COMP 3700: Software Modeling and Design

(Interaction Modeling)

Topics

- Interaction Modeling
- Use Case Models
- Sequence Models (Sequence and Collaboration Diagrams
- Activity Models
- Advanced Interaction Modeling

Interaction Modeling

- Describes how objects interact to produce useful results – holistic view of behavior across objects.
- Interactions can be modeled at different levels of abstraction.
  - Use cases describe how a system interacts with outside actors.
  - Sequence diagrams describe show the messages exchanged among a set of objects over time.
  - Activity diagrams show the flow of control among the steps of a computation – can show both control and data flow.

Use Case Model: Writing Requirements in Context

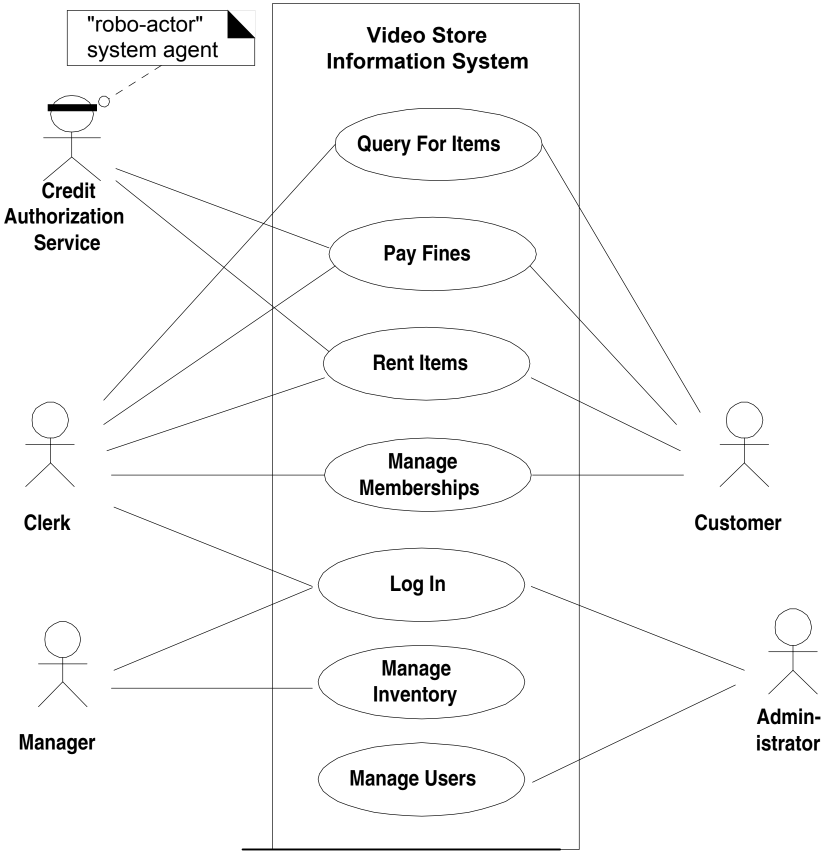
- A use case tells a story of actors using a system.
  - “Rent Videos”
  - *A use-case is a sequence of actions a system performs that yields an observable result of value to a particular actor.*
- One artifact to express (especially) *functional* requirements.
- Emphasizes thinking about the valuable objectives-oriented viewpoint of the users.

Identifying Use cases

- Major distinct, complete, end-to-end processes of using a system.
- Not usually one step, but a complete story.
- Examples
  - Rent Videos
  - Return Videos
  - Pay Fines

