Interaction Modeling

COMP 3700.002
Software Modeling and Design

Shehenaz Shaik

OO Models

- Class Model
  - Static structure of objects and relationships
  - Class diagram
- State Model
  - Changes over time or on events
  - State diagram
- Interaction Model
  - Interaction among objects
  - Use case diagram
  - Sequence diagram
  - Activity diagram

Interaction Model

- What it is?
  - Interaction model describes how objects interact to produce useful results
- Holistic view of behavior across objects
  - State model – Reductionist view (each object)
- Various levels of abstraction
  - Use cases
  - Sequence diagrams
  - Activity diagrams

Actor

- Direct external user of system
  - Communicates directly with system
    - Not indirectly
  - Not part of the system
- Represents objects having specific behavior towards system
- Has a single well-defined purpose
- Object with different facets of behavior
  - Multiple actors
- Objects of different classes that interact similarly
  - Single actor

Interaction Model

- Use cases
- Sequence diagrams
- Activity diagrams

Use case

- Coherent piece of functionality that a system can provide by interacting with actors
- Includes all relevant behavior
  - Normal mainline behavior
  - Variations on normal behavior
  - Exception conditions
  - Error conditions
  - Cancellation requests
- Involves system and actors
- Sequence of messages between system and actors

Use Case Summary (E.g.)

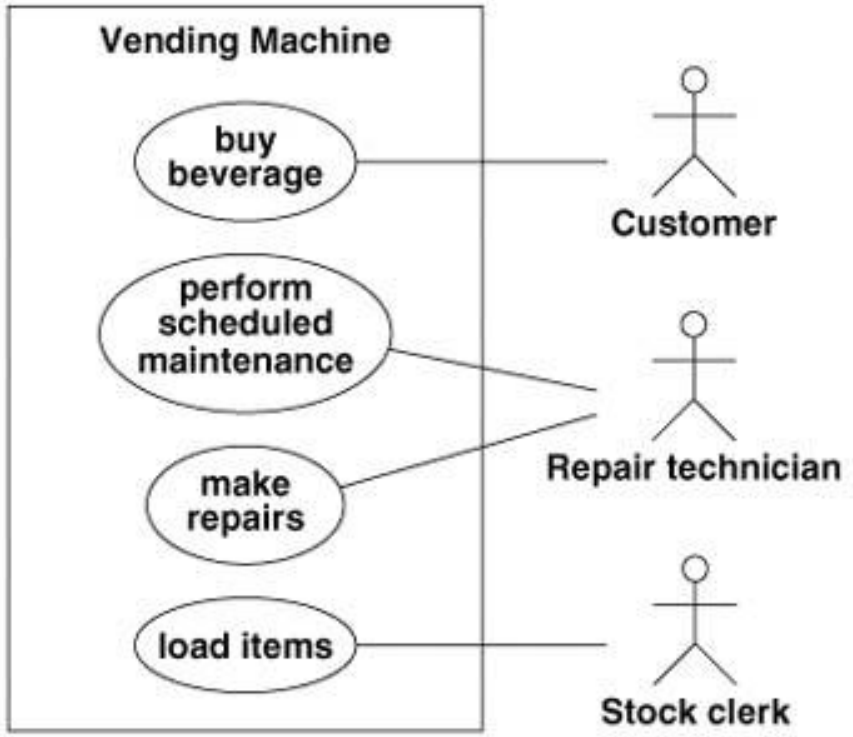
- **Buy a beverage**. The vending machine delivers a beverage after a customer selects and pays for it.
- **Perform scheduled maintenance**. A repair technician performs the periodic service on the vending machine necessary to keep it in good working condition.
- **Make repairs**. A repair technician performs the unexpected service on the vending machine necessary to repair a problem in its operation.
- **Load items**. A stock clerk adds items into the vending machine to replenish its stock of beverages.

Use Case Diagram

- Use case
  - Slice of functionality
- Set of Use cases
  - Complete functionality provided by system
- Actor
  - One kind of object, system can perform behavior
- Set of Actors
  - Complete set of objects that the system can serve

Use Case Diagram: UML Notation

- Rectangle
- Use case
  - Ellipse
  - Listed inside
- Actor
  - Stick man
  - Listed outside
- Lines connect use cases and actors
  - Many – to – Many



Use Case Description (E.g.)

Use Case: Buy a beverage

Summary: The vending machine delivers a beverage after a customer selects and pays for it.

Actors: Customer

Preconditions: The machine is waiting for money to be inserted.

Description: The machine starts in the waiting state in which it displays the message "Enter coins." A customer inserts coins into the machine. The machine displays the total value of money entered and lights up the buttons for the items that can be purchased for the money inserted. The customer pushes a button. The machine dispenses the corresponding item and makes change, if the cost of the item is less than the money inserted.

Exceptions:

Canceled: If the customer presses the cancel button before an item has been selected, the customer's money is returned and the machine resets to the waiting state.

Out of stock: If the customer presses a button for an out-of-stock item, the message "That item is out of stock" is displayed. The machine continues to accept coins or a selection.

Insufficient money: If the customer presses a button for an item that costs more than the money inserted, the message "You must insert $*nn.nn* more for that item" is displayed, where *nn.nn* is the amount of additional money needed. The machine continues to accept coins or a selection.

No change: If the customer has inserted enough money to buy the item but the machine cannot make the correct change, the message "Cannot make correct change" is displayed and the machine continues to accept coins or a selection.

Postconditions: The machine is waiting for money to be inserted.

Use Cases Vs. Traditional Requirements Lists (TRL)

- Use cases organize system functionality from user perspective
  - TRLs may be vague to users
  - TRLs may overlook supporting functionality
    - E.g. Initialization / Termination
  - Use cases list complete transactions and less likely to omit details
- TRLs necessity
  - Global constraints and other non-localized functionality

Use Case Relationships

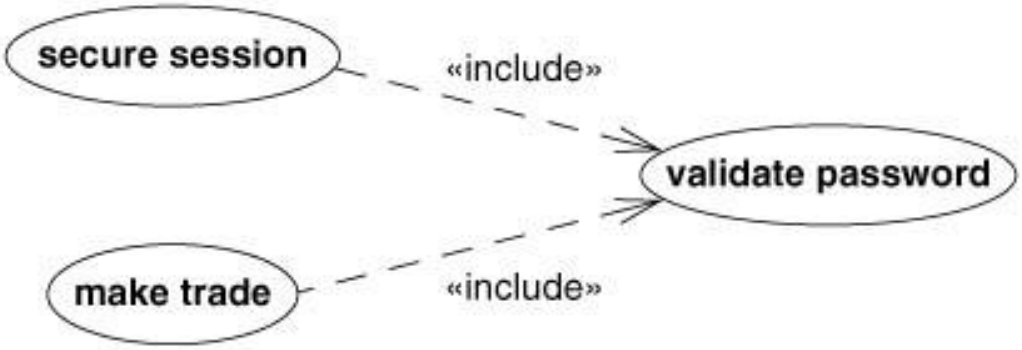
- Facilitates building complex use cases
  - Include relationship
  - Extend relationship
  - Generalization relationship

Include Relationship

- Incorporates one use case within behavior sequence of another use case
  - Inserted at a specific location within behavior sequence
  - Included use case may(/not) be usable on its own
- Notation:
  - Include use-case-name

Include Relationship: UML Notation

- Dashed arrow
  - from source (including) use case
  - to target (included) use case
  - <<include>> keyword



Extend Relationship

- Adds incremental behavior to use case
- Base use case
  - Valid use case
- Extension use case
  - Fragment
  - Cannot appear alone as a behavior sequence

Extend Relationship: UML Notation

- Dashed arrow
  - from extension use case
  - to base use case
  - <<extend>> keyword

Generalization

- Shows specific variations on general use case
  - Child use case
    - Specializes parent by inserting additional steps or by refining steps
  - Parent use case
    - General behavior sequence

