UML Modeling

UML diagrams

- UML (Unified Modeling Language) is a general purpose visual modeling language that provides different types of diagrammatic techniques and notations to specify, visualize, analyze, construct, and document the artifacts of a software system.

- Software artifacts include: SRS, SDS, test cases, source code, technical/user manual, software architecture, etc.

- UML diagrams are used to understand, design, browse, configure, maintain, and control information about software system.

- UML is intended for use with all development methods, lifecycle stages, application domains, and media.

- In these slides we cover use-case diagram, sequence diagram, collaborative diagram, class diagram, and component diagram from UML.

UML History

- UML was developed in an effort to unify and simplify a number of OO development methods that use popular OO languages.

- Simula 67 was the first OO language

- Smalltalk was introduced in early 1980s followed by Objective C, C++, Eiffle, CLOS, Java.

- First OO development method was Shlaer-88, then Yourdon 91, and Booch 91

- Unification effort:
  - UML 1995 by Booch, Rumbaugh, Jacobson
  - OMG 1996 proposed a standard for OO modeling

Modeling Software System

- What is model?
  - A model captures the important aspects of the artifact being modeled from a certain point of view, and simplifies or omits the rest.

- What are models for?
  - To capture and precisely state requirements and domain knowledge so that all stakeholders may understand and argue on them.
  - To think about the design of a system
  - To capture design decisions
  - To organize, find, filter, retrieve, examine, and edit information about large systems.

UML Views

- Views are the result of applying separation of concern on the development process in order to classify the knowledge about the system into more understandable and manageable forms.

- In the Zachman and 4+1 view models, the views are orthogonal, i.e., each view consists of different set of concepts.

- There is no sharp line between different views in UML. A view is a subset of UML modeling constructs and concepts that represents one aspect of a system that is also intuitive. Three categories:
  - Structural classification: things and in-between relations
    - Static view, use case view, implementation view
  - Dynamic behavior: behavior of the system over time
    - State machine view, activity view, interaction view
  - Model management: organization of the models themselves
  - Extensibility: to constrain or describe attributes of the views

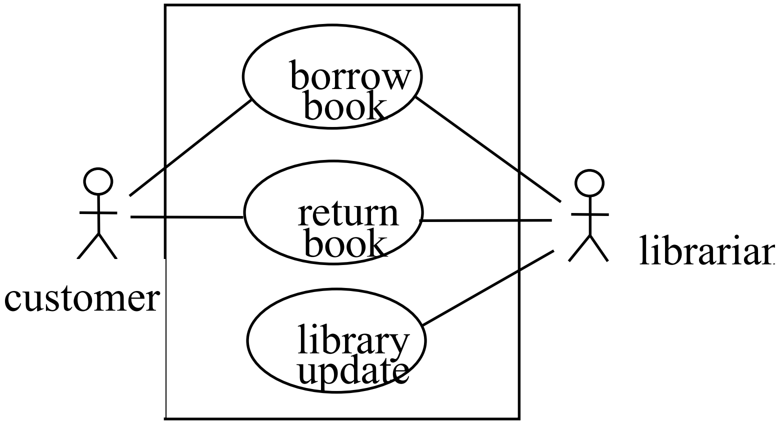
	Zachman	Views		
	Framework	Data view	Function view	Network view
Perspectives	General scope (Ballpark)	**List of entities important to business**	**List of functions the business performs**	**List of locations the business operates**
	Owner's perspective	**Entity-relation diagram**	**Function flow diagram**	**Logistic network**
	Designer's perspective (Architect's plan)	**Data model**	**Data flow diagram**	**Distributed system architecture**
	Developer's perspective (Contractor's plan)	**Data design**	**Structure chart**	**System architecture**
	Programmer's perspective (Builder's product)	**Data description**	**Program**	**Network architecture**

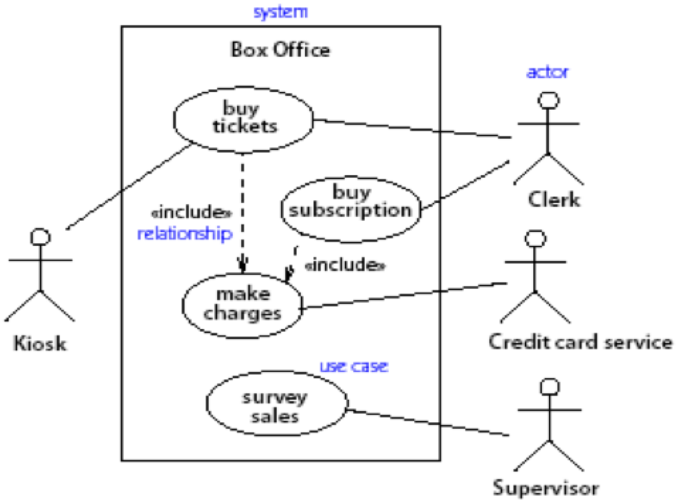
Table 3-1: *UML Views and Diagrams*

Major Area	*View*	*Diagrams*	*Main Concepts*
structural	static view	class diagram	class, association, generalization, dependency, realization, interface
	use case view	use case diagram	use case, actor, association, extend, include, use case generalization
	implementation view	component diagram	component, interface, dependency, realization
	deployment view	deployment diagram	node, component, dependency, location
dynamic	state machine view	statechart diagram	state, event, transition, action
	activity view	activity diagram	state, activity, completion transition, fork, join
	interaction view	sequence diagram	interaction, object, message, activation
		collaboration diagram	collaboration, interaction, collaboration role, message
model management	model management view	class diagram	package, subsystem, model
extensibility	all	all	constraint, stereotype, tagged values

