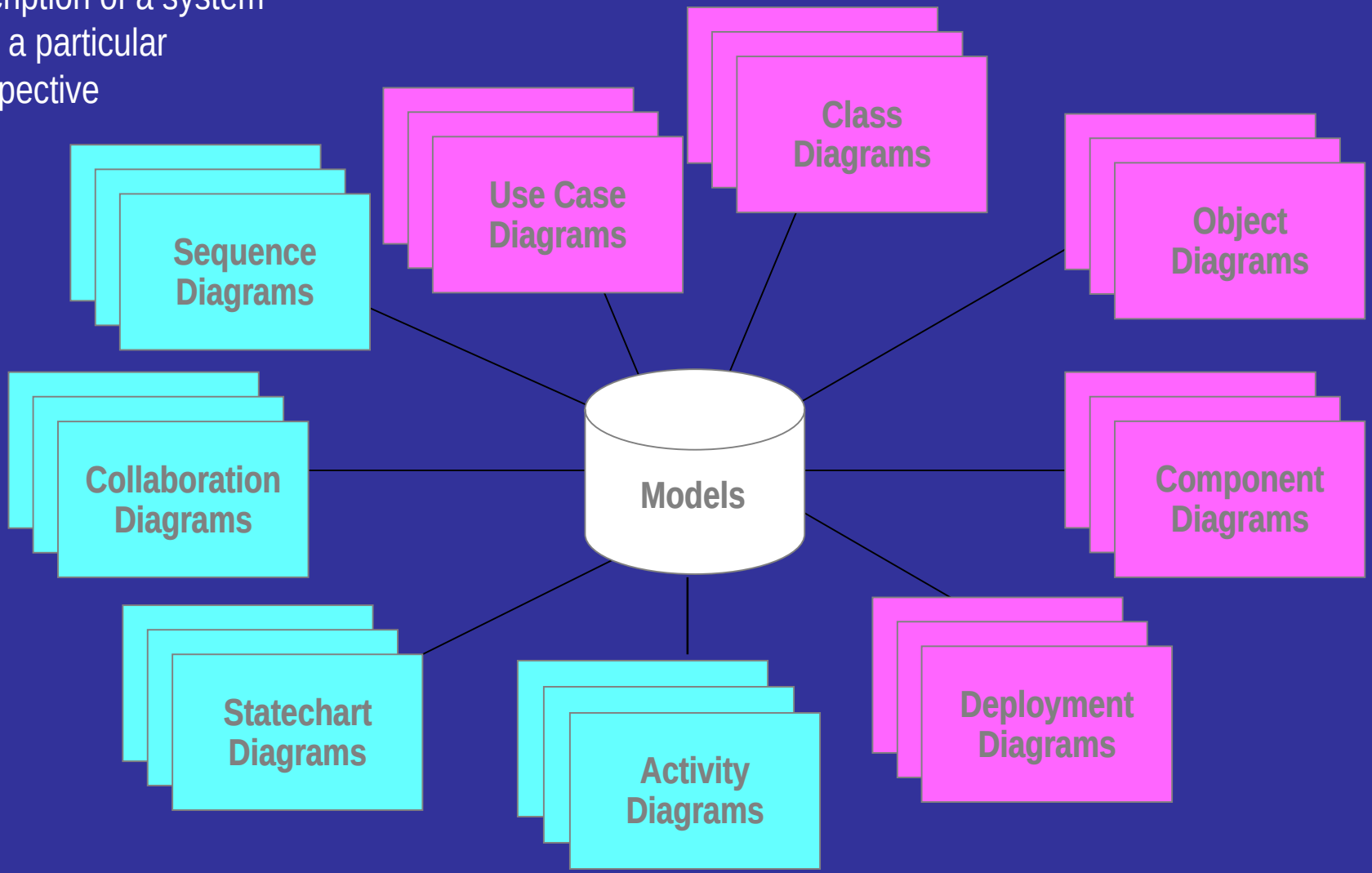
Architecture and the UML

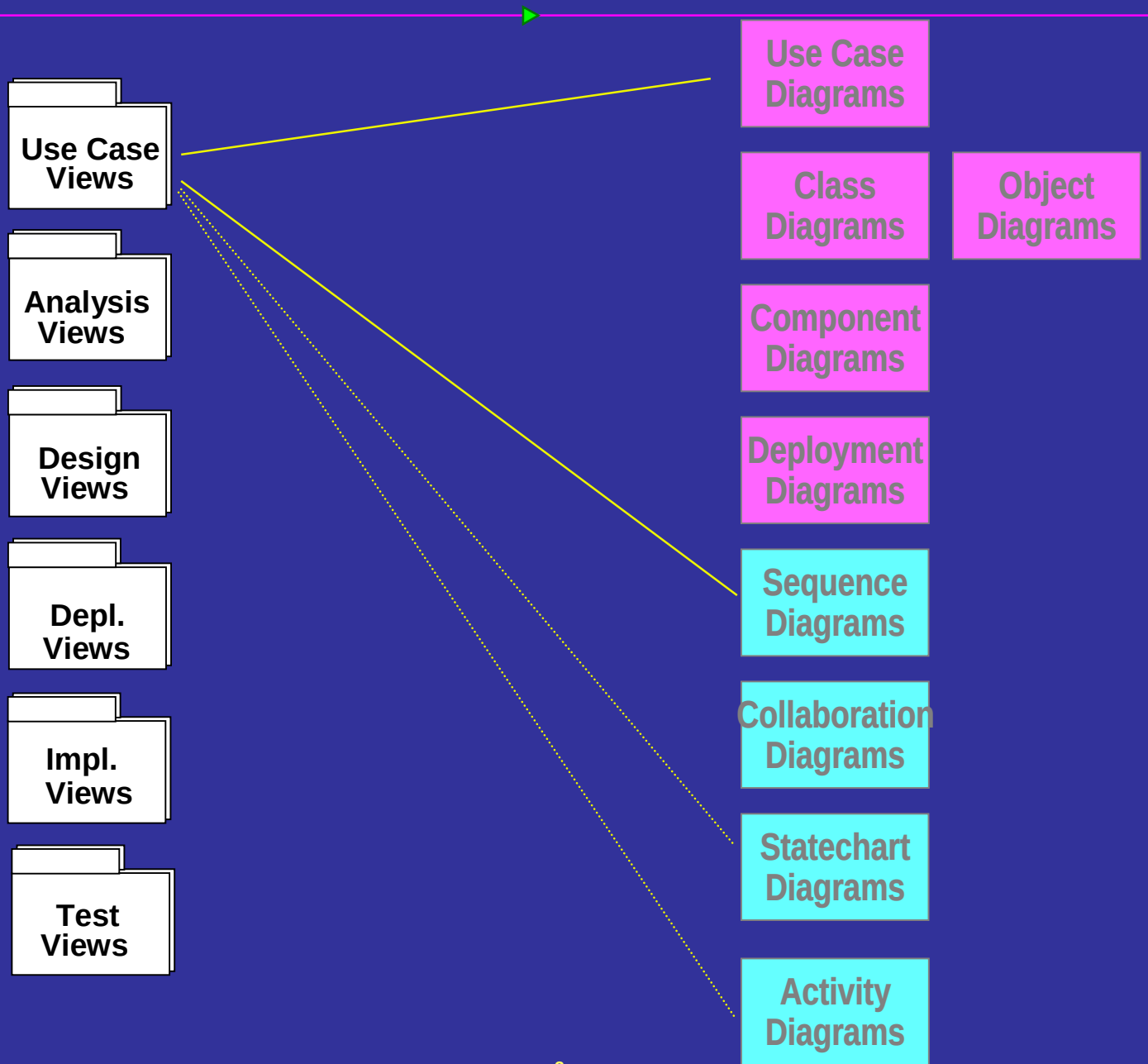


Models, Views, and Diagrams

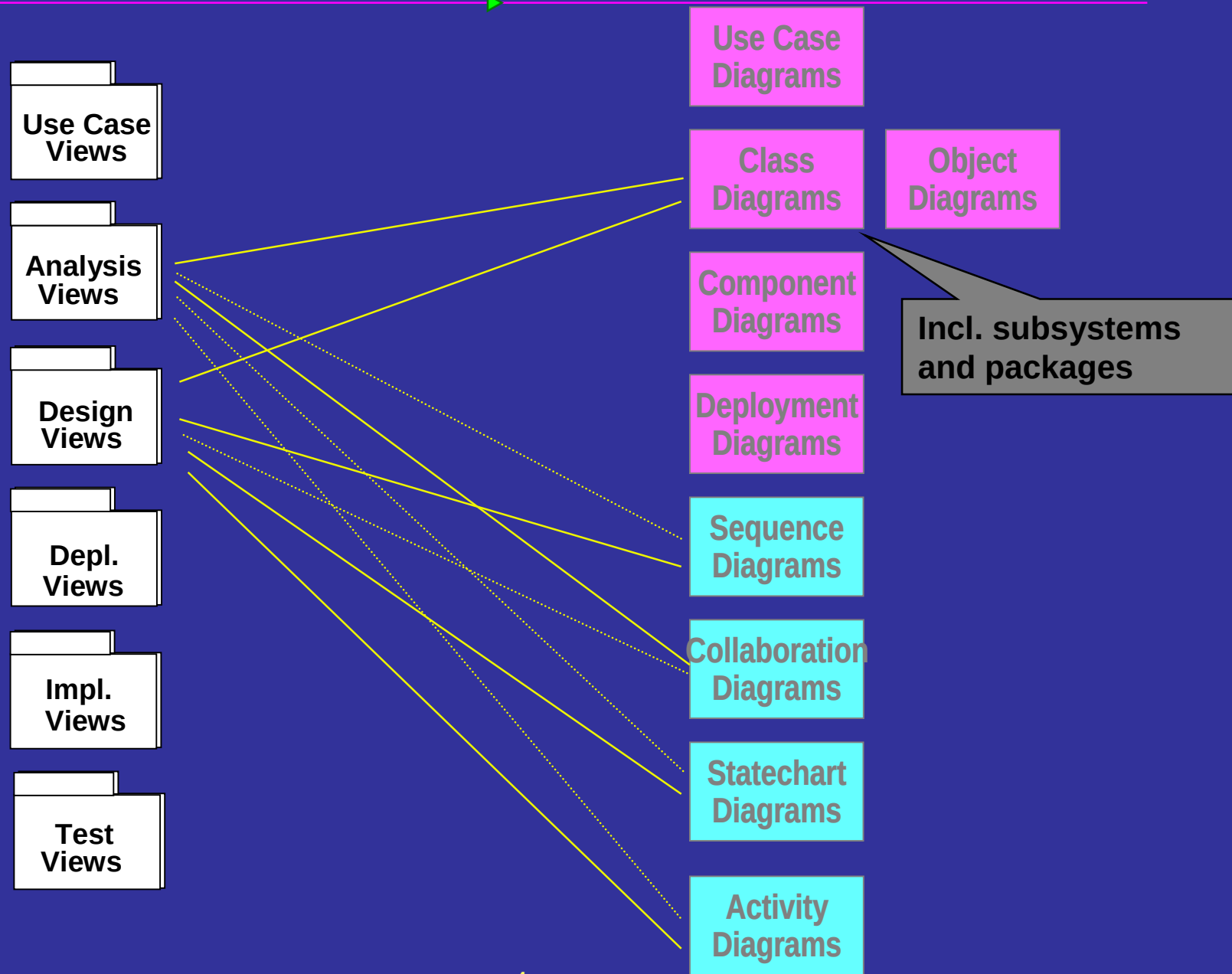
A ***model*** is a complete description of a system from a particular perspective