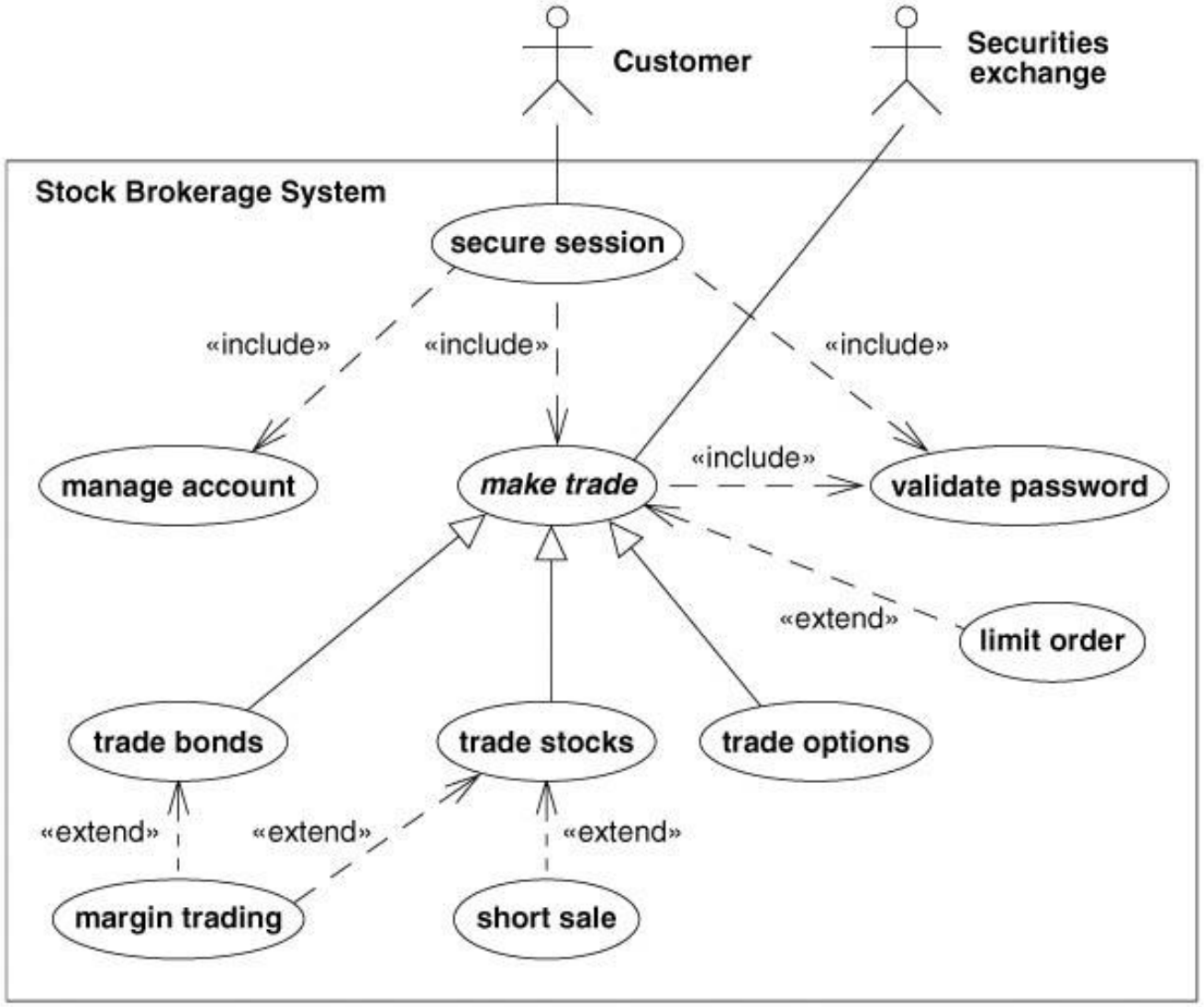
Use case Generalization Vs. Class Generalization

- Similarities
  - Parent may be abstract or concrete
  - Abstract parent cannot be used directly
  - Abstract parent recommended
  - Polymorphism supported
- Differences
  - Use case generalization more complicated
  - Order of subclass attributes not important
    - Child use case adds behavior steps; May be at multiple points; Must appear in proper locations within behavior sequence of parent
  - Multiple inheritance is not supported

Generalization: UML Notation

- Arrow
  - Tail on child use case
  - Triangular arrowhead on parent use case

Use Case Relationships: Combinations



Use Case Relationships: Guidelines

- Use case generalization
  - If use case has several variations
    - Model common behavior as abstract use case
    - Specialize each of the variations
- Use case inclusion
  - If use case includes well-defined behavior fragment, define a use case for it & include
  - Meaningful activity, but not an end in itself
    - E.g. password validation

Use Case Relationships: Guidelines

- Use case extension
  - If use case with optional features
    - Model base behavior as use case
    - Add features with extend relationship
  - Permits system to be tested / debugged without extensions

Use Case Model: Guidelines

- First determine system boundary
- Ensure that actors are focused
  - Single, coherent purpose
  - Multiple purposes → Multiple actors
- Each use case must provide value to users
- Relate use cases and actors
  - At least one actor per use case
  - At least one use case per actor
- Use cases may be distinct or structured
- Use cases are informal

Interaction Model

- Use cases
- Sequence diagrams
- Activity diagrams

Sequence Model

- Elaborates themes of use cases
  - Adds detail
- Two kinds
  - Scenario
    - Unstructured format
  - Sequence Diagram
    - Structured format

Scenario

- Sequence of events that occurs during one particular execution of a system, such as for a use case
- List of text statements containing
  - Messages between objects
  - Activities performed by objects
- Steps of writing a scenario
  - Identify the objects exchanging messages
  - Determine sender & receiver of each message
  - Determine sequence of messages
  - Add activities for internal computations

Scenario (E.g. A session)

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.

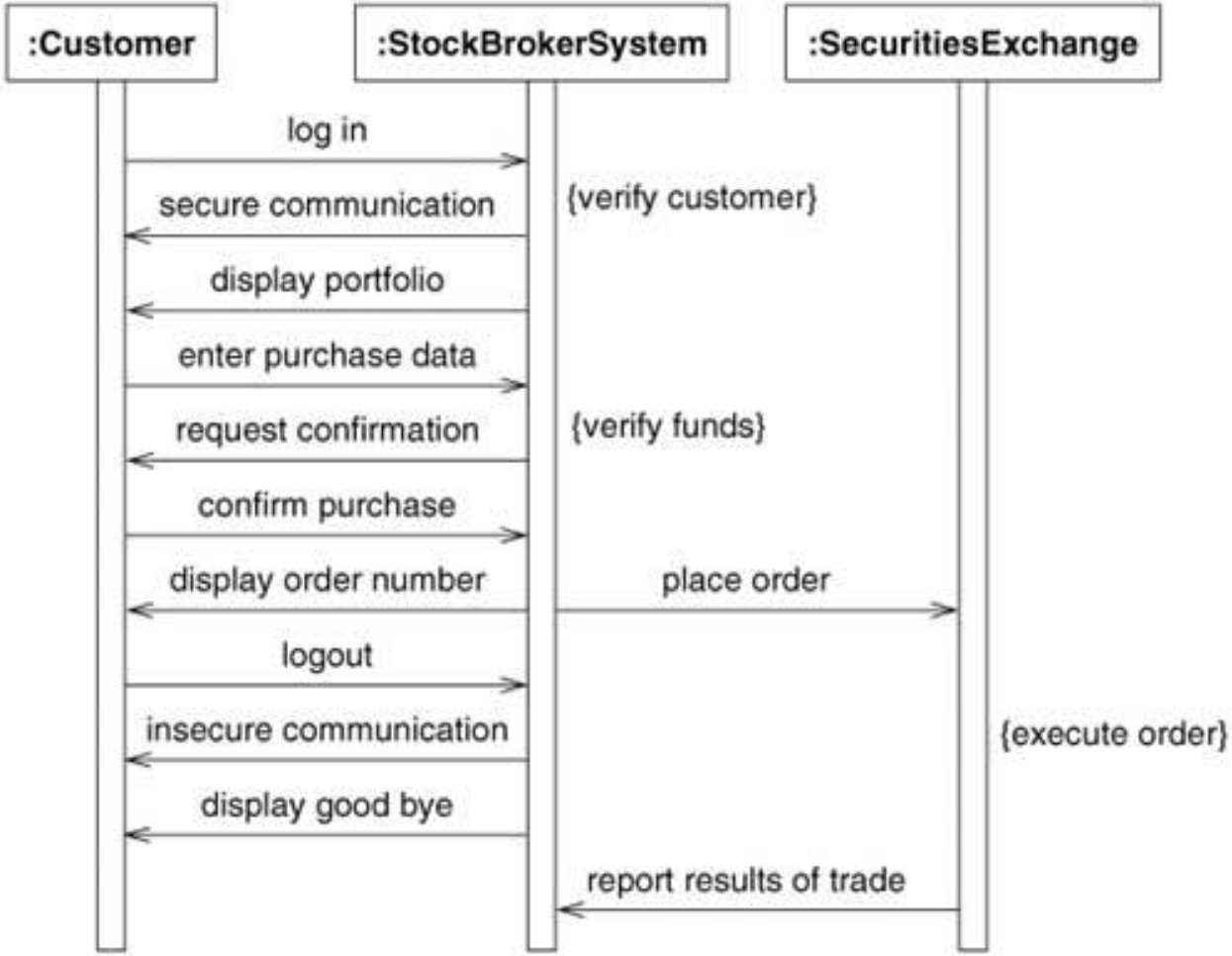
Sequence Diagram

- Shows participants in an interaction and sequence of messages among them
- Shows a particular behavior sequence of use case
- Use case
  - Multiple sequence diagrams to describe behavior
  - One sequence diagram for each major flow of control

Sequence Diagram (Contd.)