UML use-case diagrams

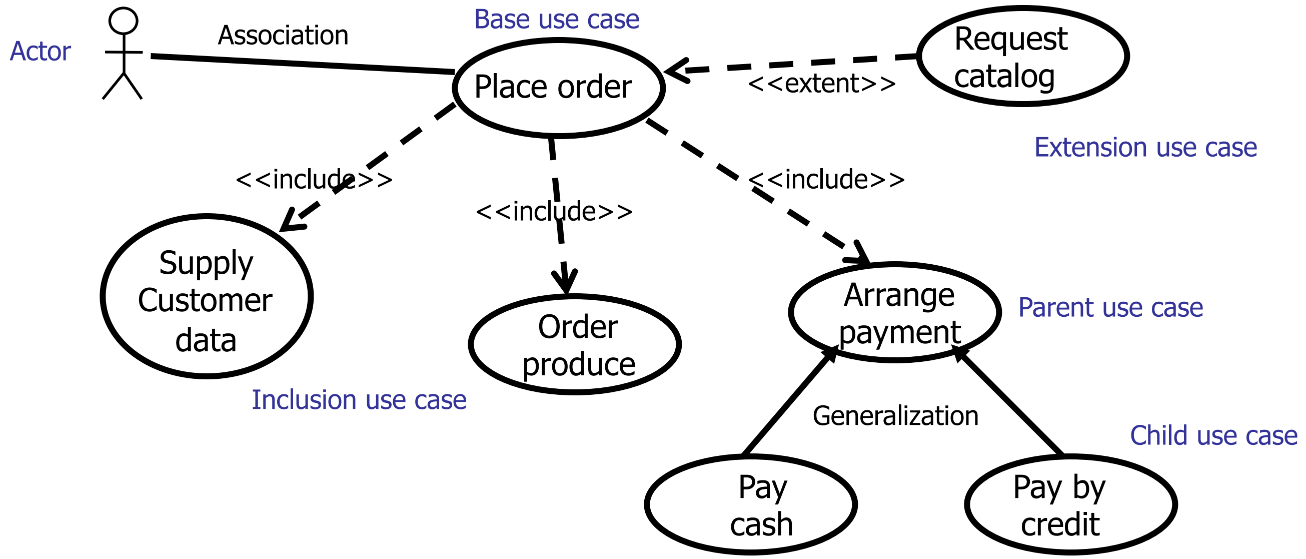
- Defines a global view of the actors involved in a system and the actions that the system performs, which in turn provides an observable result that is of value to the actors.
- Partitions the overall functionality of the system into transactions with respect to the actor and illustrates how actors interact with them.
- Actors define different roles such as: people, computer systems, environment





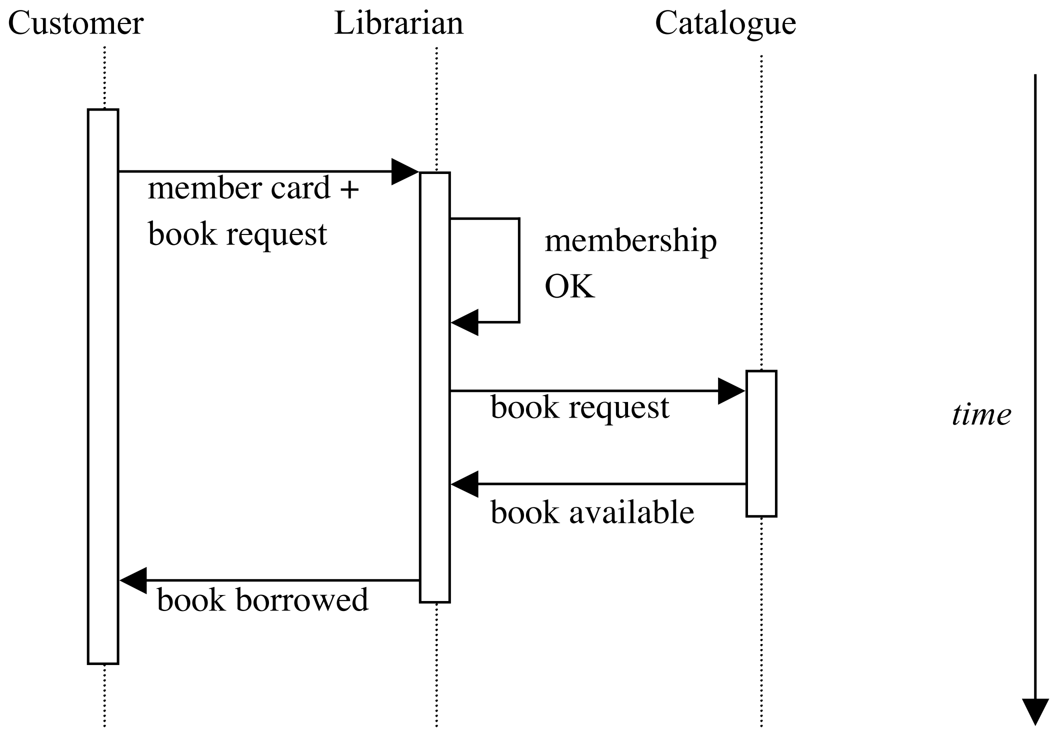
Use case diagram notations

- Association: the communication path between an actor and a use case that the actor participates in
- Extend: the insertion of additional behavior into a base use case that extends its operation.
- Generalization: relation between a general use case and a more specific use case that inherits from it.
- Include: the insertion of additional behavior into a base use case that describes the details of the base use case.

