

# UML

- *UML* - Unified Modeling Language
- Championed by [Rational Corp.](#) (Now IBM)
- Standardized by [Object Management Group](#)
- Supported by CASE tools like [Rational Rose](#) and [Enterprise Architect](#)
- Conceived of by Rumbaugh, Booch, Jacobson
- Diagrams + Object Constraint Language (OCL)
- Can be used for analysis or design models
- Example diagrams, unless otherwise indicated are from the [UML Reference Manual](#) and UML [User Guide](#)

# Diagram Types

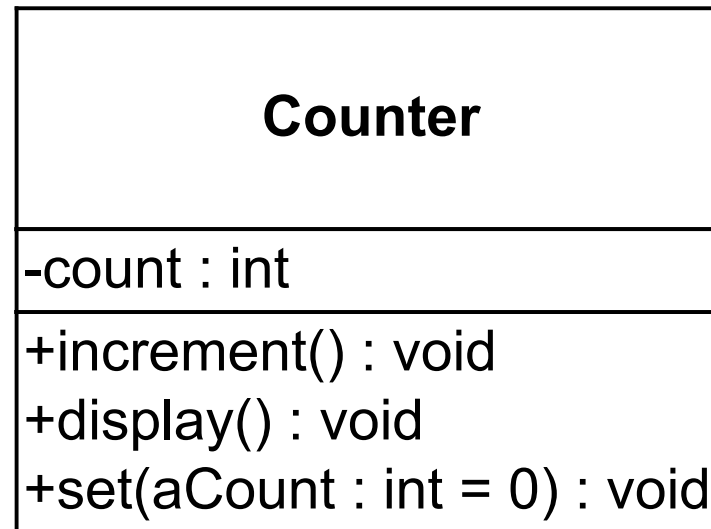
- Structure Diagrams
  - Class: components and structural properties
  - Composite Structure: internal structure and possible interactions
  - Component: organization of physical software components
  - Deployment: physical system resources and how they map to hardware
  - Object: static structure at a particular time
  - Package: logical groupings and dependencies
- Behavioral Diagrams
  - Activity: flow of control from activity to activity
  - Sequence: interaction of classes in terms of message exchange
  - Communication: object interaction in terms of numbered messages
  - Interaction Overview: synthesis of lower-level Activity Diagrams
  - Timing: rotated sequence diagram
  - Use Case: system functionality provided to external actors
  - State Machine: dynamic behavior in response to stimuli

# Class Models

- Also called *static models*
- Structural properties
- Classes and relationships
- Numerous embellishments

# UML Classes

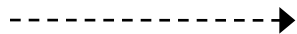
- Three-part box
  - Name of class
  - Attributes
  - Methods



# UML Relationships

- Dependencies: *X uses Y*

Dashed, directed line



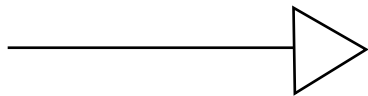
- Associations: *X affects Y; X has a Y*

