

UML

Diagrams

Diagrams in UML

A *Diagram* is the graphical presentation of a set of elements, most often rendered as a connected graph of things and relationships.

1. Class Diagram, Package diagram, Object diagram

2. Use Case Diagram.

3. Sequence Diagram.

4. Collaboration Diagram.

5. State Chart Diagram.

6. Activity Diagram.

Diagram Types

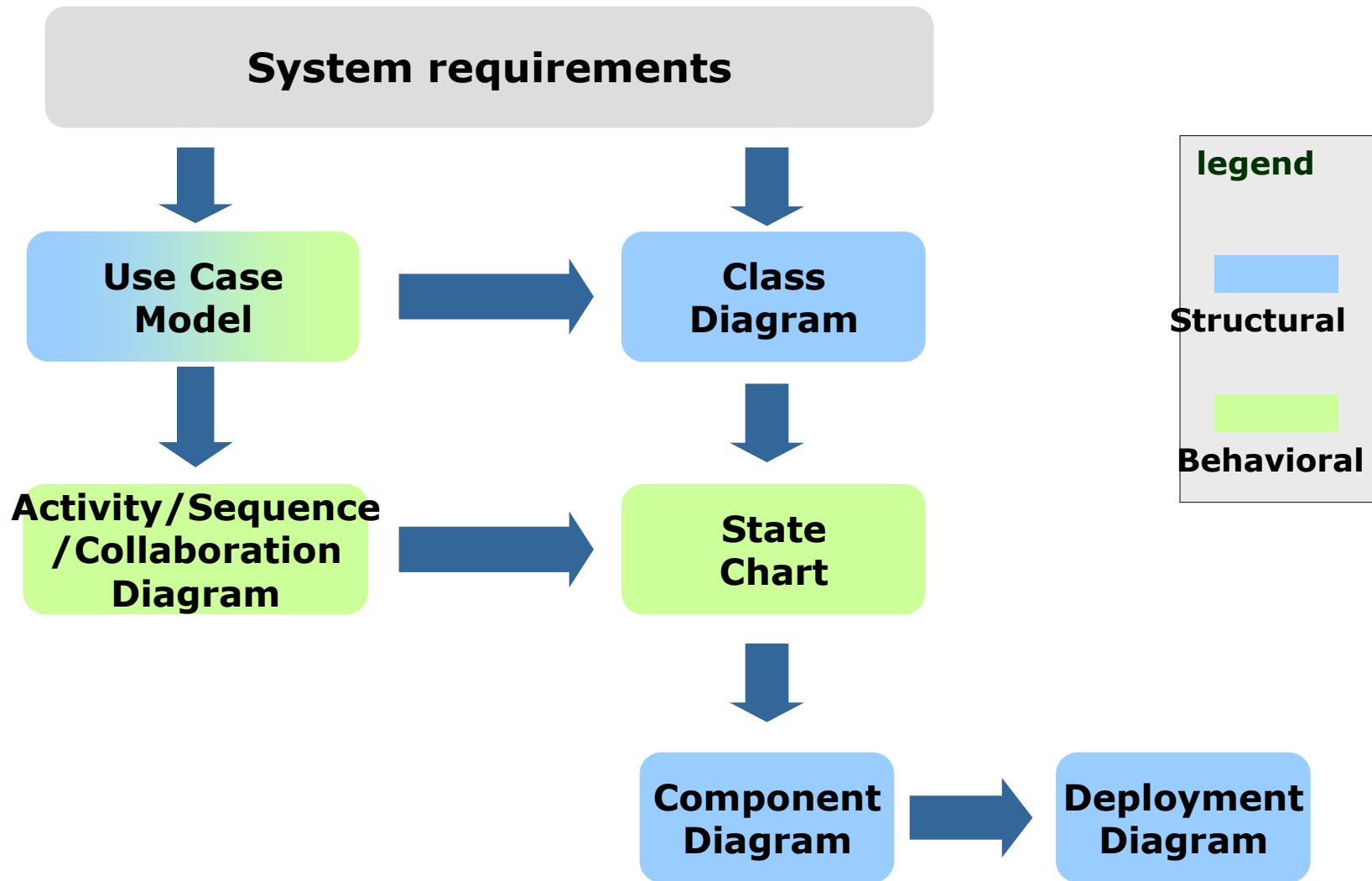
■ Structural Diagrams

- focus on static aspects of the software system
- Class, Object, Package, Component, Deployment

■ Behavioral Diagrams

- focus on dynamic aspects of the software system
- Use-case, Sequence, Collaboration, State Chart, Activity

Design Process



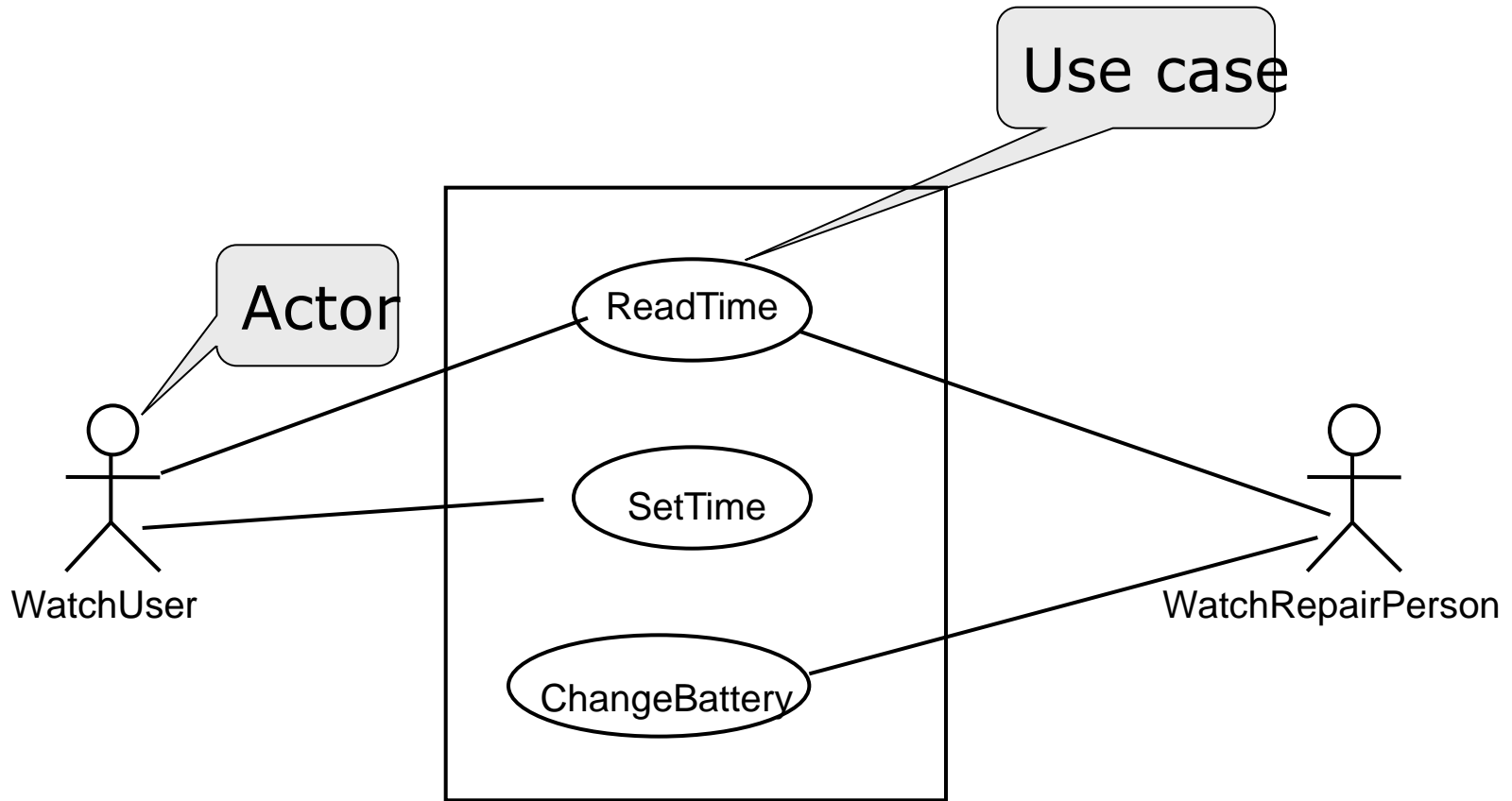
Interaction Diagrams

- Show how the software can be used/maintained
 - Usecase diagram
- Show how objects interact with one another
 - Sequence diagram
 - Collaboration diagram
- Show the major activities
 - Activity diagram
- Show the states of a few important classes/objects
 - State/Statechart diagram

UML First Pass

- Use case Diagrams
 - Describe the functional behavior of the system as seen by the user.
- Class diagrams
 - Describe the static structure of the system: Objects, Attributes, Associations
- Sequence diagrams
 - Describe the dynamic behavior between actors and the system and between objects of the system
- Statechart diagrams
 - Describe the dynamic behavior of an individual object (essentially a finite state automaton)
- Activity Diagrams
 - Model the dynamic behavior of a system, in particular the workflow (essentially a flowchart)