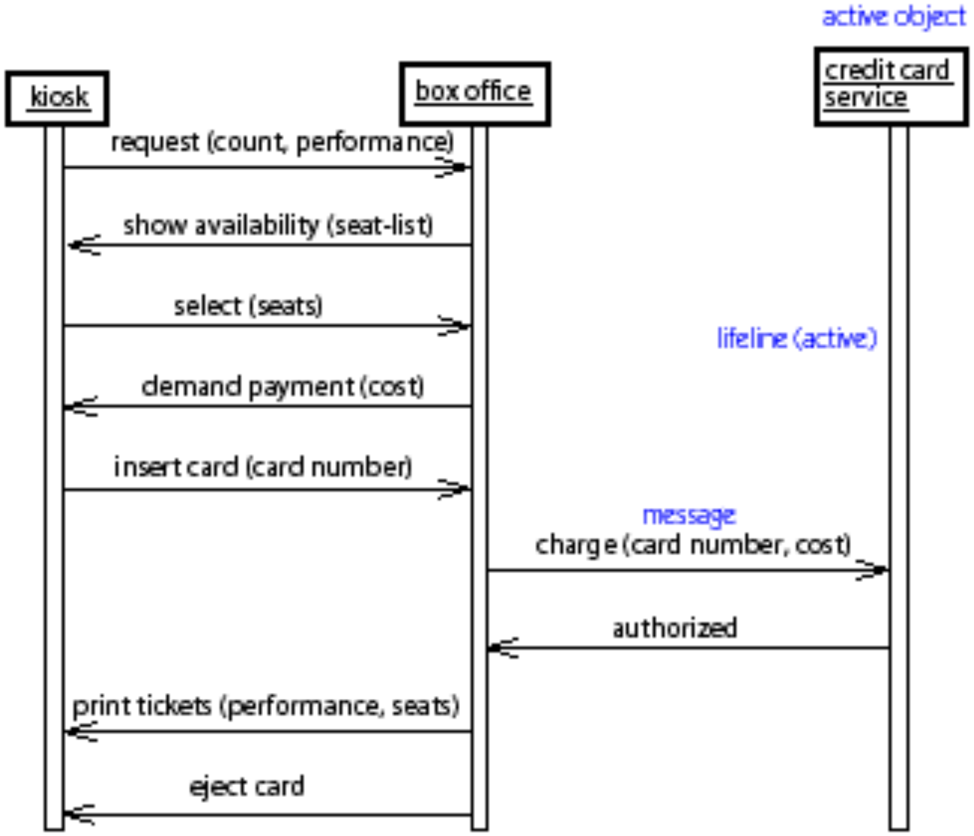


UML sequence diagram

- Describes how different objects in the system interact by exchanging messages
- Provides a dynamic and temporal view
- Emphasizes on time sequence of message exchange



Sequence Diagram


active object
kiosk
box office
credit card service
request (count, performance)
show availability (seat-list)
select (seats)
lifeline (active)
demand payment (cost)
insert card (card number)
message
charge (card number, cost)
authorized
print tickets (performance, seats)
eject card

UML sequence diagram example: Ticket selling box office

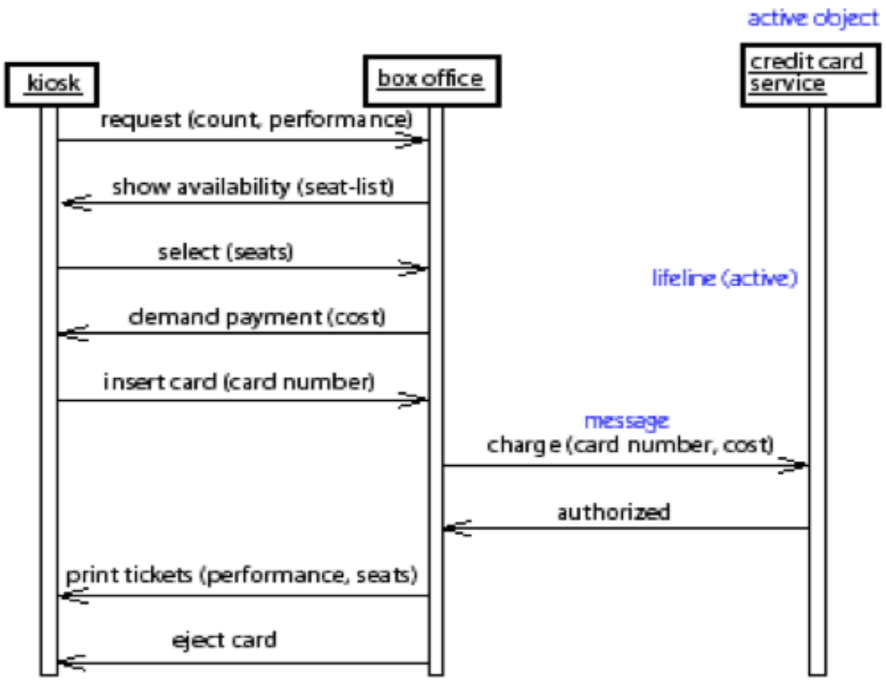

active object
kiosk
box office
credit card service
request (count, performance)
show availability (seat-list)
select (seats)
lifeline (active)
demand payment (cost)
insert card (card number)
message
charge (card number, cost)
authorized
print tickets (performance, seats)
eject card

Figure 3-3. *Sequence diagram*

Sequence diagram

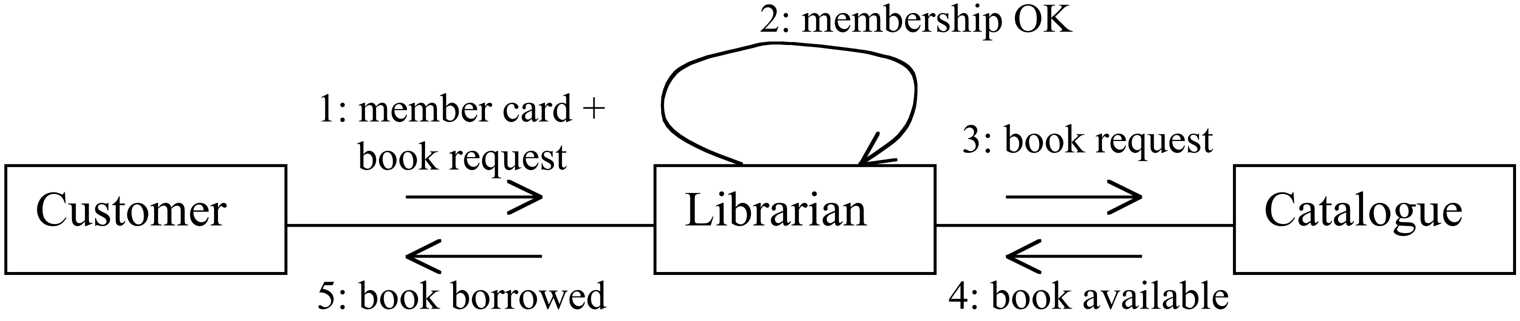
A sequence diagram shows a set of messages arranged in time sequence. Each classifier role is shown as a lifeline—that is, a vertical line that represents the role over time through the entire interaction. Messages are shown as arrows between lifelines. A sequence diagram can show a scenario—that is, an individual history of a transaction.

One use of a sequence diagram is to show the behavior sequence of a use case. When the behavior is implemented, each message on a sequence diagram corresponds to an operation on a class or an event trigger on a transition in a state machine.