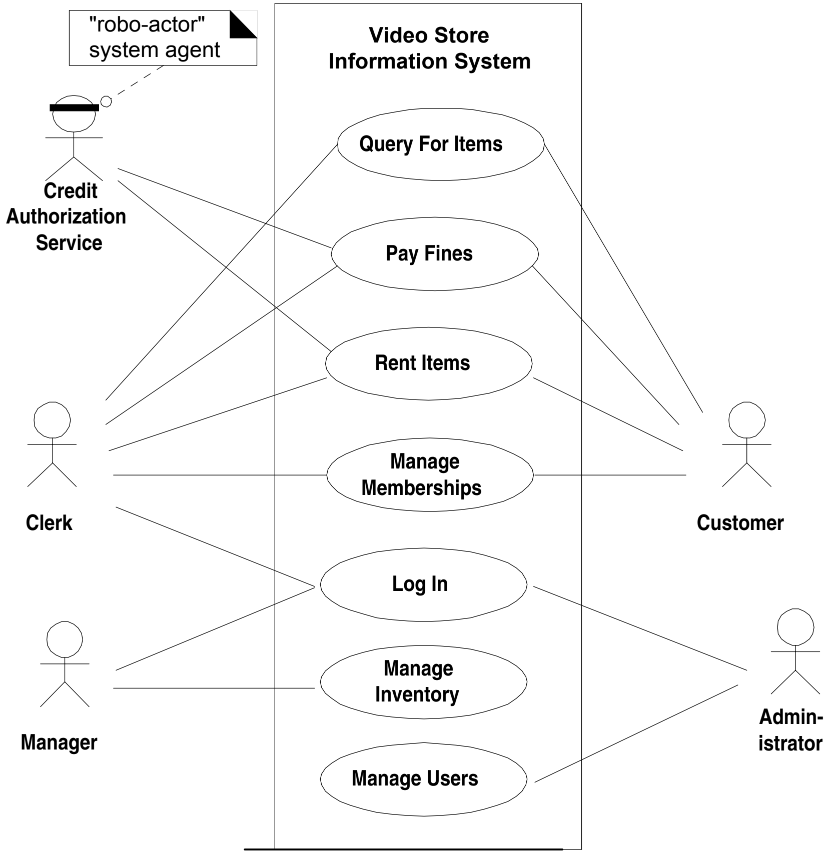
Use Case Diagram

- A way to conceive and illustrate the use cases.
- Usually created during the initial use case analysis.
- An actor is a direct external user of a system.
- A use case is a coherent piece of functionality that a system can provide by interacting with actors.

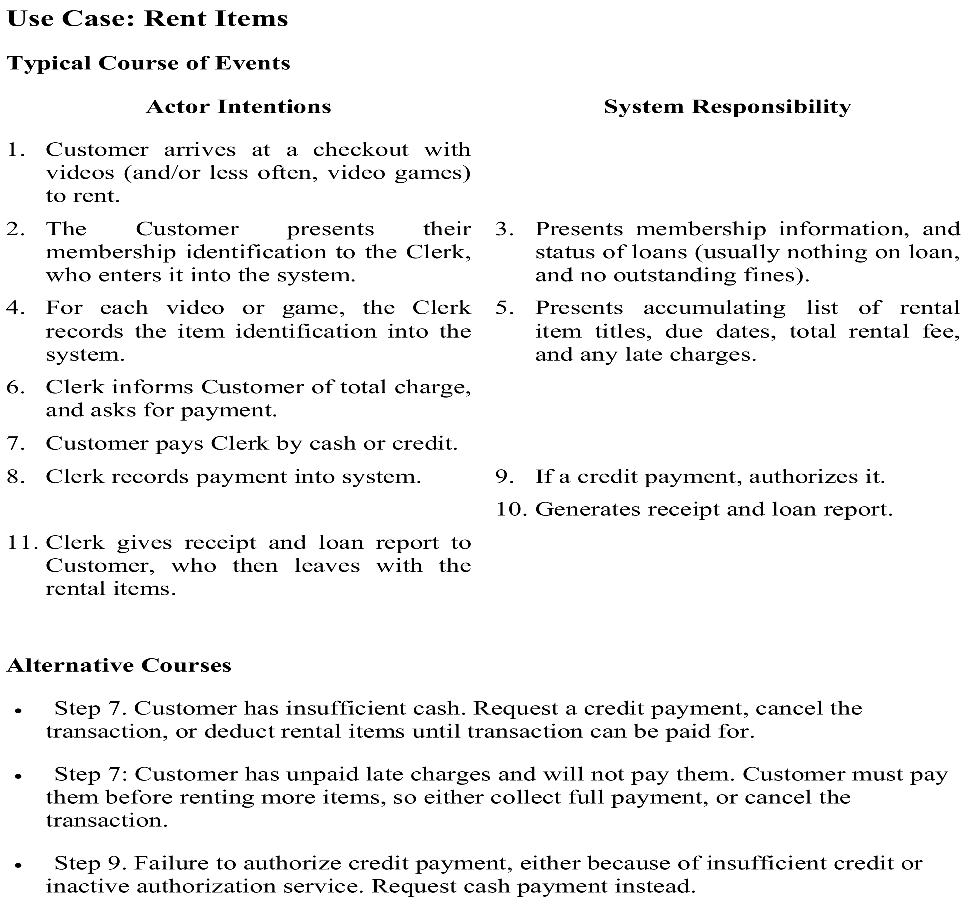


Use Case Diagram

- A way to conceive and illustrate the use cases.
- Usually created during the initial use case analysis.
- An actor is a direct external user of a system.
- A use case is a coherent piece of functionality that a system can provide by interacting with actors.



A Sample Detailed Use Case



Use Case: Rent Items

Typical Course of Events

Actor Intentions	System Responsibility
1. Customer arrives at a checkout with videos (and/or less often, video games) to rent.	
2. The Customer presents their membership identification to the Clerk, who enters it into the system.	3. Presents membership information, and status of loans (usually nothing on loan, and no outstanding fines).
4. For each video or game, the Clerk records the item identification into the system.	5. Presents accumulating list of rental item titles, due dates, total rental fee, and any late charges.
6. Clerk informs Customer of total charge, and asks for payment.	
7. Customer pays Clerk by cash or credit.	
8. Clerk records payment into system.	9. If a credit payment, authorizes it.
	10. Generates receipt and loan report.
11. Clerk gives receipt and loan report to Customer, who then leaves with the rental items.	