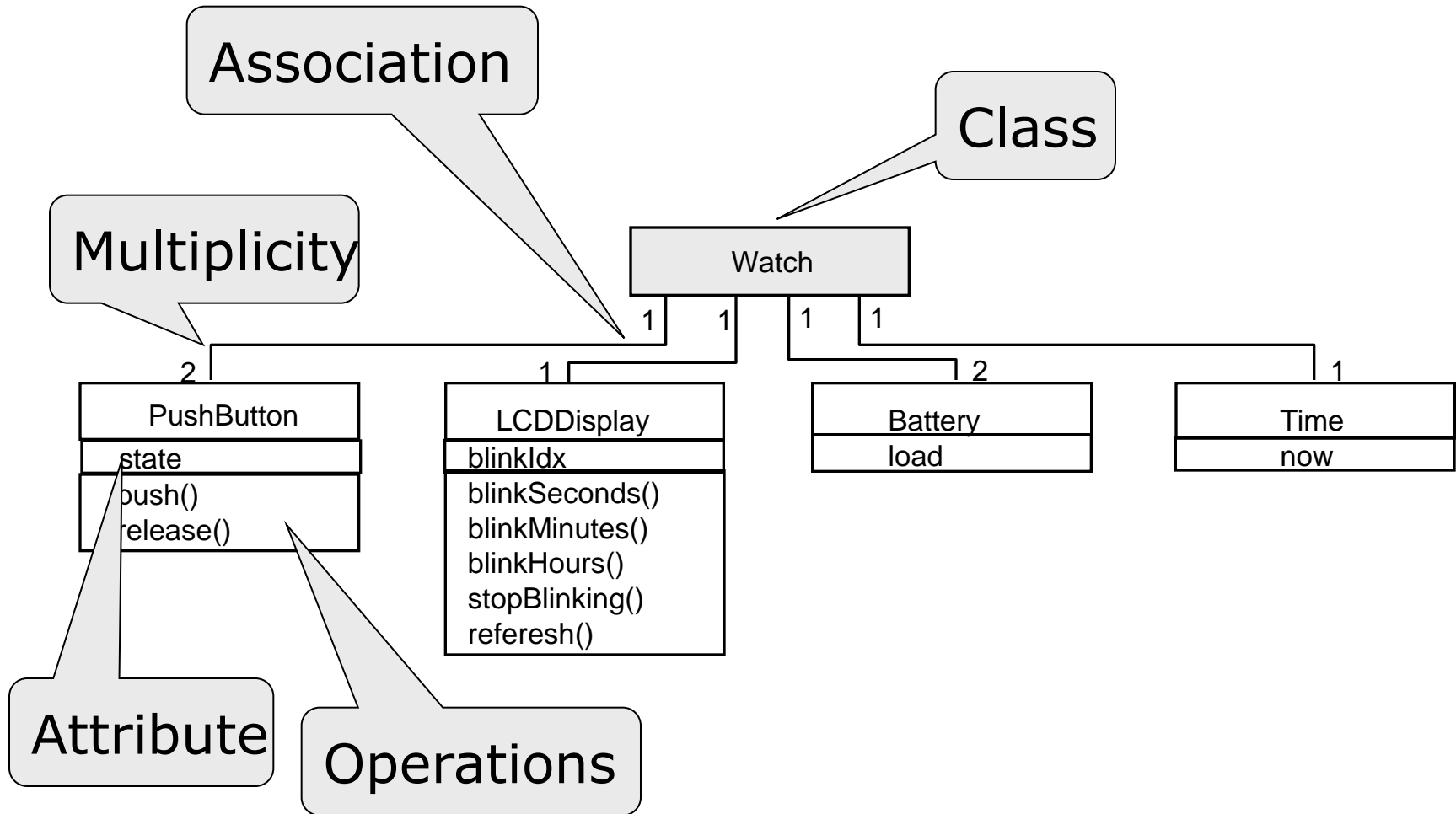
UML first pass: Use case diagrams



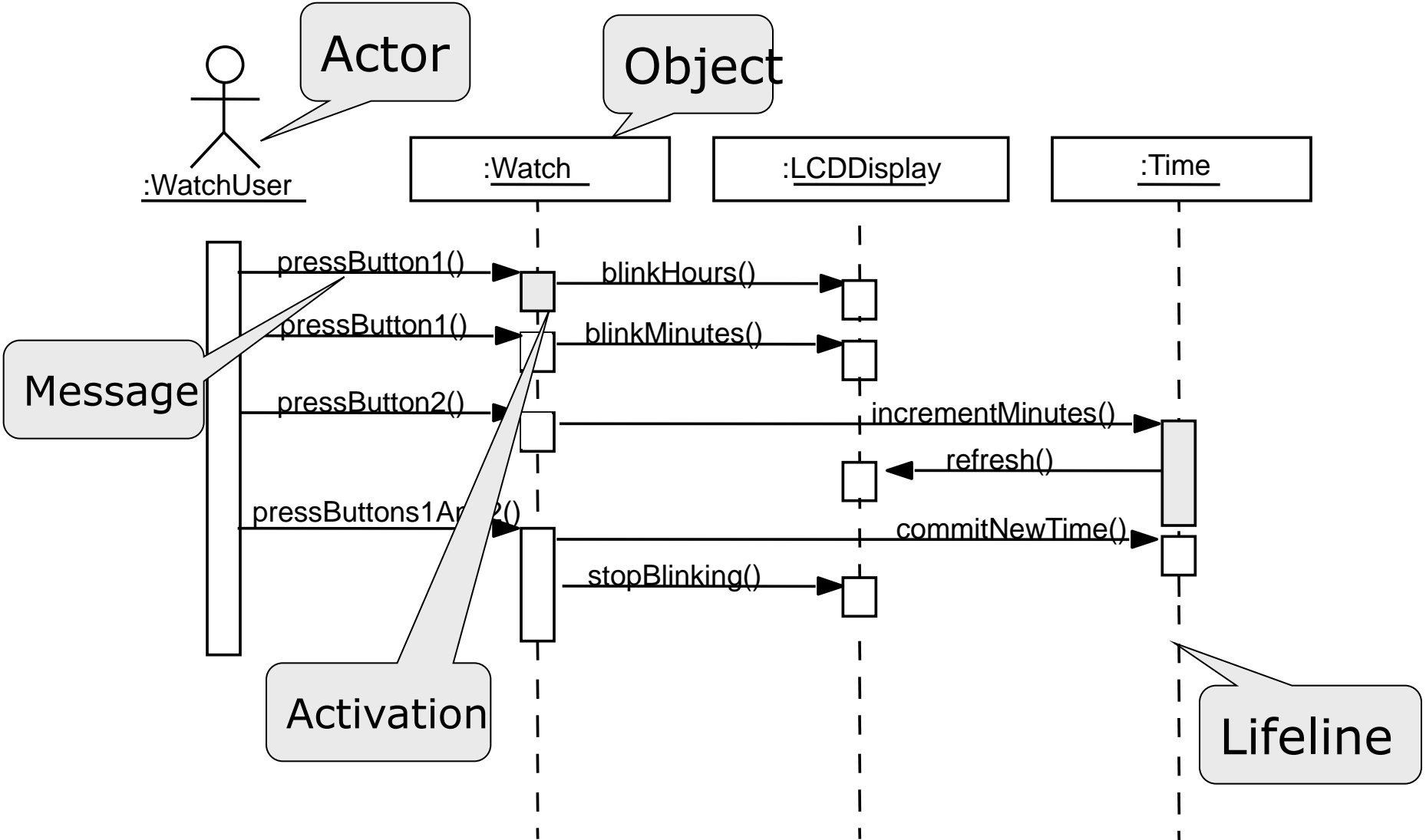
Use case diagrams represent the functionality of the system from user's point of view

UML first pass: Class diagrams

Class diagrams represent the structure of the system

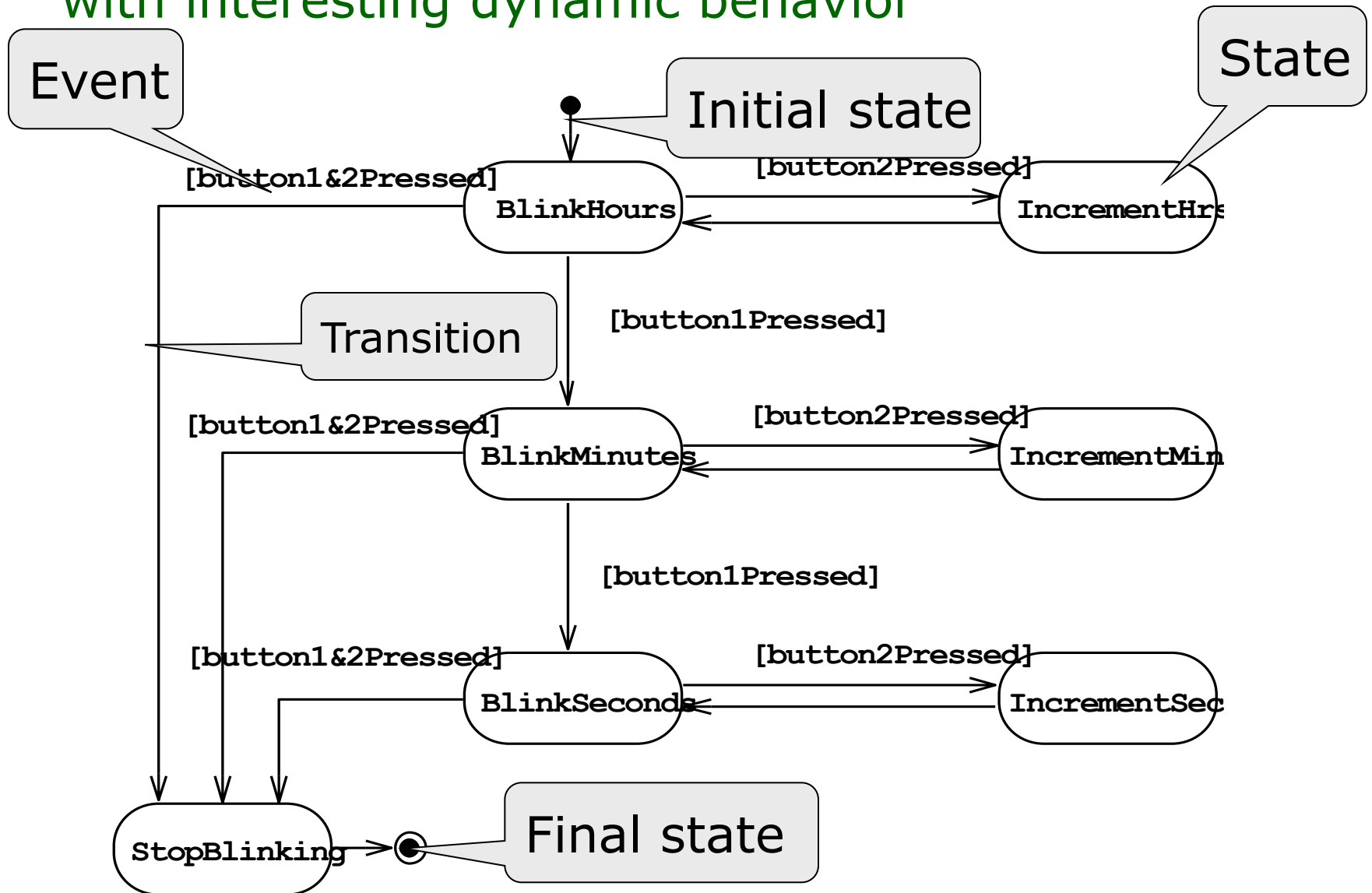


UML first pass: Sequence diagram



Sequence diagrams represent the behavior as interactions. Here it is giving details of incrementing minutes in setTime module

