Karlstad University
Department of Information Systems

Adapted for a textbook by Blaha M. and Rumbaugh J.

Object Oriented Modeling and Design

Pearson Prentice Hall, 2005

INTERACTION MODELING

Remigijus GUSTAS
Phone: +46-54 700 17 65
E-mail: Remigijus.Gustas@kau.se
http://www.cs.kau.se/~gustas/

Interaction Diagrams

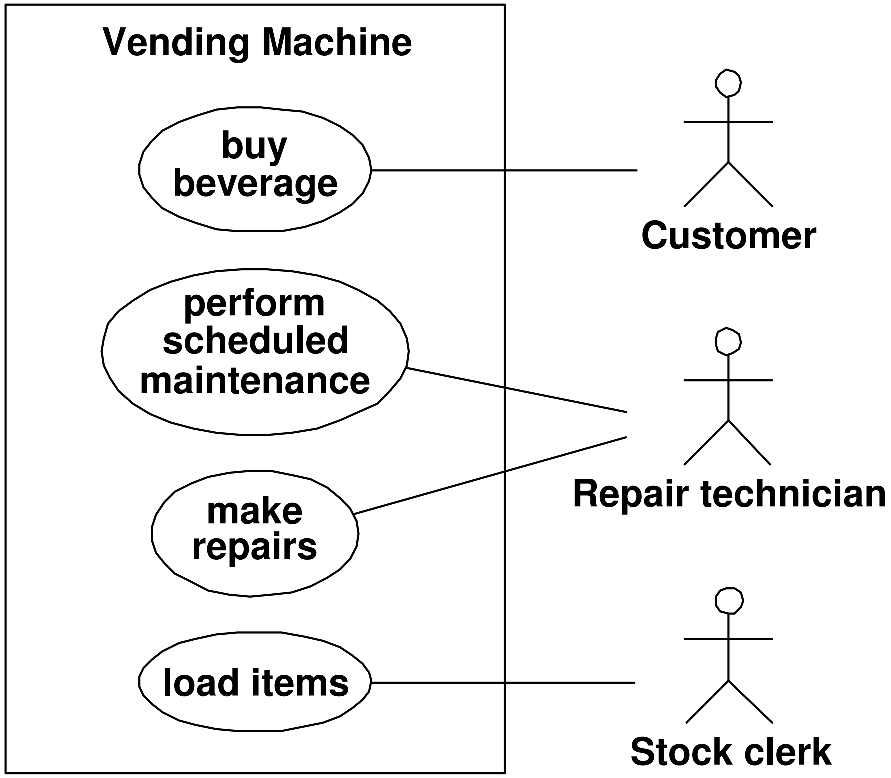
- ✓ Both state model and interaction model are needed to describe behavior fully.
- ✓ Interactions can be modeled on different levels of abstraction.
- ✓ Use case, activity and sequence diagrams are used to document interaction.
- ✓ State diagrams and interaction complement each other.

Use Cases and Actors

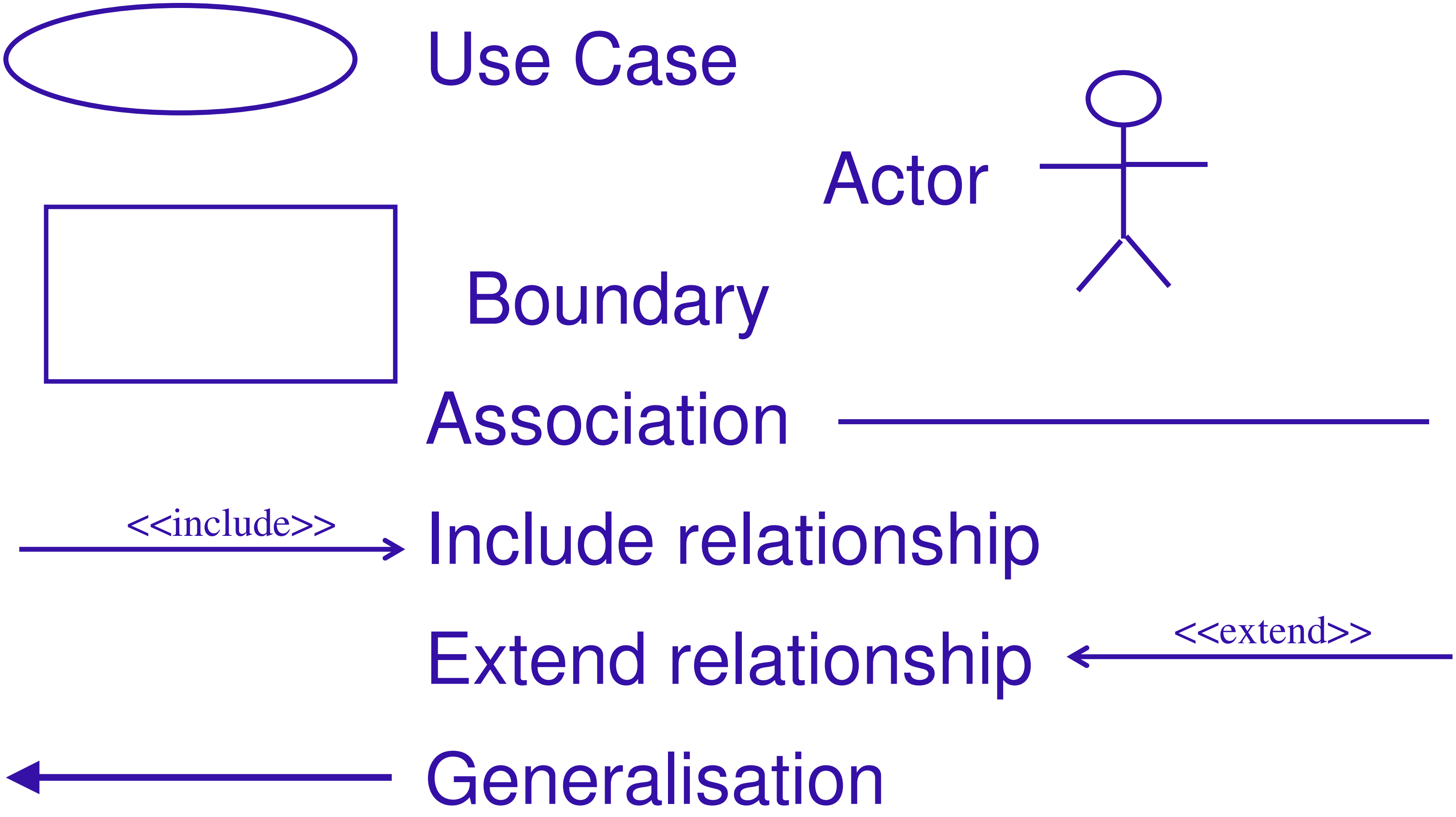
- ✓ Use cases represent functionality that a system can provide by interacting with actors.
- ✓ Use case involves a sequence of messages among the system and its actors.
- ✓ An actor is a direct external user of a system, but it is not part of the computerized system.
- ✓ The system responds to requests actors.
- ✓ An actor and use case has a single well-defined purpose (objects and classes serve many different purposes).
- ✓ Modeling the actors helps to define system boundary.

Use Case Diagrams

- ✓ A system involves a set of use cases and a set of actors.
- ✓ A set of use cases represent decomposition of functionality within system boundary.
- ✓ An organizational system part is represented by human actors.



UML Use Case Diagram Notation



Use Case

Actor

Boundary

Association

<<include>>

Include relationship

Extend relationship

<<extend>>

Generalisation

Guidelines for Use Case Models

- ✓ Determine clear system boundary,
- ✓ Each actor should be focused on a single, coherent purpose,
- ✓ Each use case must provide value to users,
- ✓ Actors and use cases must be related (use case should have an actor and visa versa),
- ✓ Use cases can be decomposed by using special relationships.