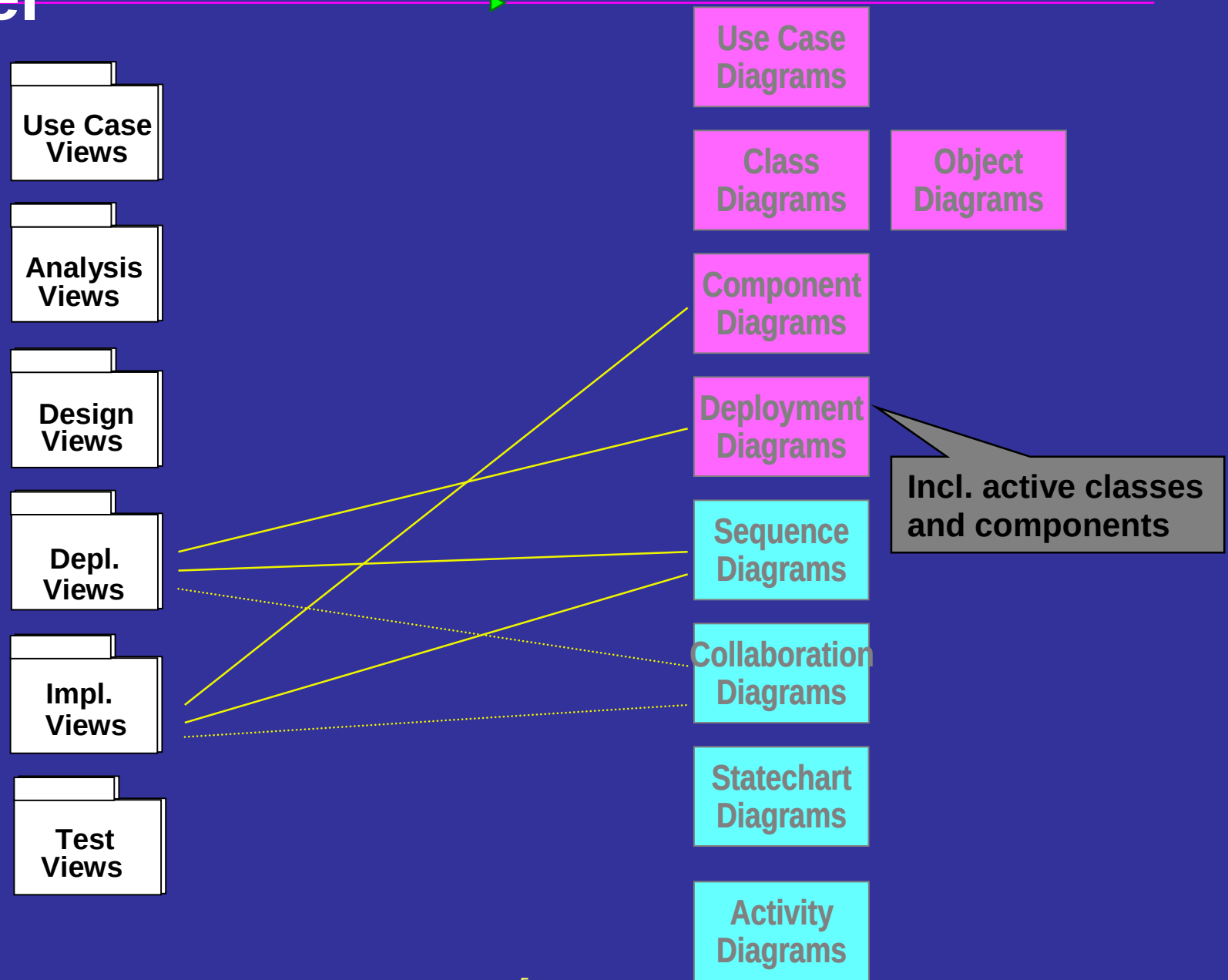
Use Case Model



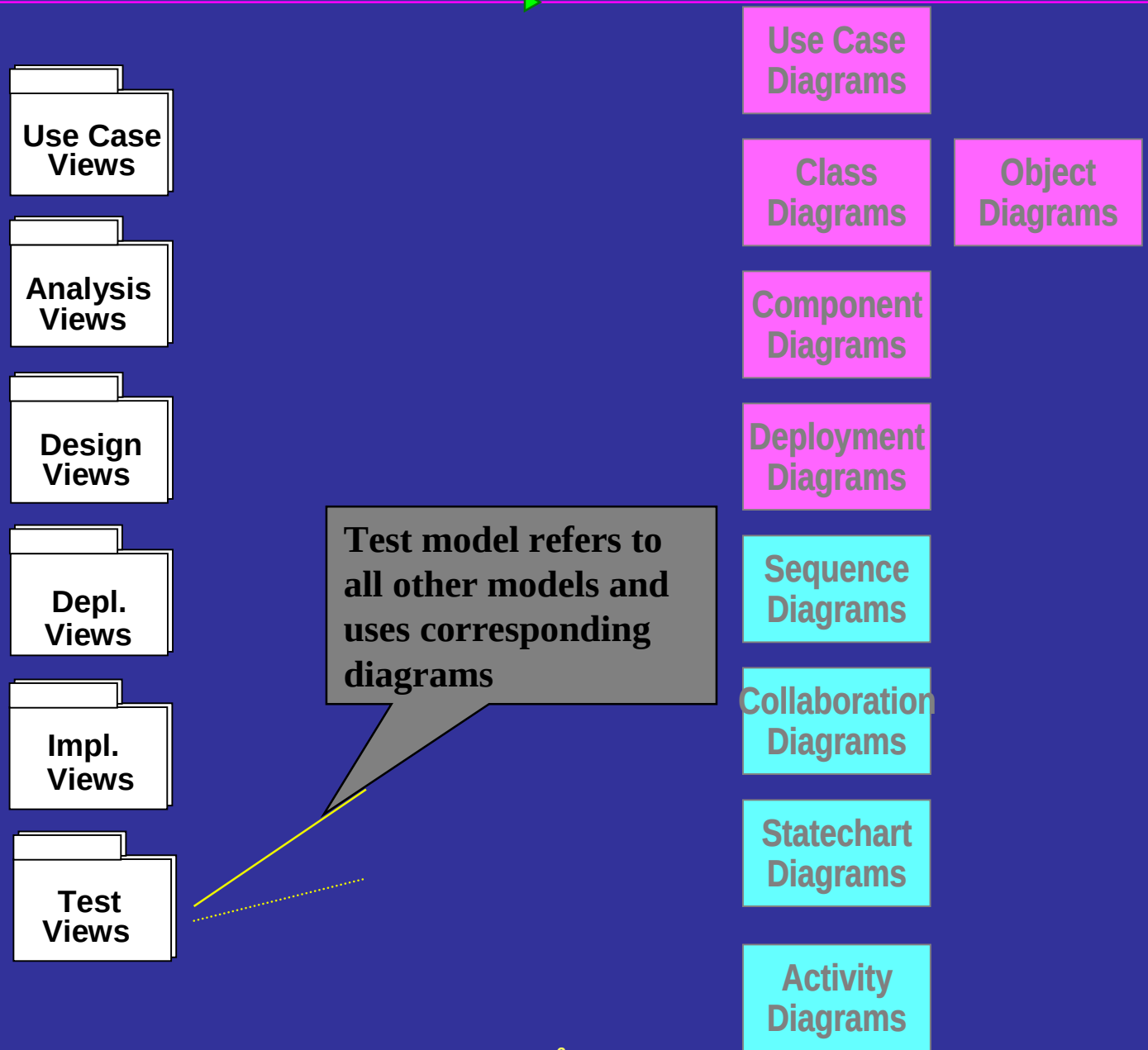
Analysis & Design Model



Deployment and Implementation Model



Test Model


Use Case Views
Analysis Views
Design Views
Depl. Views
Impl. Views
Test Views
Test model refers to all other models and uses corresponding diagrams
Use Case Diagrams
Class Diagrams
Object Diagrams
Component Diagrams
Deployment Diagrams
Sequence Diagrams
Collaboration Diagrams
Statechart Diagrams
Activity Diagrams

Use case - functions of a system from the user's point of view

Sequence diagrams -illustrates object interactions arranged in a time sequence.

Class diagrams -static structure in terms of classes and relationships

Activity diagrams -behavior of an operation as a set of actions

State chart diagrams -behavior of a class in terms of states

Collaboration diagrams -spatial representation of objects, links, and interactions

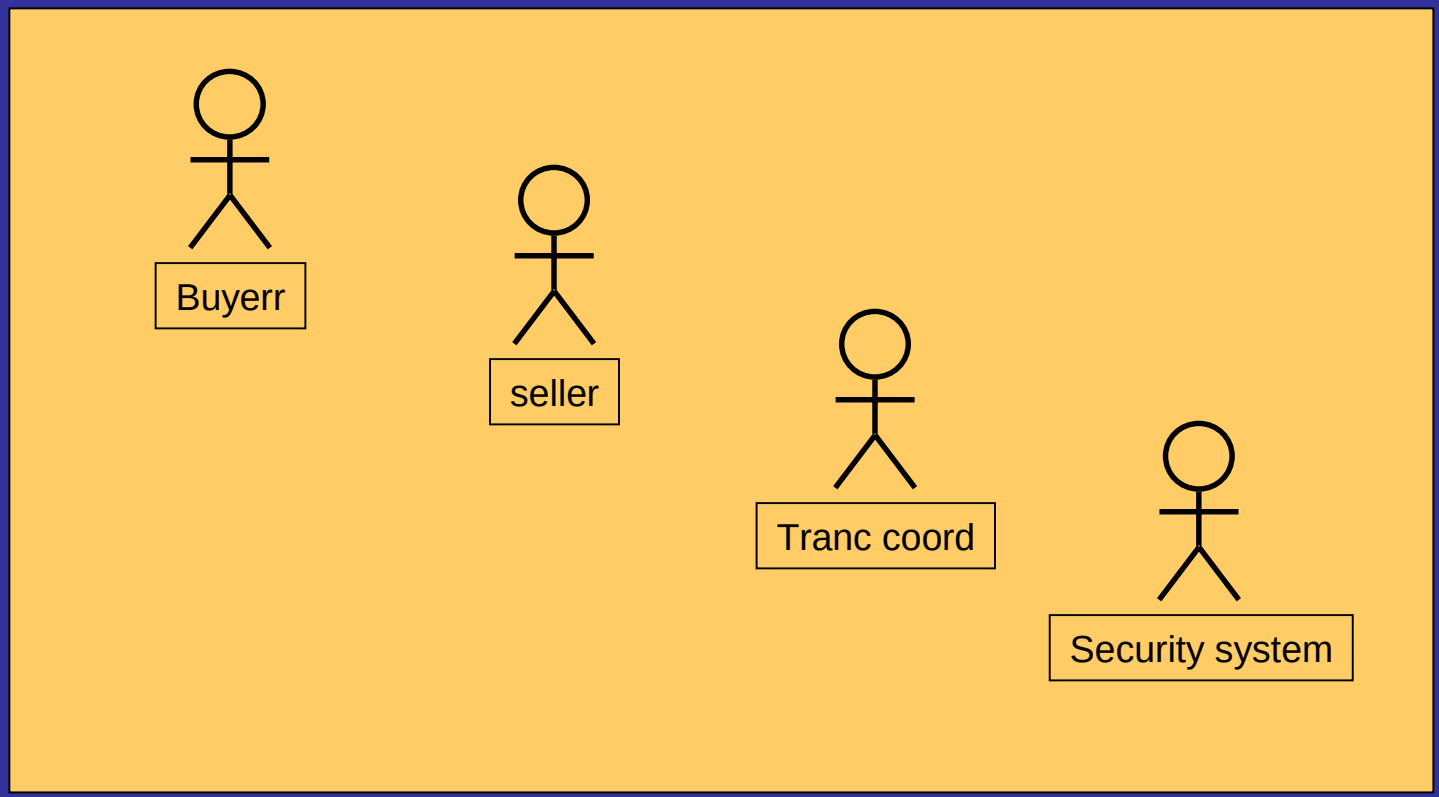
Object diagrams -objects and their relationships and correspond to (simplified collaboration diagrams that do not represent message broadcasts)