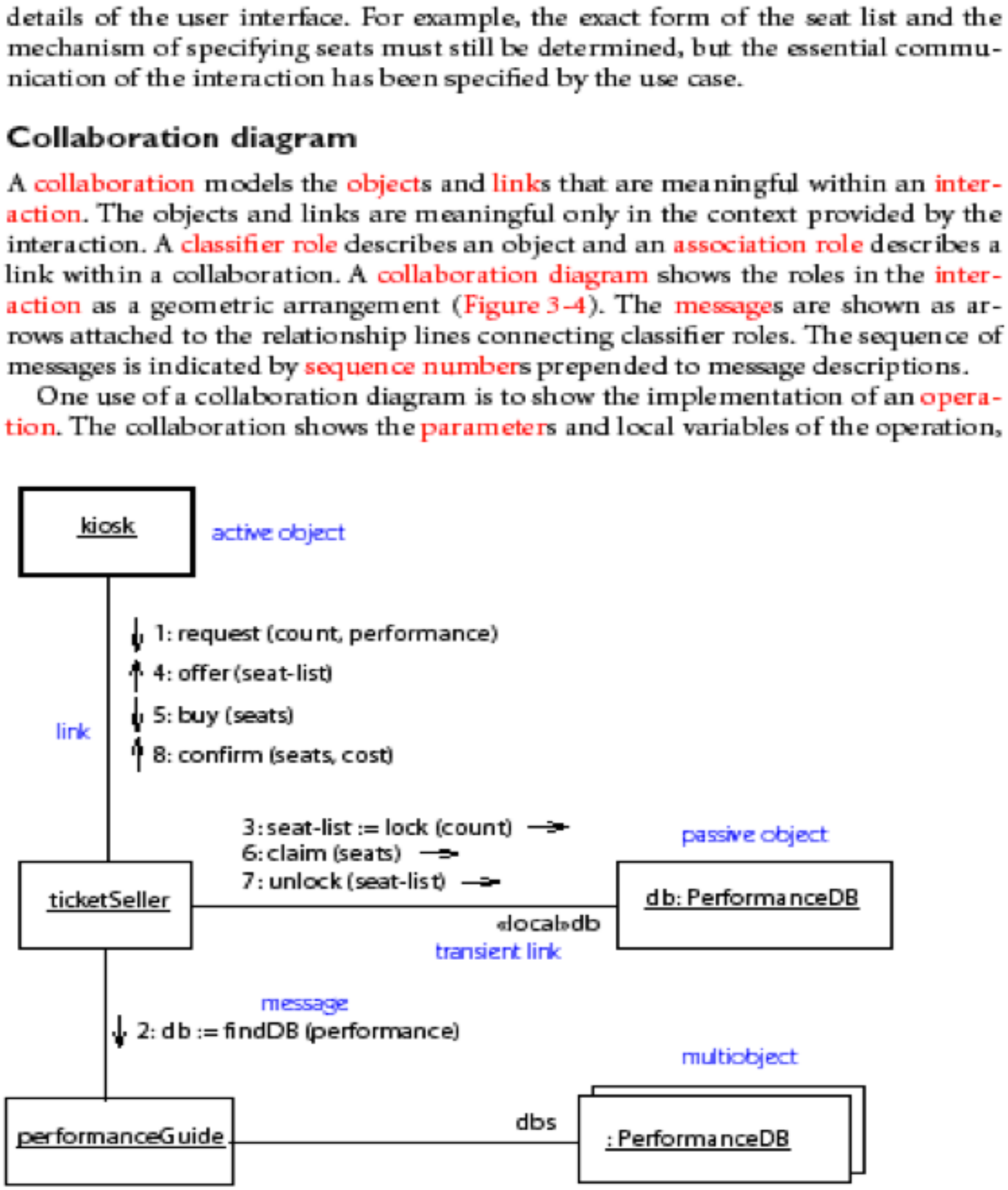
Figure 3-3 shows a sequence diagram for the buy tickets use case. This use case is initiated by the customer at the kiosk communicating with the box office. The steps for the make charges use case are included within the sequence, which involves communication with both the kiosk and the credit card service. This sequence diagram is at an early stage of development and does not show the full

UML collaboration diagrams

- Represents object interactions and their order
- Equivalent to sequence diagrams
- Sequence diagram is intended for time ordering considerations, whereas collaboration is emphasizes on the structural aspect of the system.



UML collaborative diagram example:
Ticket selling box office



details of the user interface. For example, the exact form of the seat list and the mechanism of specifying seats must still be determined, but the essential communication of the interaction has been specified by the use case.