UML first pass: Statechart diagrams for objects with interesting dynamic behavior

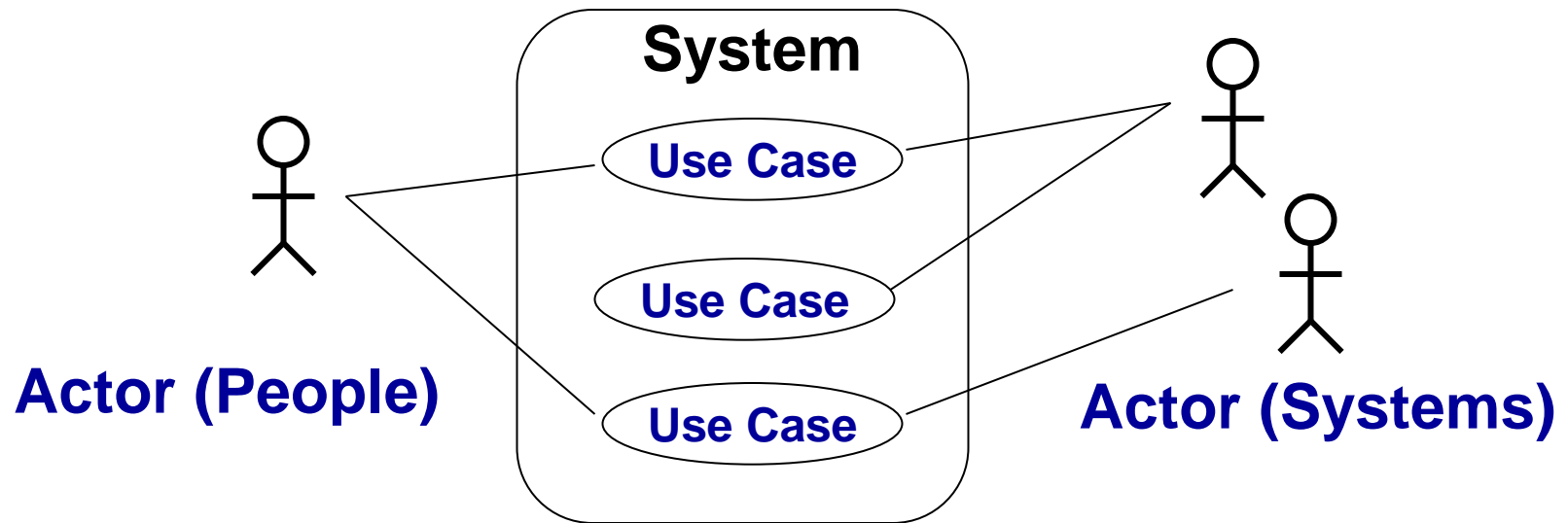


Represent behavior as states and transitions

Usecase diagram

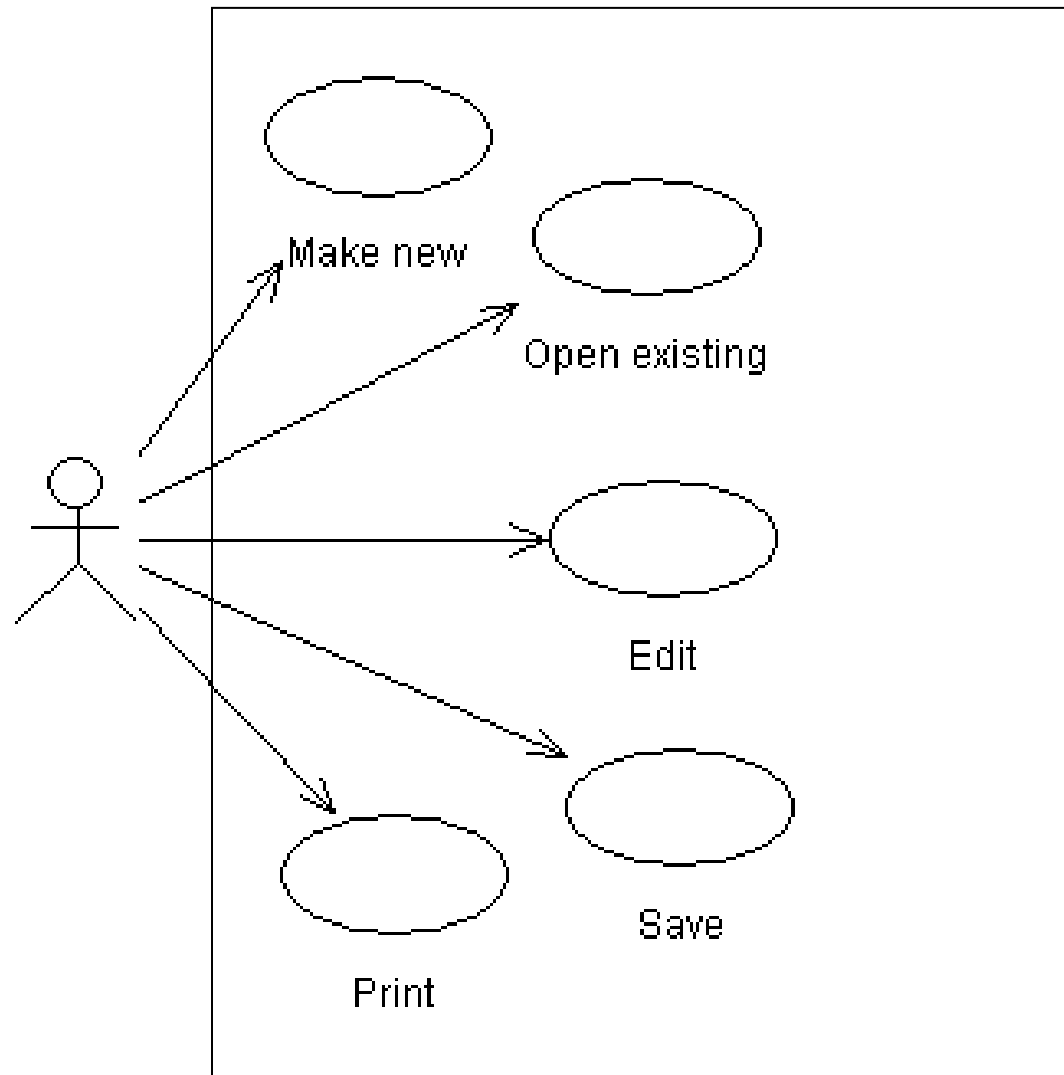
Use Cases

- Two types of Actors: Users and System administrators



Use case examples

(use cases for powerpoint.)