Alternative Courses

- Step 7. Customer has insufficient cash. Request a credit payment, cancel the transaction, or deduct rental items until transaction can be paid for.
- Step 7: Customer has unpaid late charges and will not pay them. Customer must pay them before renting more items, so either collect full payment, or cancel the transaction.
- Step 9. Failure to authorize credit payment, either because of insufficient credit or inactive authorization service. Request cash payment instead.

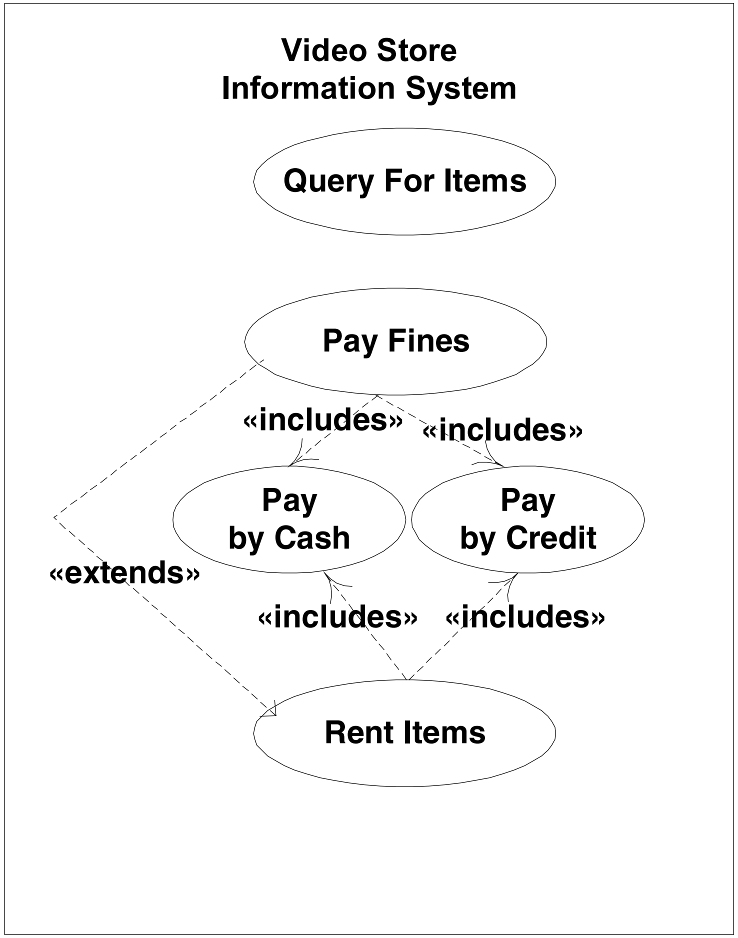
Essential vs. Concrete Use Cases

- *Essential* use cases defer the details of the UI, and focus on the *intentions* of the actors, and responsibilities of the system.
  - Concrete (AKA Real) do not.
- Essential: "The AccountHolder identifies themselves to the ATM"
- Real: "The AccountHolder inserts their card in the reader. Window A is displayed. They enter their PIN on the numeric keypad, …"
- As we move from analysis to design, we are more inclined to move from essential to concrete use case descriptions.

Relating Use Cases

- When creating the use case diagram, it can be useful (in terms of comprehension and simplification) to:
  - factor out shared sub-processes
    - use the <<includes>> relationship
  - show precedence order
    - use the <<extends>> relationship