Collaboration diagram

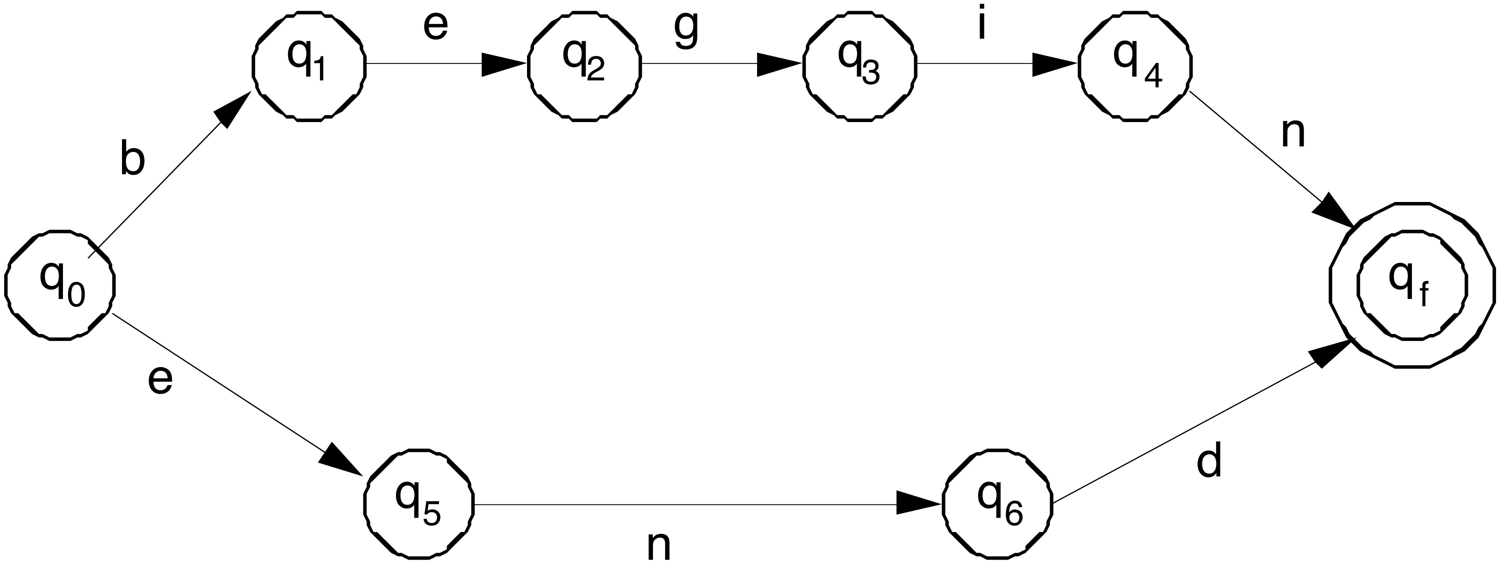
A collaboration models the objects and links that are meaningful within an interaction. The objects and links are meaningful only in the context provided by the interaction. A classifier role describes an object and an association role describes a link within a collaboration. A collaboration diagram shows the roles in the interaction as a geometric arrangement (Figure 3-4). The messages are shown as arrows attached to the relationship lines connecting classifier roles. The sequence of messages is indicated by sequence numbers prepended to message descriptions.

One use of a collaboration diagram is to show the implementation of an operation. The collaboration shows the parameters and local variables of the operation,

Classes of FSMs

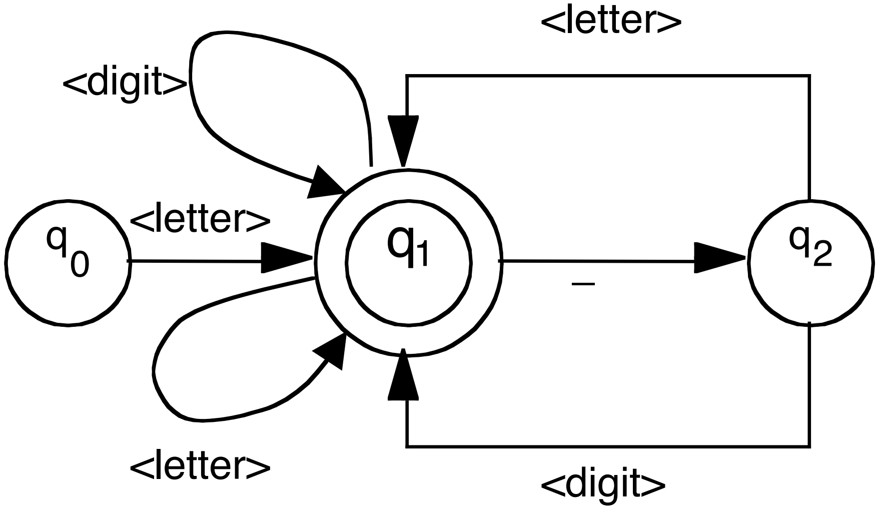
- Deterministic/nondeterministic
- FSMs as recognizers
  - introduce final states
- FSMs as transducers
  - introduce set of outputs
- . . .

FSMs as recognizers



q_f is a final state

FSMs as recognizers



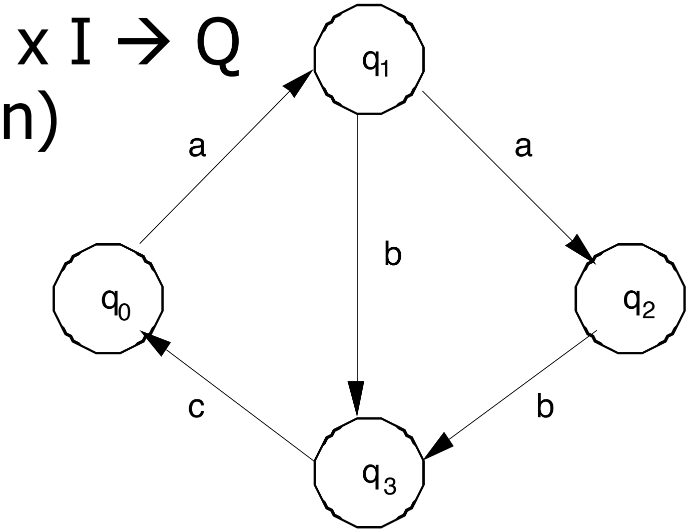
Legend: $\xrightarrow{\text{<letter>}}$ is an abbreviation for a set of arrows labeled a, b,..., z, A,..., Z, respectively

$\xrightarrow{\text{<digit>}}$ is an abbreviation for a set of arrows labeled 0, 1,..., 9, respectively

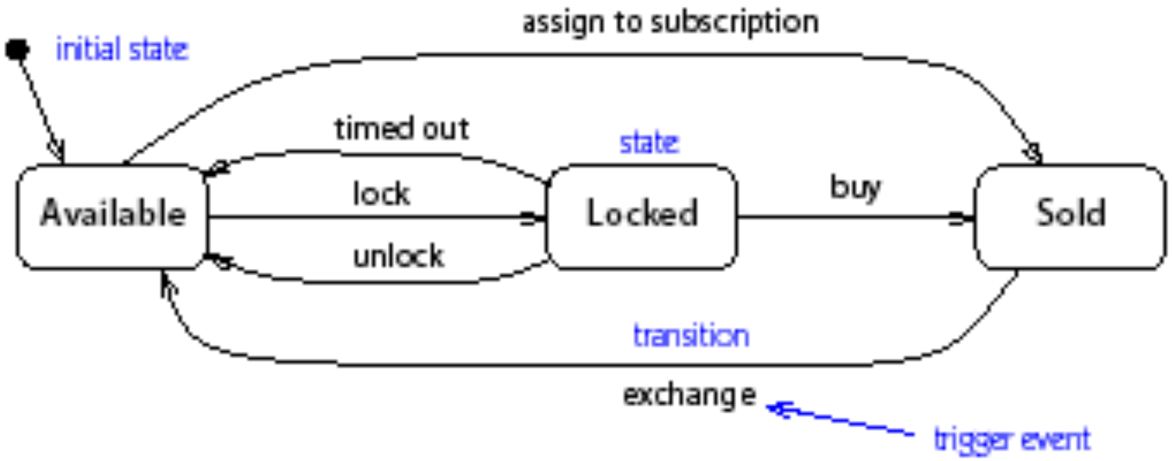
Finite state machines (FSMs)