What is an Actor?

- ✓ Actor is an external entity that interacts with the system.
- ✓ Most actors represent user roles, but actors can also be external systems.
- ✓ An actor is a role, not a specific user; one user may play many roles, and an actor may represent many users.

What is a Boundary?

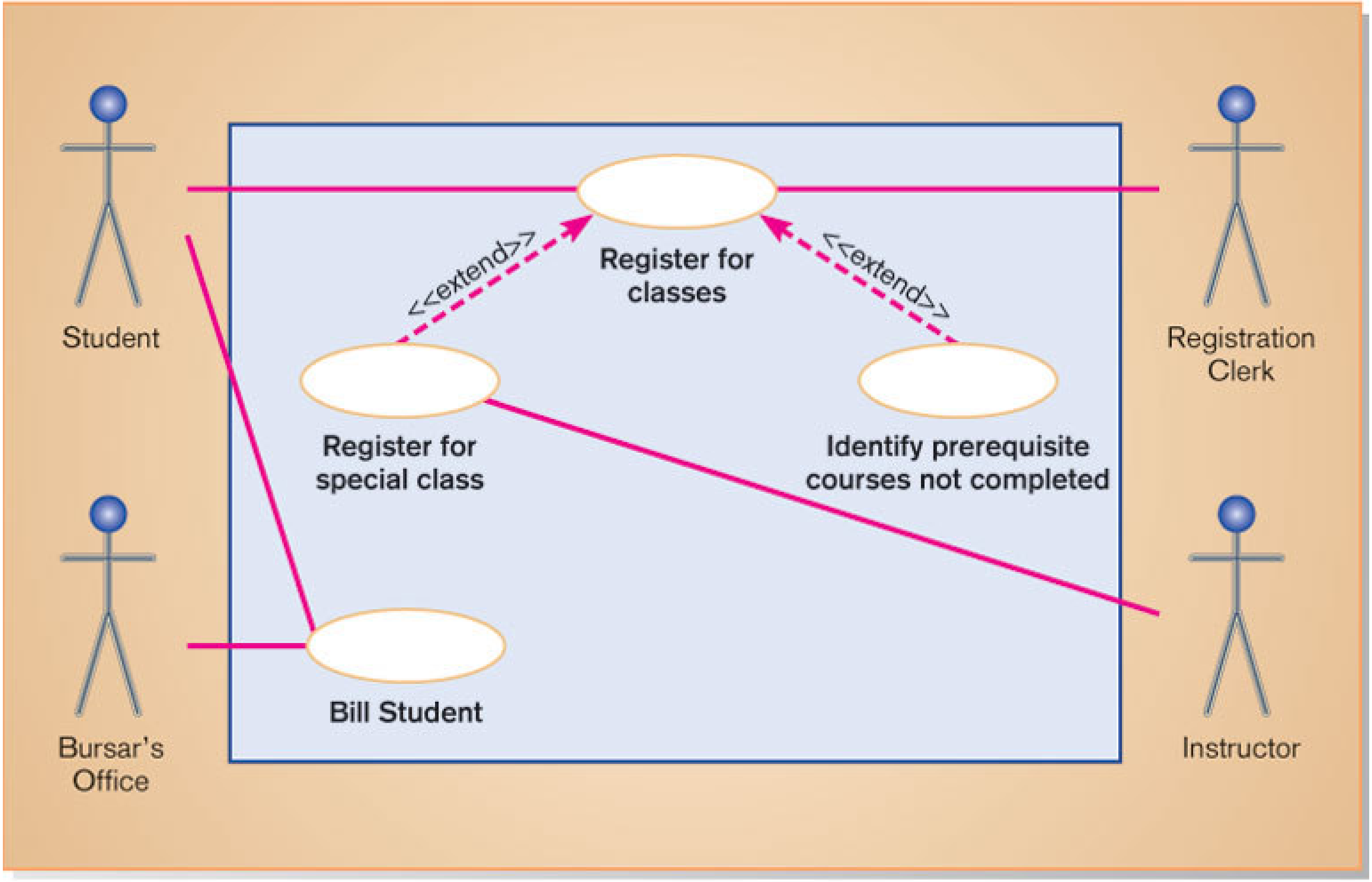
- ✓A boundary is the dividing line between the system and its environment.
- ✓Use cases are within the boundary.
- ✓Actors are outside of the boundary.

What is a Use Case Connection?

- ✓A connection is an association between an actor and a use case.
- ✓Depicts a usage relationship
- ✓Connection does not indicate data flow

What is an <<extends>> Relationship?

- ✓The extends relationship adds new incremental behavior to a use case.
- ✓A (optional) connection between two use cases.
- ✓Specialized use case extends the general use case.

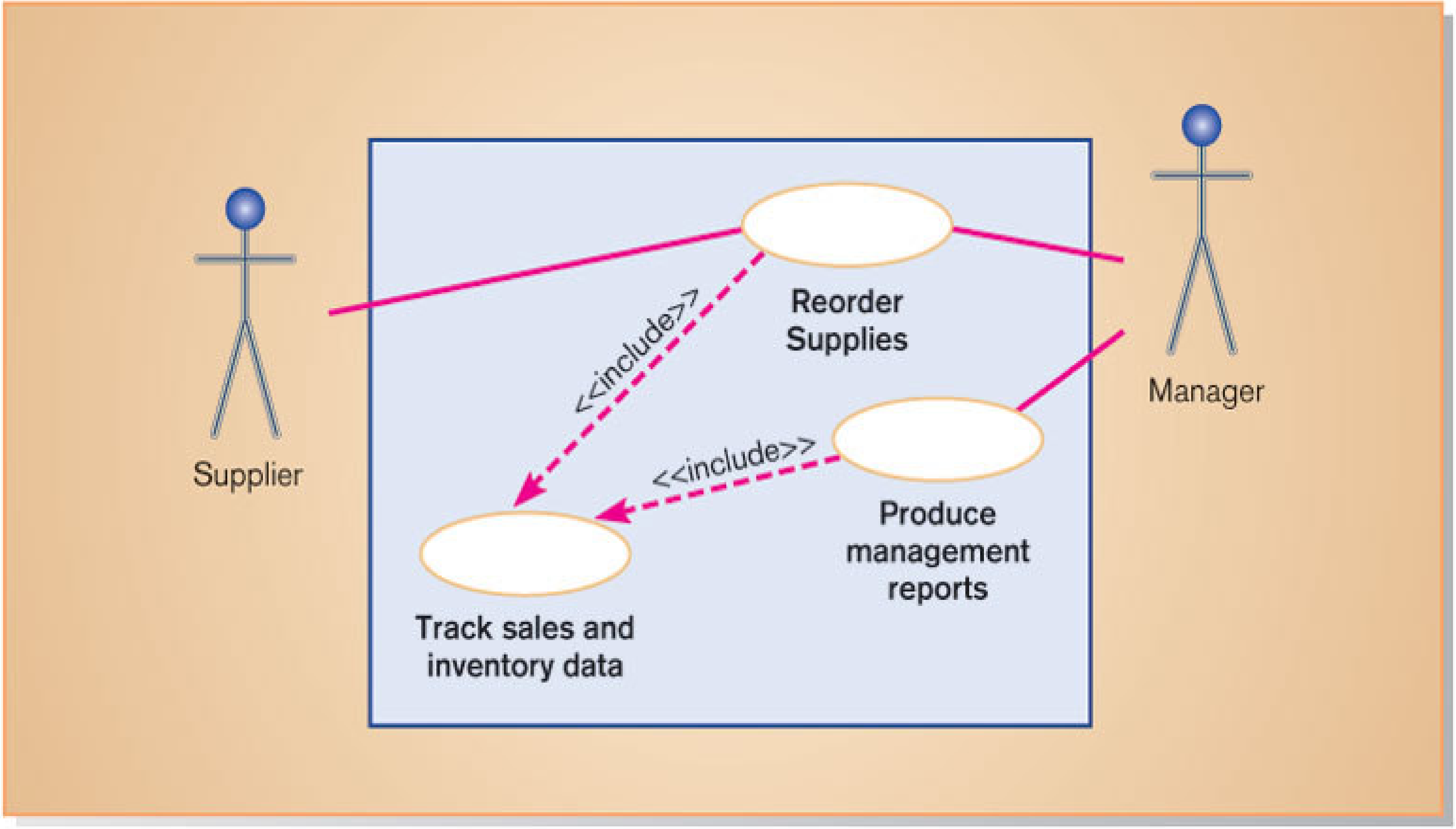

Student
Registration Clerk
Bursar's Office
Instructor
Register for classes
<<extend>>
<<extend>>
Register for special class
Identify prerequisite courses not completed
Bill Student

What is an <<include>> Relationship?