Solid line or line with diamond

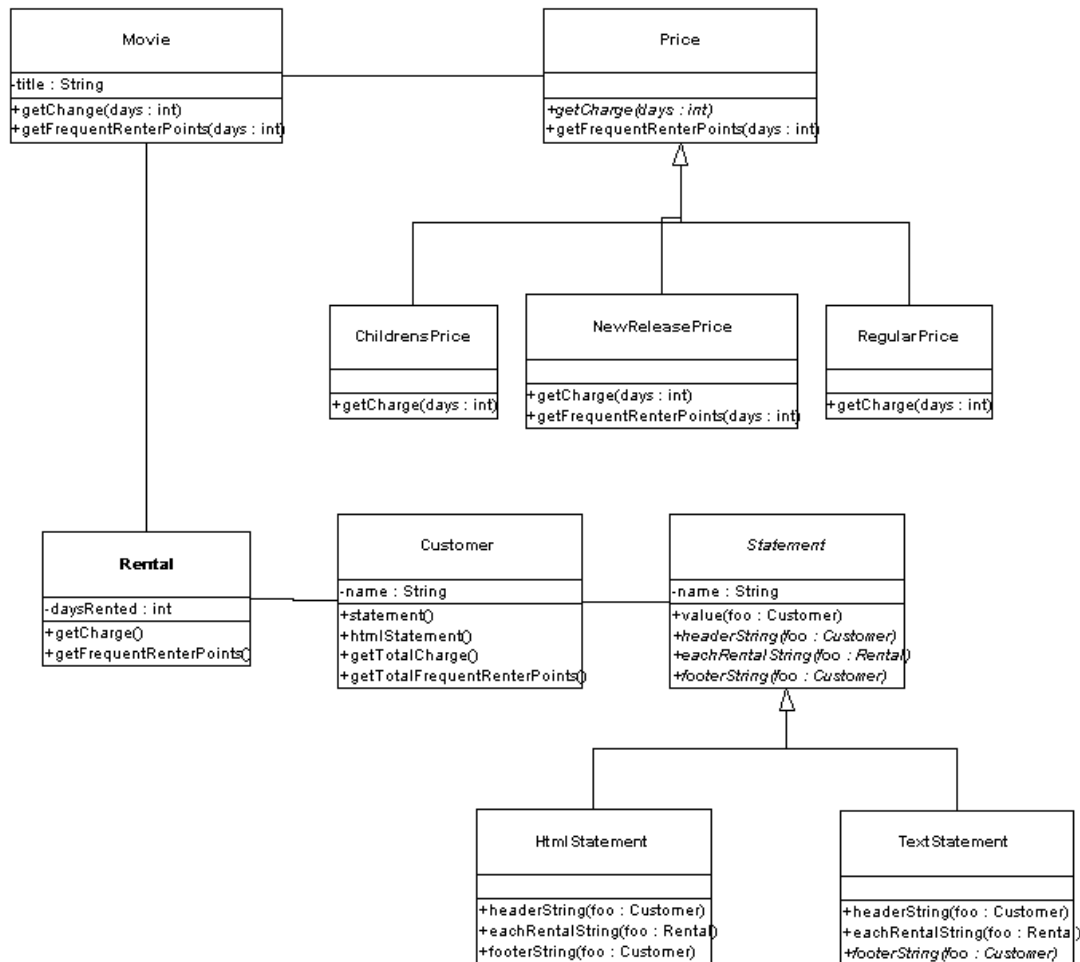


- Generalization: *X is a kind of Y*

Solid directed line with large, open arrowhead



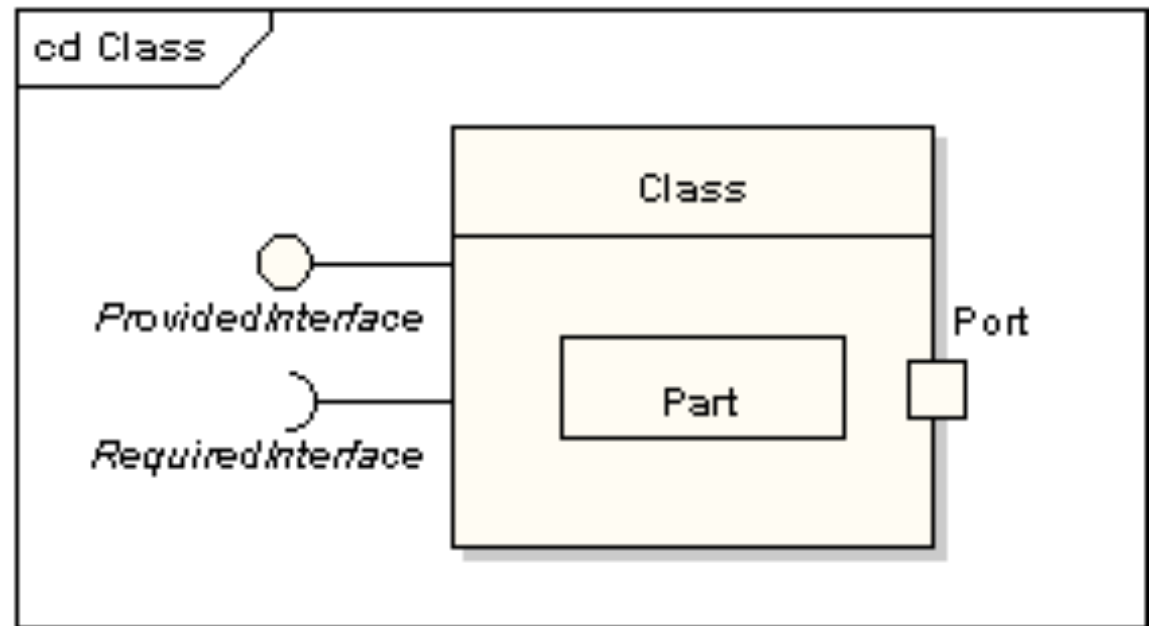
# Example Class Diagram



# Composite Structure Diagram

(from SparxSystems)

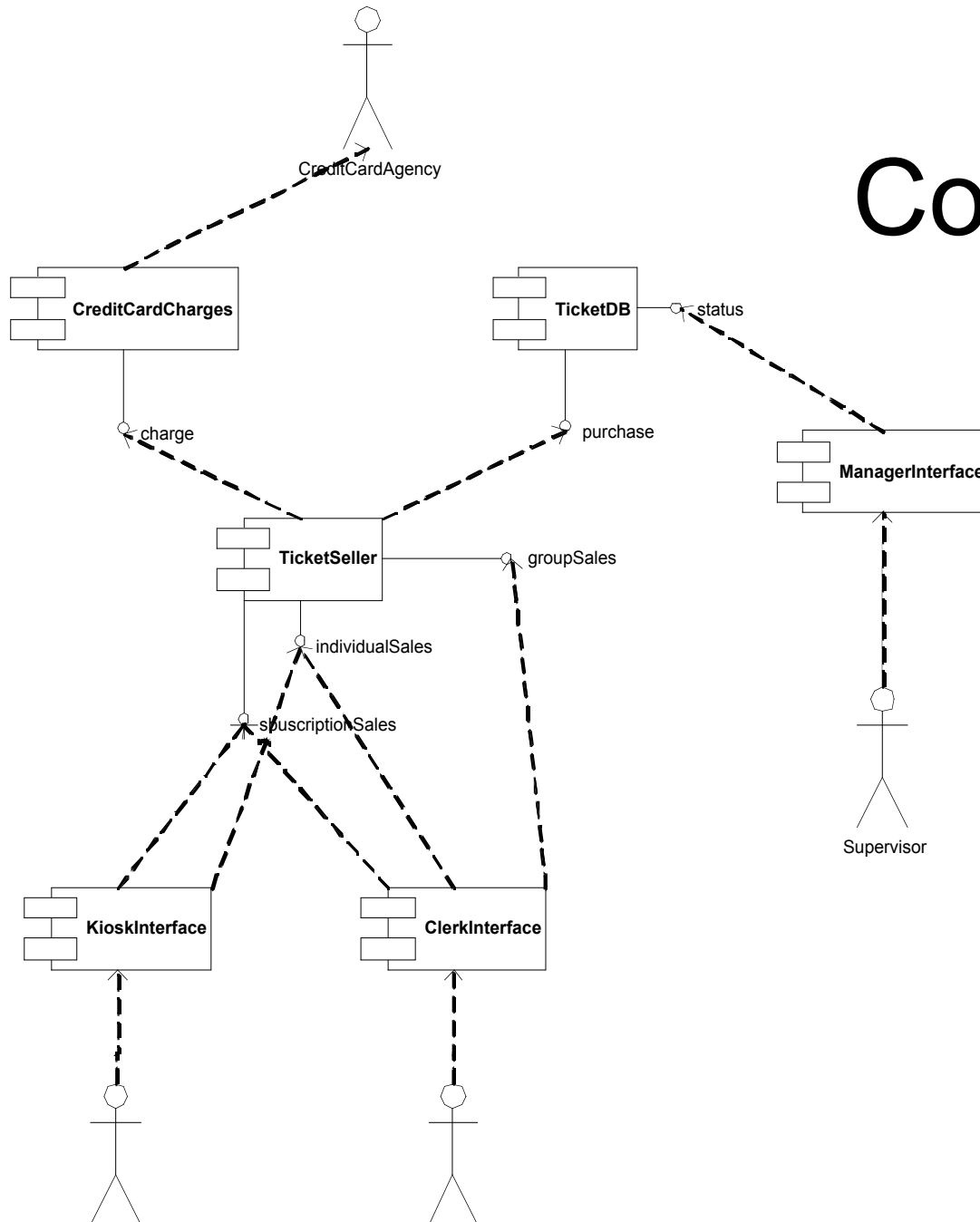
- Internal structure of a class
- Parts
- Ports (environment interactions)
- Interfaces



# Component Diagram

- Static implementation view
- A component is "A physical, replaceable part of a system that packages implementation and conforms to and provides the realization of a set of interfaces" - *UML Reference Manual* (URM)
- Models code entities
  - Binary (including libraries) or source (including scripts)
  - Relationships indicate *uses services of*
  - May be nested
- Can be used to convey architecture