- Can specify control flow aspects
- Defined as

a finite set of states, Q;
a finite set of inputs, I;
a transition function d : Q x I → Q
(d can be a partial function)



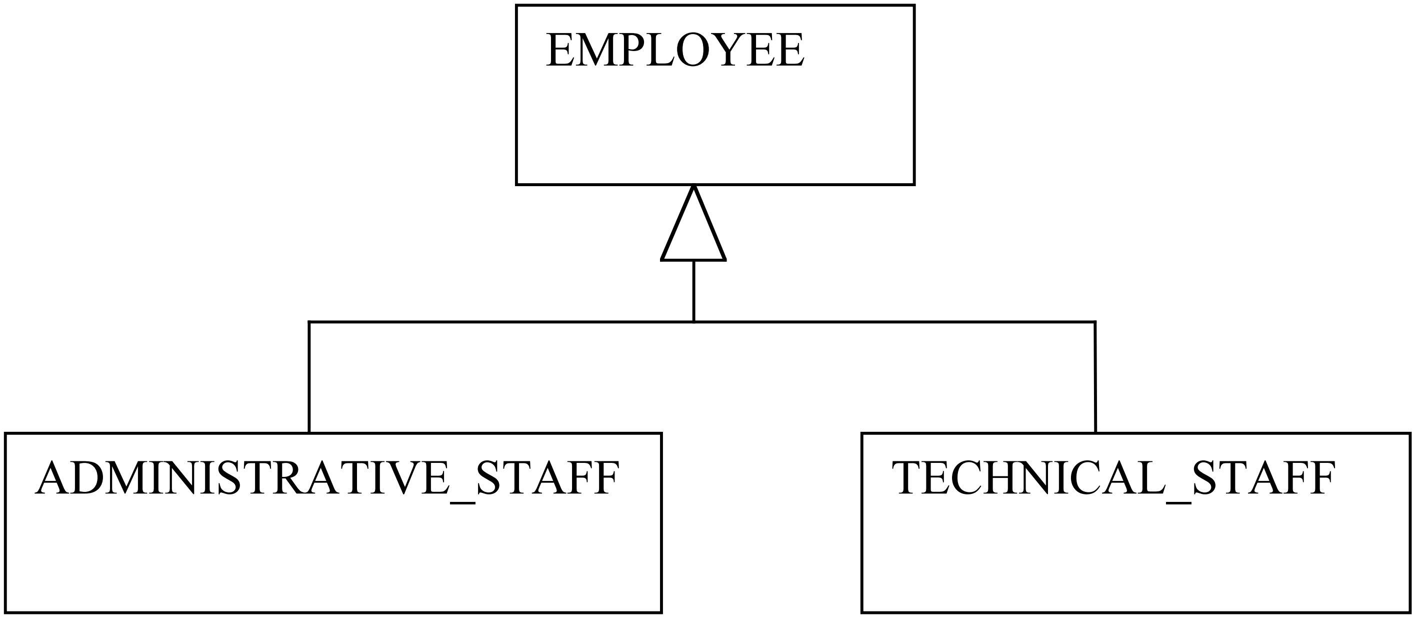
Statechart diagram
State machine view



Shows the history of a ticket to a performance

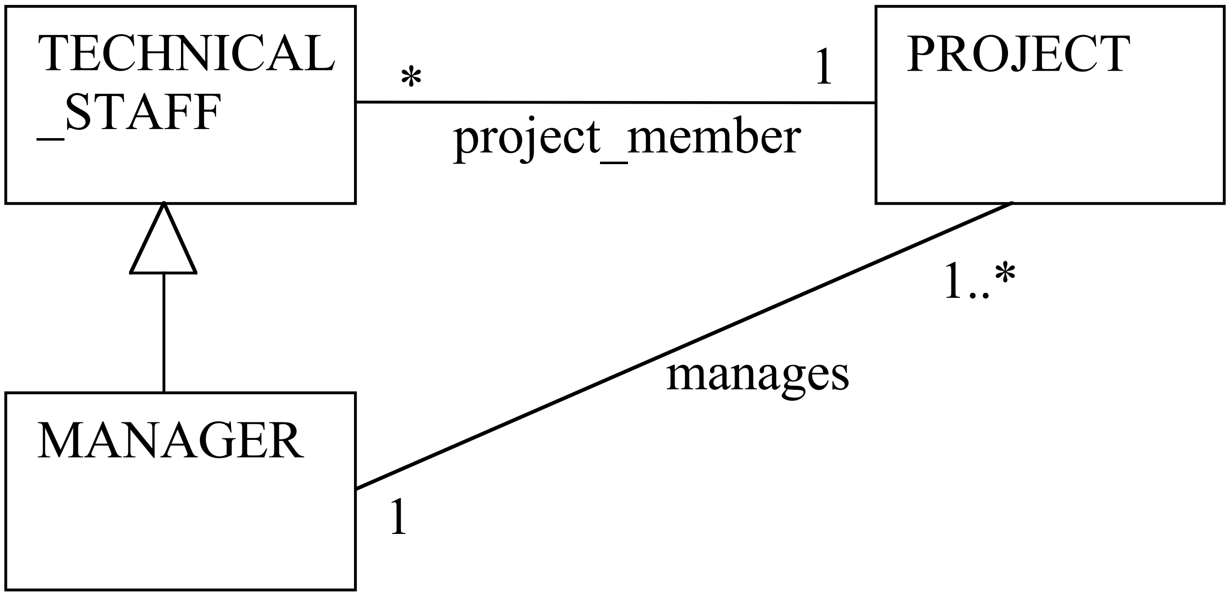
UML Class Diagram

UML representation of inheritance



UML associations

- Associations are relations that the implementation is required to support
- Can have multiplicity constraints