Component diagrams -physical components of an application

Deployment diagrams -deployment of components on particular pieces of hardware

Actors

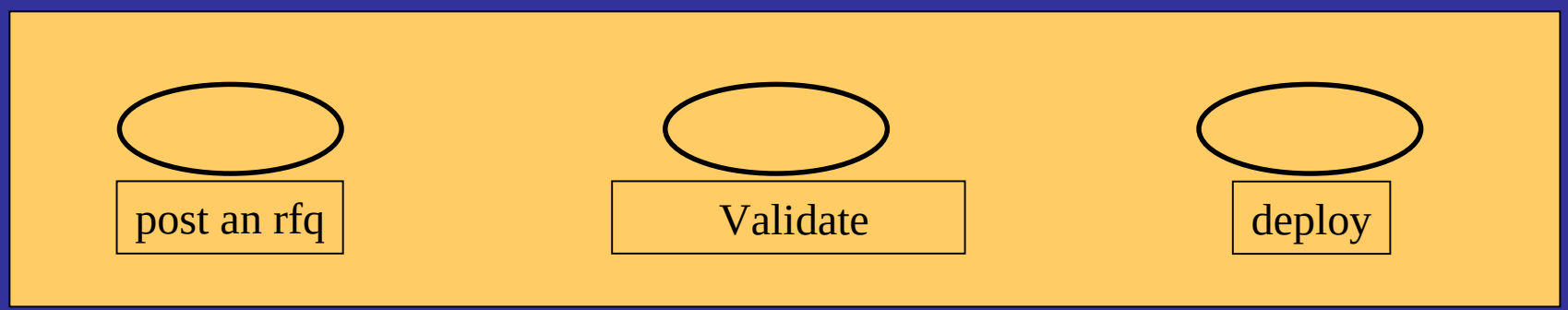
- An actor is someone or some thing that must interact with the system under development



Use Cases

- A use case is a pattern of behavior the system exhibits
  - Each use case is a sequence of related transactions performed by an actor and the system in a dialogue
- Actors are examined to determine their needs
  - Buyer – post an rfq
  - seller – respond to rfq
  - Data validator – validate
  - Dep manager -- deploy

Principal actors
Secondary actors
External hardware
Other systems

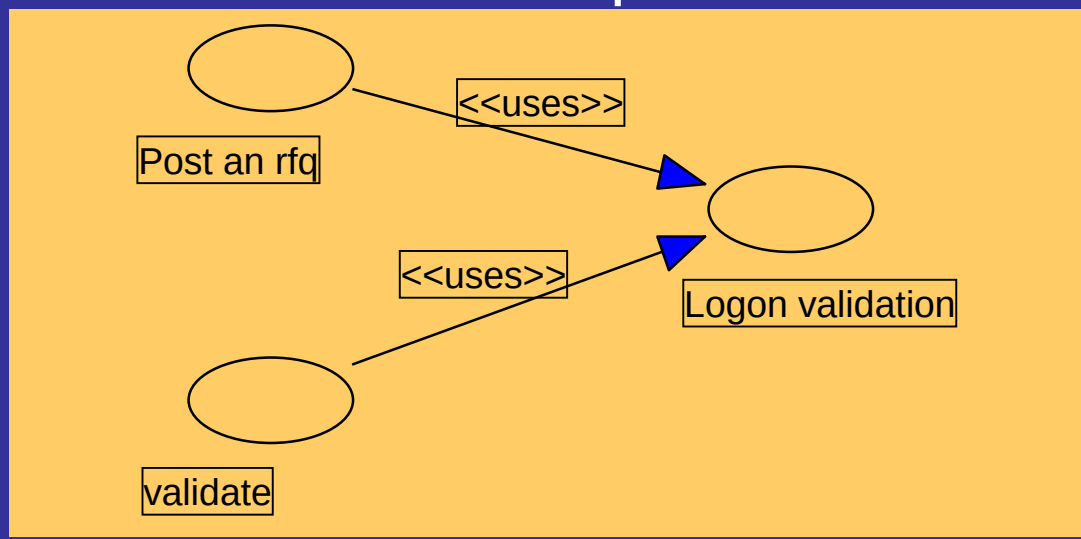


Use Cases

- A flow of events document is created for each use cases
  - Written from an actor point of view
- Details what the system must provide to the actor when the use cases is executed
- Typical contents
  - How the use case starts and ends
  - Normal flow of events
  - Alternate flow of events
  - Exceptional flow of events

Uses and Extends Use Case Relationships

- As the use cases are documented, other use case relationships may be discovered
  - A uses relationship shows behavior that is common to one or more use cases
  - An extends relationship shows optional behavior
  - Communicates shows specific functions

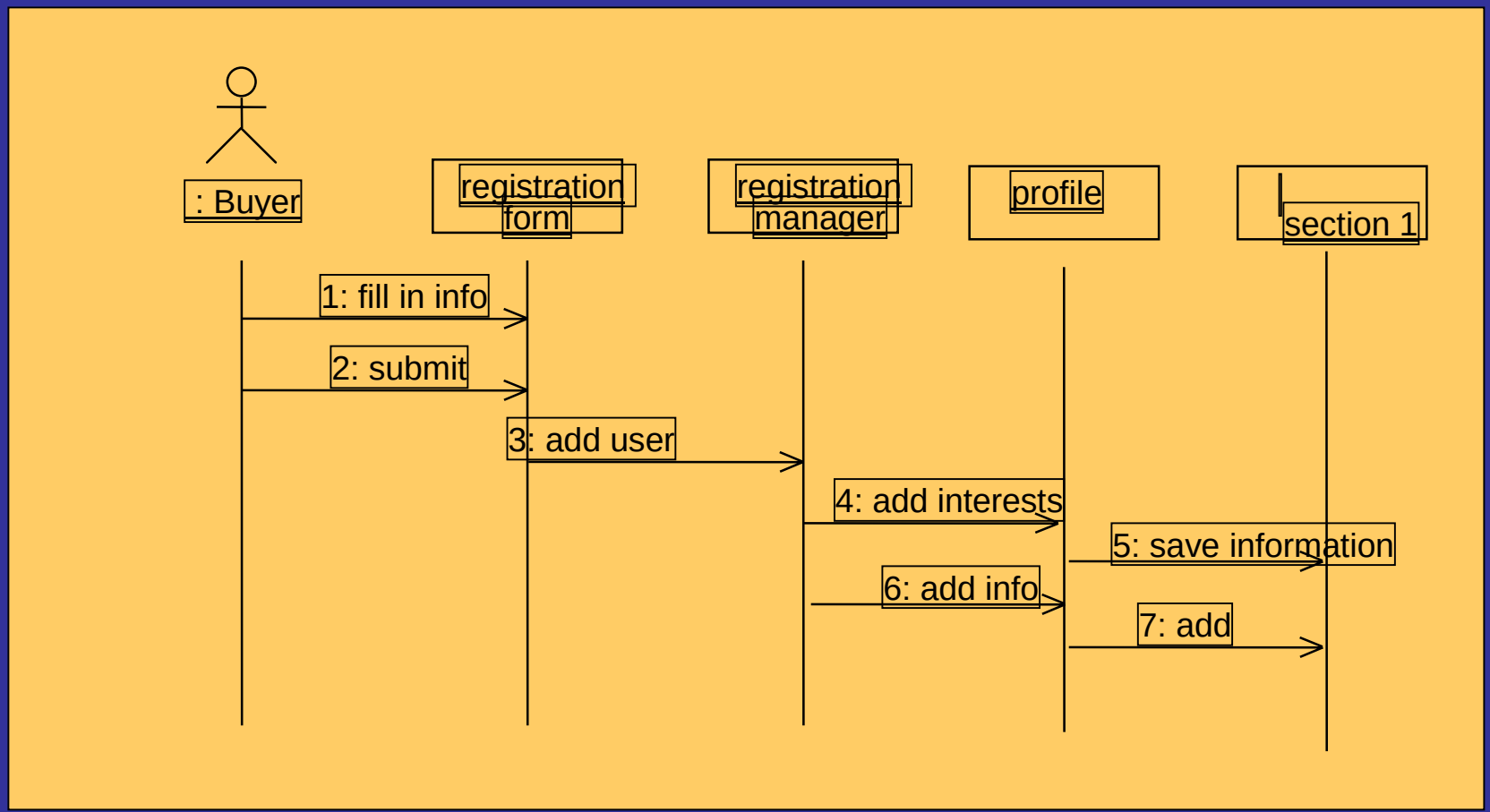


Use Case Realizations

- The use case diagram presents an outside view of the system
- Interaction diagrams describe how use cases are realized as interactions among societies of objects
- Two types of interaction diagrams
  - Sequence diagrams
  - Collaboration diagrams

Sequence Diagram

- A sequence diagram displays object interactions arranged in a time sequence



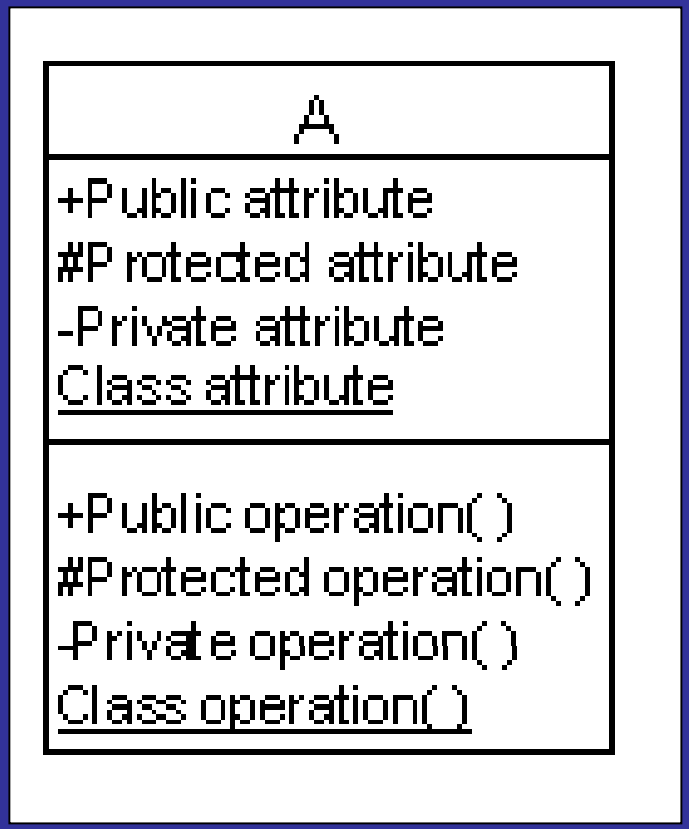
Class Diagrams

- A class diagram shows the existence of classes and their relationships in the logical view of a system
- UML modeling elements in class diagrams
  - Classes and their structure and behavior
  - Association, aggregation, dependency, and inheritance relationships
  - Multiplicity and navigation indicators
  - Role names

Classes

- A class is a collection of objects with common structure, common behavior, common relationships and common semantics
- Classes are found by examining the objects in sequence and collaboration diagram
- A class is drawn as a rectangle with three compartments
- Classes should be named using the vocabulary of the domain
  - Naming standards should be created
  - e.g., all classes are singular nouns starting with a capital letter