Relating Use Cases



Guidelines for Use Case Models

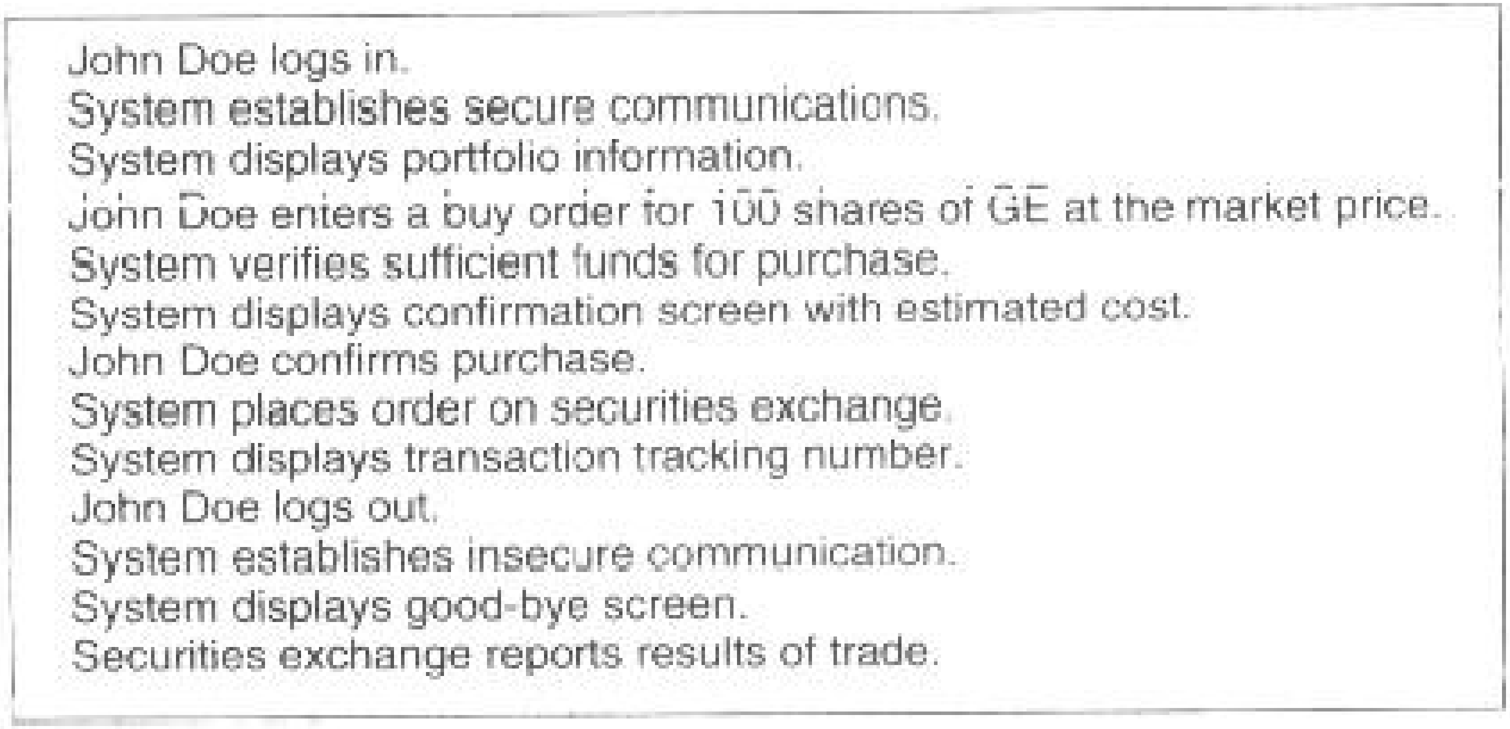
- First determine the system boundary.
- Ensure that actors are focused.
- Each use case must provide value to users.
- Relate use cases and actors.
- Remember use cases are informal.
- Use cases can be structured.

Sequence Models

- The sequence model elaborates the themes of use cases.
- Two types of sequence models:
  - A scenario is a sequence of events that occurs during one particular execution of a system, such as for a use case.
  - An interaction diagram shows the participants in an interaction and the sequences of messages among them.
    - Sequence diagram uses a fence format.
    - Collaboration diagrams use the graph format.

Scenarios

- Scenario for a session with an online stock broker.



John Doe logs in.
System establishes secure communications.
System displays portfolio information.
John Doe enters a buy order for 100 shares of GE at the market price.
System verifies sufficient funds for purchase.
System displays confirmation screen with estimated cost.
John Doe confirms purchase.
System places order on securities exchange.
System displays transaction tracking number.
John Doe logs out.
System establishes insecure communication.
System displays good-bye screen.
Securities exchange reports results of trade.

Sequence Diagrams

- Each use case requires one or more sequence diagrams to describe its behavior

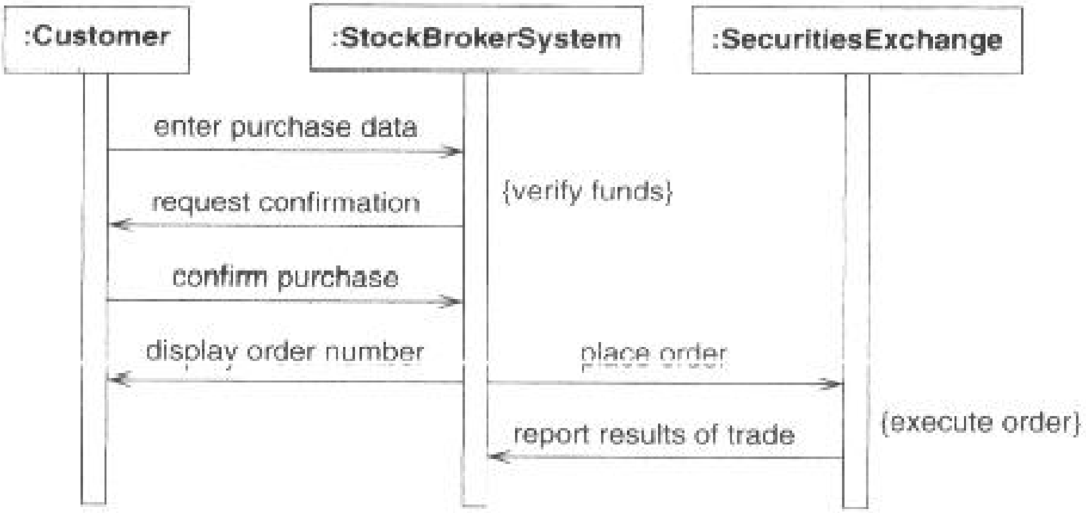


Figure 7.6 Sequence diagram for a stock purchase. Sequence diagrams can show large-scale interactions as well as smaller, constituent tasks.