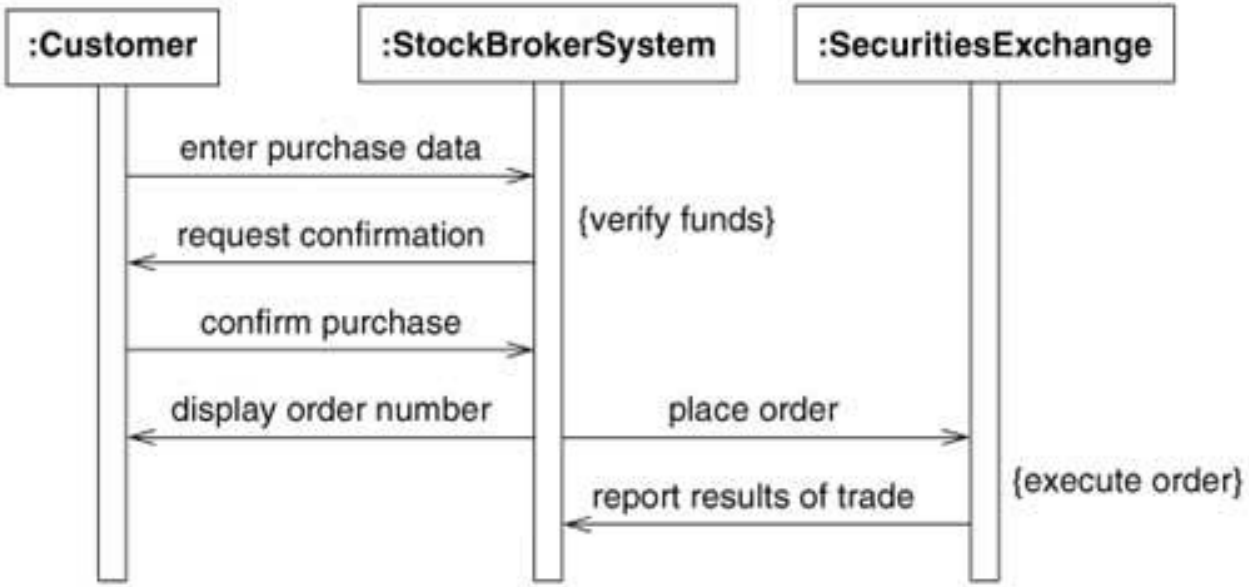
- UML Notation
  - Actor / System: Vertical line (Lifeline)
  - Message: Horizontal arrow from sender to receiver
  - Time proceeds from top to bottom
- Show concurrent signals
- Signals between participants need not alternate

Sequence Diagram (E.g. Session with an online stock broker)



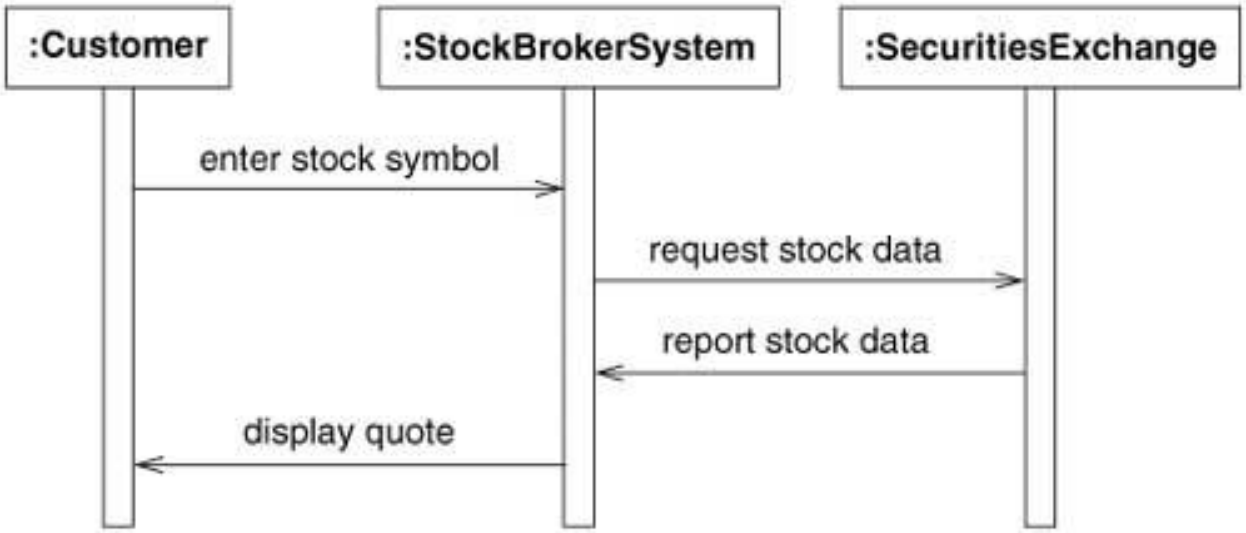
Sequence Diagram (E.g. Stock purchase)

- Separate sequence diagram for each task



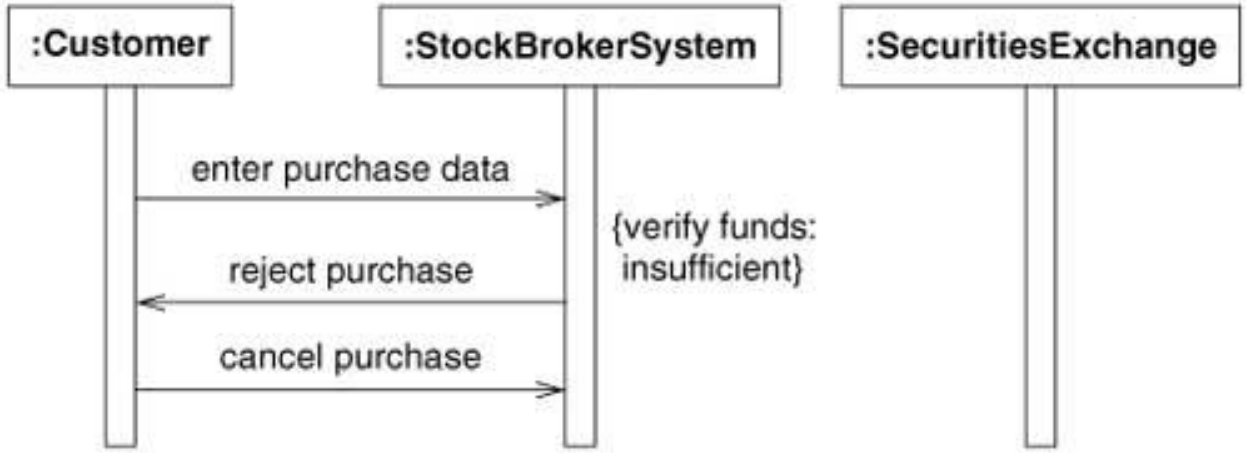
Sequence Diagram (E.g. Stock quote)

- Separate sequence diagram for each task



Sequence Diagram (E.g. Stock purchase that fails)