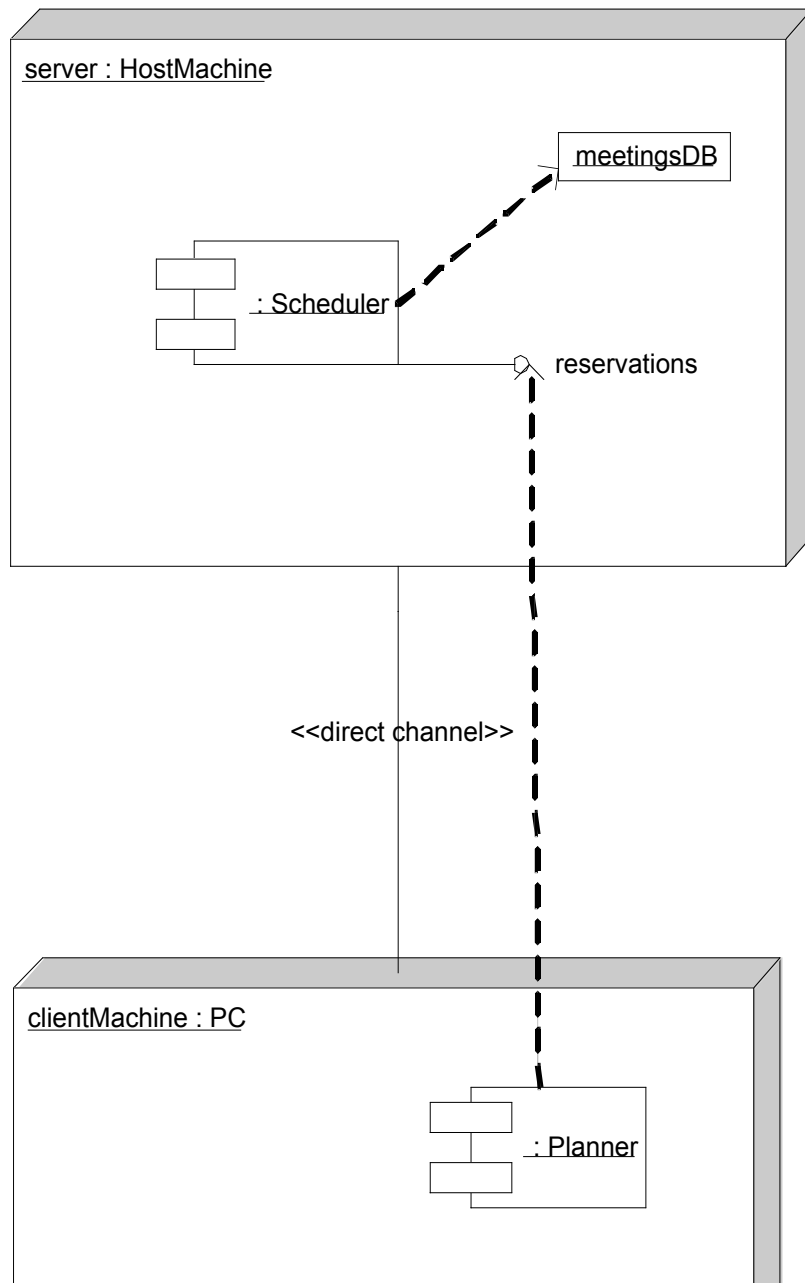
# Example Component Diagram



# Deployment Diagram

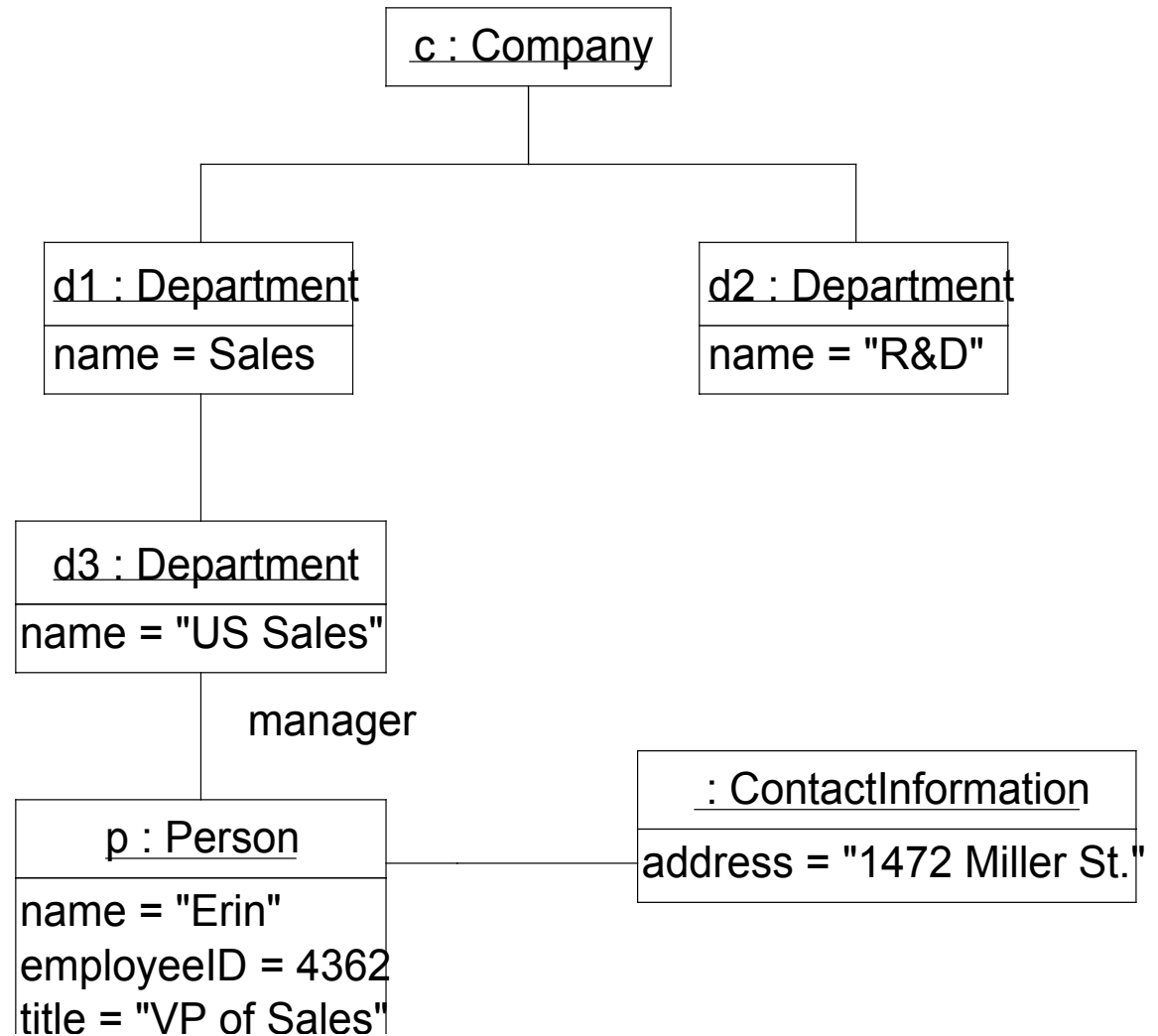
- "Configuration of run-time processing nodes and the component instances and objects that live on them." - URM
- *Node* denotes a computational device
- Arcs indicate communications

# Example Deployment Diagram



# Object Diagram

- Objects and *links*  
(instances of relationships)
- Shows specific instances rather than classes