Sequence Diagrams

- Each use case requires one or more sequence diagrams to describe its behavior

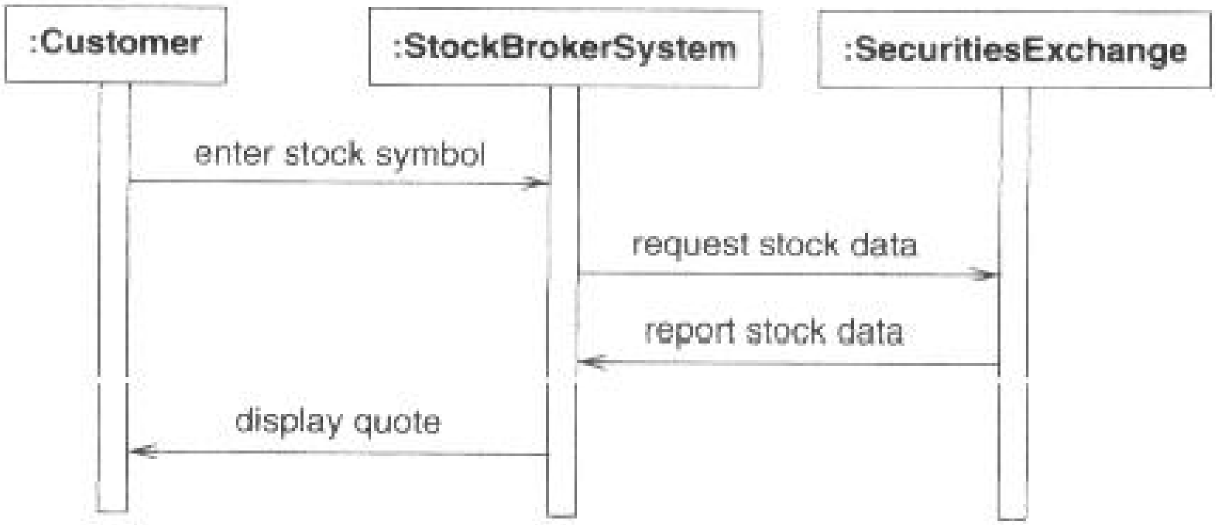


Figure 7.7 Sequence diagram for a stock quote

Sequence Diagrams

- Each use case requires one or more sequence diagrams to describe its behavior

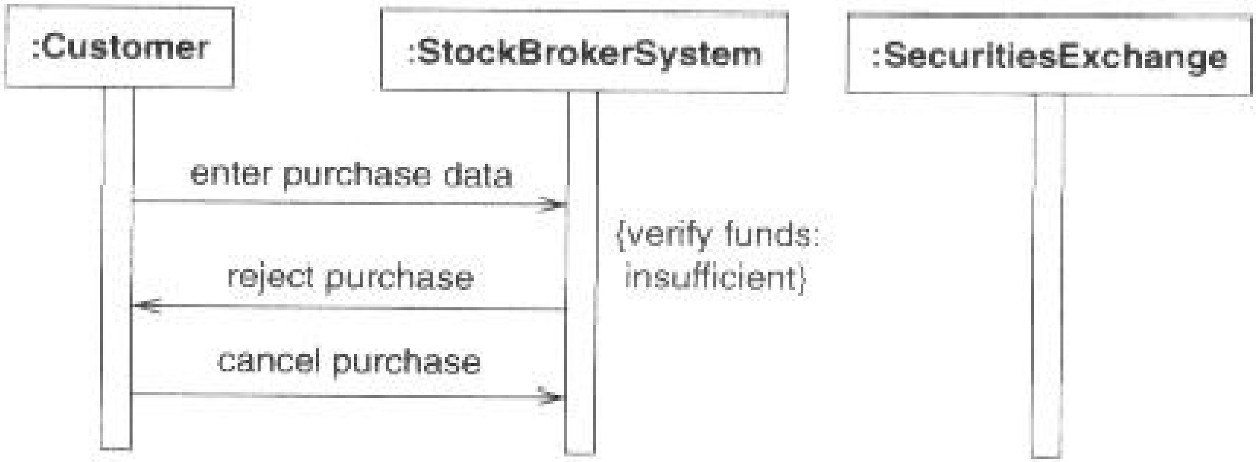


Figure 7.8 Sequence diagram for a stock purchase that fails

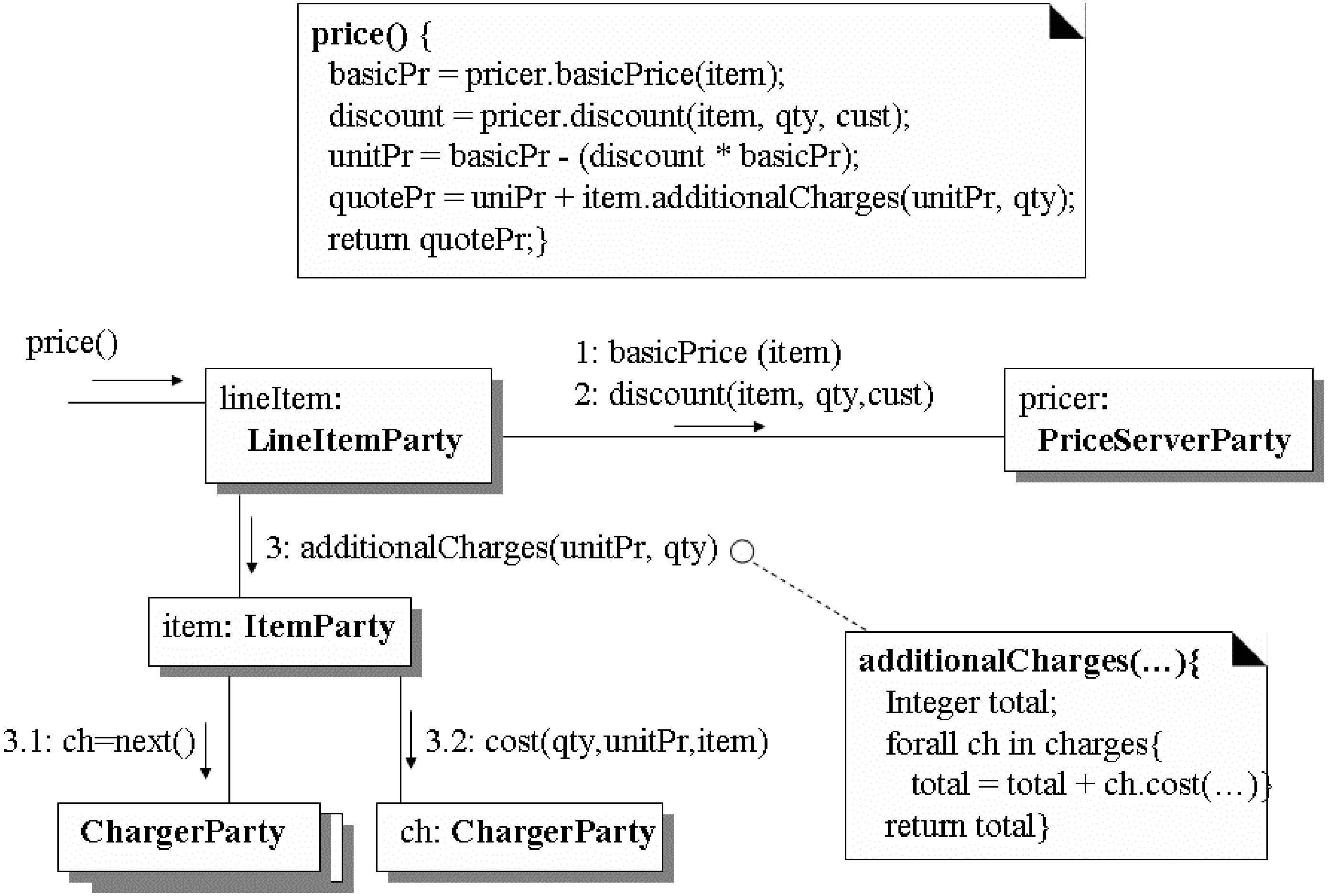
Sequence Diagram Notation

- Object lifetimes
- Illustrating reply and returns
- Loops
- Conditional messages
- Alternative
- Iteration over a collection

Guidelines for Sequence Diagrams

- Prepare at least one scenario for one use case
- Abstract the scenarios into sequence diagrams
- Divide complex interactions
- Prepare a sequence diagram for each error condition

Collaboration Diagrams



```
price() {
  basicPr = pricer.basicPrice(item);
  discount = pricer.discount(item, qty, cust);
  unitPr = basicPr - (discount * basicPr);
  quotePr = uniPr + item.additionalCharges(unitPr, qty);
  return quotePr;}
```

price()

lineItem: **LineItemParty**

1: basicPrice (item)

2: discount(item, qty,cust)

pricer: **PriceServerParty**

3: additionalCharges(unitPr, qty)

item: **ItemParty**

3.1: ch=next()

3.2: cost(qty,unitPr,item)