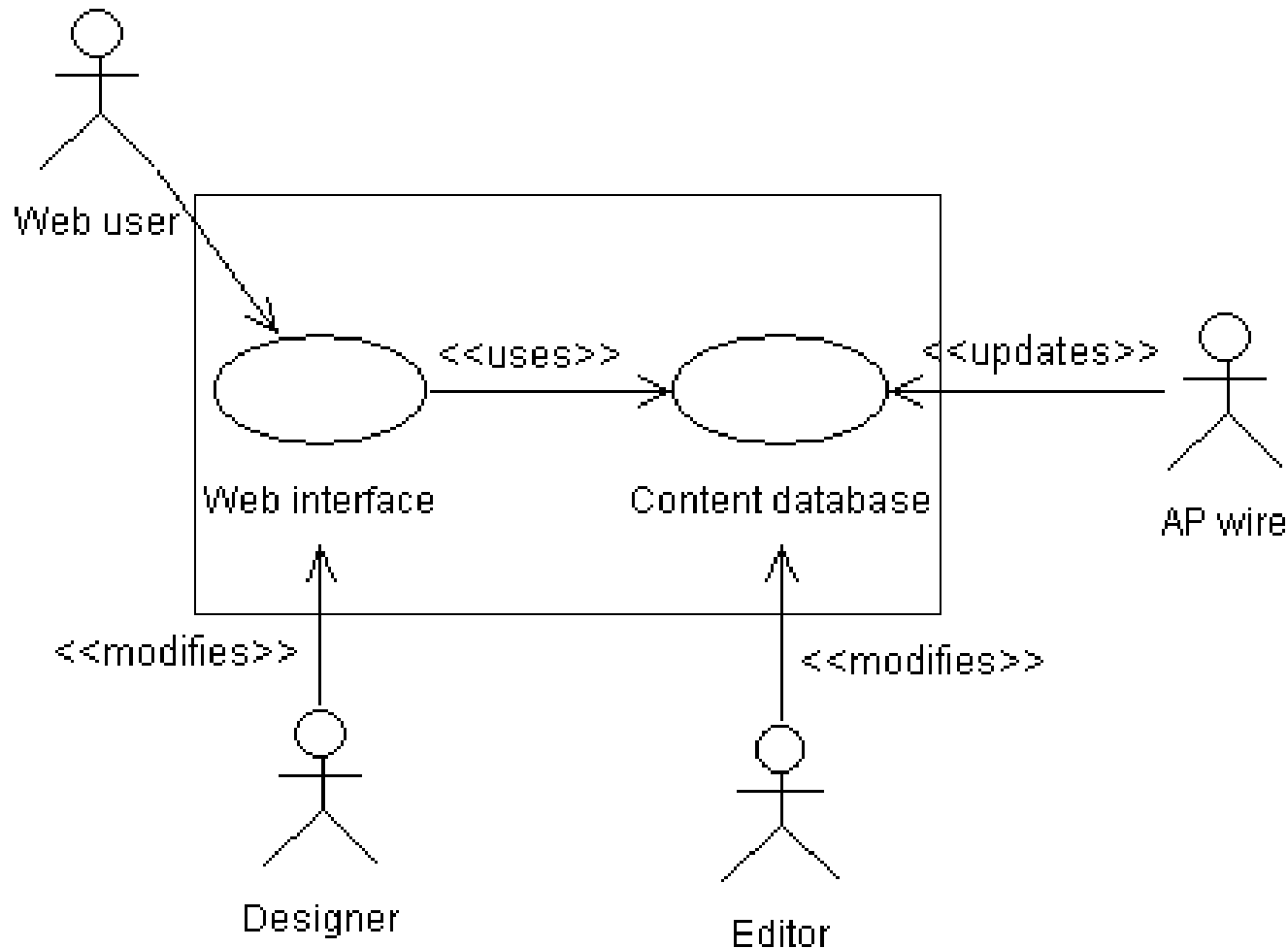


About the last example...

- Gives a view of powerpoint.
- focusses your attention to the key features

Use case examples

(Relationships in a news web site.)



About the last example...

- The last is more complicated and realistic use case diagram. It captures several key use cases for the system.
- Note the multiple actors. In particular, 'AP wire' is an actor, with an important interaction with the system, but is not a person (or even a computer system, necessarily).
- The notes between << >> marks are *stereotypes*: make the diagram more informative.

Usecase diagram

Give a Usecase diagram for an ATM machine:

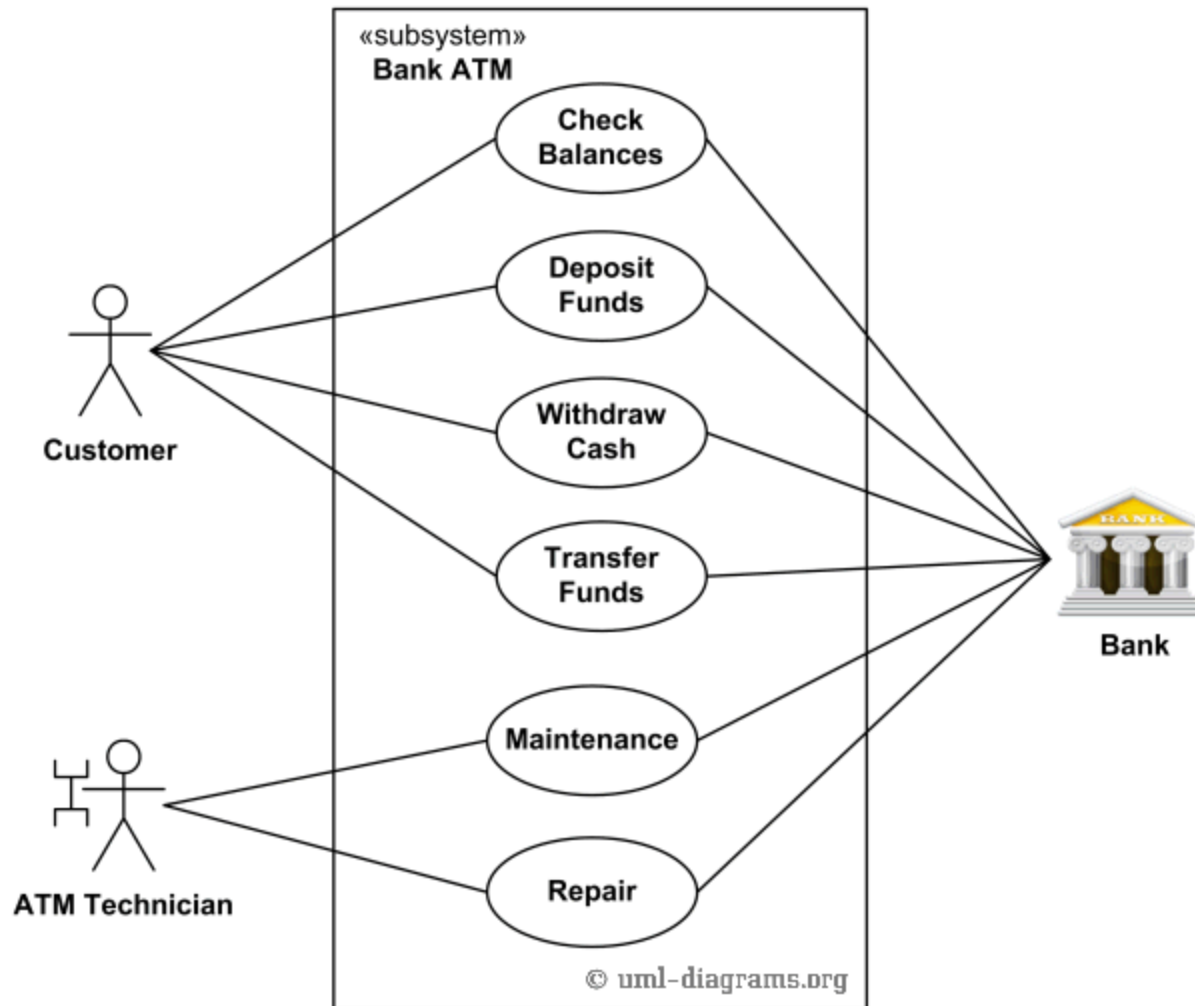
An automated teller machine (**ATM**) provides bank customers with access to financial transactions.

Customer uses bank ATM to *Check Balances* of his/her bank accounts, *Deposit Funds*, *Withdraw Cash* and/or *Transfer Funds*

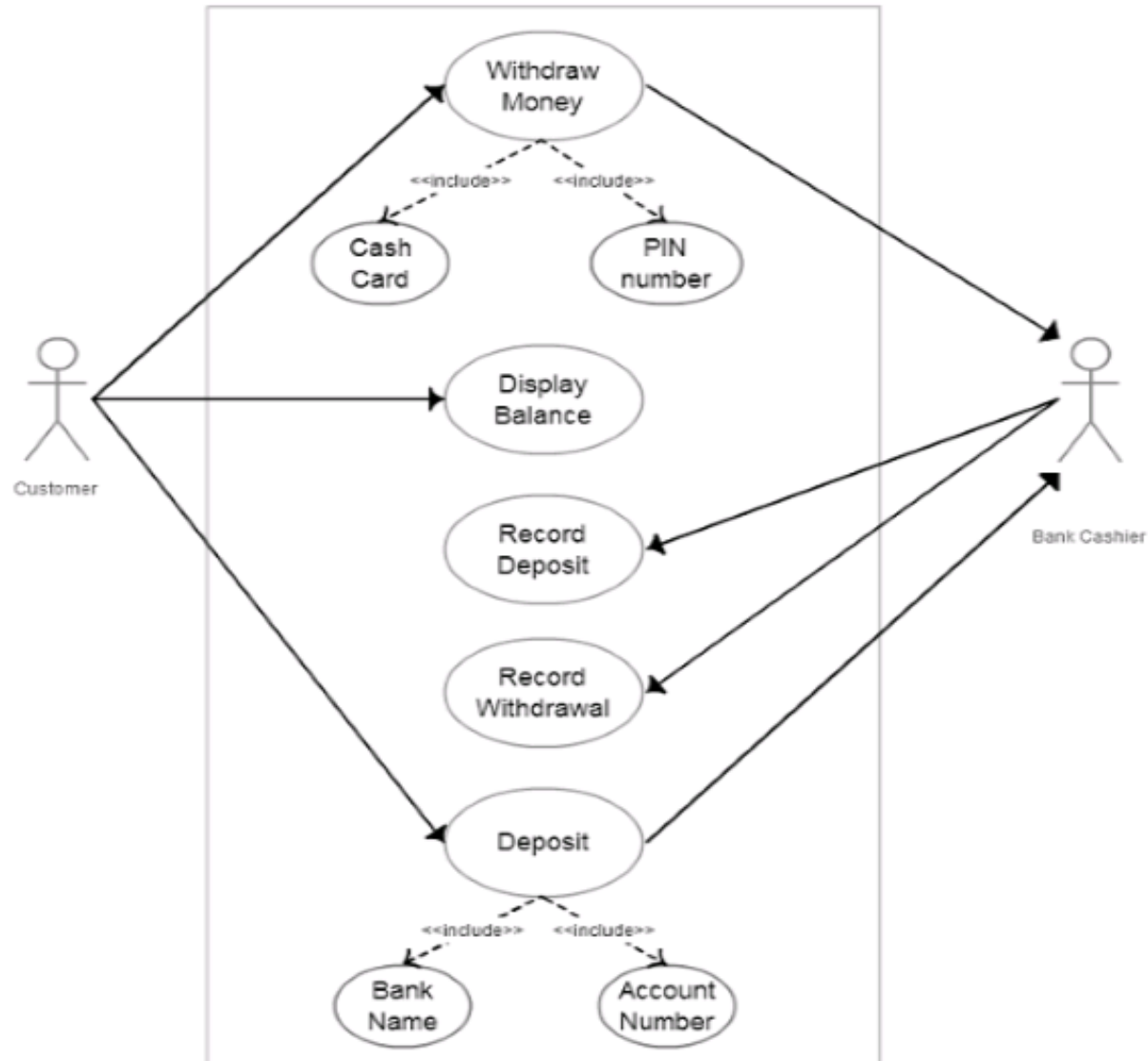
ATM Technician provides *Maintenance* and *Repairs*.