- ✓ The **include relationship** incorporates one use case within the behavior sequence of another use case.

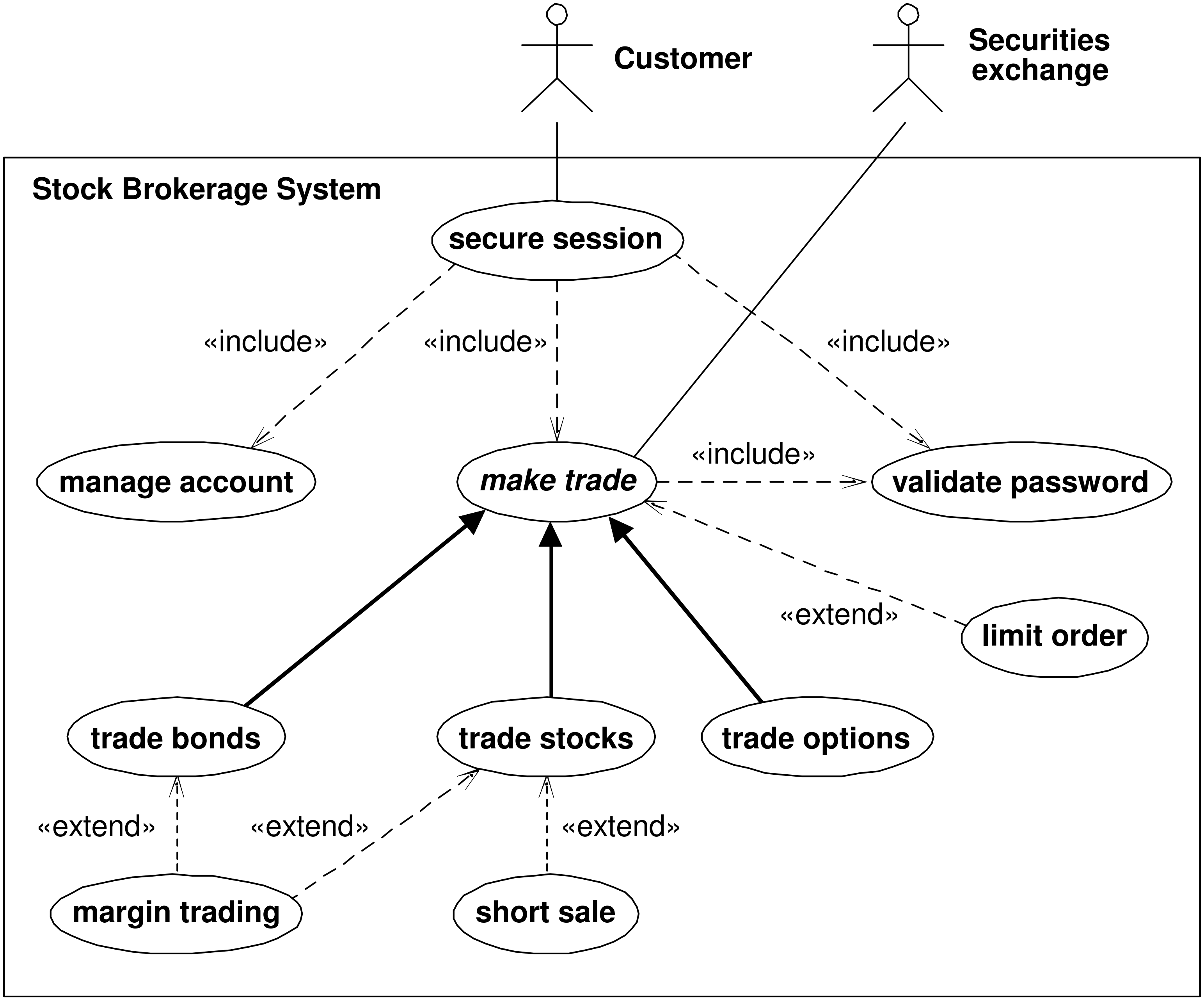
- ✓ A (mandatory) connection between two use cases. Indicates a use case is always used (invoked) by another use case

- ✓ Links to general purpose functions, used by many other use cases


Supplier
Manager
Reorder Supplies
<<include>>
<<include>>
Produce management reports
Track sales and inventory data

What is Use Case Generalization?

- ✓ Analogous to generalization among classes.
- ✓ A more specific use cases show variations of more general behavior sequence.
- ✓ A parent use case may be abstract or concrete.
- ✓ Concrete use cases exhibit polymorphism - a more specific use case freely substitutes for a more general use case (overriding).
- ✓ Multiple inheritance is not allowed in use case generalization.


Customer
Securities exchange
Stock Brokerage System
secure session
«include»
«include»
«include»
manage account
make trade
«include»
validate password
«extend»
limit order
trade bonds
trade stocks
trade options
«extend»
«extend»
«extend»
margin trading
short sale

Sequence Models

- ✓ The sequence models are used to elaborate use cases.

- ✓ Different kinds of sequence models:
  - ■ Scenarios,
  - ■ Sequence diagrams,
  - ■ Activity diagrams.