ChargerParty

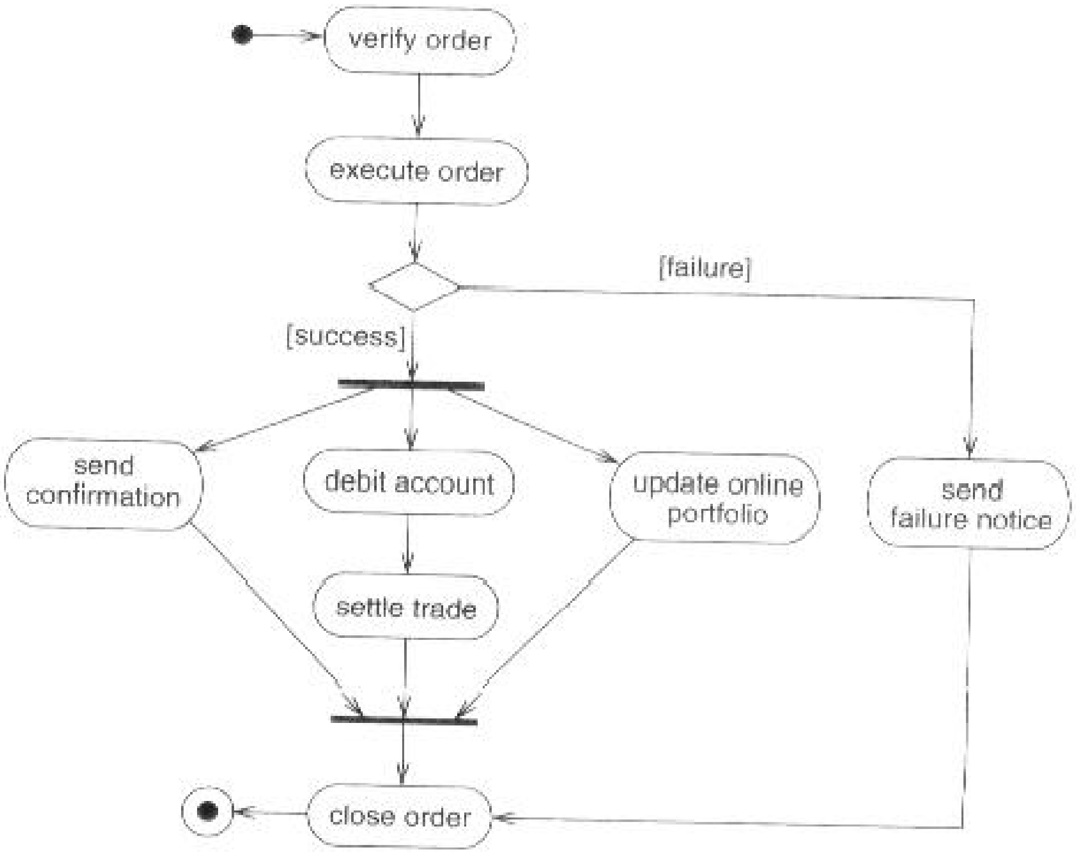
ch: **ChargerParty**

```
additionalCharges(…){
  Integer total;
  forall ch in charges{
    total = total + ch.cost(…)}
  return total}
```