Bank actor: customer transactions or to the ATM servicing.

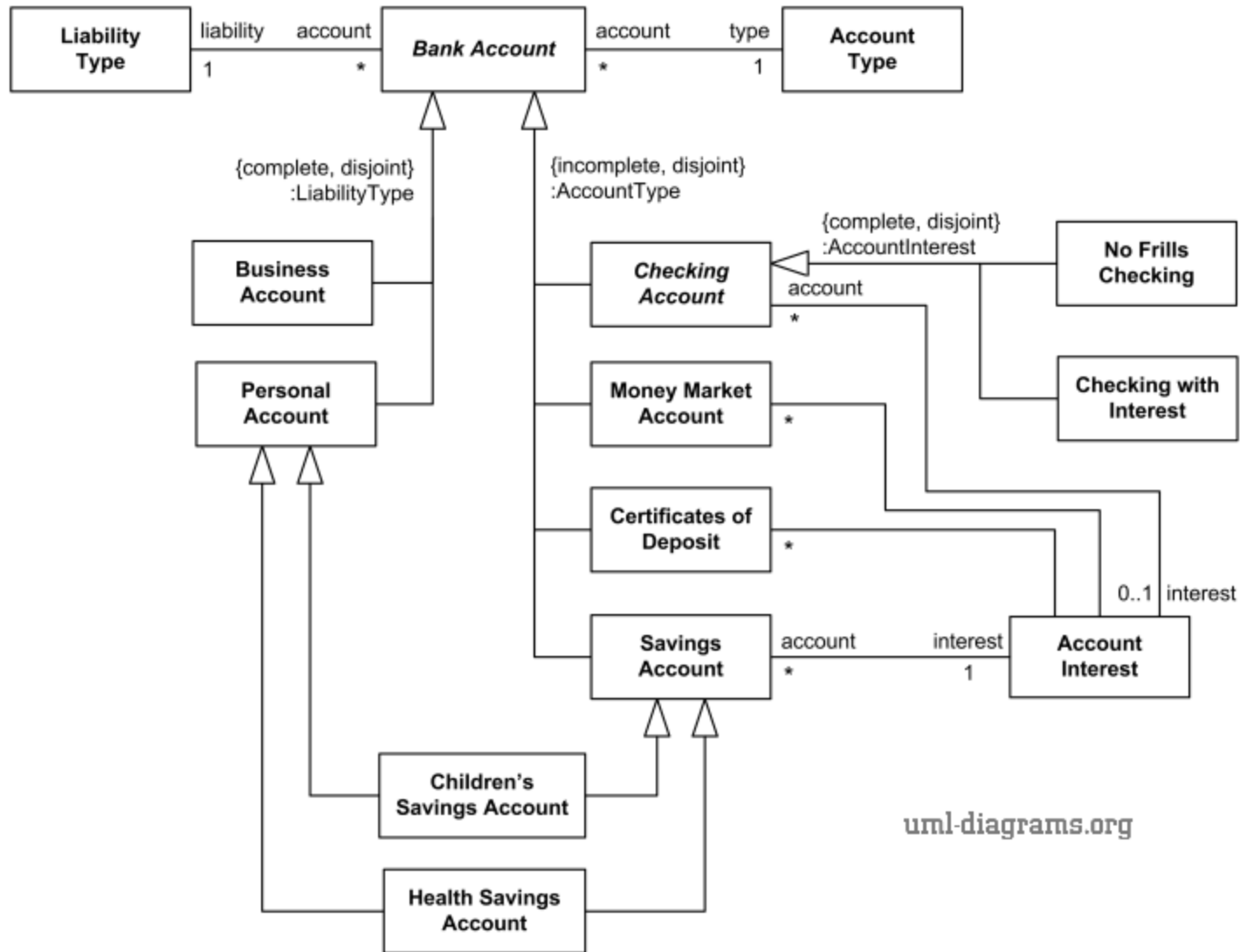
Usecase diagram for an ATM machine



Usecase diagram for an ATM machine



Class diagram



Practice problems

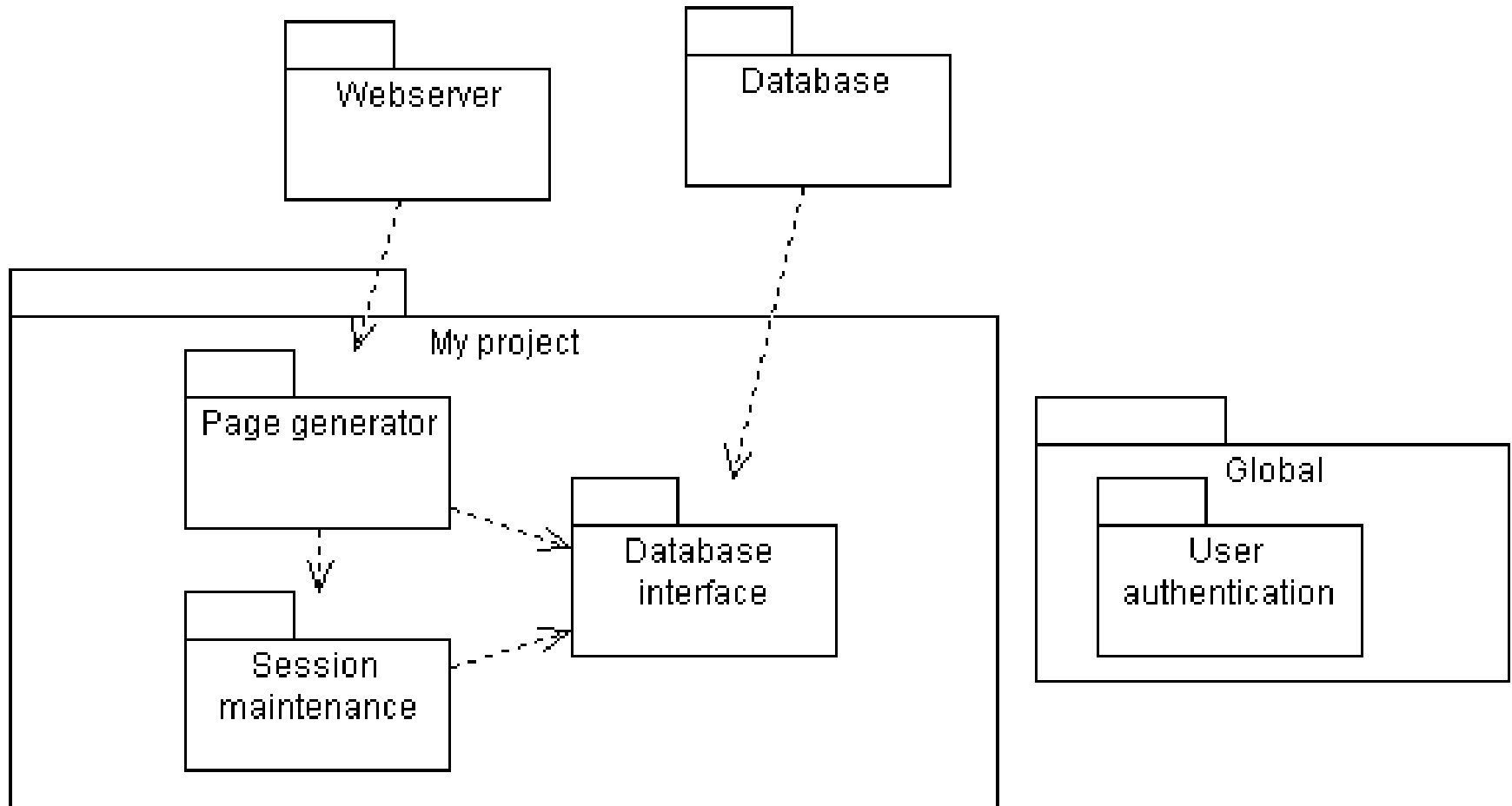
- Modify the Usecase diagram in Slide 7 to incorporate:
 - SetHours, setMinutes and setSeconds
 - SetTime will be a usecase which <<uses>> the usecases listed above
- Modify the usecase diagram in slide 18 to incorporate:
 - Two type of users: Bank customers and non-bank customers. Non-Bank customers can only withdraw and check balance
- Modify the Class diagram in slide 20 to make it possible to make code in Java. This requires multiple inheritance to be removed. Possible mechanism: Have multilayered hierarchy with duplication of classes.

Package diagram

Package diagram

- A type of class diagram, package diagrams show dependencies between high-level system component.
- A “package” is usually a collection of related classes, and will usually be specified by it’s own class diagram.

Package diagram example



About the last example...

- This package diagram indicates that:
 - there are three dependent but decoupled software components that will be developed in “My Project”, which is itself a package or component.
 - Parts of my software depend on some existing software packages, which I won’t be developing, but just using (“Webserver” and “Database”).
 - There is a globally available package “User authentication” which all the other packages depend on.