Scenarios

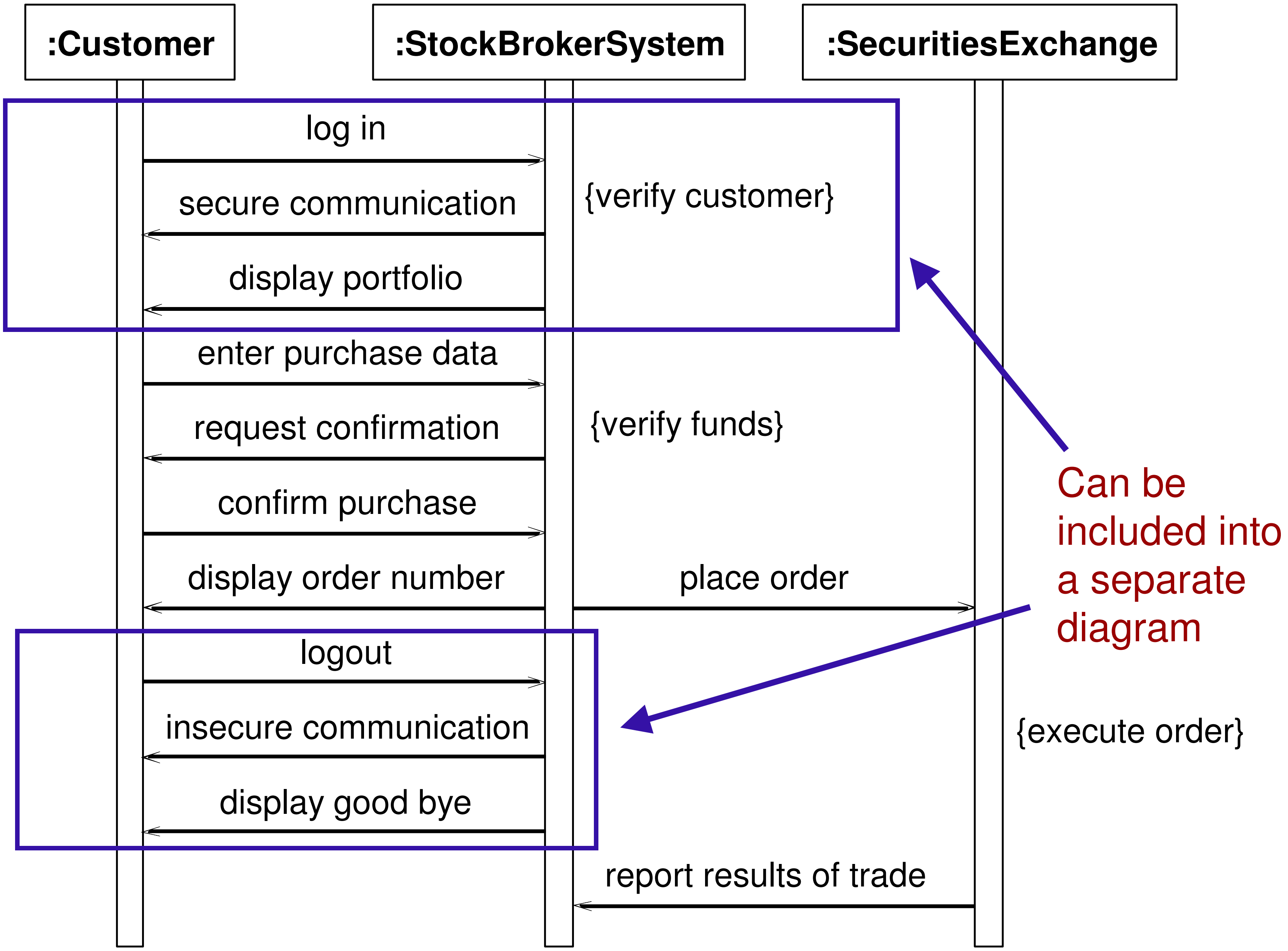
- ✓ Scenario is a sequence of events that occurs during a particular execution (for a system or use case).
- ✓ The scope of scenario can vary (all events or events generated by certain objects).
- ✓ Steps of writing a scenario:
  - ■ Identify objects exchanging messages,
  - ■ Determine the sender and receiver of each message as well as the message sequence,
  - ■ Internal computing operations are added later.

Scenario between User (John Doe) and 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

- ✓ A sequence diagram shows the participants of interaction and the message sequences.
- ✓ Each actor is represented by a lifeline and each message by a horizontal arrow from sender to receiver.
- ✓ Time proceeds from left to right and from top to bottom.
- ✓ Use case requires one or more diagrams to describe the behavior.
- ✓ Diagram can show an entire session or a separate task (rather than repeating the same sequences in many diagrams).



:Customer

:StockBrokerSystem

:SecuritiesExchange

log in

secure communication

display portfolio

{verify customer}

enter purchase data

request confirmation

{verify funds}

confirm purchase

display order number

place order

logout

insecure communication

display good bye

{execute order}

report results of trade

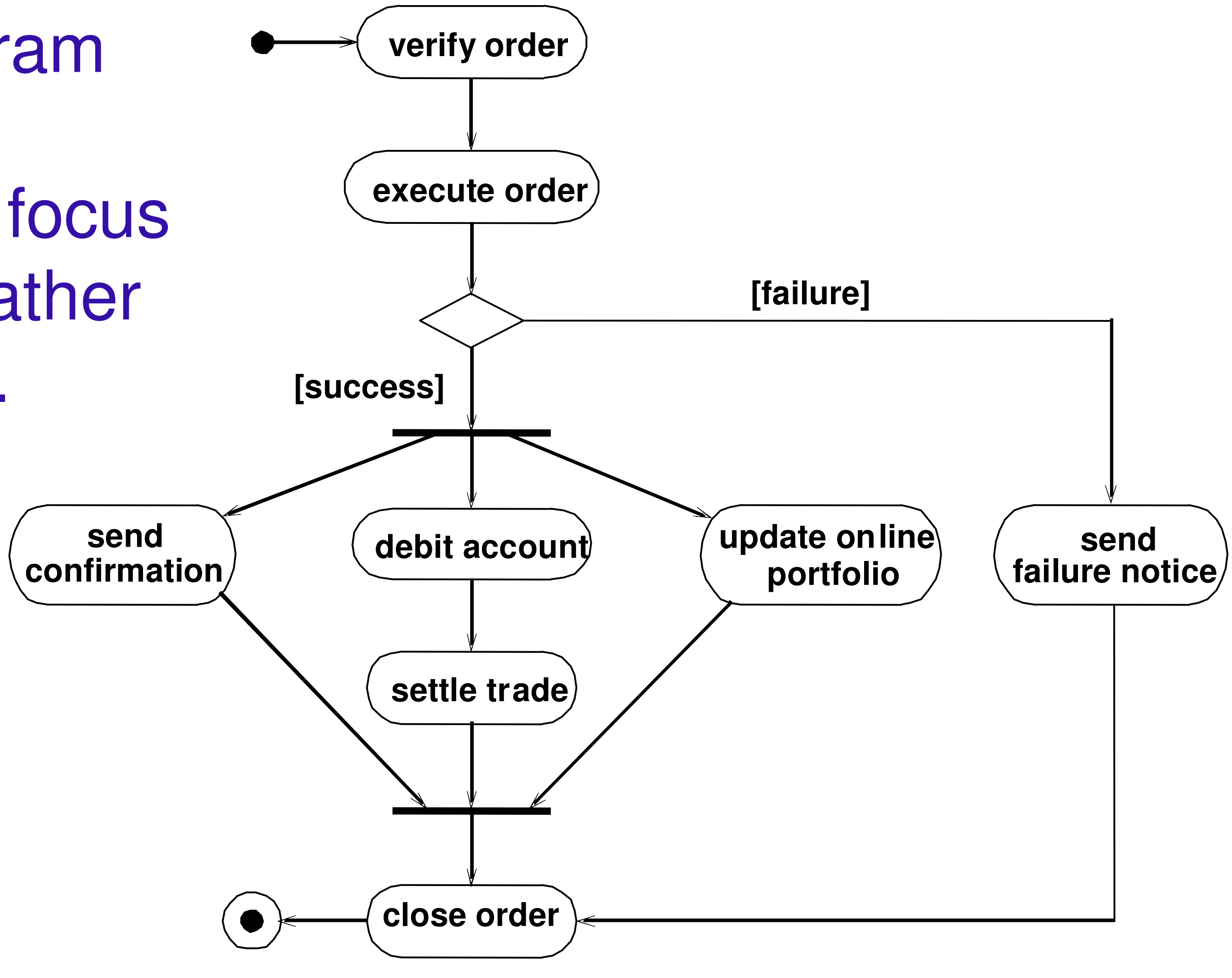
Can be included into a separate diagram

Guidelines for Sequence Modeling

- ✓ Define at least one scenario per use case,
- ✓ Abstract the scenarios into graphical descriptions (such as sequence diagrams),
- ✓ Break down complex interactions into smaller parts and define a separate diagram for each of them,
- ✓ Define a diagram for each alternative action (for instance, error condition).

Activity Diagrams

An activity diagram defines flow of control with the focus on operations rather than on objects.