- Separate sequence diagram for each exception condition within use case



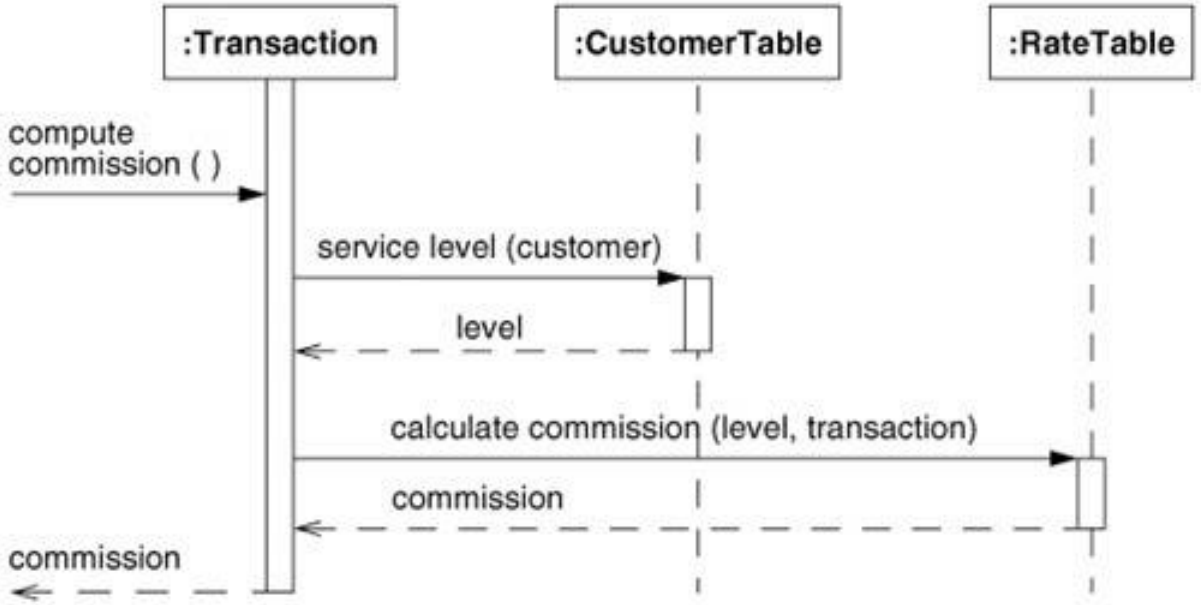
Sequence Diagrams with Passive Objects

- Implementation may limit number of active objects at an instant
- Active object
  - Always active
  - Has own focus of control
- Passive object
  - Lack their own thread of control
  - Not activated until it has been called
  - Operation execution completes
    - → Control returns to caller
    - → Passive object becomes inactive

Sequence Diagrams with Passive Objects: UML Notation

- Active period
  - Thin rectangle (Activation / Focus of control)
- Inactive period
  - Dashed line (Lifeline)
- Call
  - Line with thick arrowhead
- Return
  - Dashed line with thick arrowhead

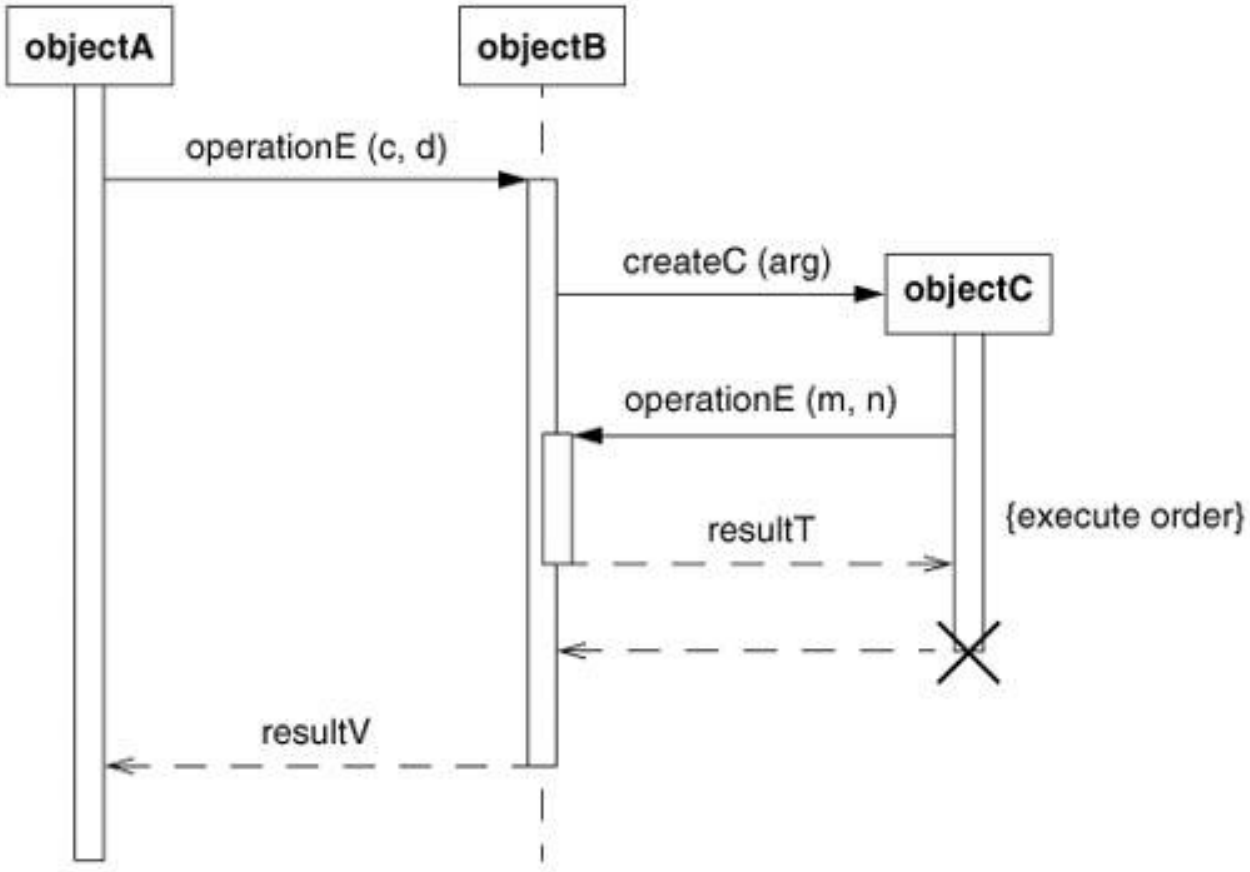
Sequence Diagrams with Passive Objects (E.g.)



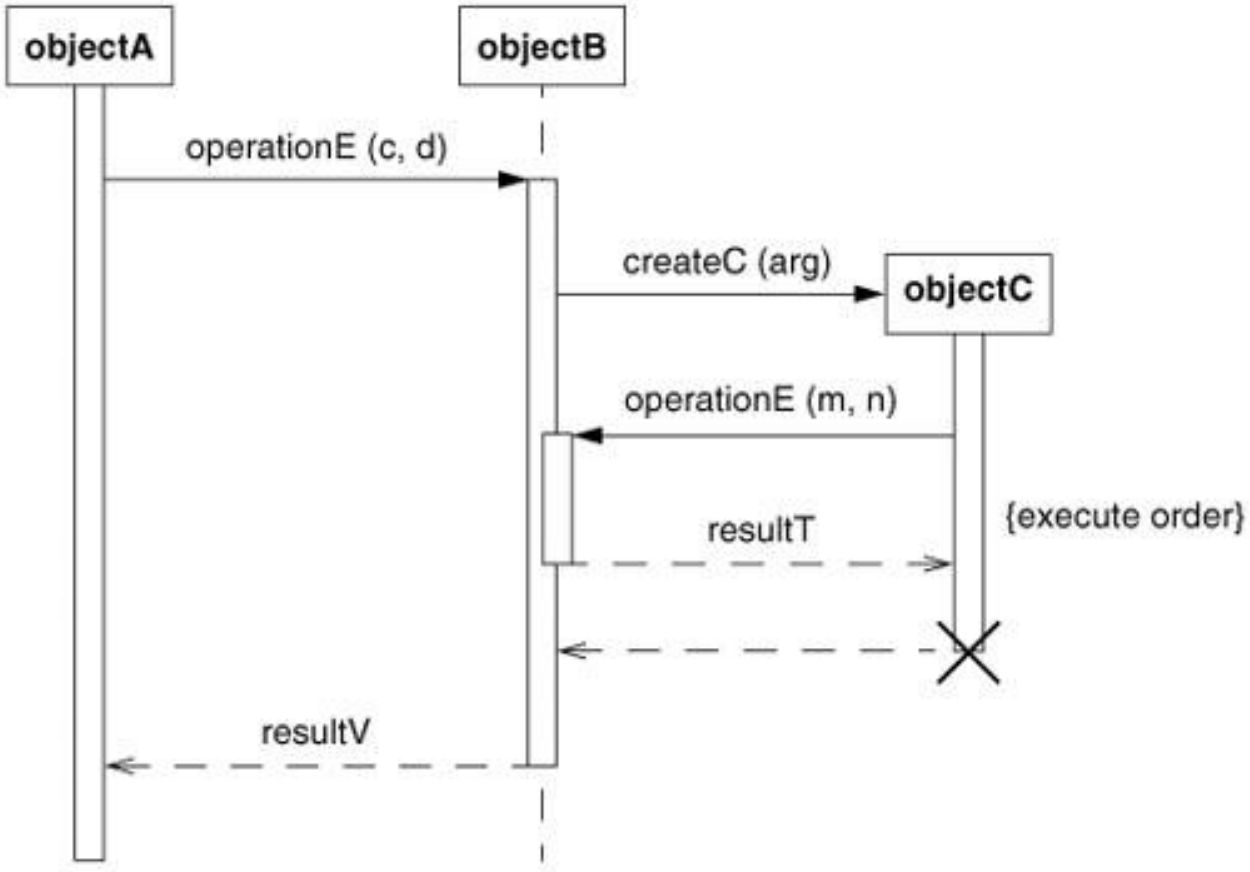
Sequence Diagrams with Transient Objects

- Object created and destroyed
  - Limited lifespan
- UML Notation
  - Object Creation: Name at arrowhead of call
  - Object Deletion: X at end of lifeline

Sequence Diagrams with Transient Objects (E.g.)