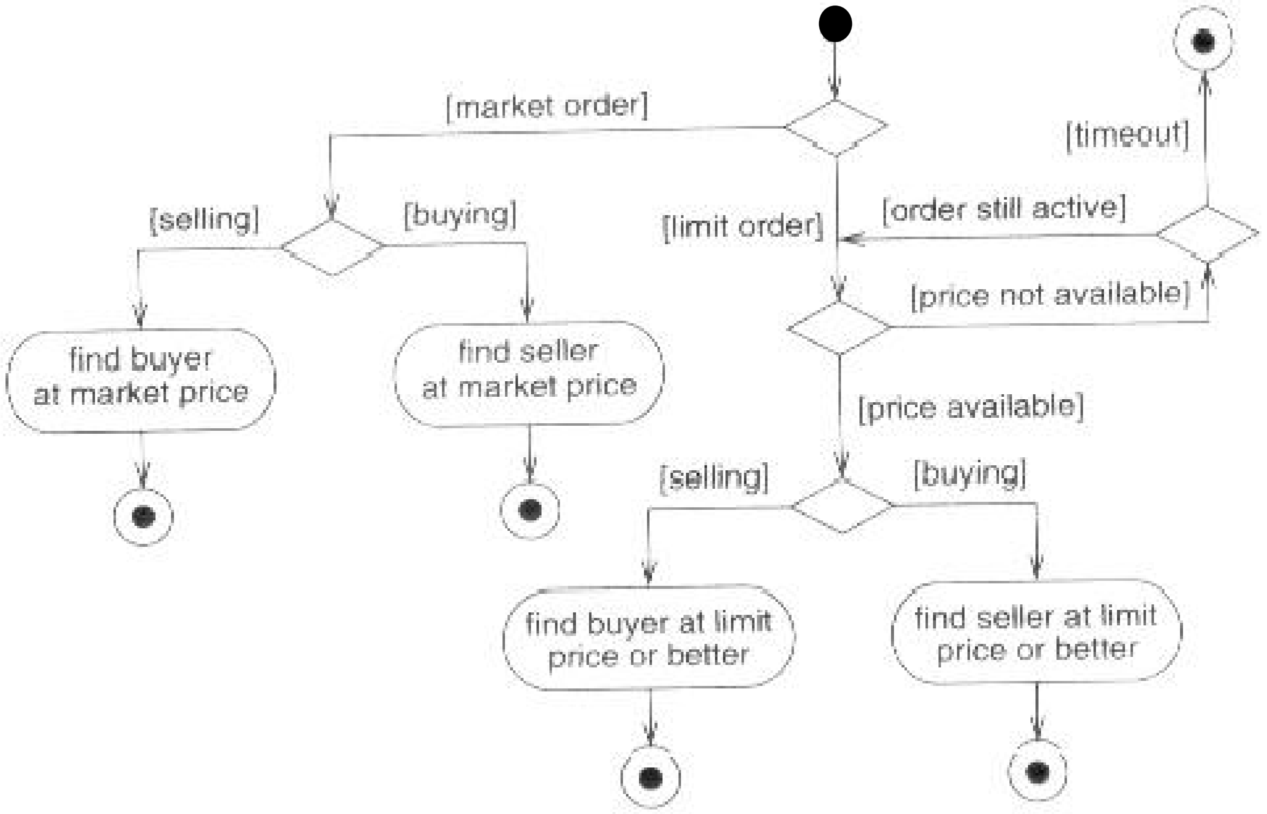
Activity Models

- Activity diagrams show the sequence of steps that make up of a complex process.
- Most useful during the early stages of designing algorithms and workflows.
- Can show both sequential and concurrent flow of control.
- Activities: The steps of an activity diagram are operations, specifically activities from the state model.

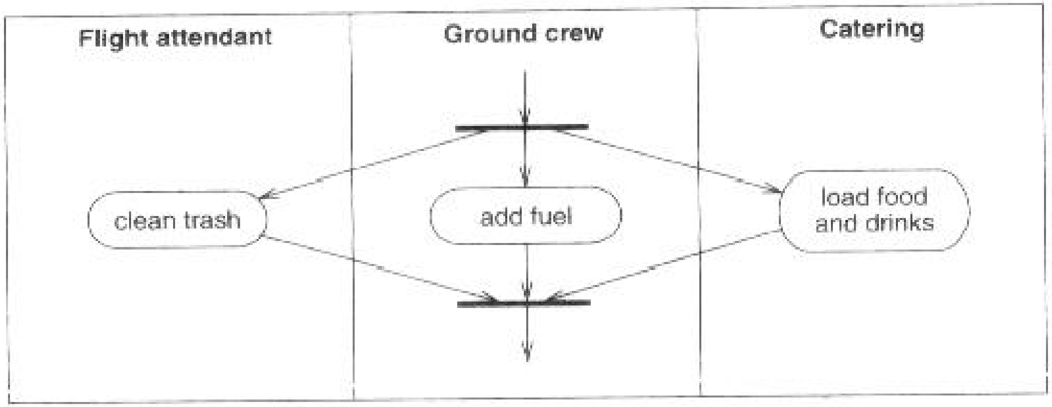
Activity Diagram (Top-level)



Activity Diagram (Execute Order)

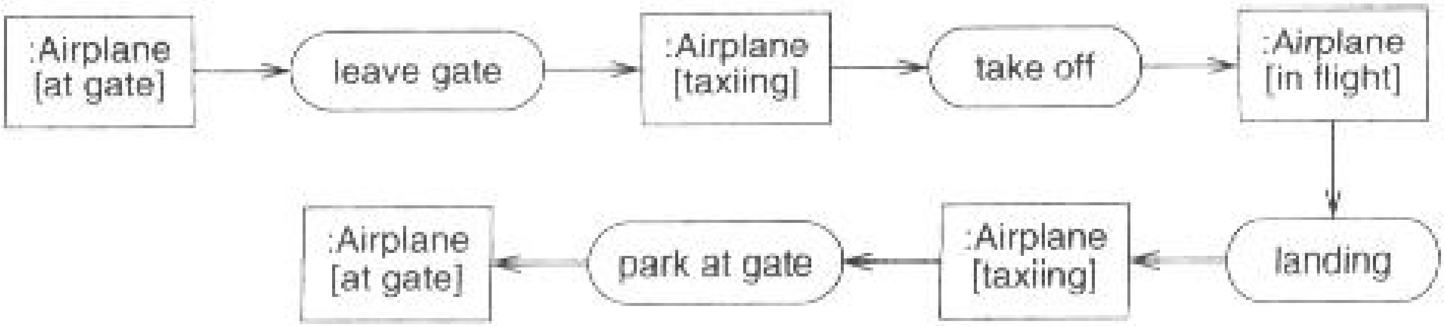


Activity Diagram (Swimlanes)



- It is often useful to know which organization is responsible for an activity.
- Lines across swimlane boundaries indicate interactions among different organizations.

Activity Diagram (Objectflows)



- It is often helpful to see relations between an operation and the objects that are its arguments.
- Activity diagram can show inputs to or outputs from the activities.