Object diagrams

Object Diagrams

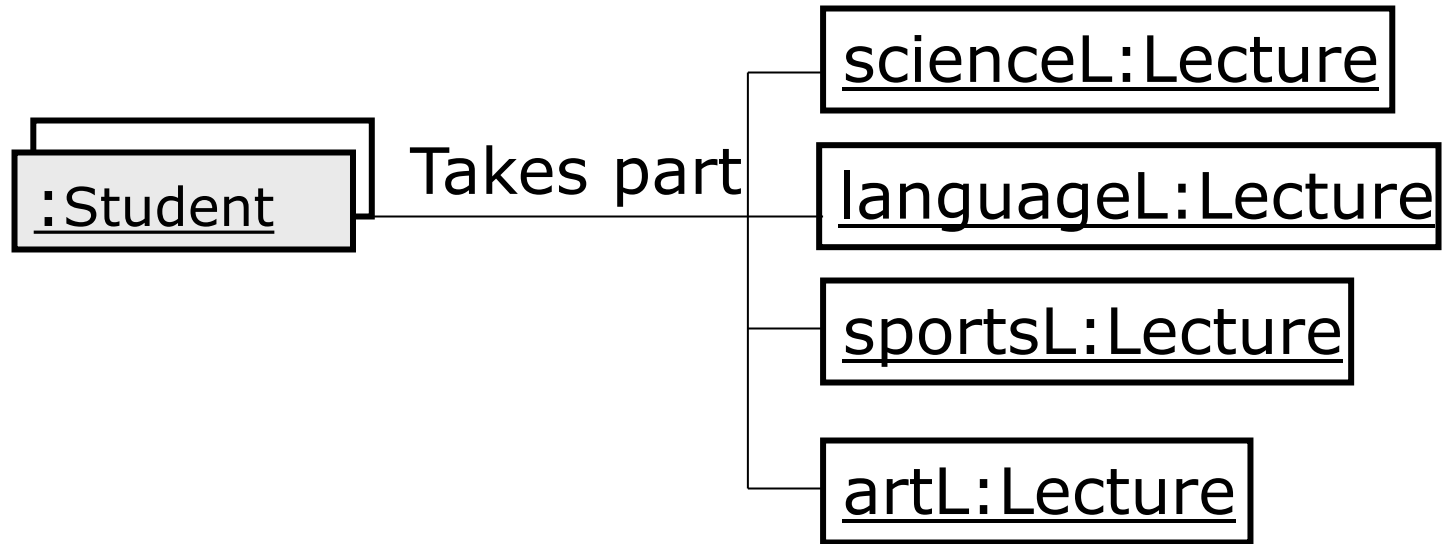
Class model describes all possible situations

Object model describes a particular situation.

Object Diagrams

- Objects are also represented in rectangles, their names underlined & written with first letter in small case.
- Names of objects can be written in 4 ways:
 - write only the class name preceded by a colon and underlined : Student
 - write the name of specific object with it's class oneStudent : Student
 - write only the object name oneStudent:
 - multiple objects :  : Student

Object Diagrams



Sequence diagram

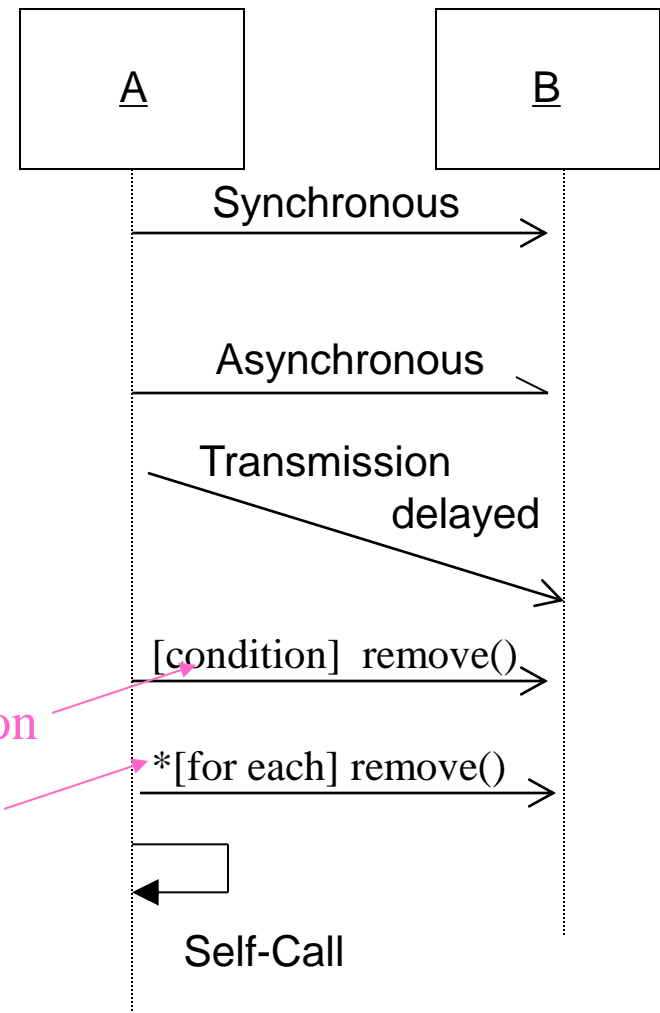
Sequence Diagram: Object interaction

Self-Call: A message that an Object sends to itself.

Condition: indicates when a message is sent. The message is sent only if the condition is true.

Condition

Iteration



Sequence Diagrams – Object Life Spans

■ Creation

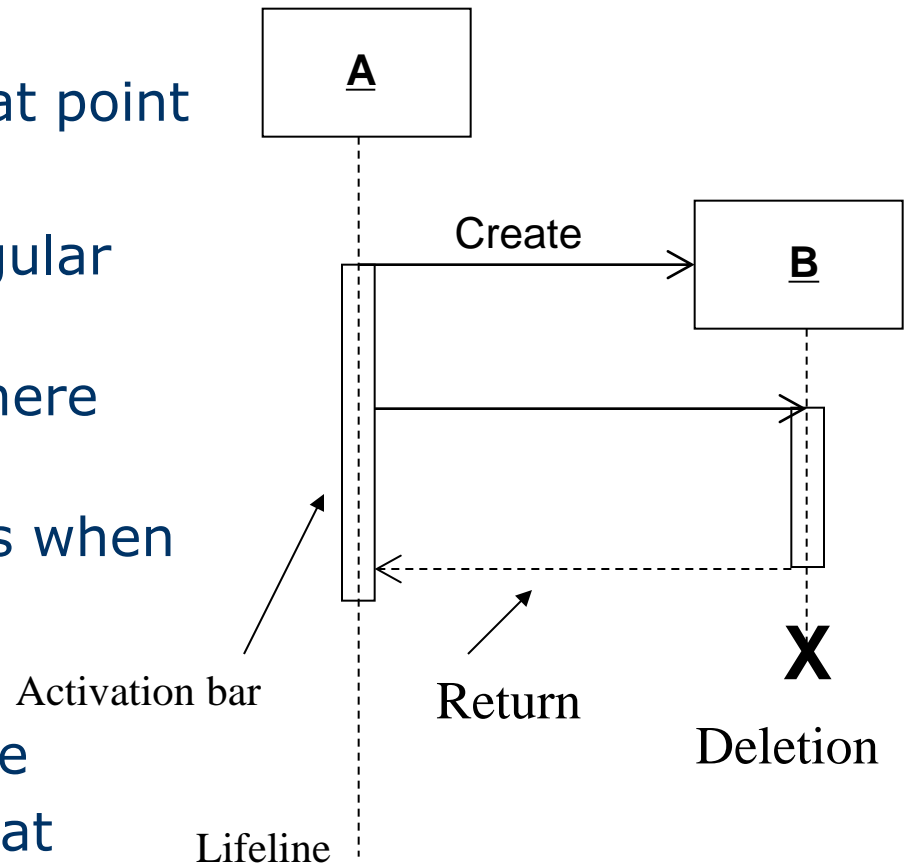
- Create message
- Object life starts at that point

■ Activation

- Symbolized by rectangular stripes
- Place on the lifeline where object is activated.
- Rectangle also denotes when object is deactivated.