Classes

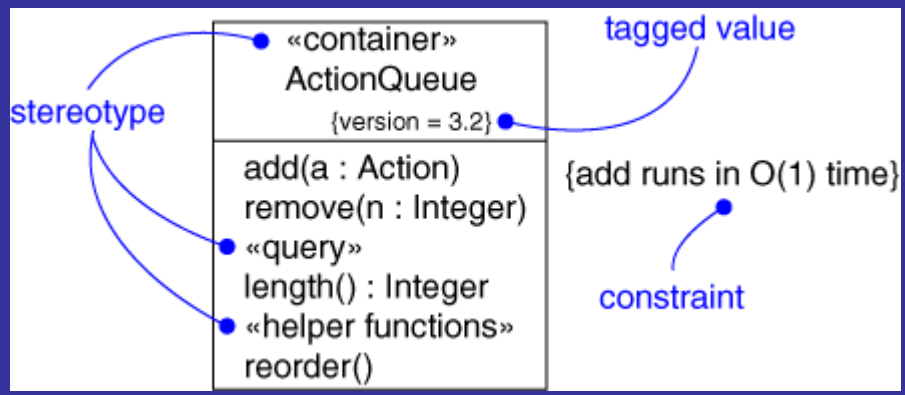


1. Attributes and Operations
2. Stereotype
3. Visibility of Attributes and Operations

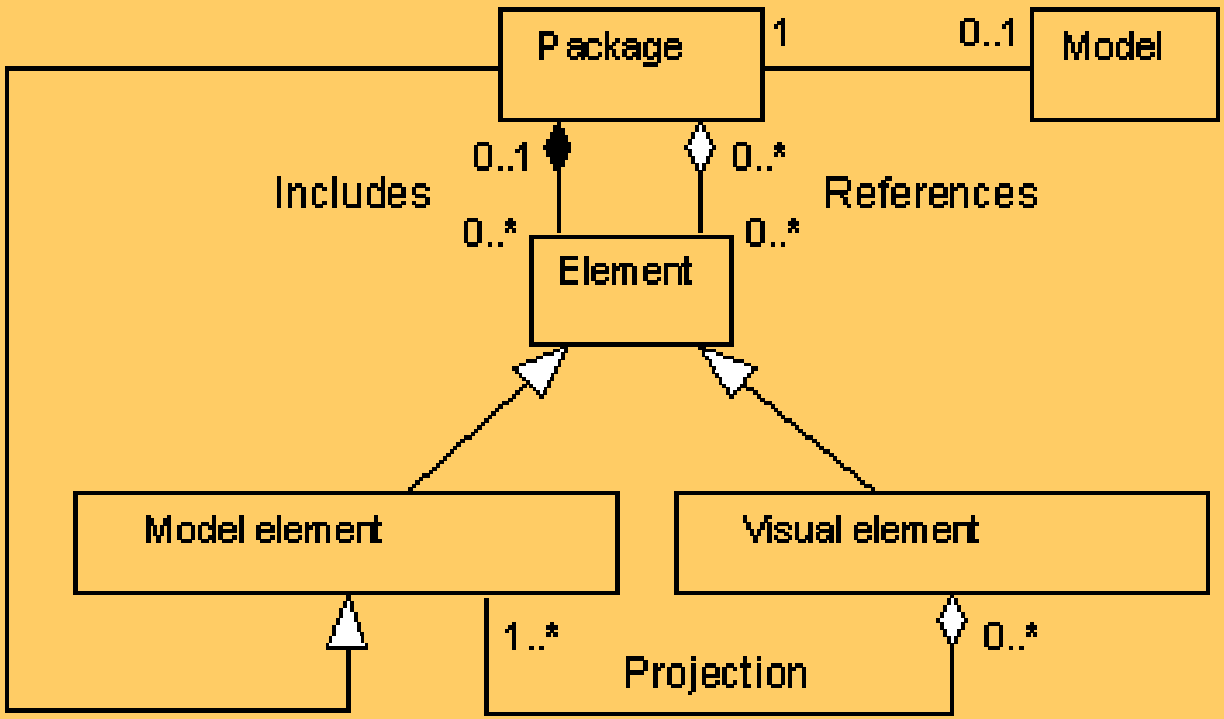
«signal» transaction within a state machine.
«interface» description of visible operations.
«metaclass» The class of a class
«utility» A class reduced to the concept of module

Extensibility Mechanisms

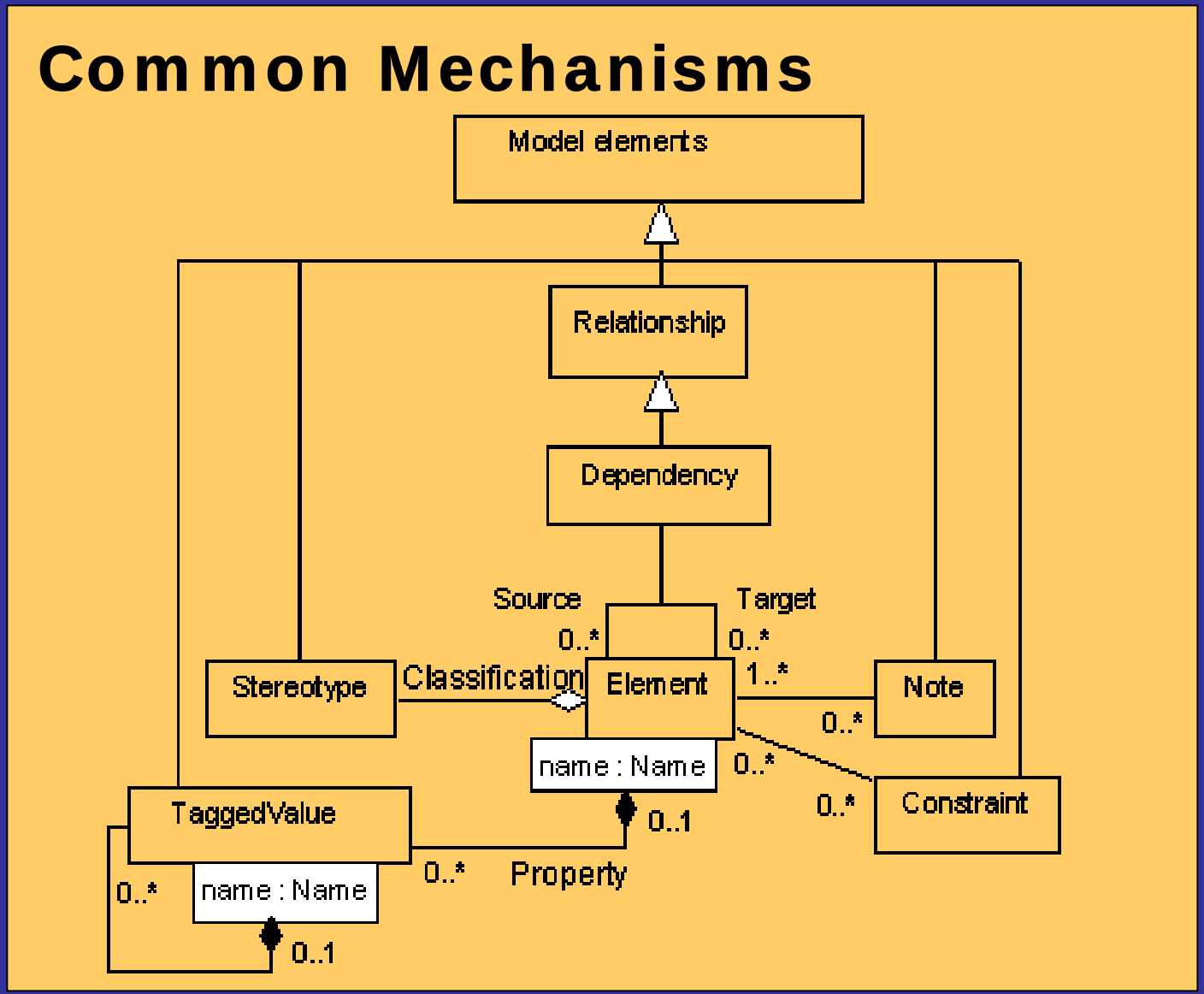
- Stereotype
- Tagged value
- Constraint