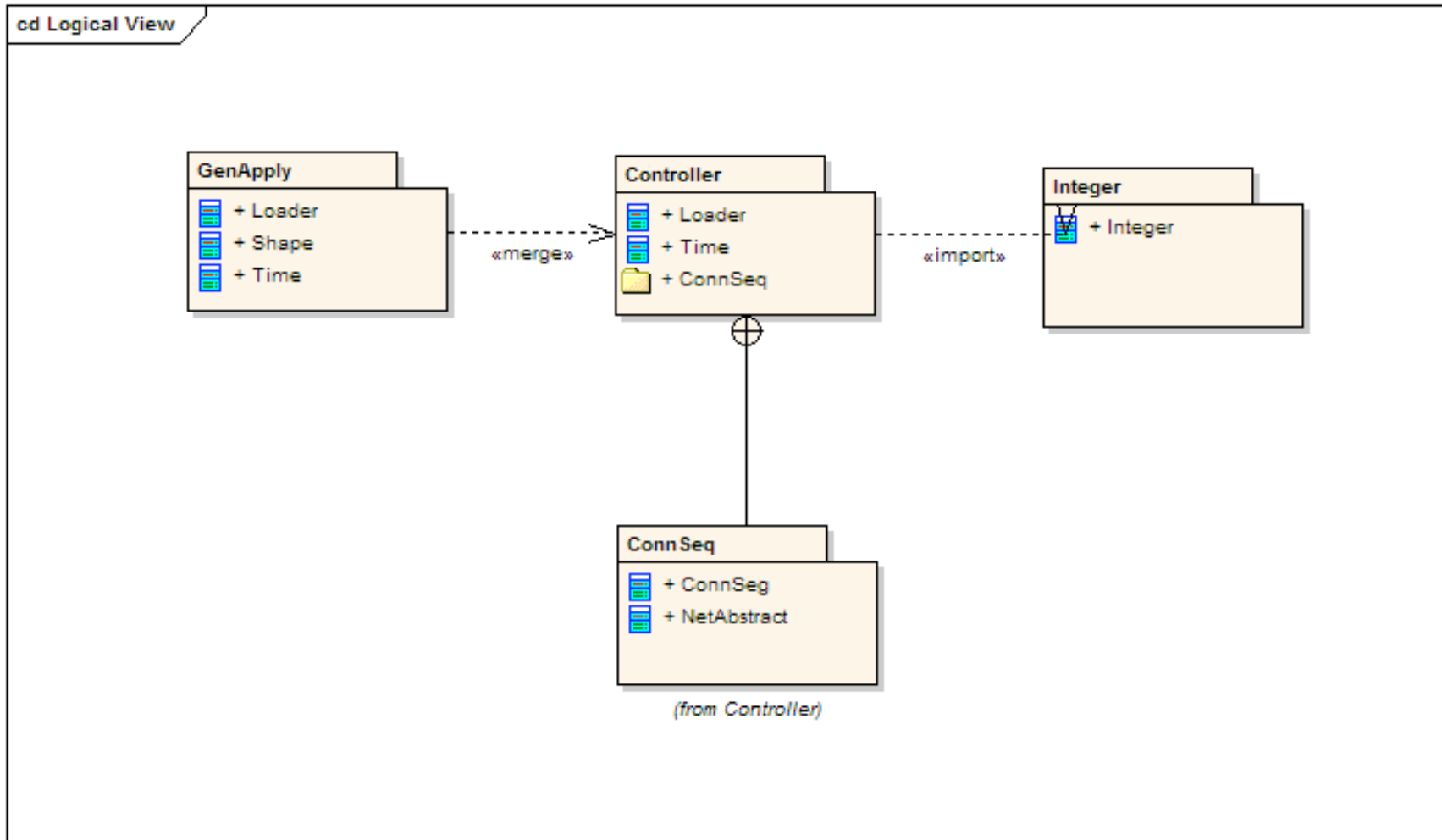


# Packages

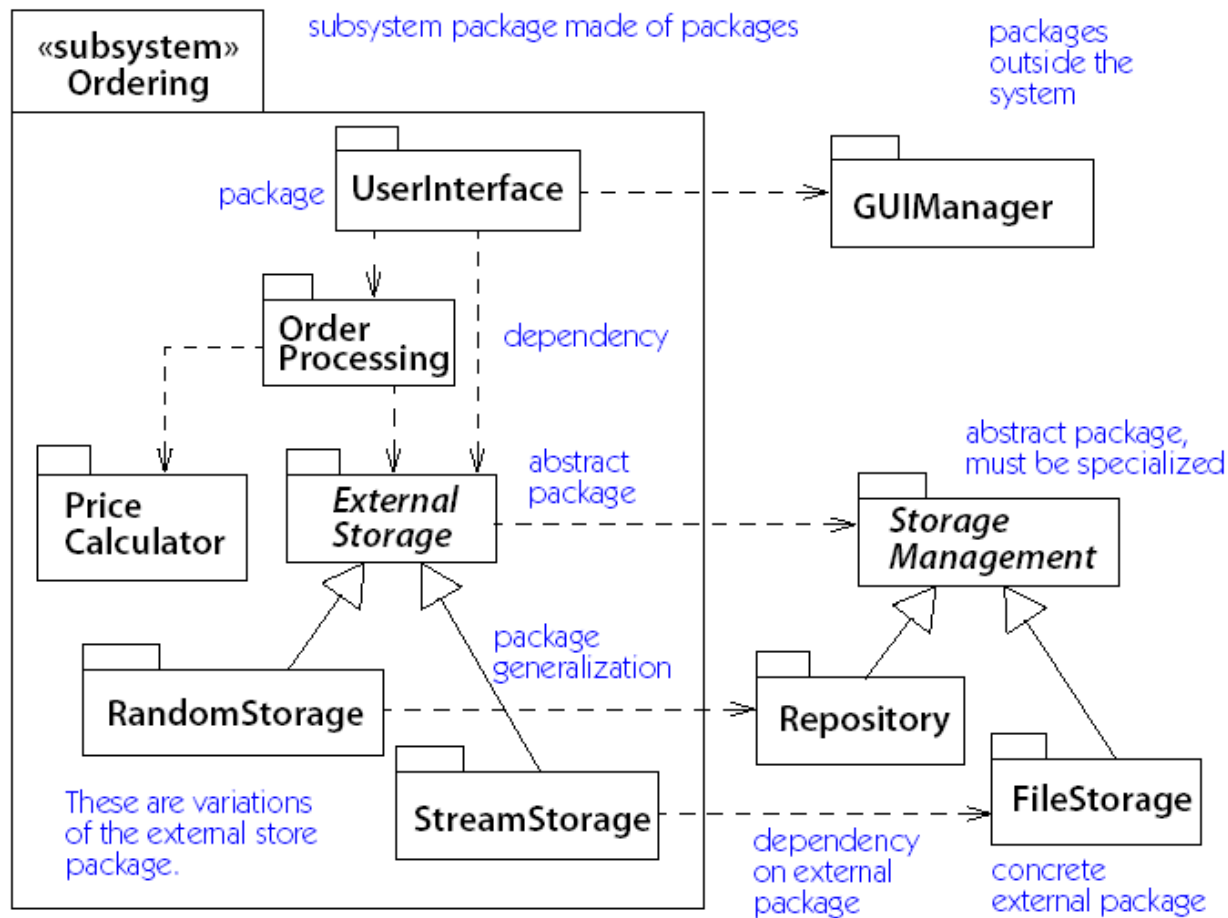
- General purpose organizing mechanism
  - May be used on other diagrams (1.5)
  - Has its own diagram (2.0)
- Provides namespace scoping
- *System* is the top-level package
- Dependency arrows between packages indicate the existence of dependencies between constituents