Aggregation

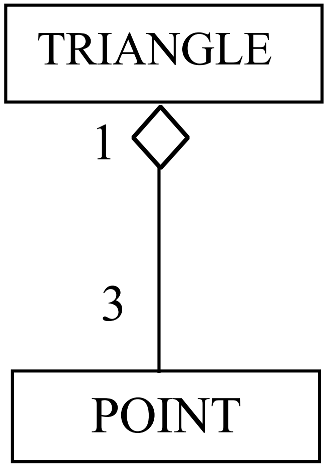
- Defines a PART_OF relation

  Differs from IS_COMPOSED_OF

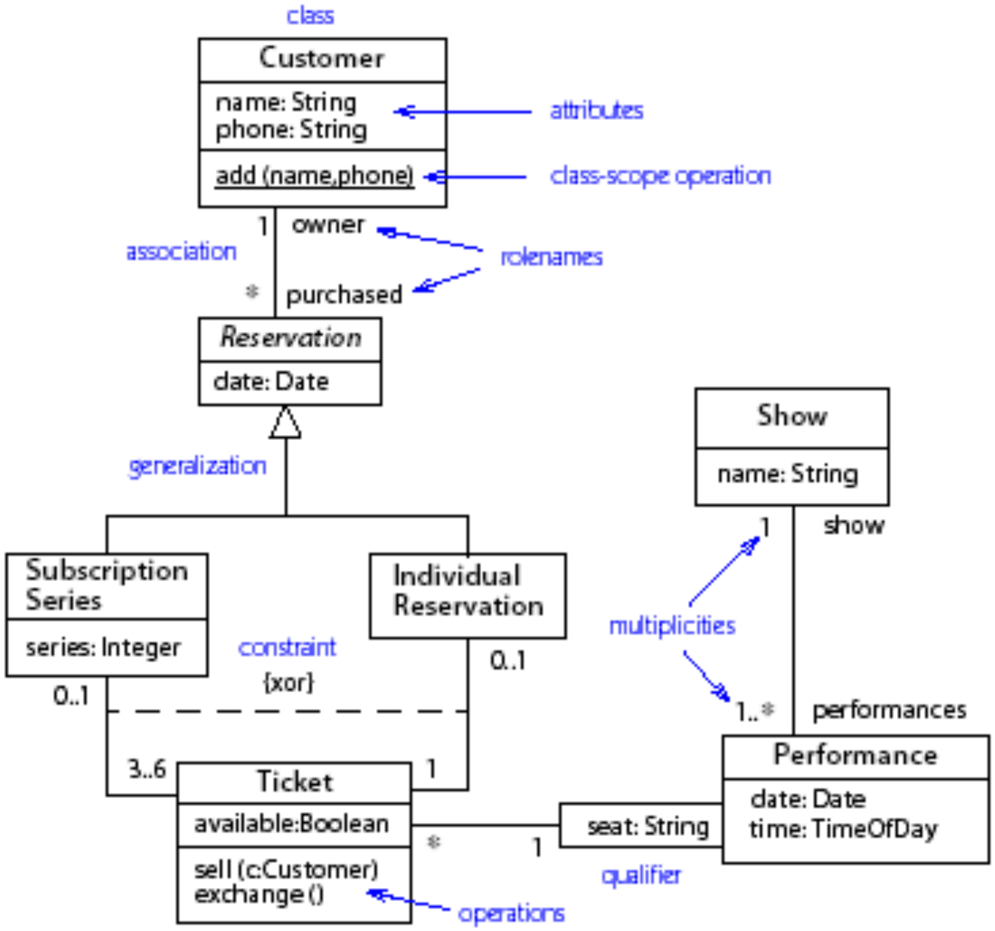
  Here TRIANGLE has its own methods

  It implicitly uses POINT to define

  its data attributes



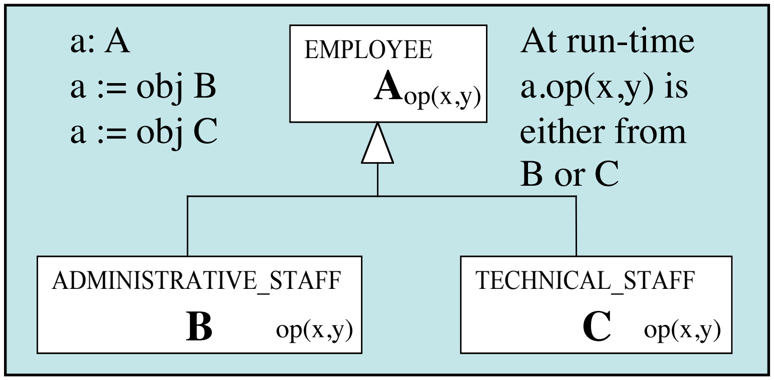
Ticket Selling Booth Class Diagram



Inheritance

- A way of building software incrementally

- A subclass defines a subtype
  - subtype is *substitutable* for parent type



a: A
a := obj B
a := obj C

EMPLOYEE
A op(x,y)