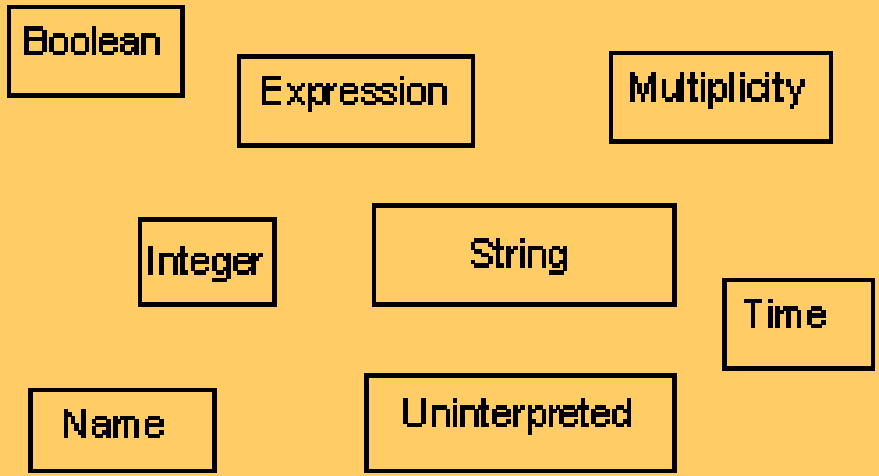
Common Elements



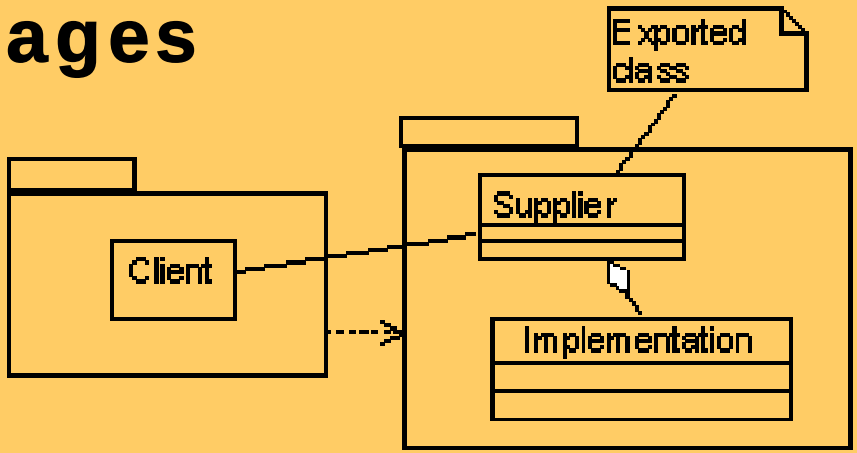
Common Mechanisms



Data Types



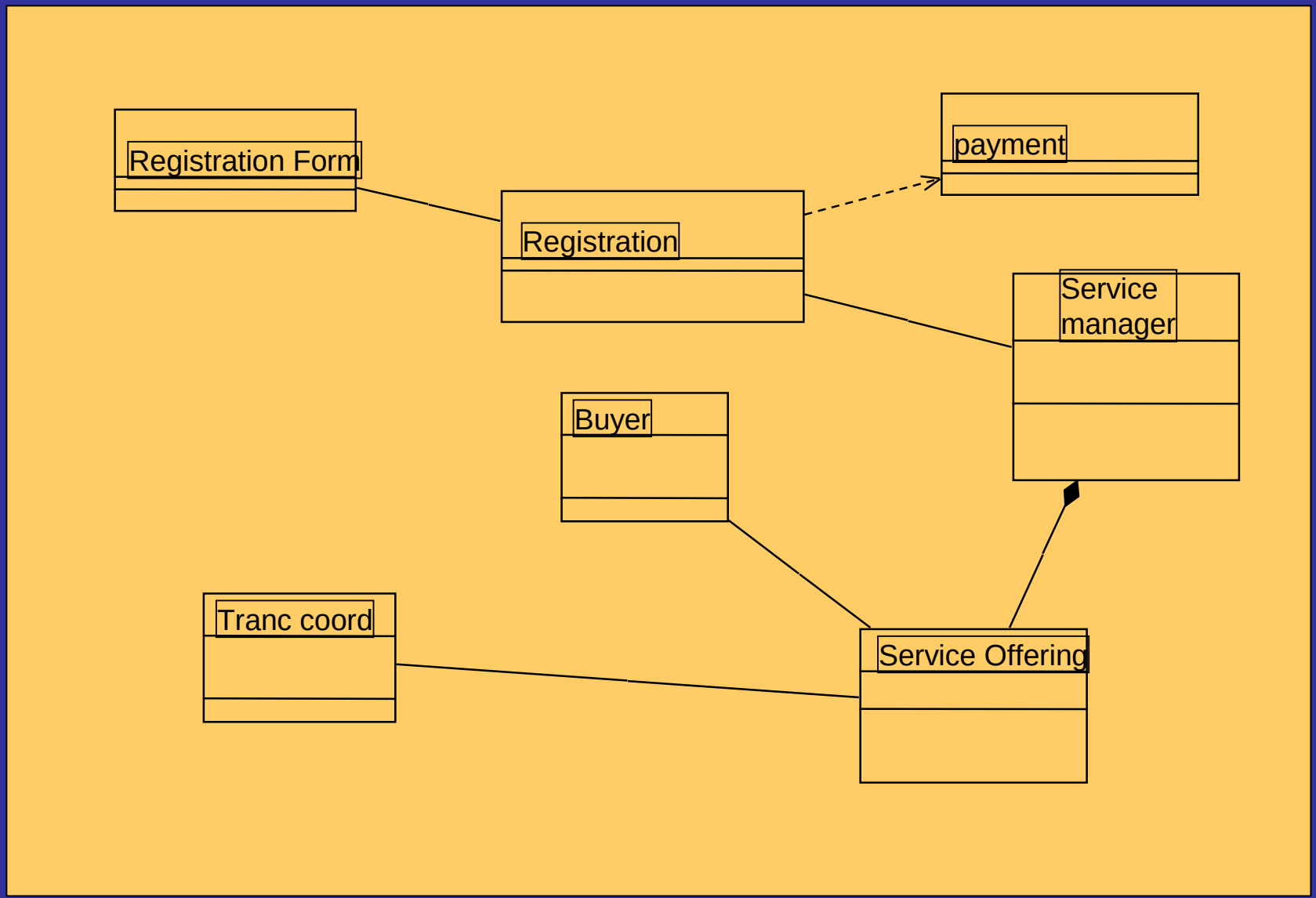
Packages



Extending the UML

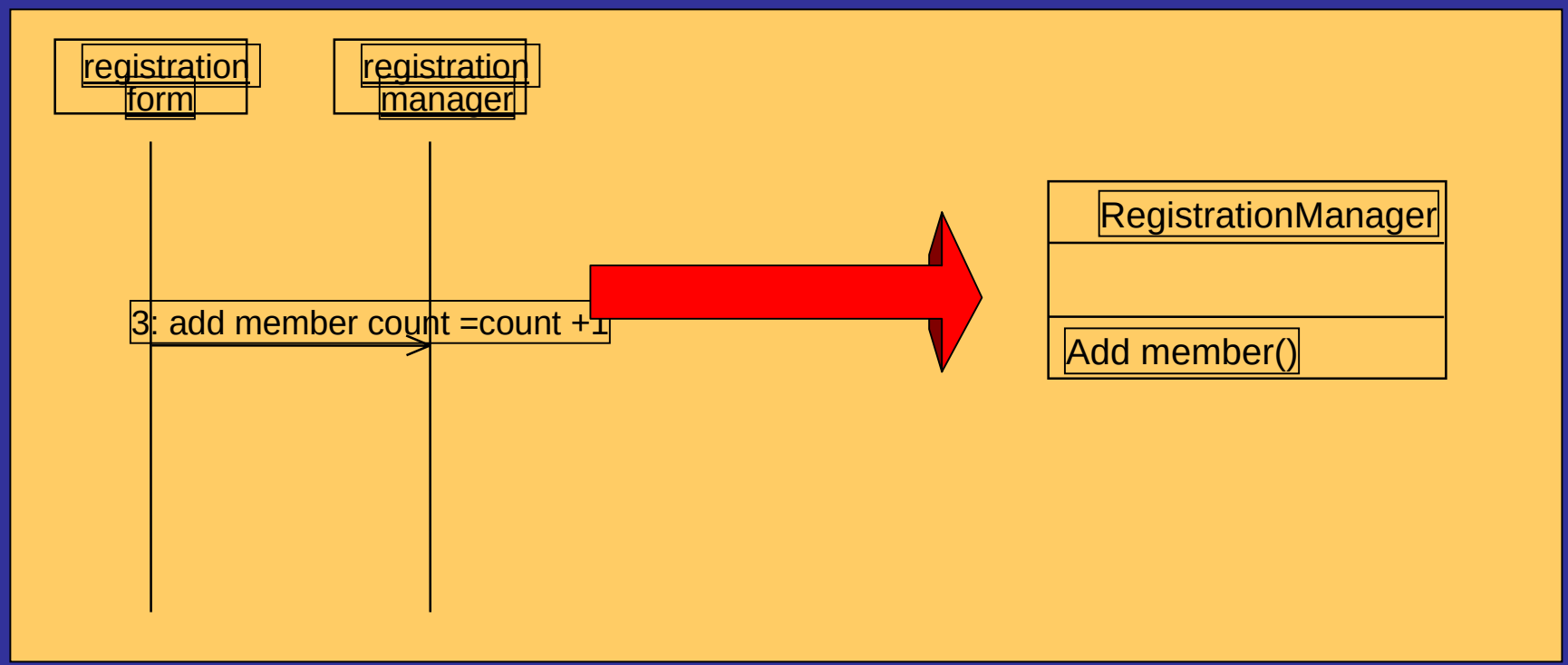
- Stereotypes can be used to extend the UML notational elements
- Stereotypes may be used to classify and extend associations, inheritance relationships, classes, and components
- Examples:
  - Class stereotypes: boundary, control, entity, utility, exception
  - Inheritance stereotypes: uses and extends
  - Component stereotypes: subsystem

Classes


Registration Form
payment
Registration
Service manager
Buyer
Tranc coord
Service Offering

Operations

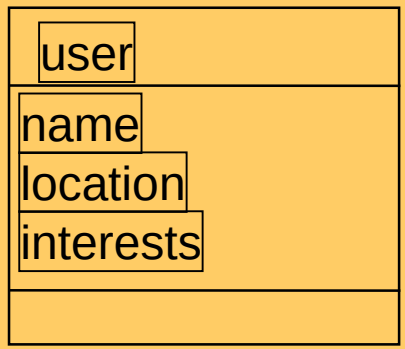
- The behavior of a class is represented by its operations
- Operations may be found by examining interaction diagrams



Attributes

- The structure of a class is represented by its attributes
- Attributes may be found by examining class definitions, the problem requirements, and by applying domain knowledge

Each User has a name , location, interests, and authentification



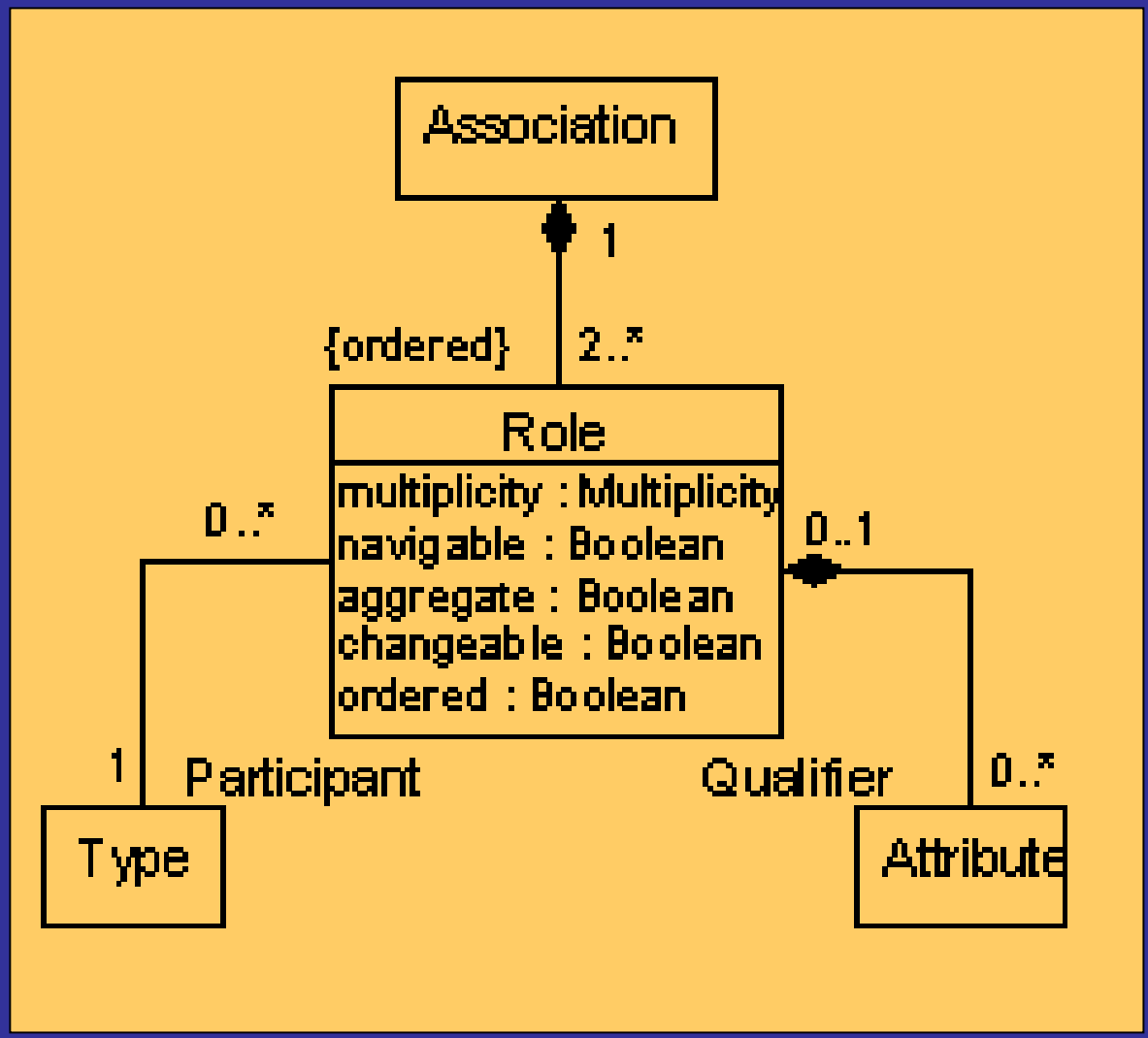
Relationships

- Relationships provide a pathway for communication between objects
- Sequence and/or collaboration diagrams are examined to determine what links between objects need to exist to accomplish the behavior -- if two objects need to “talk” there must be a link between them
- Three types of relationships are:
  - Association
  - Aggregation
  - Dependency

Relationships

- An association is a bi-directional connection between classes
- An aggregation is a stronger form of relationship where the relationship is between a whole and its parts
- A dependency relationship is a weaker form of relationship showing a relationship between a client and a supplier where the client does not have semantic knowledge of the supplier