# Example Package Diagram

(SparxSystems)



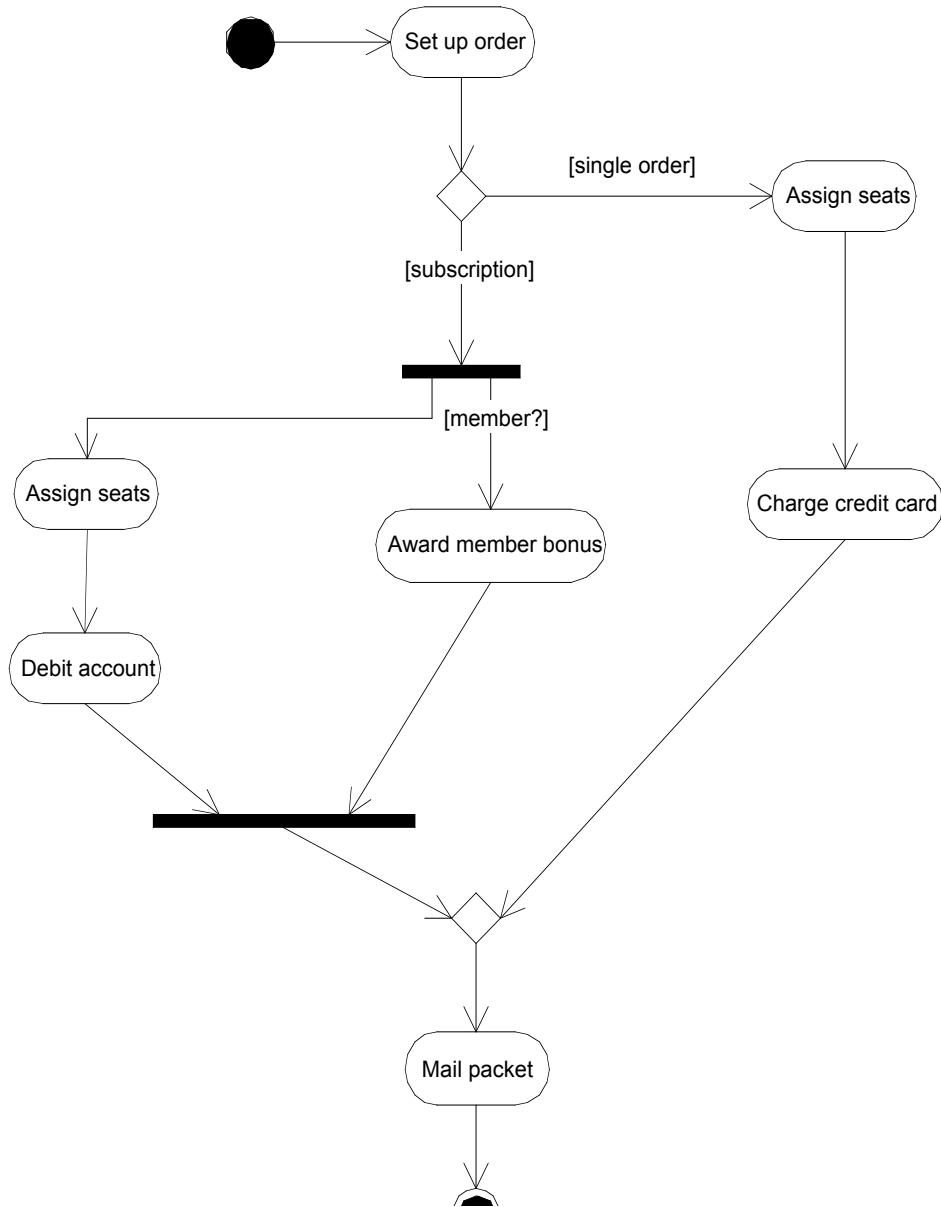
# Example Class Diagram with Packages (UML 1.5)



# Activity Diagram

- Variant of a state machine in which many states are active (have their own threads)
  - Similar to Petri Nets
- Transitions typically triggered by activity completion rather than by external events
- Used to model workflows, process synchronization, concurrency