At run-time a.op(x,y) is either from B or C

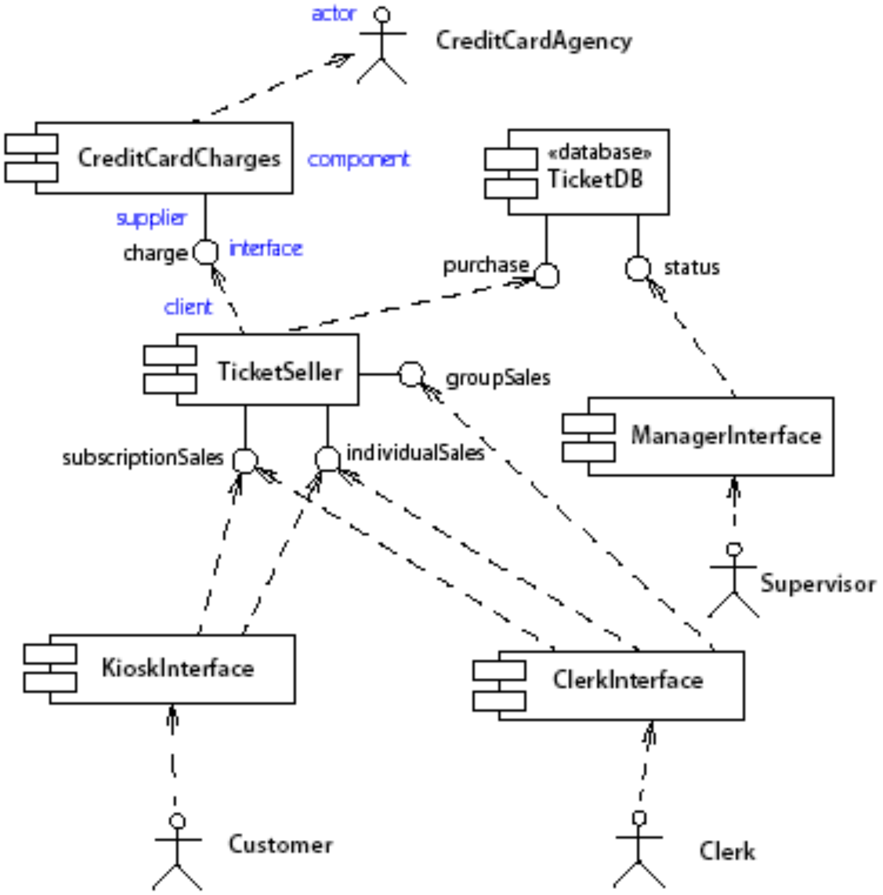
ADMINISTRATIVE_STAFF
B op(x,y)

TECHNICAL_STAFF
C op(x,y)

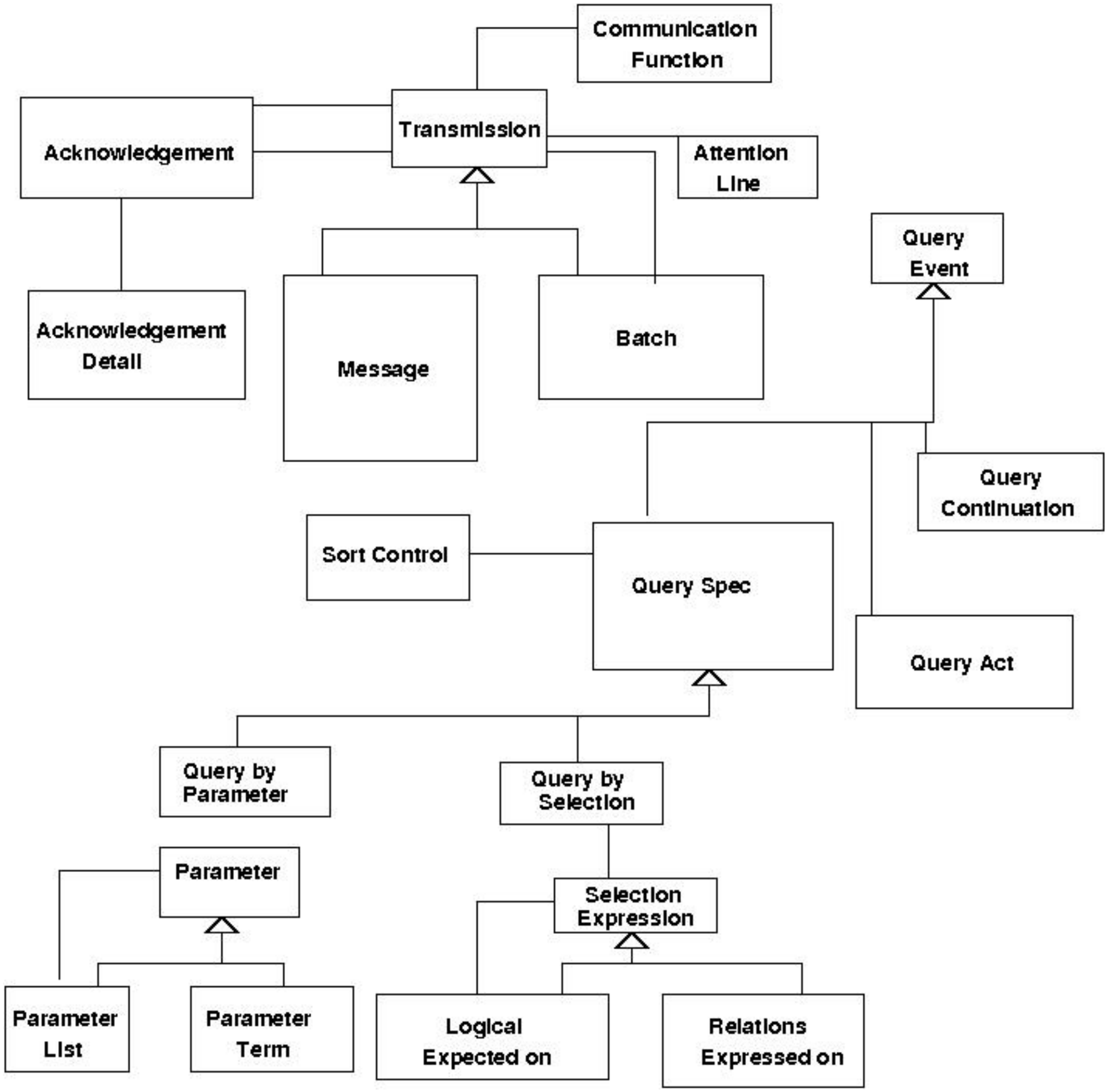
- Polymorphism
  - a reference-variable of type A can refer to an object of type B if B is a subclass of A

- Dynamic binding
  - the method invoked through a reference depends on the type of the object associated with the reference at runtime

Component Diagram



RIM: Message Control Classes


Communication Function
Transmission
Acknowledgement
Attention Line
Query Event
Acknowledgement Detail
Message
Batch
Query Continuation
Sort Control
Query Spec
Query Act
Query by Parameter
Query by Selection
Parameter
Selection Expression
Parameter List
Parameter Term
Logical Expected on
Relations Expressed on