Relationships

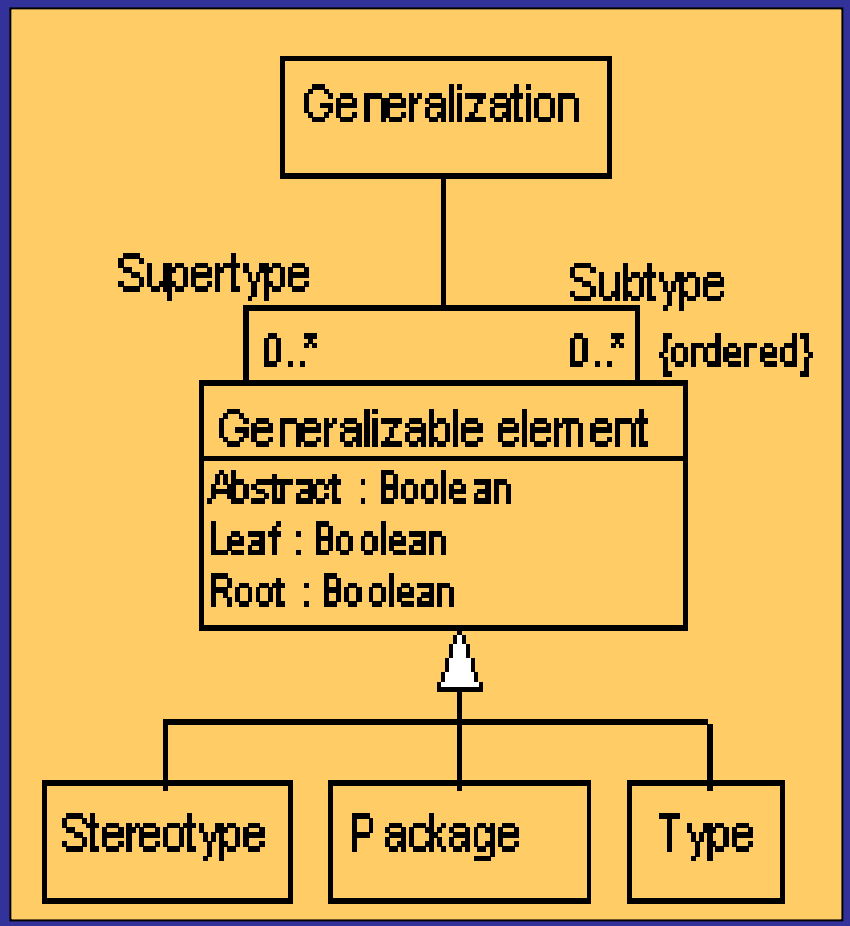


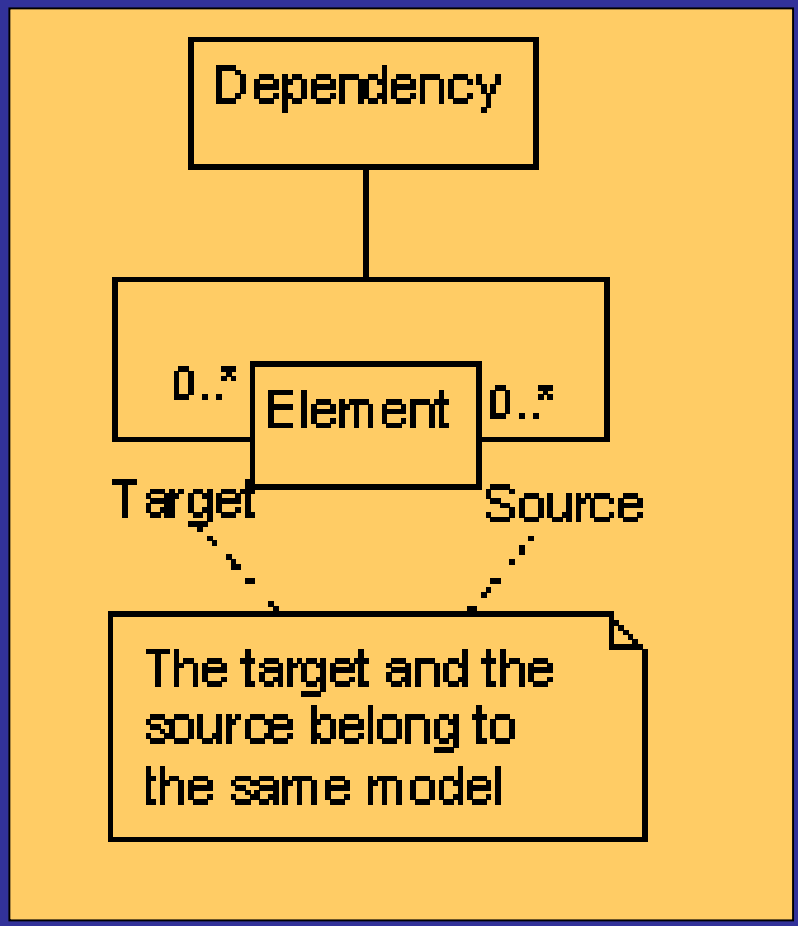
Association

Role(instances**)**

- **Multiplicity**
- **Navigability**
- **Aggregation**
- **Changeability**
- **Ordering**

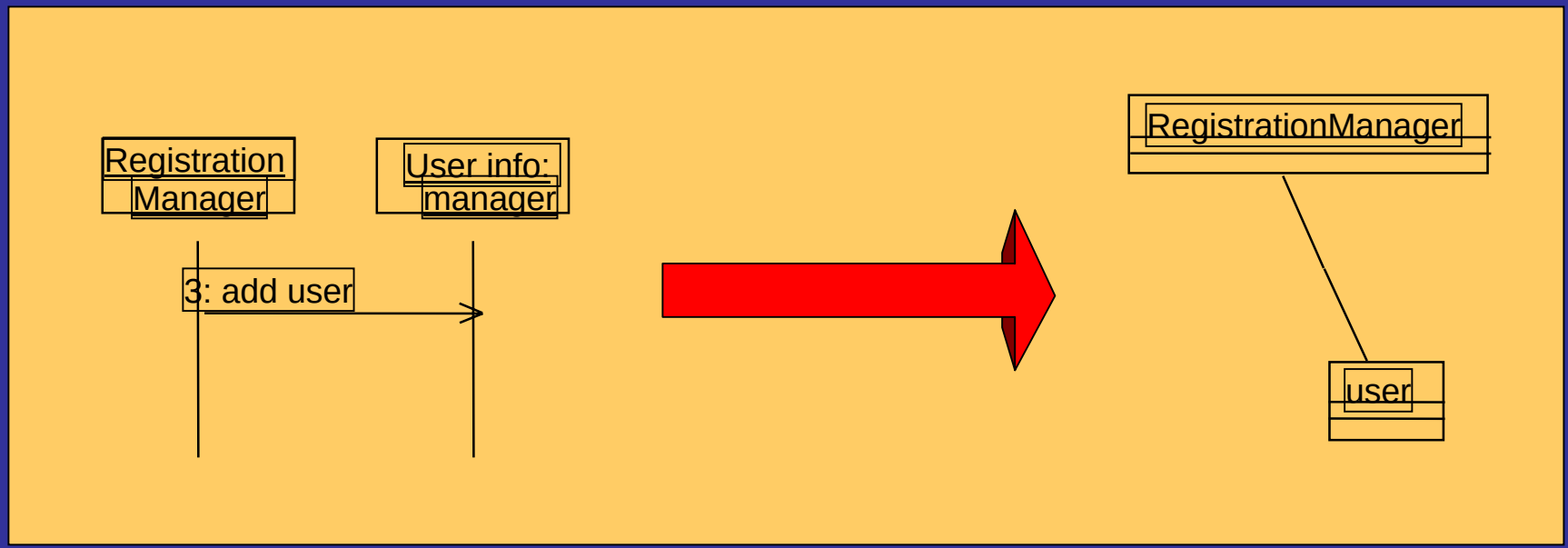
Relationships

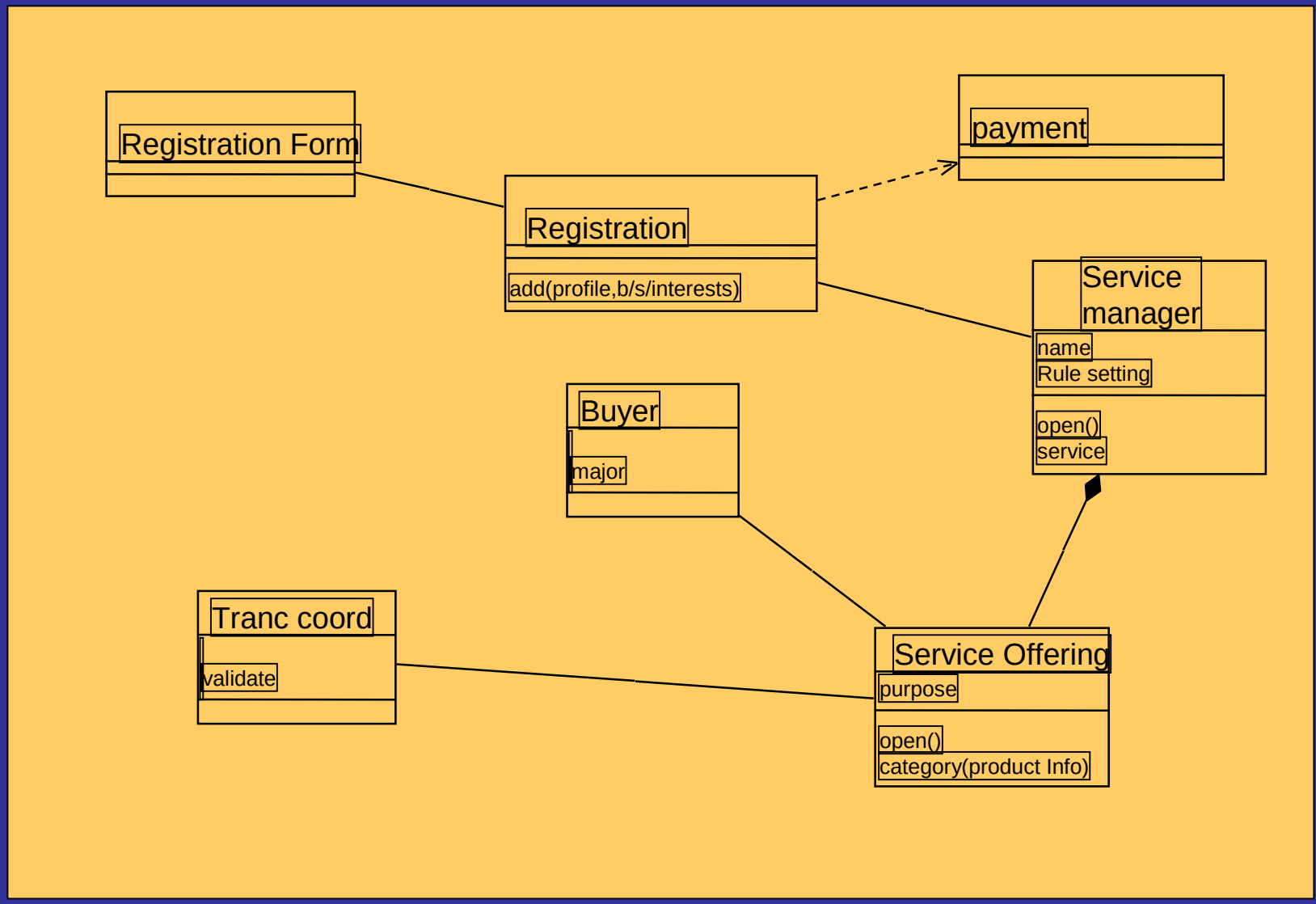




Finding Relationships

- Relationships are discovered by examining interaction diagrams
  - If two objects must “talk” there must be a pathway for communication

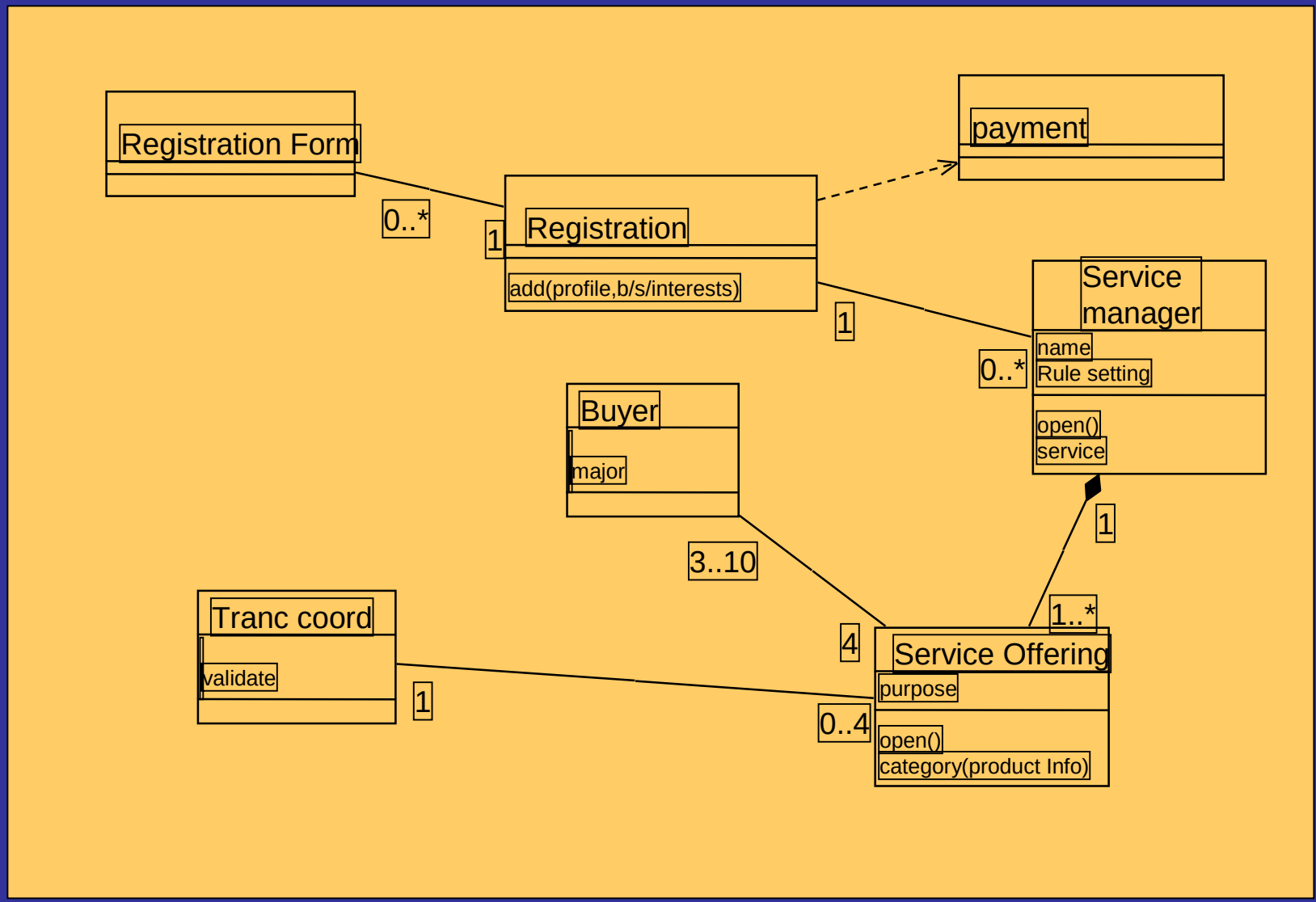



Registration Form
payment
Registration
add(profile,b/s/interests)
Service manager
name
Rule setting
open()
service
Buyer
major
Tranc coord
validate
Service Offering
purpose
open()
category(product Info)