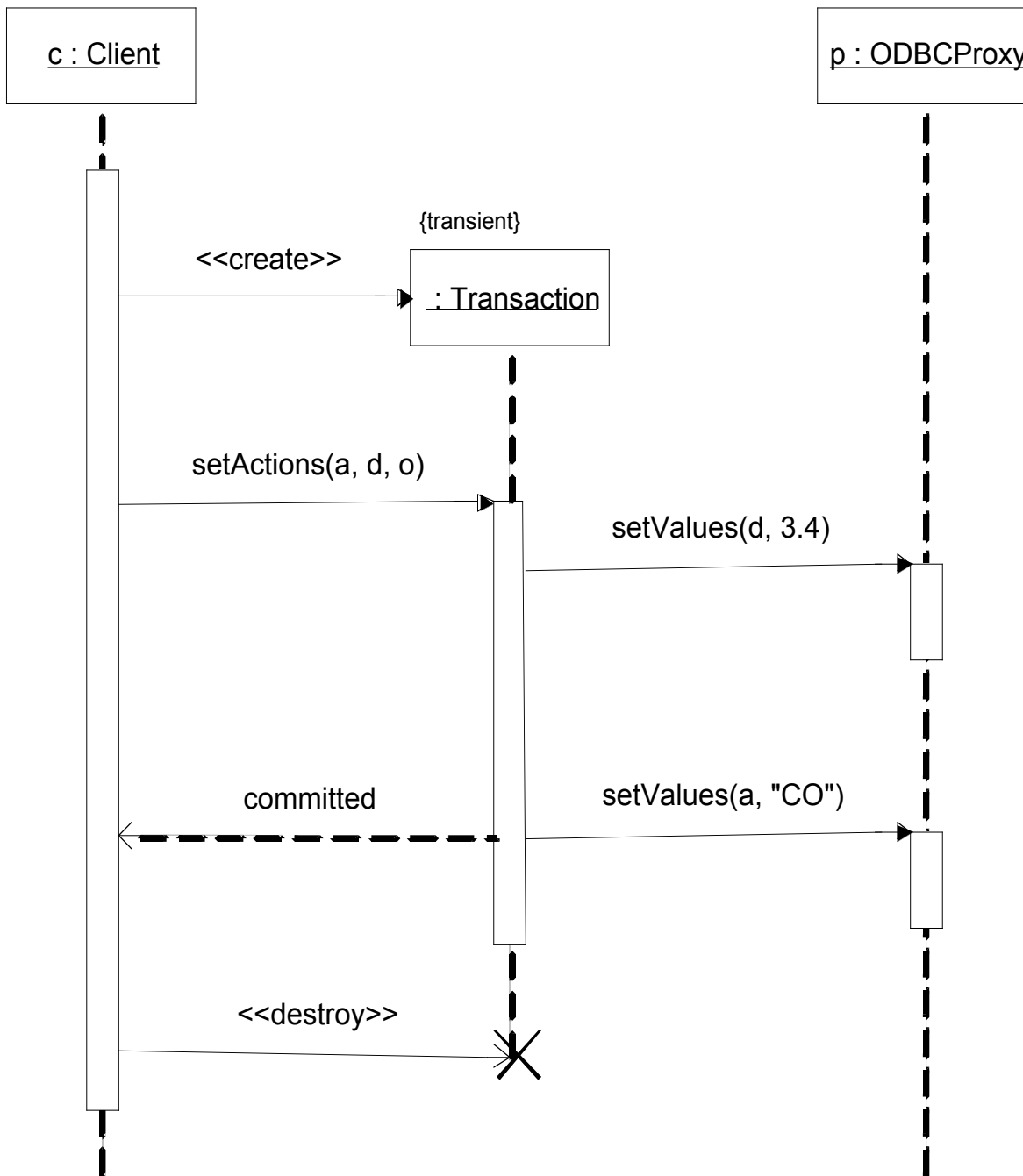
BoxOffice::ReceiveOrder



# Example Activity Diagram

# Sequence Diagram

- Sometimes called Message Sequence Chart
- Each column corresponds to an object
- Lines indicate interobject communication
- Semantically equivalent to Communication Diagram

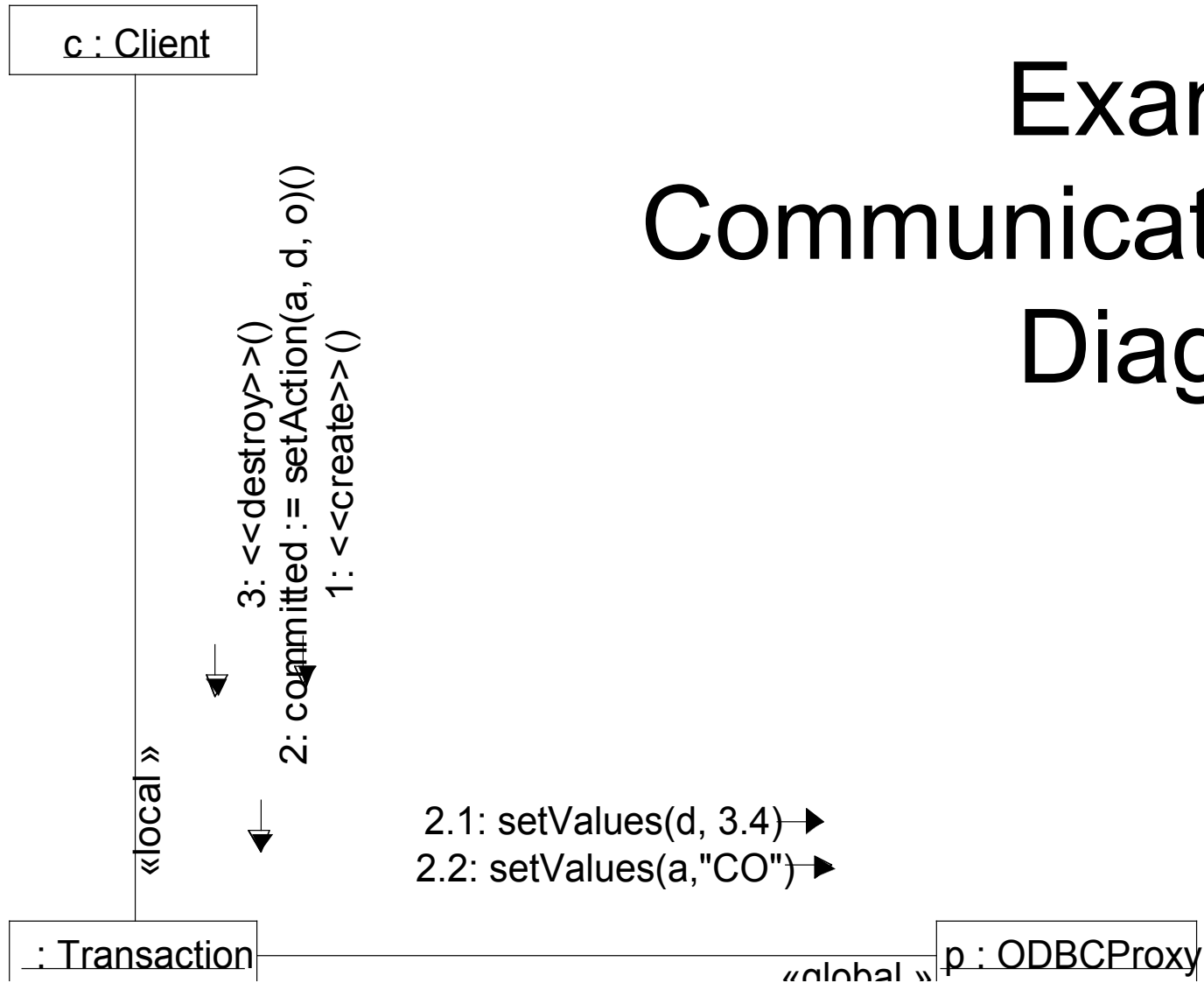


# Example Sequence Diagram

# Communication Diagram

- Object diagram annotated with ordered interactions instead of links
- Semantically equivalent to Sequence Diagram