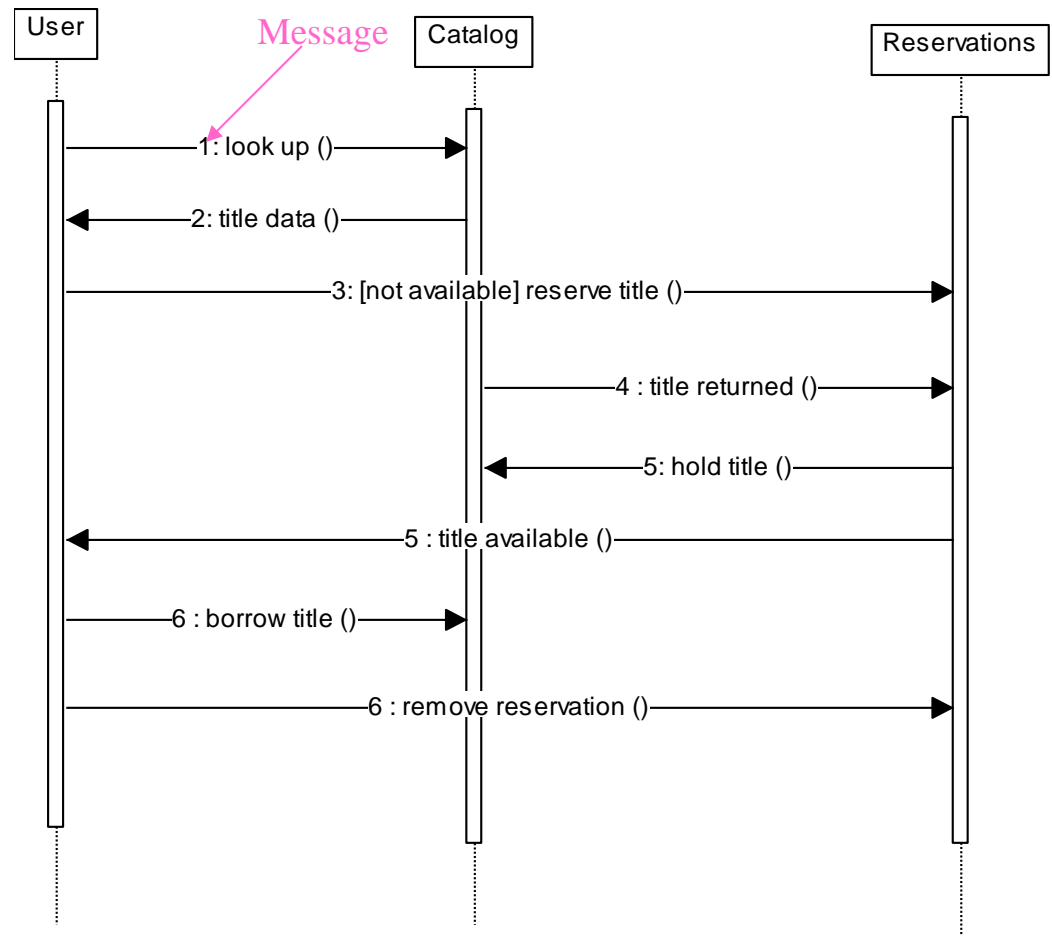
■ Deletion

- Placing an 'X' on lifeline
- Object's life ends at that point

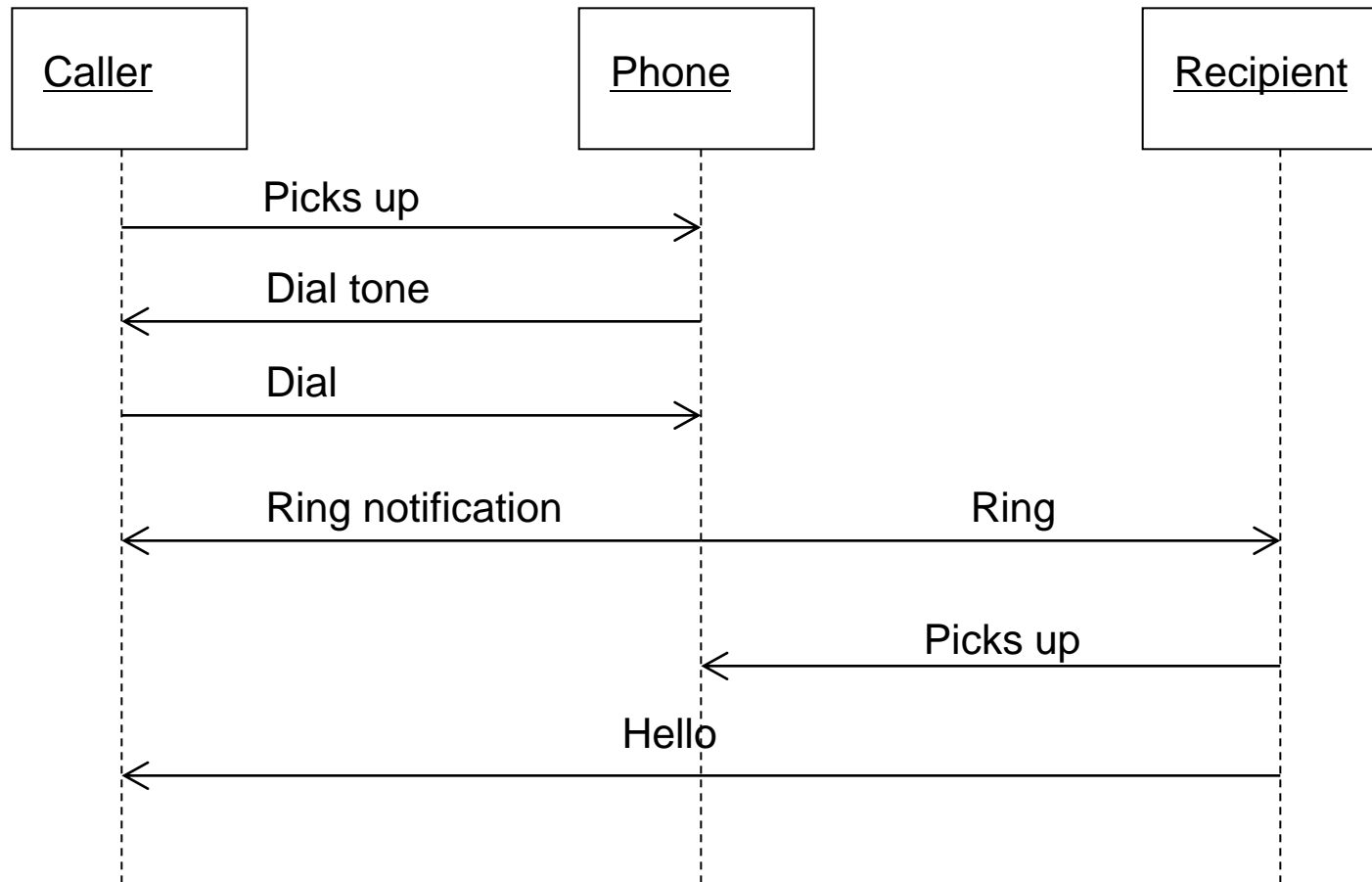


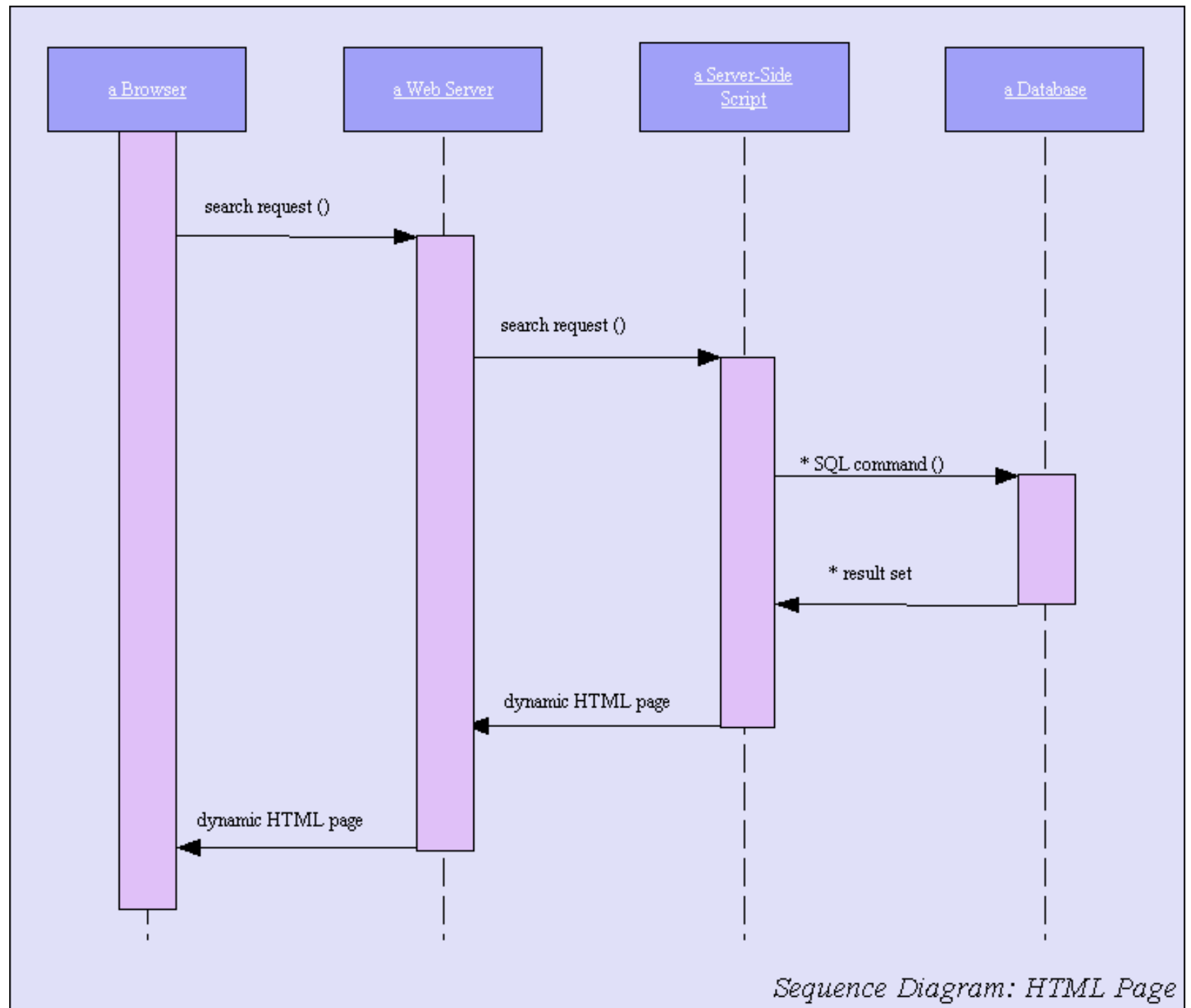
Sequence Diagram

- Sequence diagrams demonstrate the behavior of objects in a use case by describing the objects and the messages they pass.
- The horizontal dimension shows the objects participating in the interaction.
- The vertical arrangement of messages indicates their order.
- The labels may contain the seq. # to indicate concurrency.