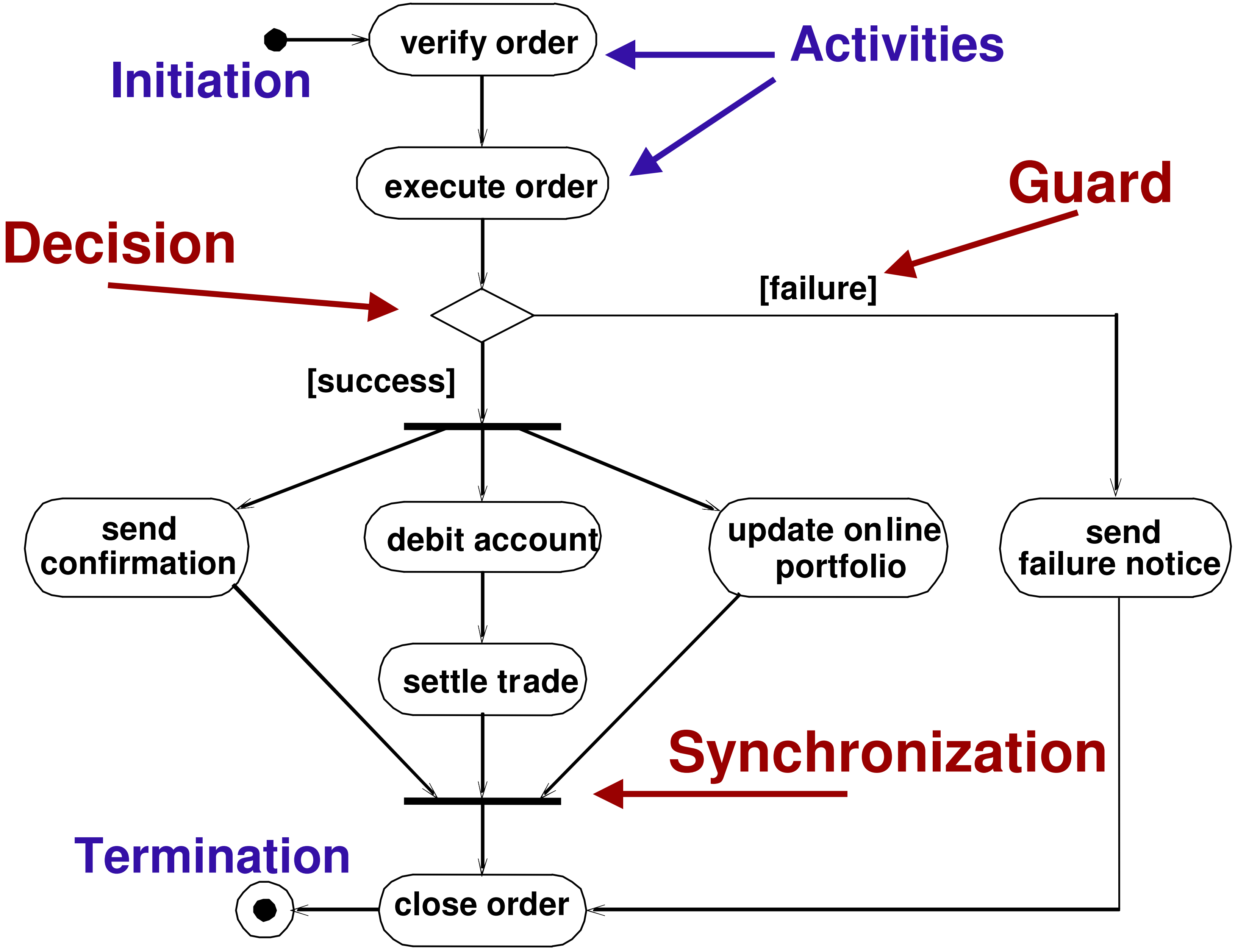


Activities and Events

- ✓ An activity can be decomposed into finer activities or operations.
- ✓ The steps of an activity diagram at the bottom level of abstraction are operations from the state model.
- ✓ The completion of an activity emits a completion event.
- ✓ An arrow from one activity to another indicates that the completion event of the first activity will trigger the second activity.

Control Conditions

- ✓ Is a mechanism that evaluates a specified to determine how processing flow should proceed.
- ✓ Activity diagram notation is using the following control conditions:
  - ■ Decisions (branching),
  - ■ Synchronizations (concurrency) and
  - ■ Guards.
- ✓ If condition is never satisfied, then it might be ill formed.



Initiation

verify order

Activities

execute order

Guard

Decision

[failure]

[success]

send confirmation

debit account

update online portfolio

send failure notice

settle trade

Synchronization

Termination

close order

Branching

- ✓ A **decision diamond** indicates branching into multiple successors (specialization of completion events).
- ✓ All subsequent **guard conditions** are tested when an activity completes.
- ✓ A particular execution chooses only one path of control.
- ✓ If several arrows enter an activity, then alternate execution paths merge. This situation can be represented by using decision diamond merges several arrows into one exit (generalization of completion events).
- ✓ Logical ’or’ of completion events.