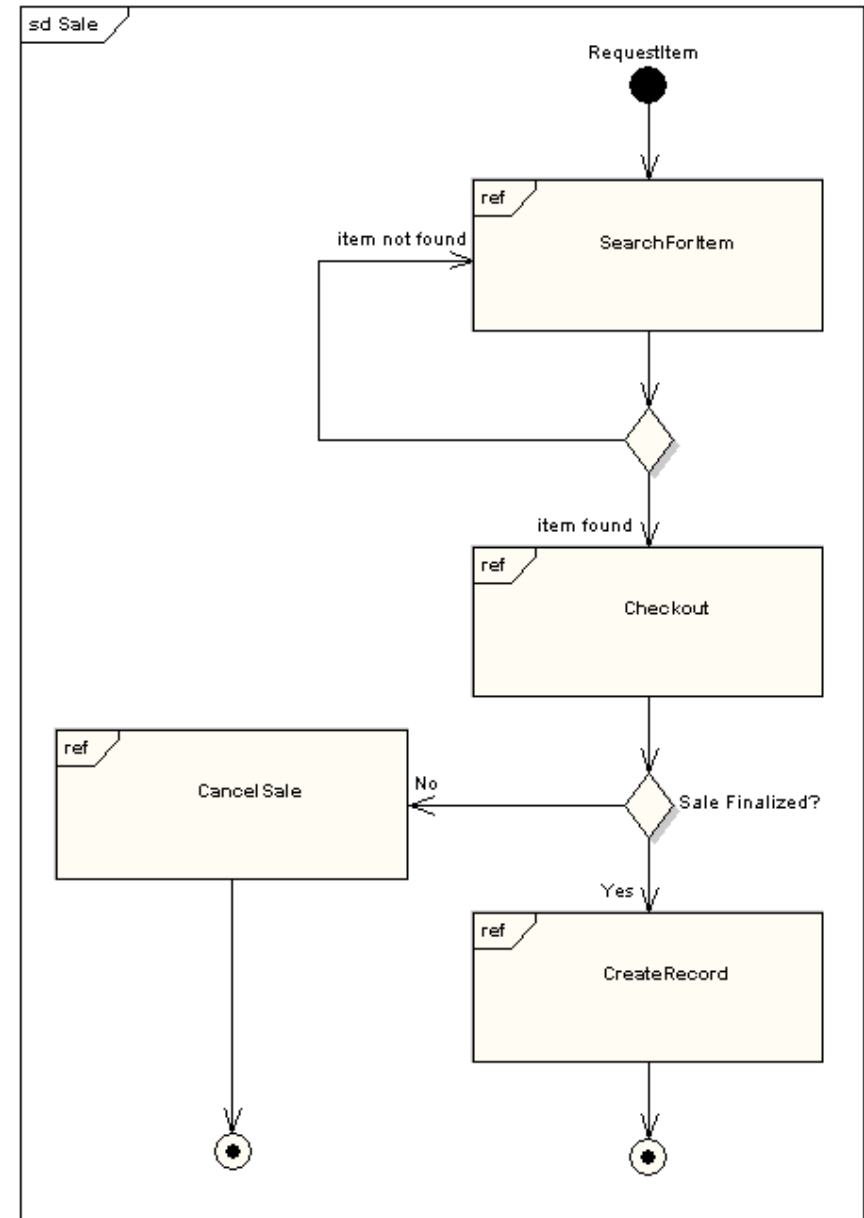
# Example Communications Diagram



# Interaction Overview Diagram

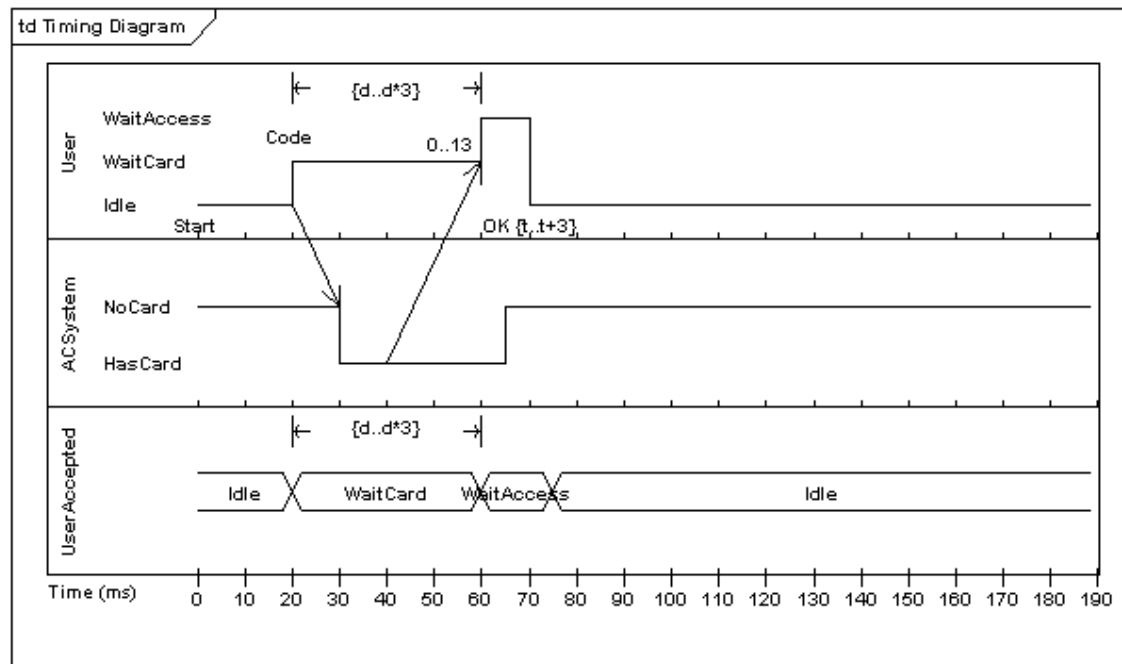
(SparxSystems)

- Type of Activity Diagram
- Nodes denote lower-level Interaction Diagrams
  - Sequence, communication, interaction overview and timing
- Adds interaction occurrences and interaction elements