Sequence Diagrams with Recursive Call (E.g.)



Sequence Model: Guidelines

- Prepare at least one scenario per use case
  - One scenario per mainline interaction
- Abstract scenarios into sequence diagrams
  - Separate contribution of each actor
  - Helps organizing behavior about objects
- Divide complex interactions into tasks
  - One sequence diagram per task
- Prepare a sequence diagram for each error condition
  - Show system response
- Differentiate active and passive objects
  - Optional implementation detail; Can be ignored

Interaction Model

- Use cases
- Sequence diagrams
- Activity diagrams

Activity Model

- Activity Diagram
- Shows sequence of steps that make up a complex process
  - E.g. algorithm or workflow
- Shows flow of control
  - Focuses on operations rather than on objects
  - Both sequential and concurrent

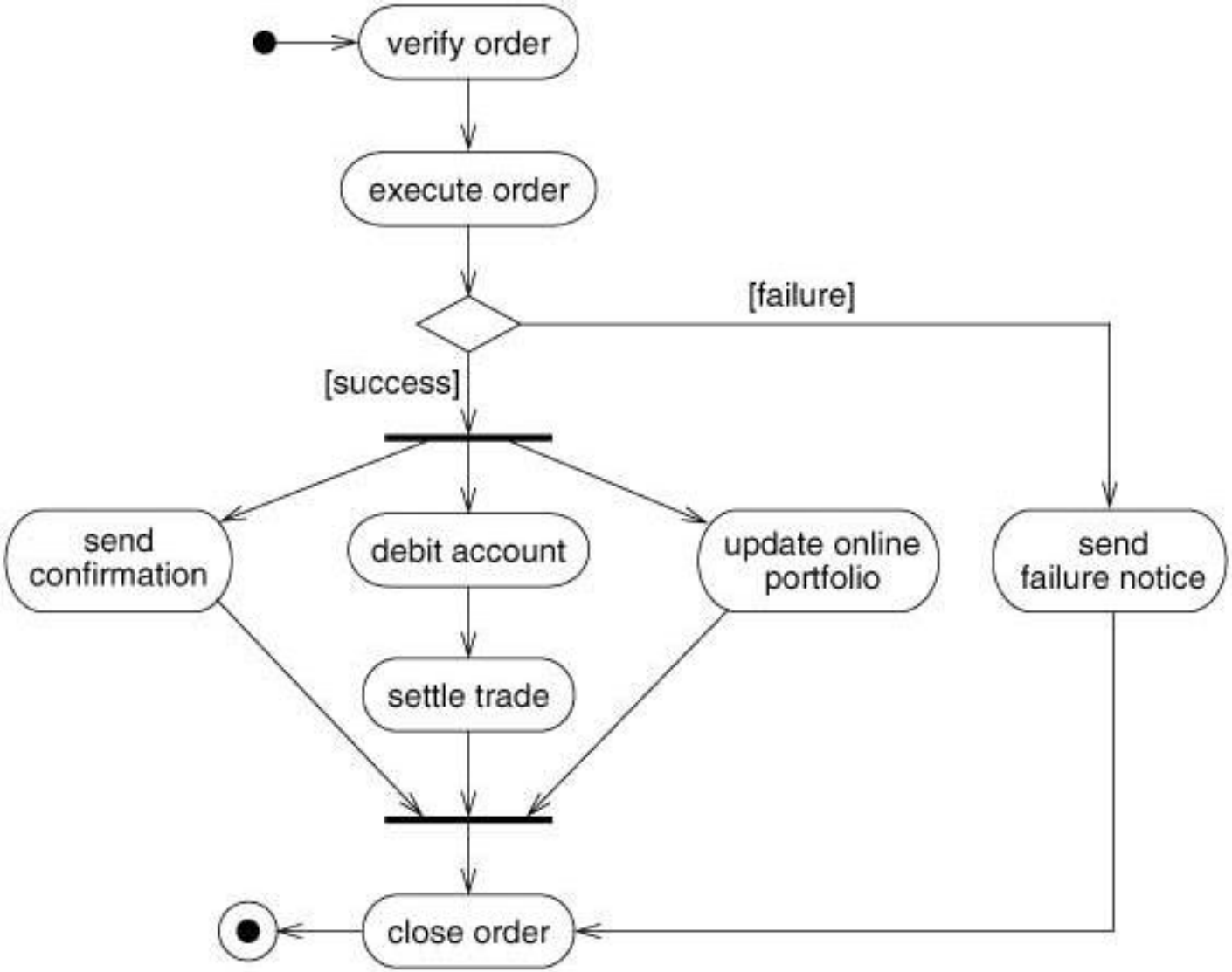
Activities

- Activity
  - Each step on activity diagram is an operation
  - Same as 'Activity' from state model
  - May be decomposed into finer activities
- Completion of activity is completion event
  - Denoted by 'Unlabeled arrow'
  - Next activity can be started