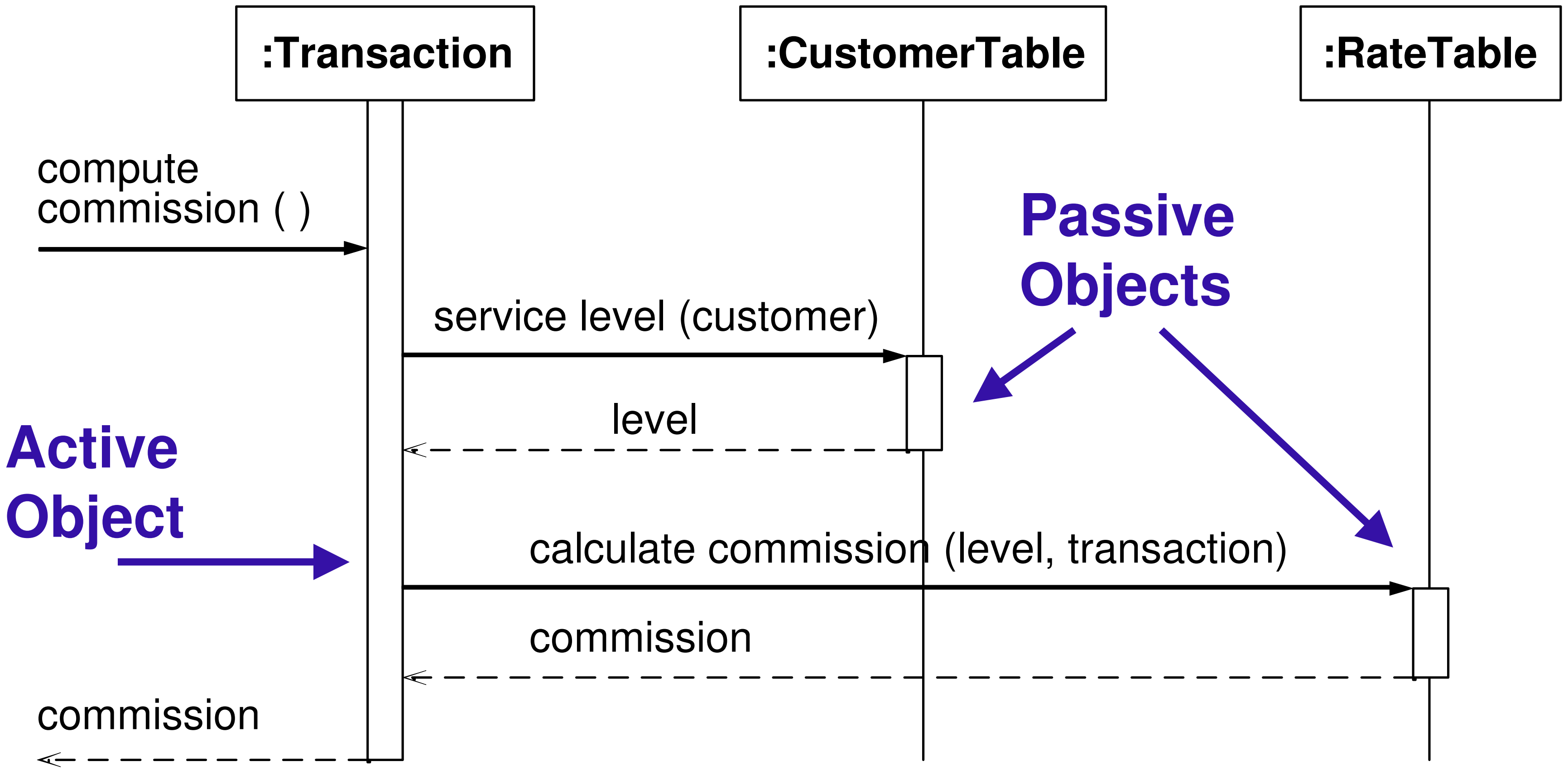
Concurrent Activities

- ✓ Organizations and computer systems can trigger more than one activity at a time.
- ✓ A synchronization bar indicates incoming or outgoing concurrent activities.
- ✓ A fork is used to split control for all outgoing activities (decomposition of completion events).
- ✓ A merge is used to aggregate control from all incoming activities (composition of completion events).
- ✓ Logical ’and’ of completion events.

Guidelines for Activity Modeling

- ✓ Activity diagrams are intended to elaborate use cases and should be used by developers to study workflow, not to document software processes,
- ✓ Complex activities should be represented on higher levels of abstraction,
- ✓ If there are guard conditions, at least one must be satisfied when activity completes,
- ✓ Concurrent activities can be completed in any order. For a merge of control, all input activities must be completed.

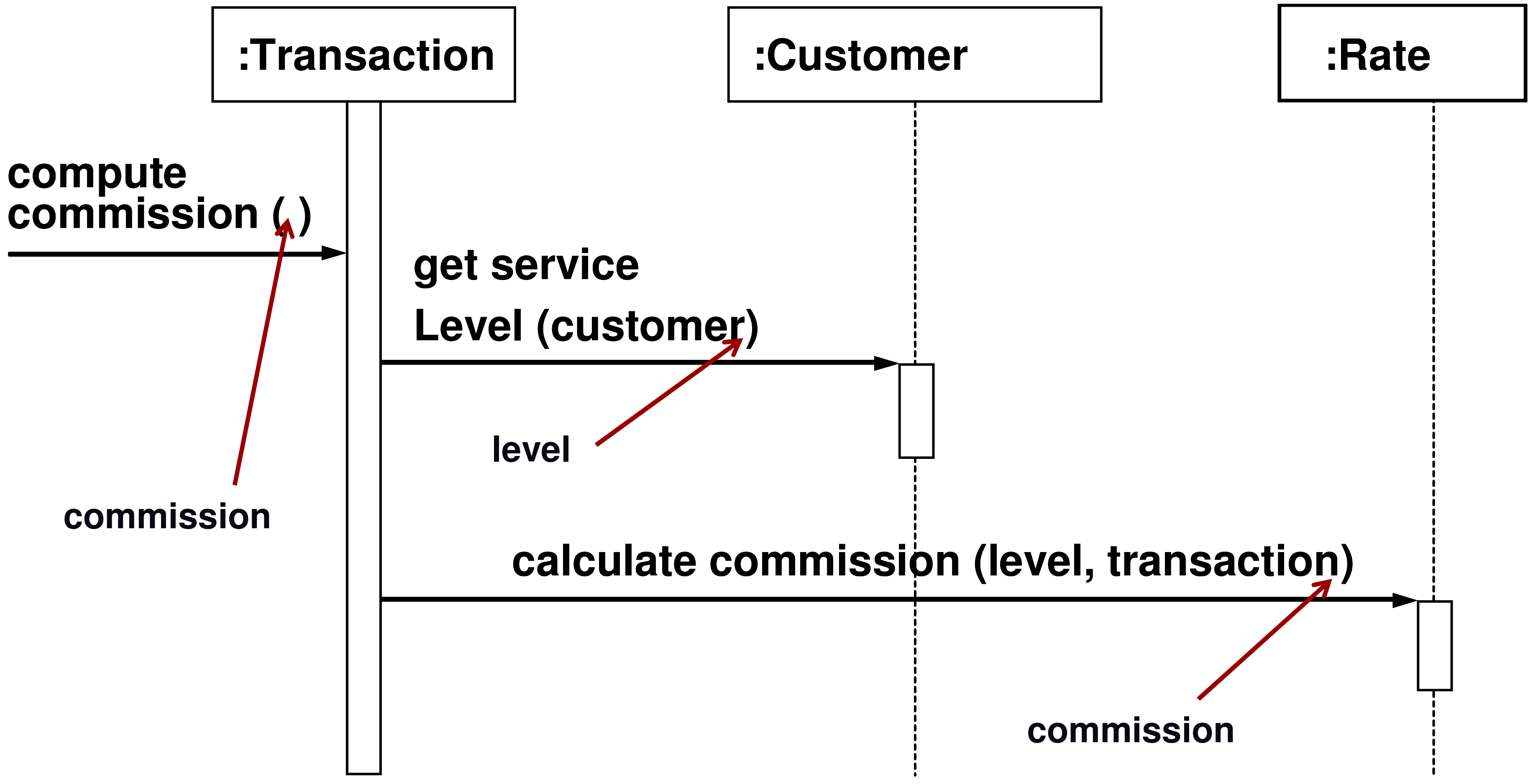
Sequence Diagrams with Passive and Active Objects



Passive and Active Objects

- ✓ **Active objects** are appropriate for higher level models. Active objects are independent. They remain active after sending message and can respond to other messages.
- ✓ A **passive object** is not activated until it has been called. Once execution of the operation completes and control returns to the caller, the passive object remains inactive.
- ✓ **Lifeline** is the entire period of object existence.
- ✓ **Activation** (thin rectangle) shows the time period during which a call is being processed.

Return arrows can be suppressed, because their location is implicit.