# Timing Diagram

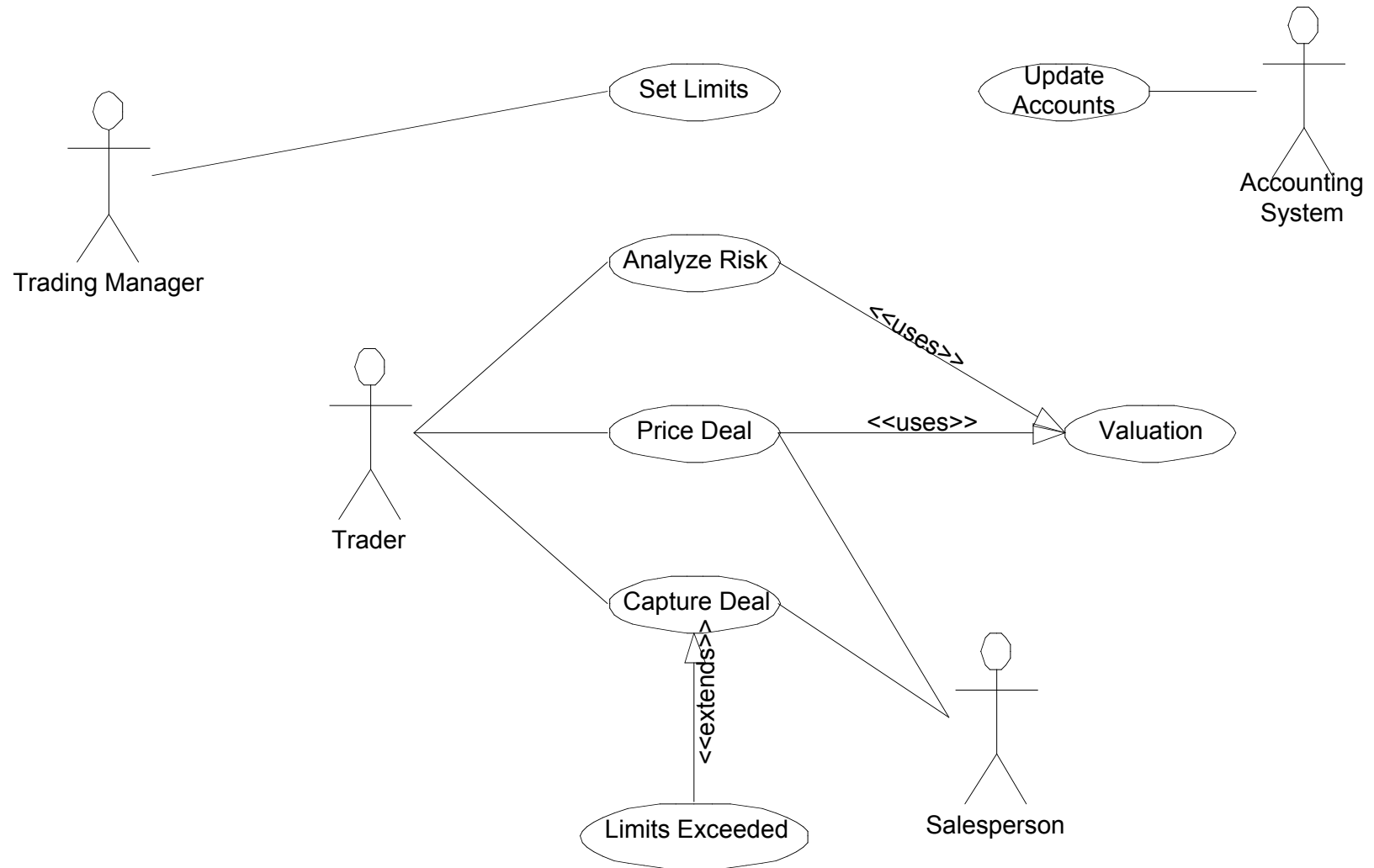
- Show change in state or value over time
- Interaction among timed events
  - Time and duration constraints



# Use Case Diagram

- Describes a set of use cases (user visible action sequences)
  - Stick figures are (external) *actors*
  - Ovals are use cases
  - Lines without annotations indicate participation
  - `<<extends>>` and `<<uses>>` *stereotype* (annotations) support factoring
- Use case diagram plays the role of dataflow diagrams, specifically, Context Diagrams, from Structured Design

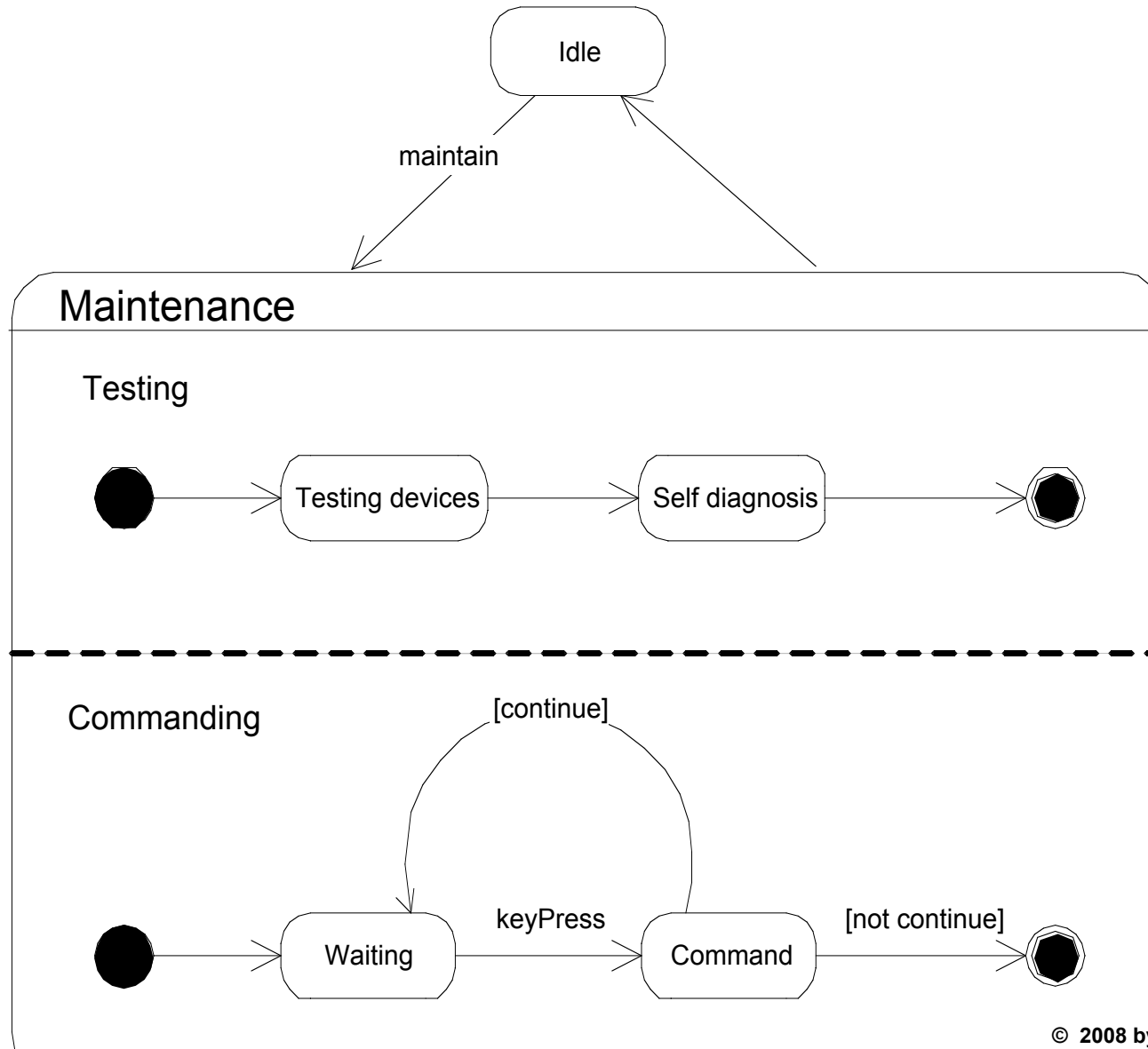
# Example Use Case Diagram



# State Machine Diagrams

- Also called StateCharts
- Extended finite state machines
  - Aggregation, concurrency, history, broadcast
- Most elaborate / complex part of UML

# Example State Machine Diagram



# Object Constraint Language

- Textual extension to UML's visual notation
- Applicable to Class and State Machine Diagrams
- First-order predicate logic + diagram navigation + collection classes
- Enables more precise specifications

# UML Metamodel

- UML itself is defined in terms of UML!
  - Recursive definition
- The definition of UML is called the UML Metamodel
- The UML Metamodel can be extended by a modeler using *profiles*
  - Stereotypes, tagged values, constraints
  - The extensions make diagrams more precise (domain specific)