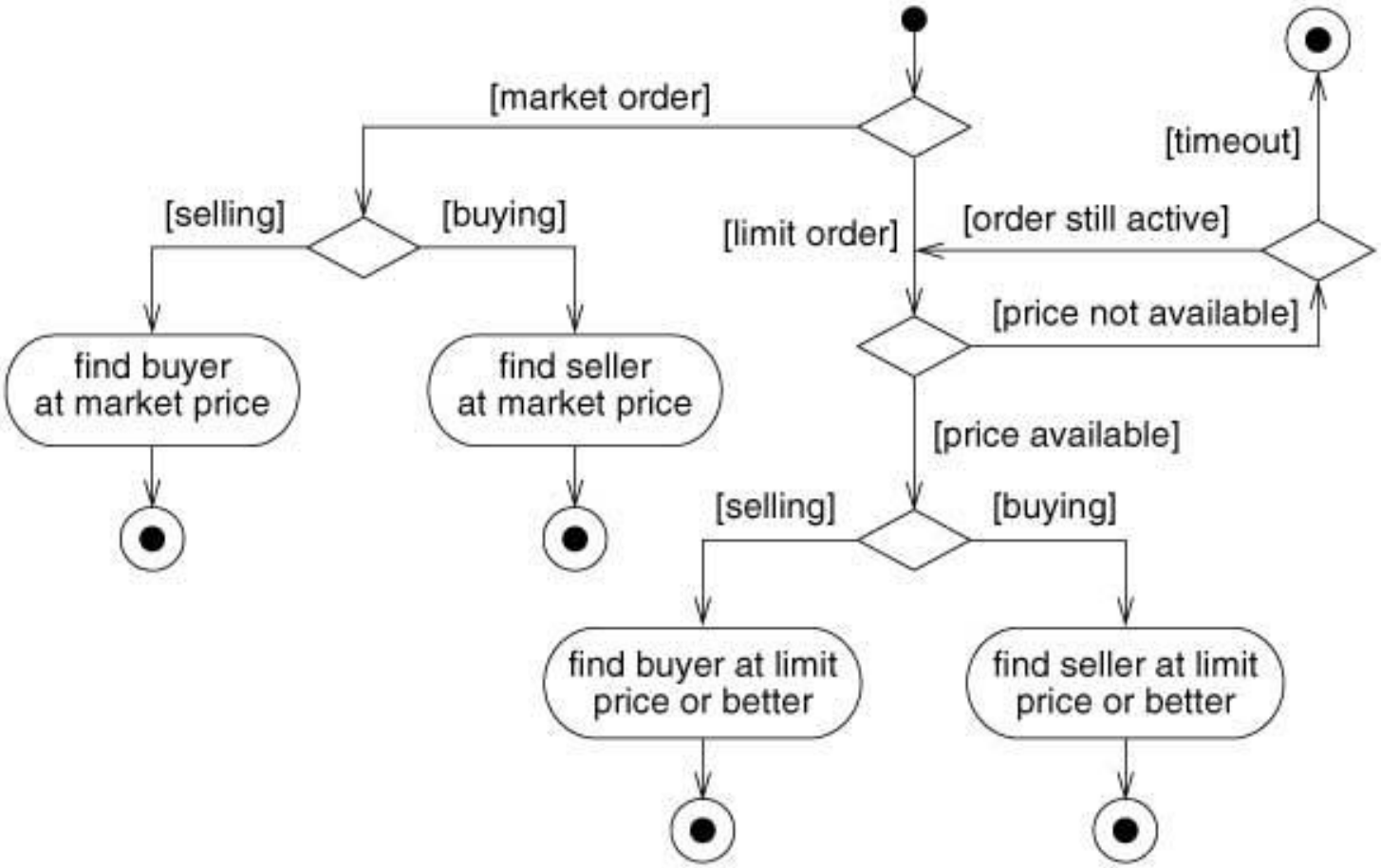
Activity Model: UML Notation

- Activity Diagram
  - Elongated Oval: Activities
  - Arrows: Sequencing
  - Diamond: Decision point
  - Heavy bar: Splitting / Merging of concurrent threads
  - Solid circle with an outgoing arrow: Initiation / Starting point
  - Bull's eye with only incoming arrows: Termination point of activity diagram

Activity Diagram (E.g. Stock trade processing)



Activity Diagram (E.g. Execute order)



Activity Diagram: Branches

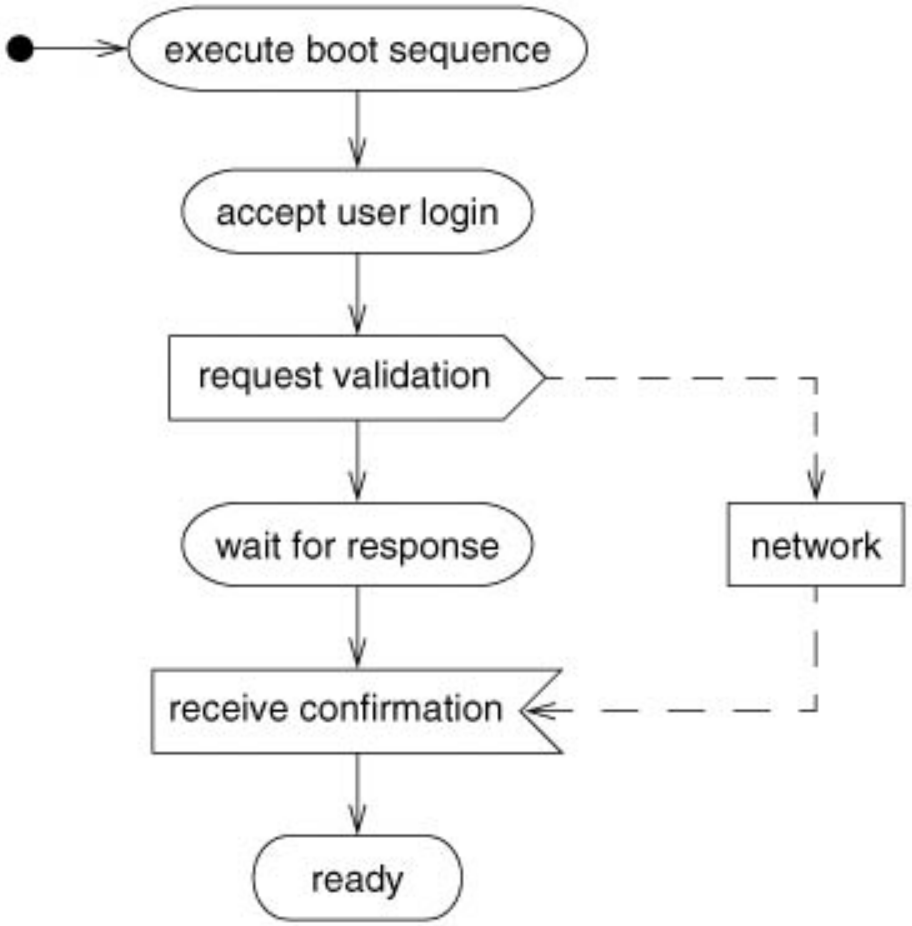
- If more than one successor to activity, then each arrow is labeled with a condition
  - If one condition is satisfied
    - Its arrow indicates next activity to perform
  - If none satisfied
    - Diagram is badly formed
    - System will hang until interrupted
    - Use 'Else' condition
  - If multiple conditions are satisfied
    - Only one successor activity executes
    - Nondeterministic selection

Executable Activity Diagrams

- Show progression of control during execution
- Activity Token
  - If executing, placed on corresponding activity symbol
  - If completed, placed on outgoing arrow
  - If condition, placed on the successful branch
  - If concurrency,
    - Multiple tokens
    - Split of control → Increase in token count
    - Merge of control → Decrease in token count

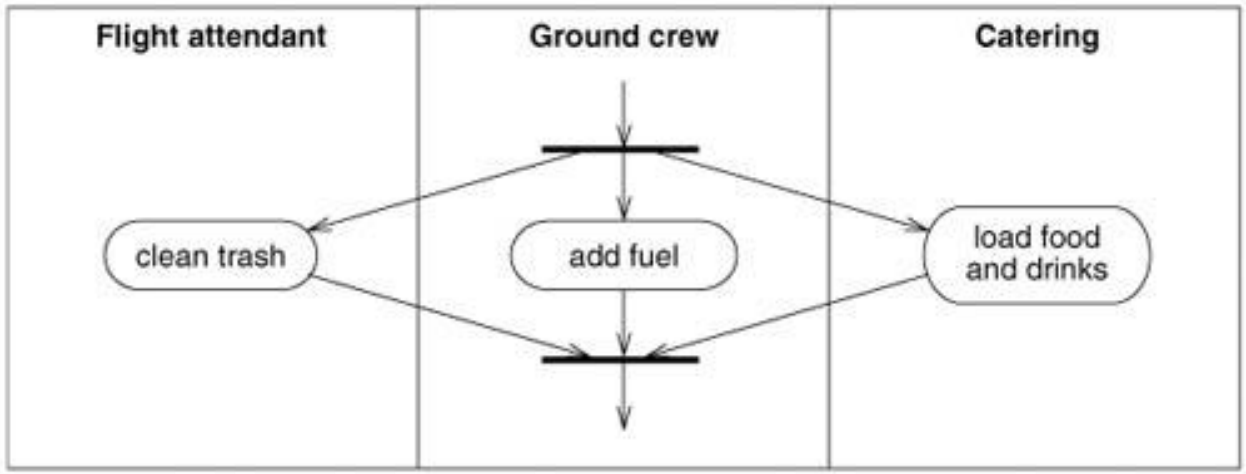
Activity Models: Sending and Receiving Signals

- UML Notation
  - Signal Send
    - Convex pentagon
  - Signal Receive:
    - Concave pentagon



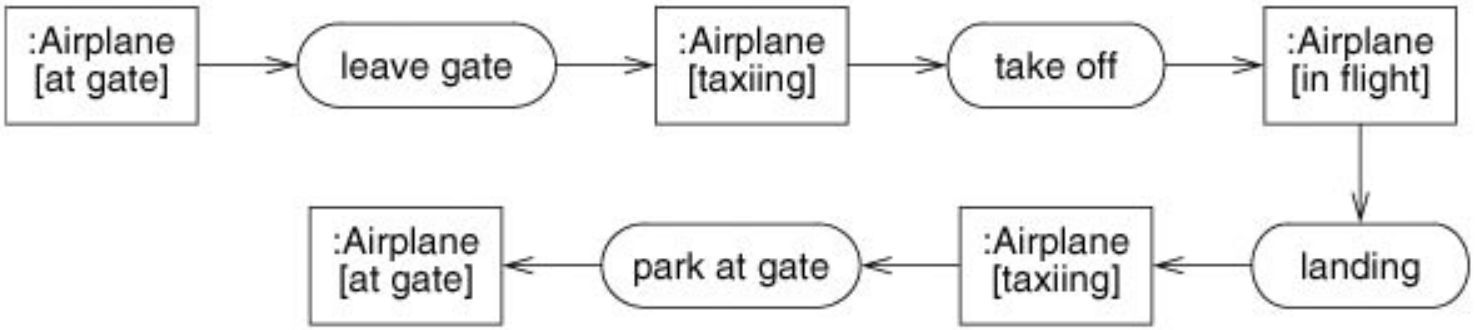
Activity Models: Swimlanes

- Partition activities among organizations
- Each column → Swimlane
- Crossing lines → interactions among organizations