Multiplicity and Navigation

- Multiplicity defines how many objects participate in a relationships
  - Multiplicity is the number of instances of one class related to ONE instance of the other class
  - For each association and aggregation, there are two multiplicity decisions to make: one for each end of the relationship
- Although associations and aggregations are bi-directional by default, it is often desirable to restrict navigation to one direction
- If navigation is restricted, an arrowhead is added to indicate the direction of the navigation

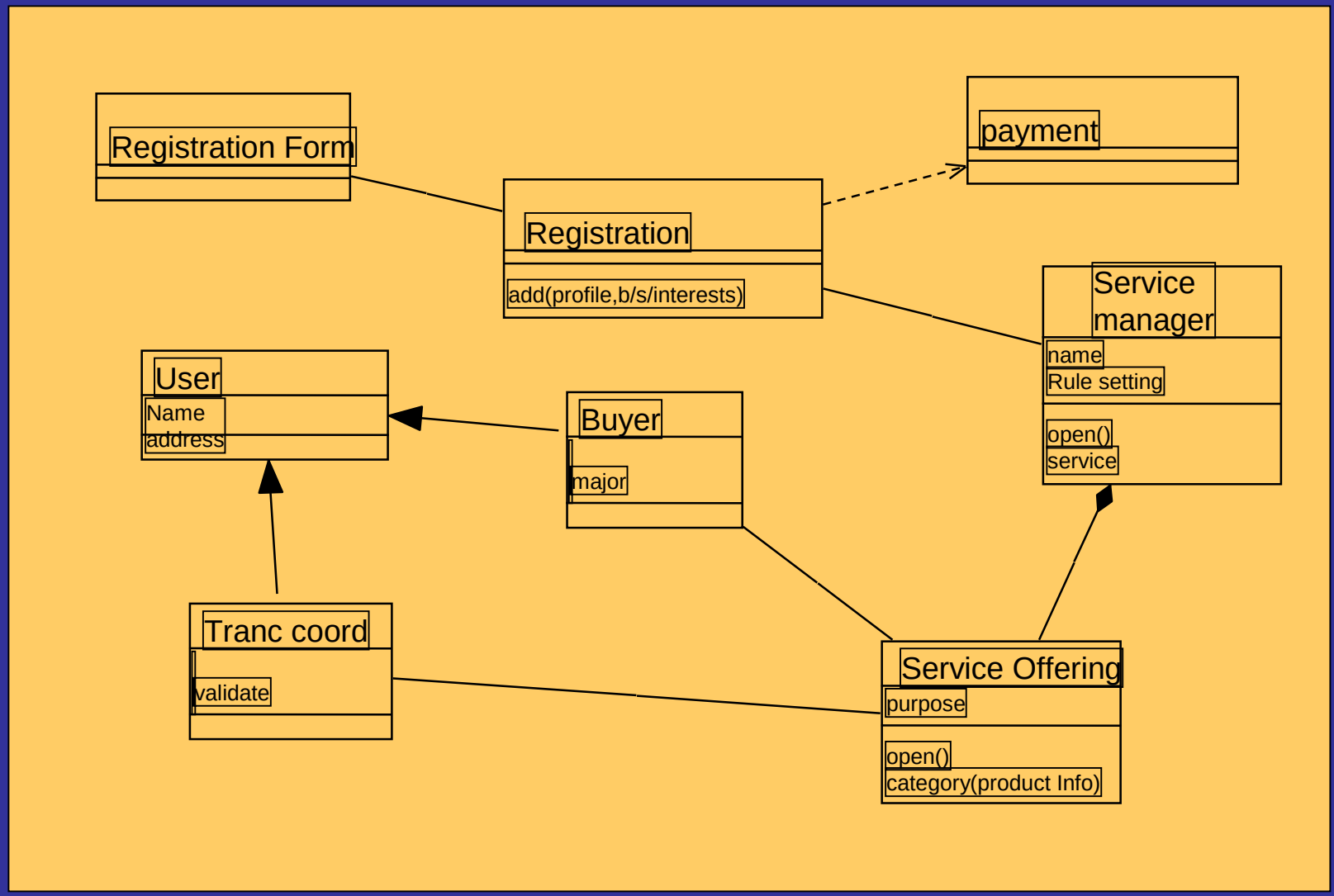
Multiplicity and Navigation



Inheritance

- Inheritance is a relationships between a superclass and its subclasses
- There are two ways to find inheritance:
  - Generalization
  - Specialization
- Common attributes, operations, and/or relationships are shown at the highest applicable level in the hierarchy

Inheritance



The State of an class

- A state chart diagram shows
  - The life history of a given class
  - The events that cause a transition from one state to another
  - The actions that result from a state change
- State transition diagrams are created for objects with significant dynamic behavior

State Chart Diagram

Action associated to the state entry transition (**Op1**)

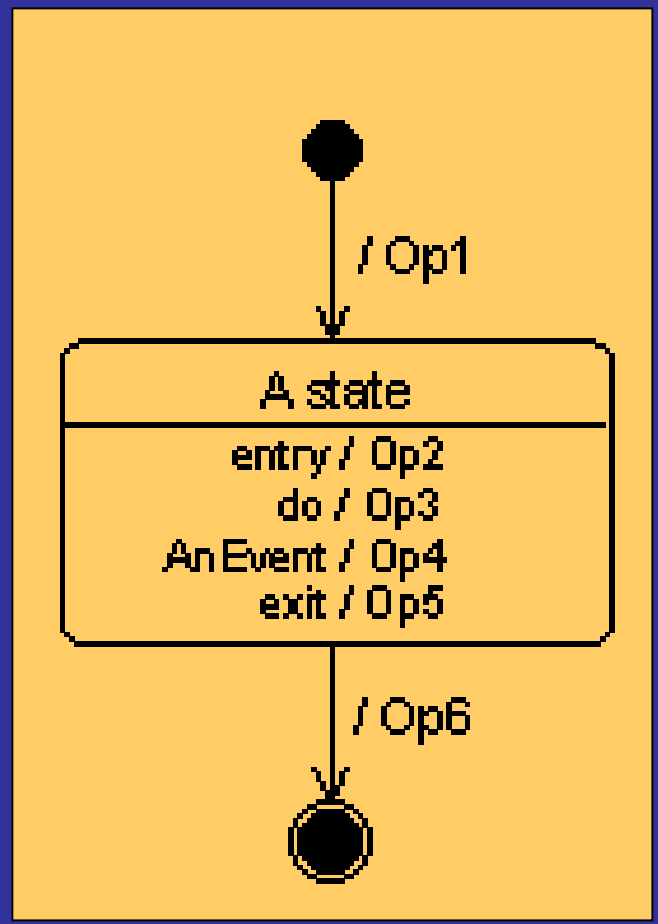
State entry action (**Op2**)

Activity within the state (**Op3**)

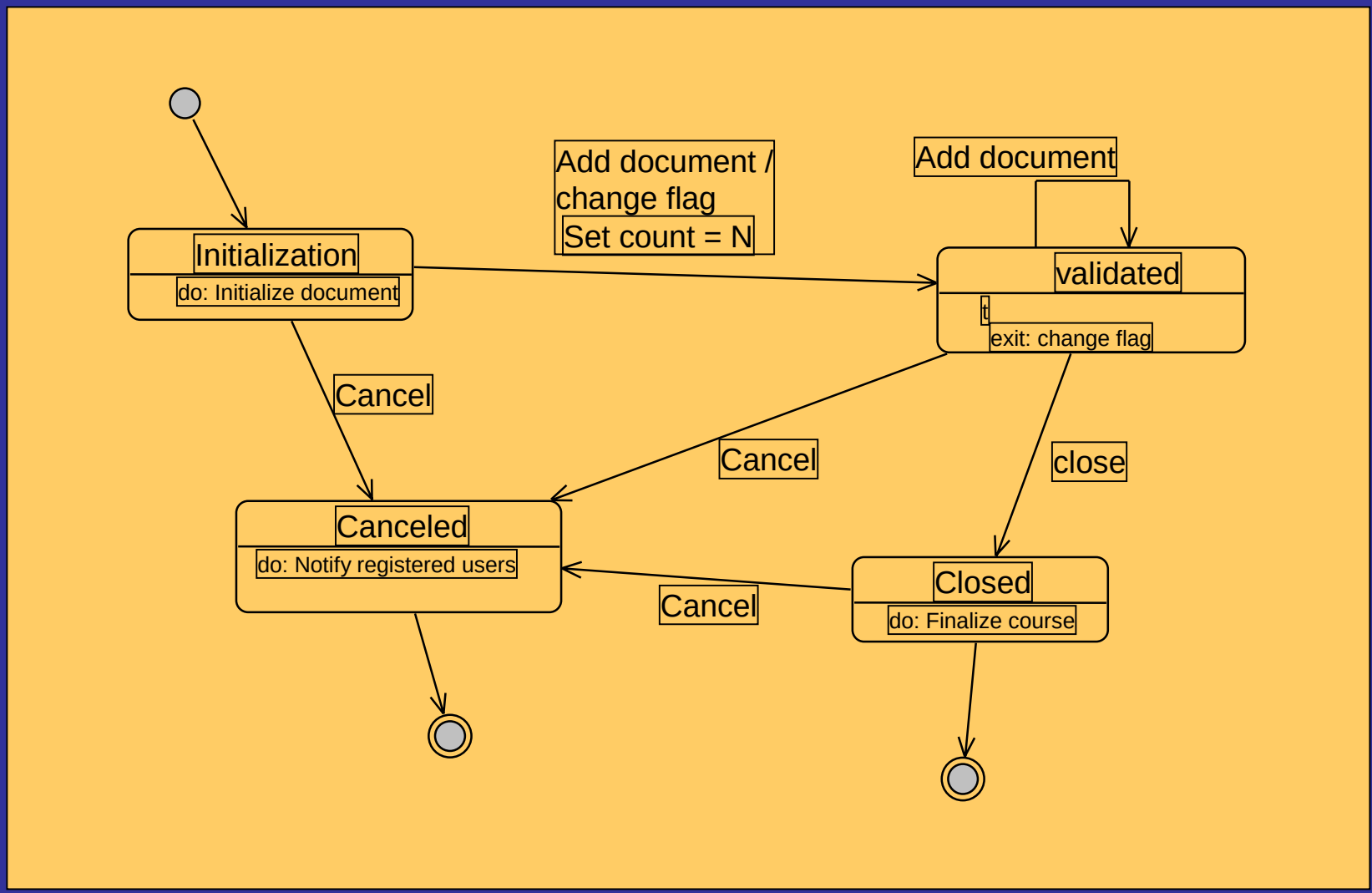
Action associated to internal events (**Op4**)

State exit action (**Op5**)