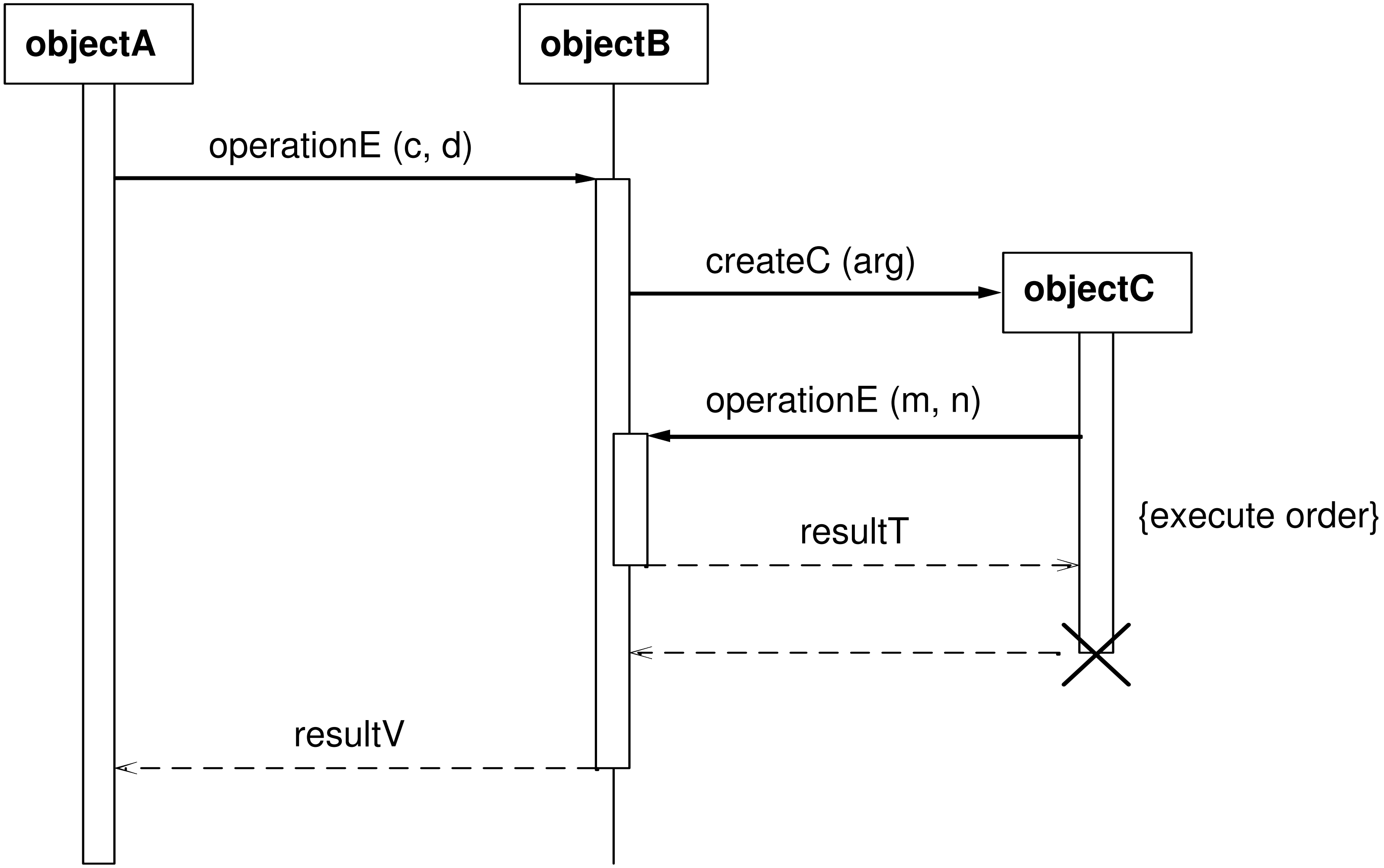


Transient Objects

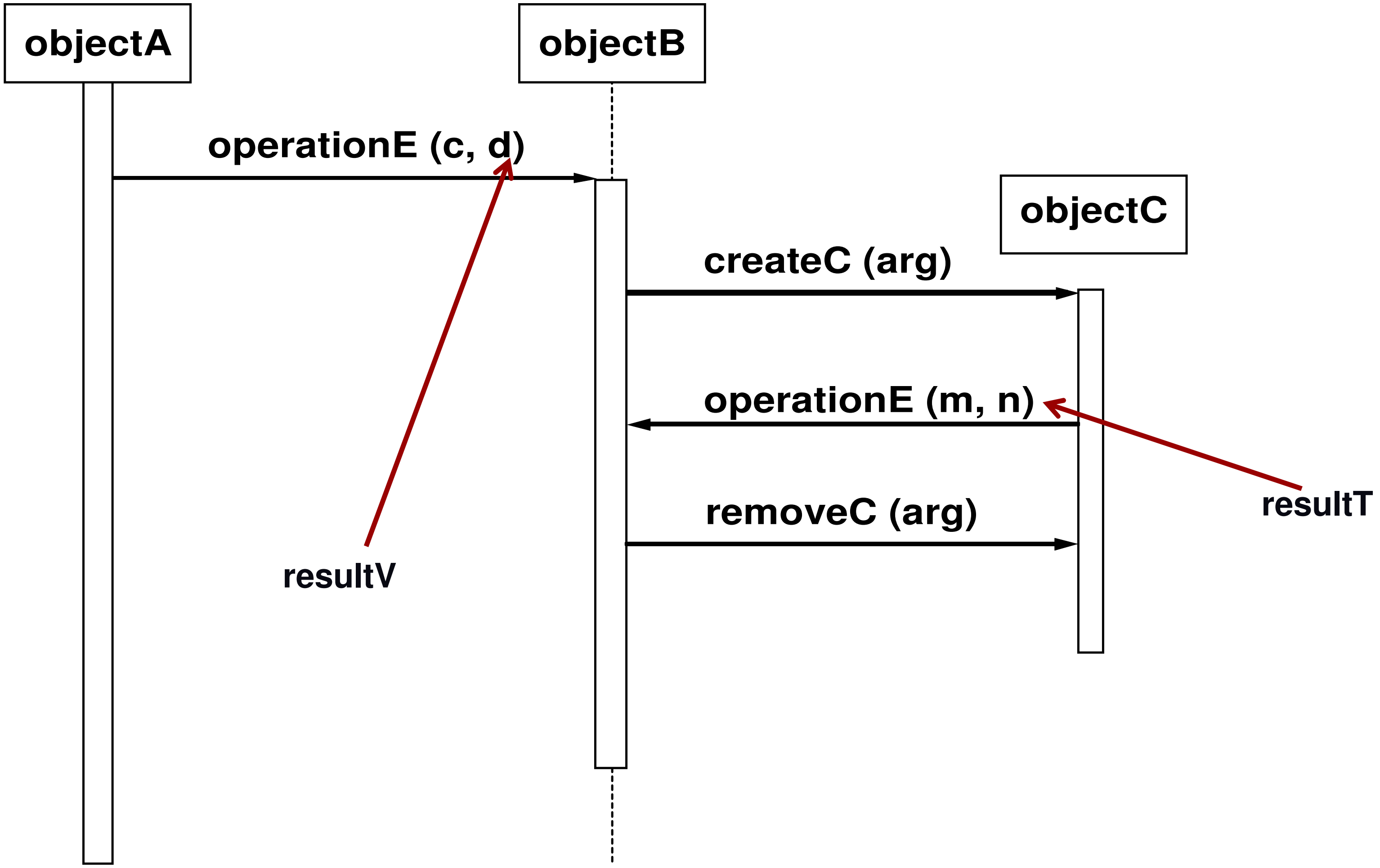
- ✓ Creation operation is shown by placing the object symbol at the head of the arrow.
- ✓ 'X' marks the end of the life cycle of an object.
  - if the object destroys itself and returns control, then 'X' is placed at the tail of return arrow.
  - 'X' can be placed at the head of the call arrow that destroys the object.

Note: The filled arrowhead indicates a call (as opposed to the stick arrowhead for asynchronous signal).