Activity Models: Object Flows

- Show objects input to / output from activities
  - Shows object states
- UML Notation
  - Object name
  - Object state name in []



Activity Model: Guidelines

- Don't misuse activity diagrams
  - Facilitate study of algorithms and workflow
  - Not an excuse to develop software via flowcharts
- Level diagrams
  - Ensure activities are at consistent level of detail
- Be careful with branches and conditions
  - Ensure at least one condition is satisfied
- Be careful with concurrent activities
  - Ensure all activities complete prior to merge
- Consider executable activity diagrams
  - Help developers understand system better
  - Help end users follow progression of a process

Interaction Model

- Use cases
- Sequence diagrams
- Activity diagrams

Interaction Modeling: Example

- Computer email system
  - Actors
  - Use cases
  - Scenarios

Computer Email System (Contd.)

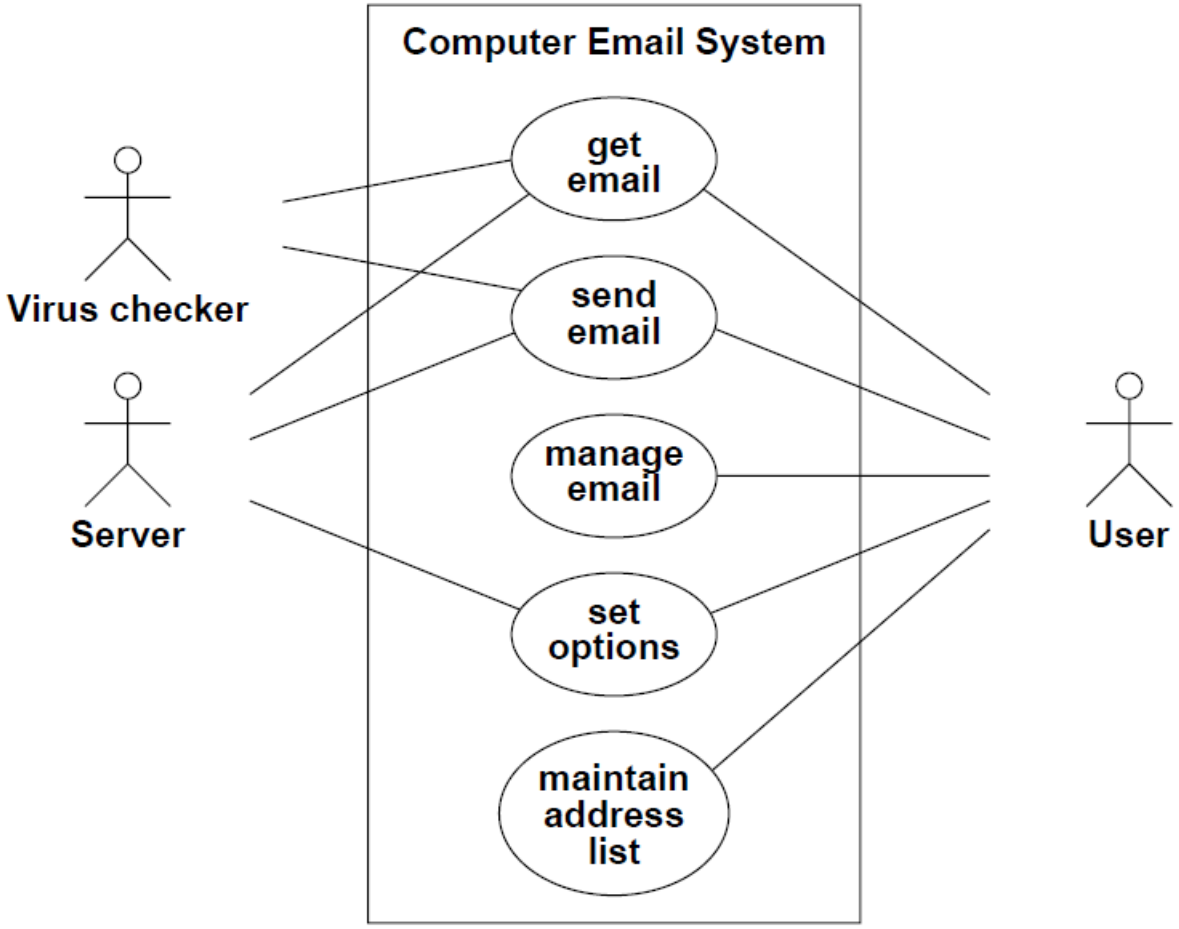
- Actors
  - User
    - Person who is the focus of services
  - Server
    - Computer that communicates with the email system and is the intermediate source and destination of email messages
  - Virus checker
    - Software that protects against damage by a malicious user.

Computer Email System (Contd.)

- Use cases
  - Get email
    - Retrieve email from the server and display it for the user
  - Send email
    - Take a message that was composed by the user and send it to the server
  - Manage email
    - Perform tasks such as deleting email, managing folders, and moving email between folders
  - Set options
    - Do miscellaneous things to tailor preferences to the user's liking
  - Maintain address list
    - Keep a list of email addresses that are important to the user

Computer Email System (Contd.)

- Use Case Diagram



Computer Email System (Contd.)

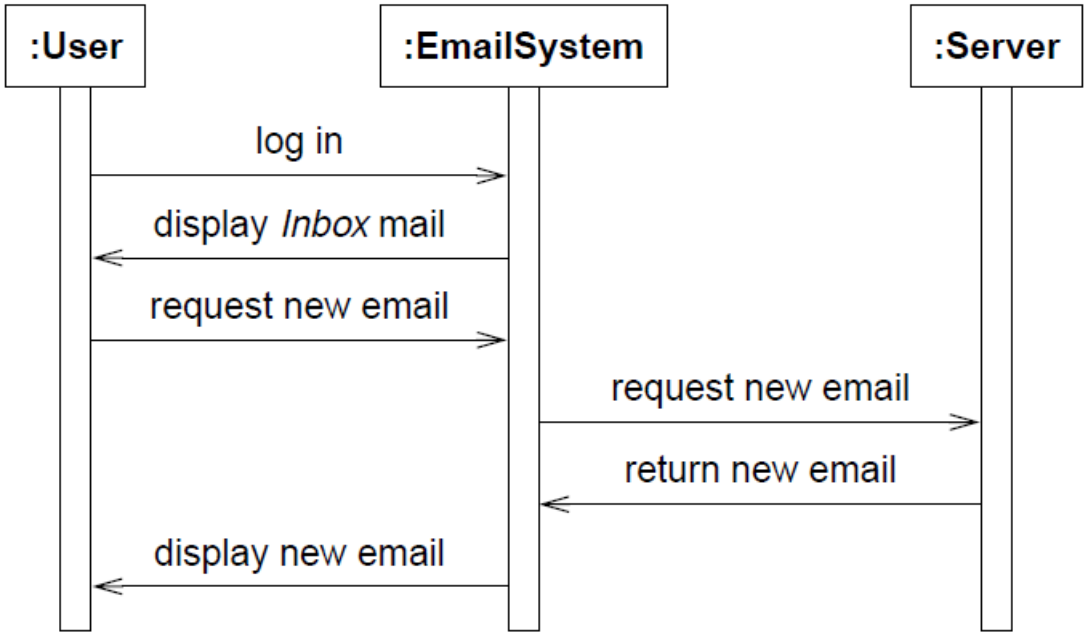
- Sequence model

Computer Email System (Contd.)

- Use case: Get email
  - Normal scenario
    - User logs in to email system.
    - System displays the mail in the Inbox folder.
    - User requests that system get new email.
    - System requests new email from server.
    - Server returns new email to system.
    - System displays the mail in the Inbox folder and highlights unread messages.
  - Exception scenario
    - User logs in to email system.
    - Local computer sends password to server.
    - Server rejects password as incorrect.

Computer Email System (Contd.)