Action associated to the state exit transition



State Transition Diagram



Collaboration Diagram

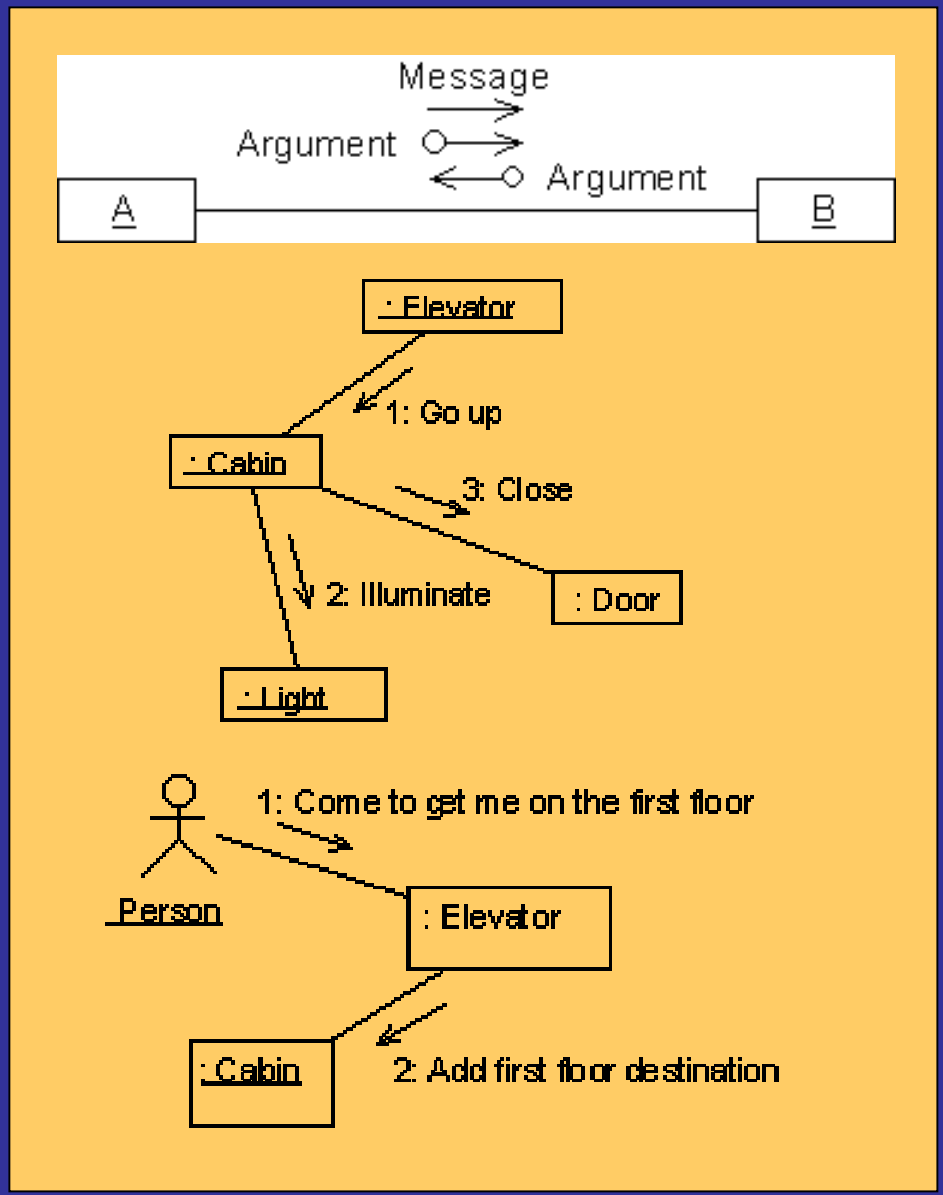
Collaboration diagrams

illustrate interactions between objects, using a static spatial structure that facilitates the illustration of the collaboration of a group of objects

Extension of object diagrams.

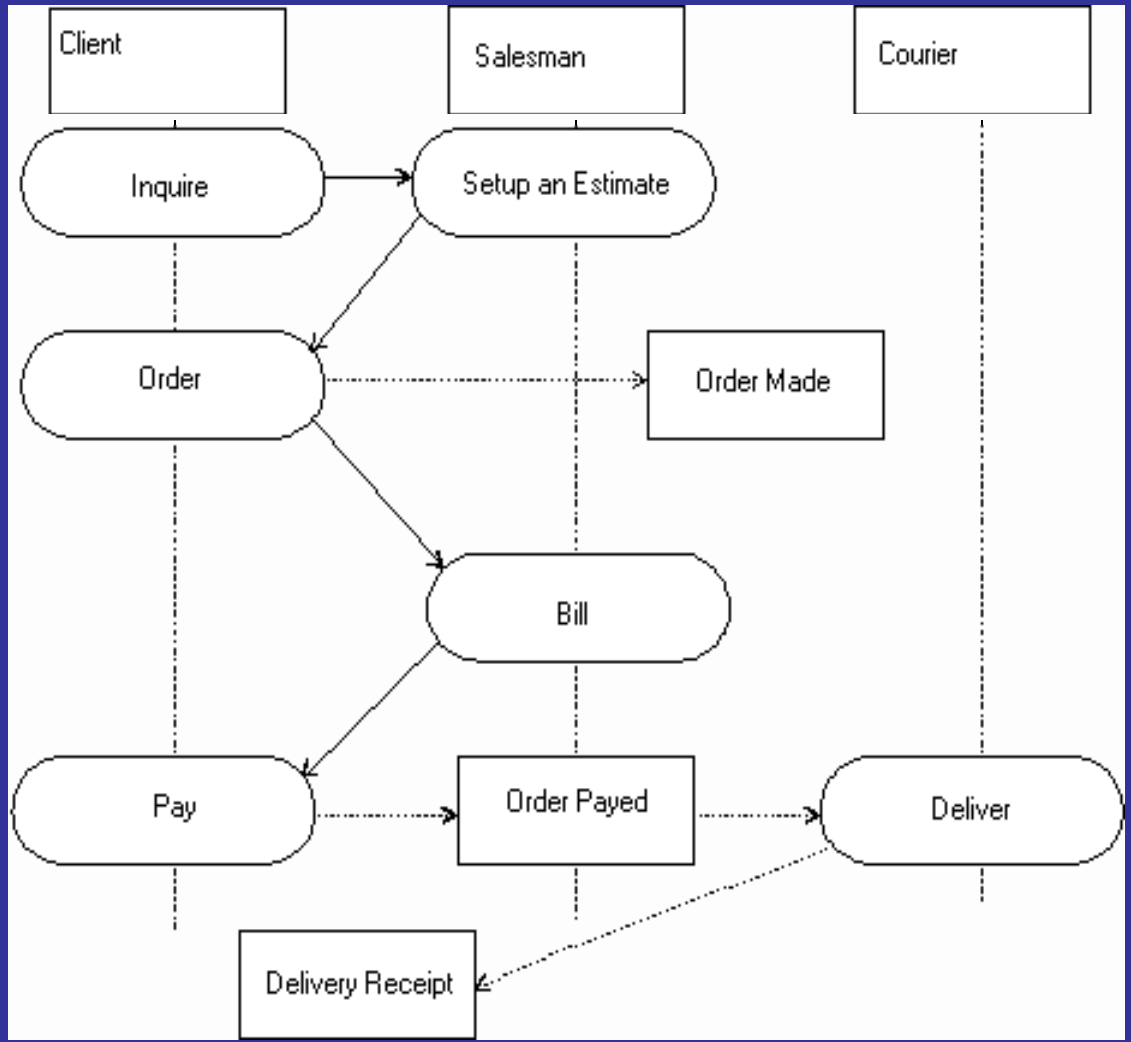
context of a group of objects

interaction between these objects



Activity Diagram

An activity diagram is a variant of statechart diagrams organized according to actions, and mainly targeted towards representing the internal behavior of a method



Component Diagram

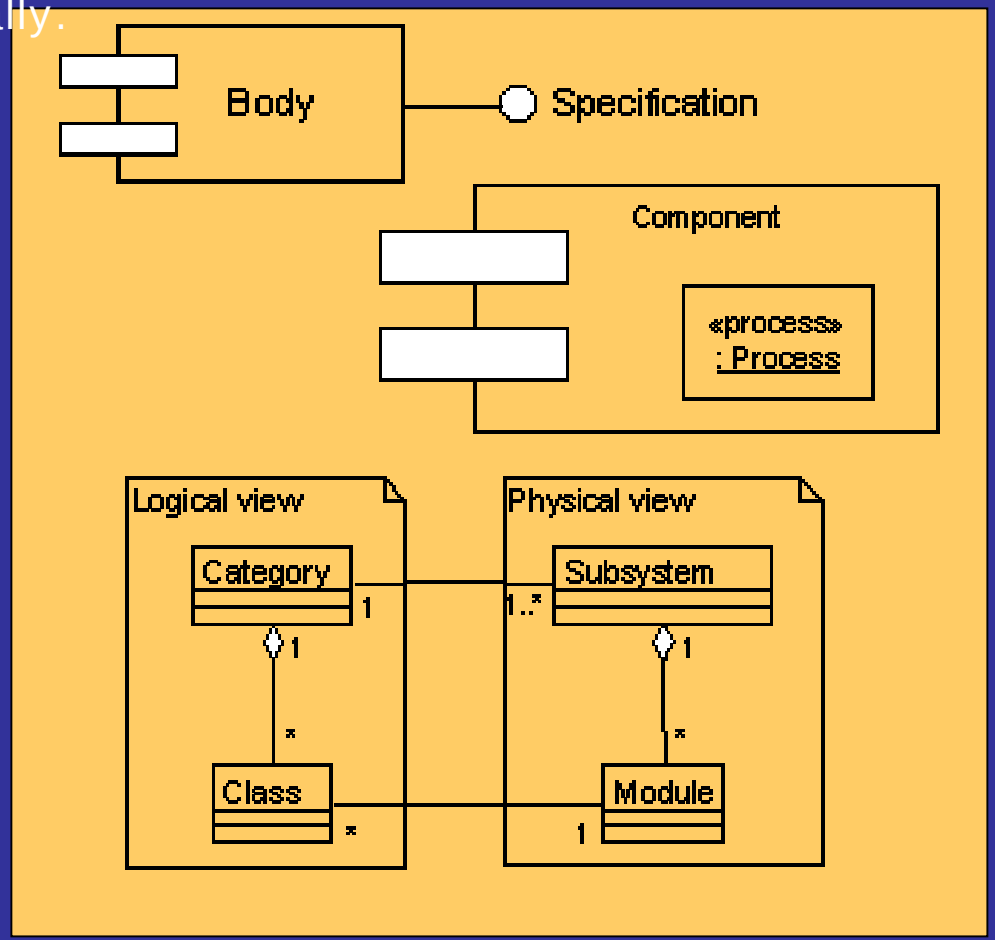
Component diagrams describe software components and their relationships within the implementation environment

Components represent all kinds of elements that pertain to the piecing together of software applications. Among other things, they may be simple files, or libraries loaded dynamically.

Processes

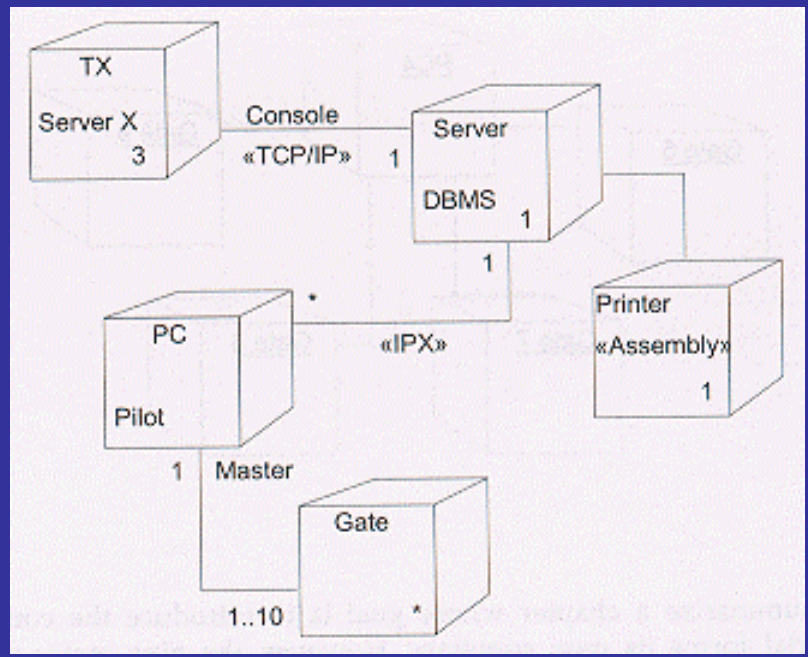
Processes are objects that have their own control flow (or *thread*), and as such are special kinds of active objects. Processes may be contained within components

Subsystems



Deployment Diagram

Deployment diagrams show the physical layout of the various hardware components (nodes) that compose a system, as well as the distribution of executable programs on this hardware.



Modeling Elements

- Structural elements
  - class, interface, collaboration, use case, active class, component, node
- Behavioral elements
  - interaction, state machine
- Grouping elements
  - package, subsystem
- Other elements
  - note