Diagram with Transient Object

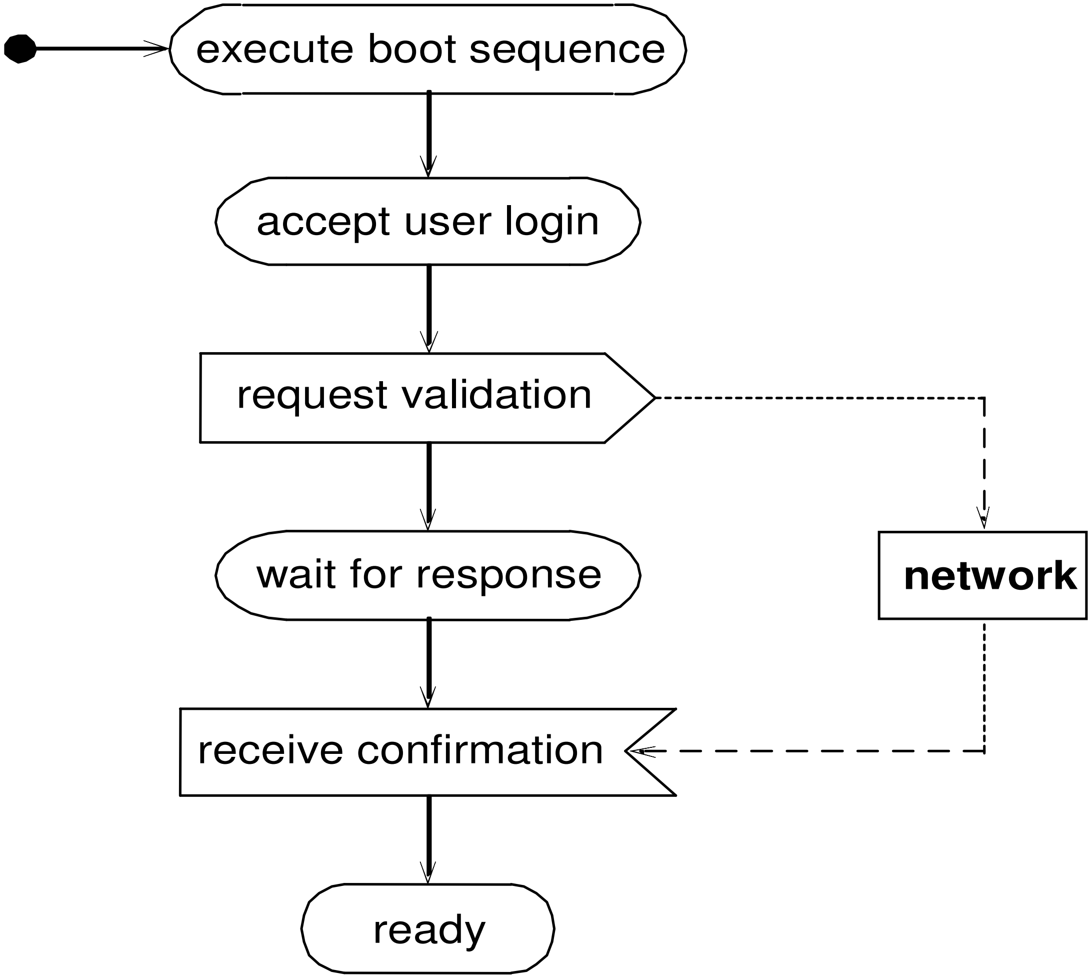


Return arrows can be suppressed, removal operation can replace ‘X’.



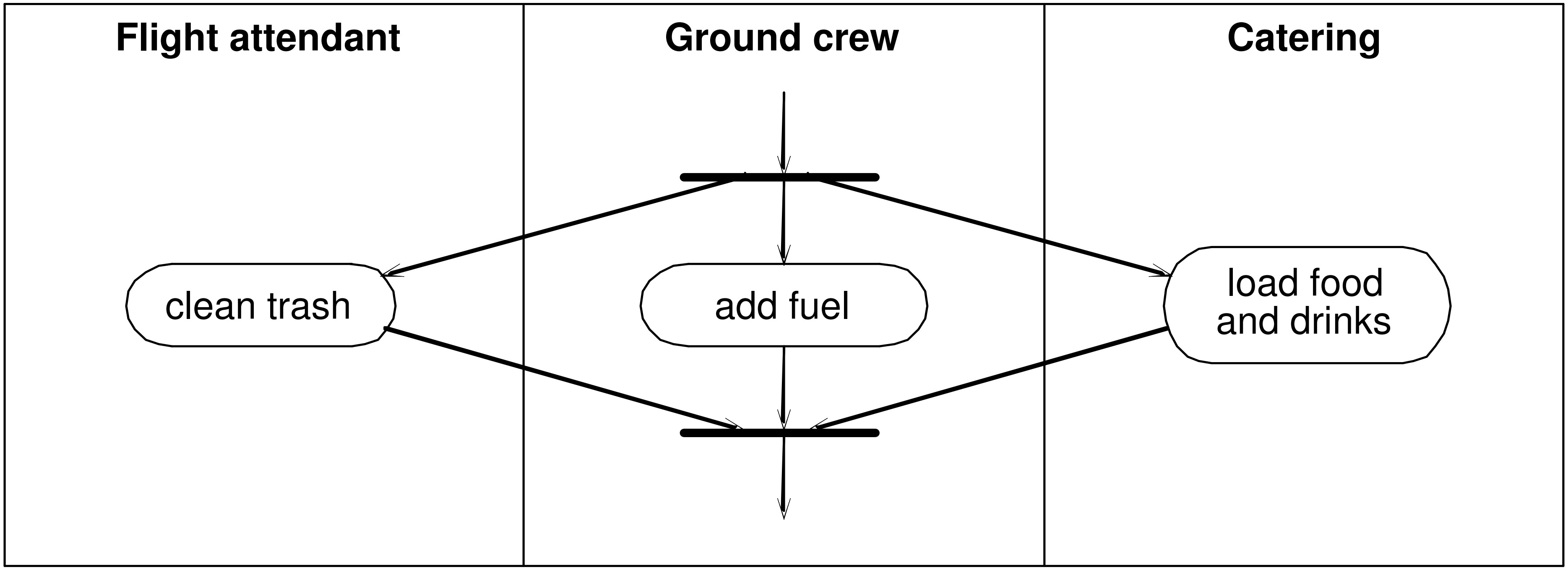
Sending and Receiving Signals

UML shows sending signals as a convex pentagon and receiving signals as concave pentagon.



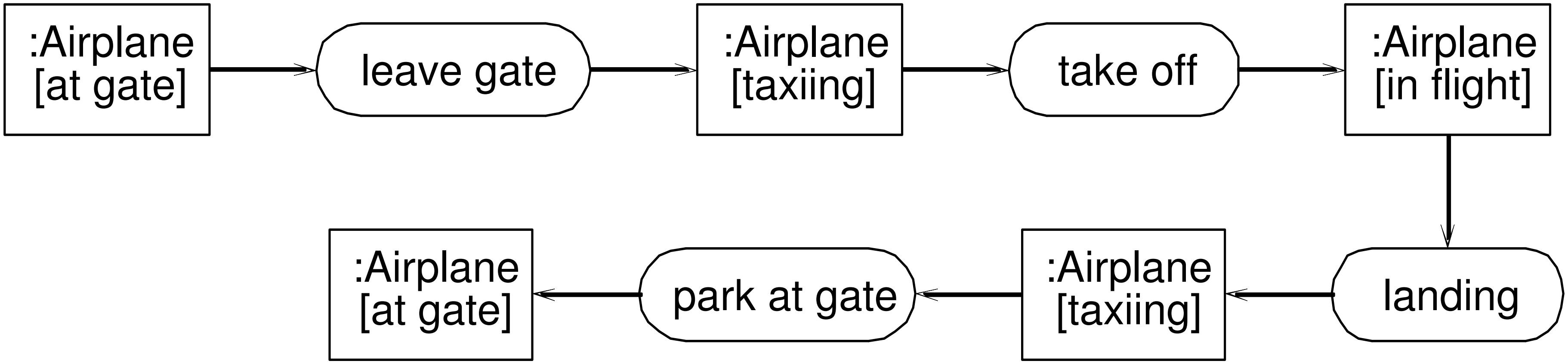
Swimlanes

Placing an activity within a swimlane indicates that it is performed by a person with the specific role or at the specific department in an organization.



Object Flows

- ✓ Activities can use or produce object in a particular state.
- ✓ Input and output arrows imply a control flow (it is unnecessary to draw control flows in object flow diagrams.


:Airplane [at gate]
leave gate
:Airplane [taxiing]
take off
:Airplane [in flight]
landing
:Airplane [taxiing]
park at gate
:Airplane [at gate]