- Sequence Diagram for getting email

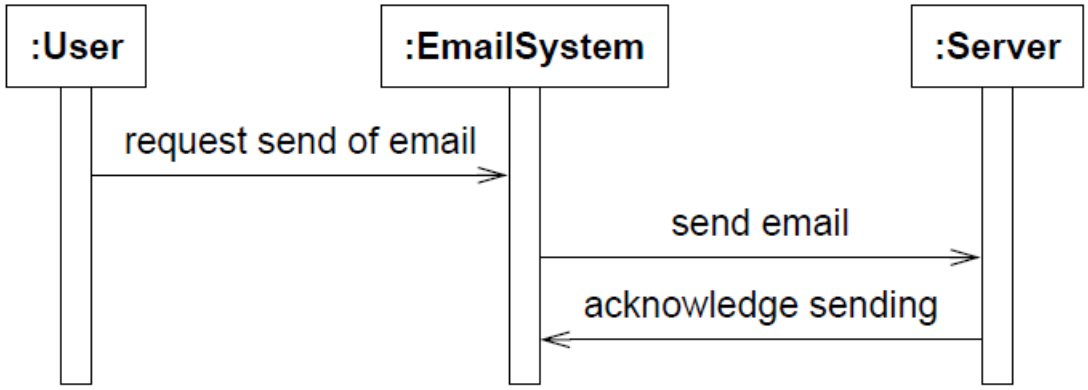


Computer Email System (Contd.)

- Use case: Send email
  - Normal scenario
    - User composes an email message and requests that system sends it.
    - Local computer sends email to server.
    - Server acknowledges receipt of email to send.
  - Exception scenario
    - User composes an email message.
    - User requests that system sends email.
    - System detects loss of Internet connection.

Computer Email System (Contd.)

- Sequence Diagram for sending email

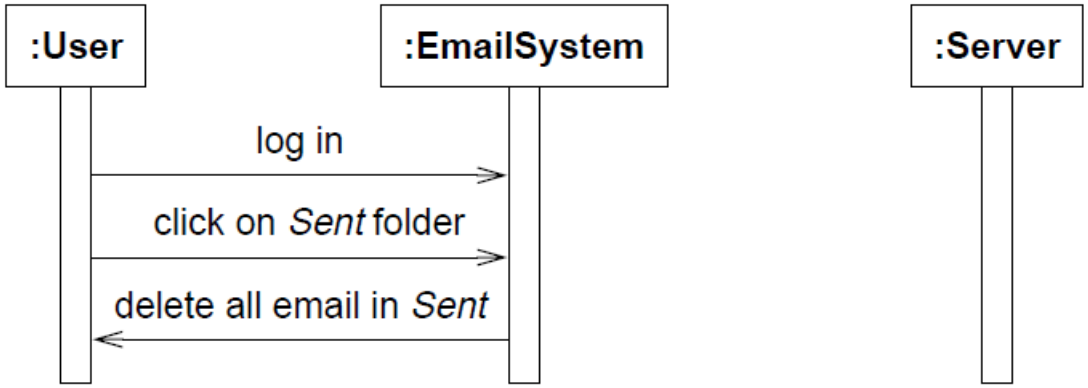


Computer Email System (Contd.)

- Use case: Manage email
  - Normal scenario
    - User logs in to email system.
    - User clicks on Sent folder.
    - User deletes all email in Sent folder.
  - Exception scenario
    - User logs in to email system.
    - User clicks on Sent folder.
    - System reports that data in folder is corrupted.

Computer Email System (Contd.)

- Sequence Diagram for managing email

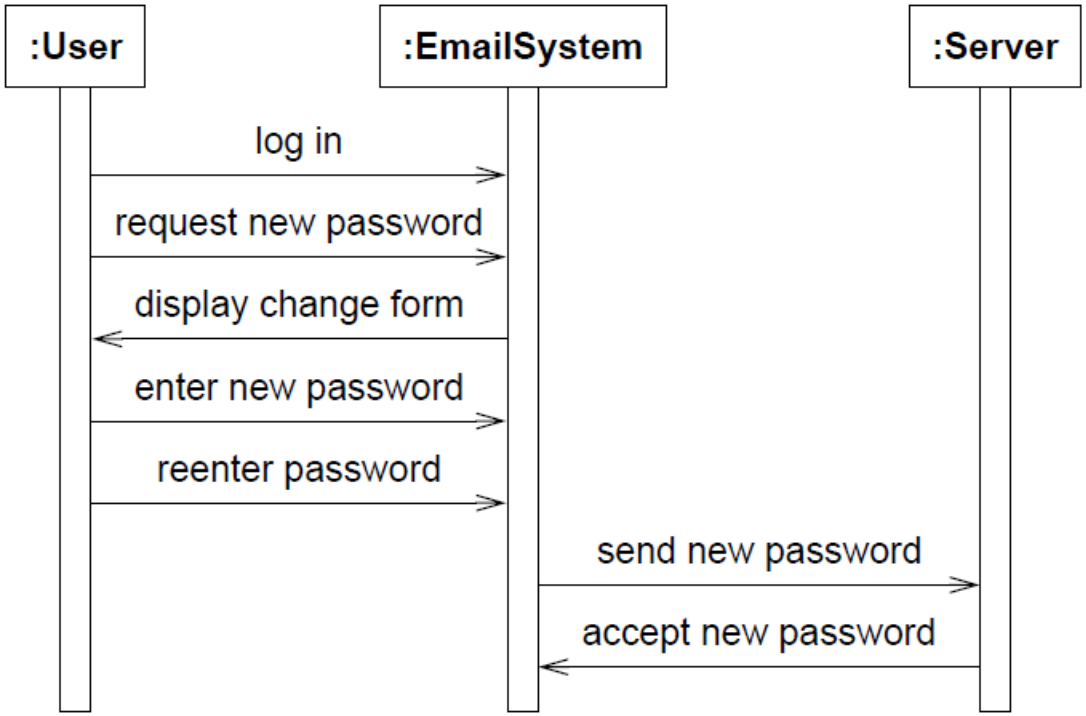


Computer Email System (Contd.)

- Set options
  - Normal scenario
    - User logs in to email system.
    - User requests change of password.
    - Local computer displays password change form.
    - User enters new password.
    - User reenters new password.
    - Local computer sends new password to server.
    - Server accepts new password.
  - Exception scenario
    - User logs in to email system.
    - User requests change of password.
    - Local computer displays password change form.
    - User enters new password.
    - Password length is illegal and password is not accepted.

Computer Email System (Contd.)

- Sequence Diagram for setting options

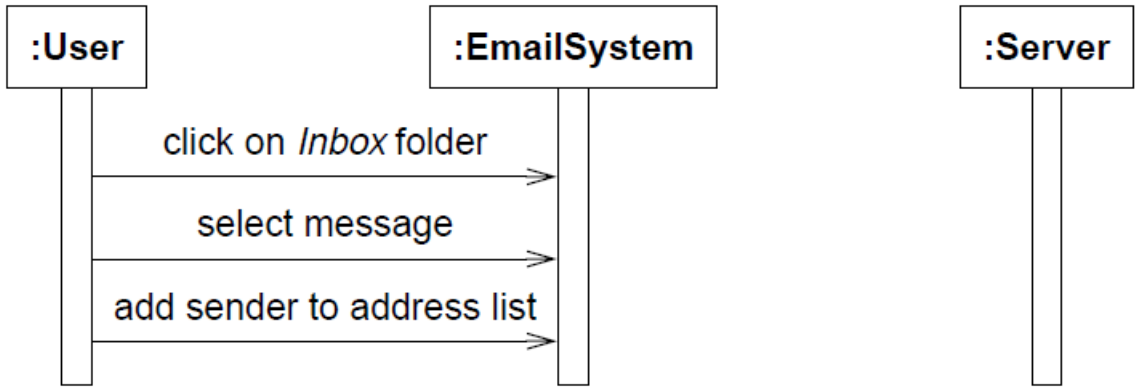


Computer Email System (Contd.)

- Maintain address list
  - Normal scenario
    - User clicks on Inbox folder.
    - User selects a message.
    - User adds sender to address list.
  - Exception scenario
    - User clicks on Inbox folder.
    - User selects a message.
    - User tries to add sender to address list.
    - System reports that sender is already in address list.

Computer Email System (Contd.)

- Sequence Diagram for maintaining address list



Next sessions…

- System Conception

Reading assignment

- [Blaha] Chp 7: Pages 131-144
- [Blaha] Chp 8: Pages 147-157