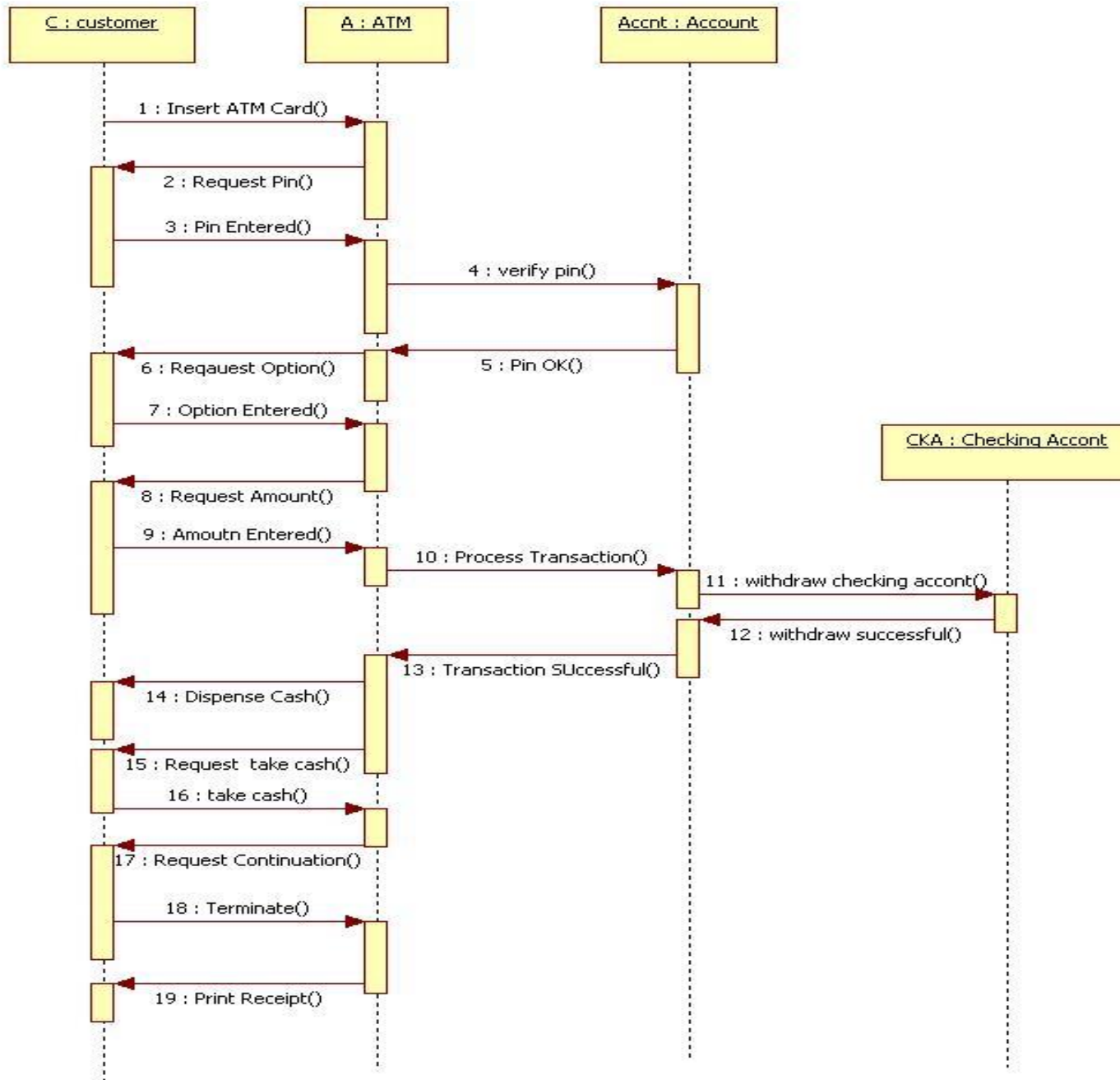


Sequence Diagram(make a phone call)

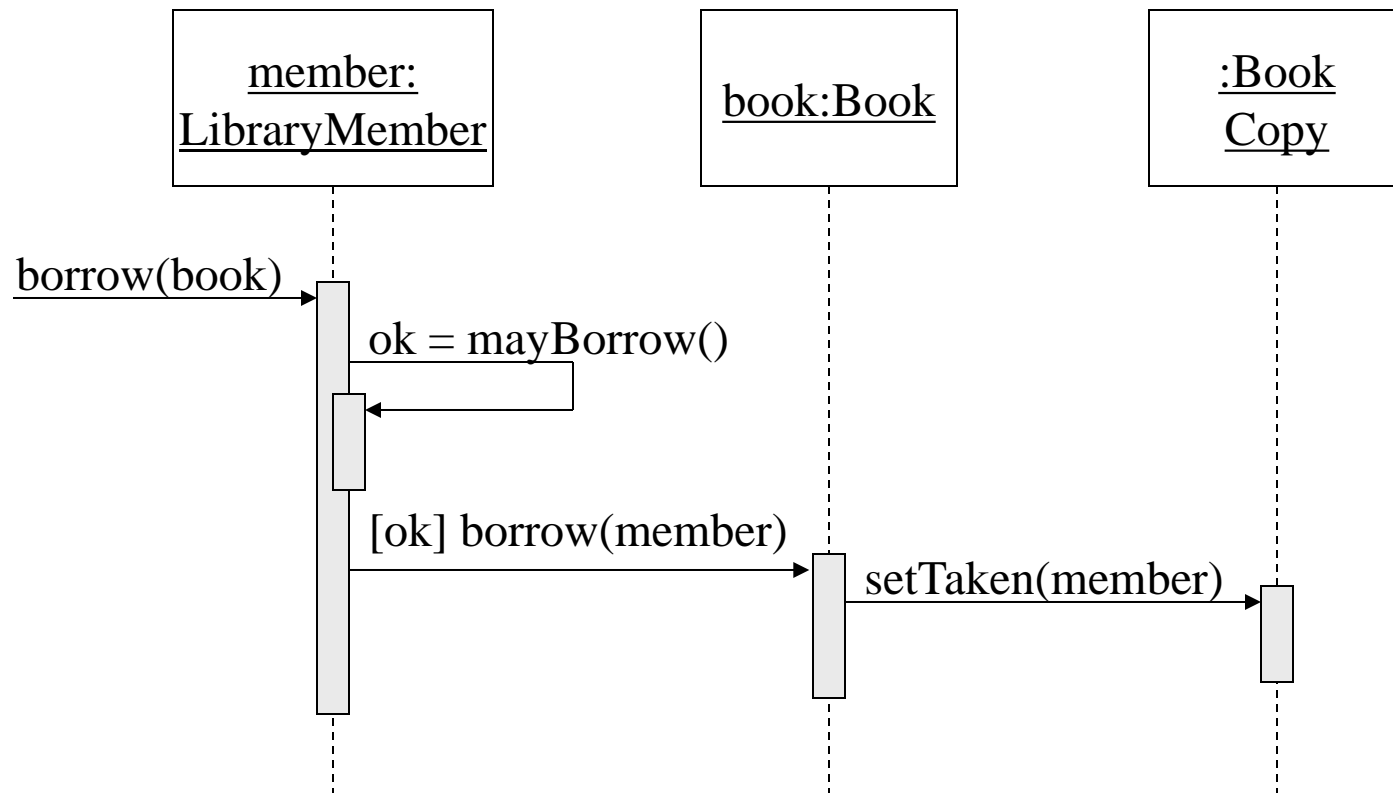




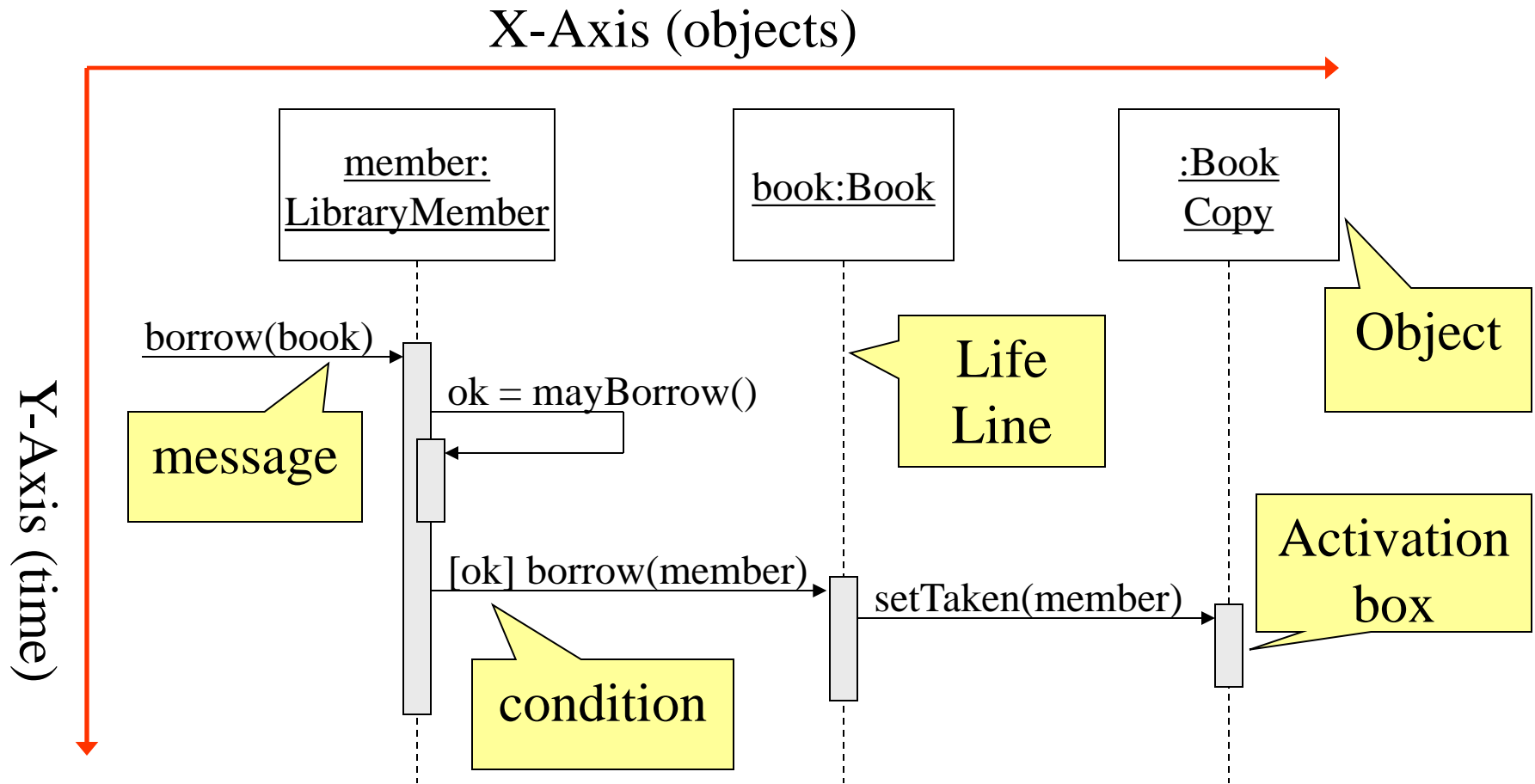
Sequence Diagram: HTML Page



A Sequence Diagram

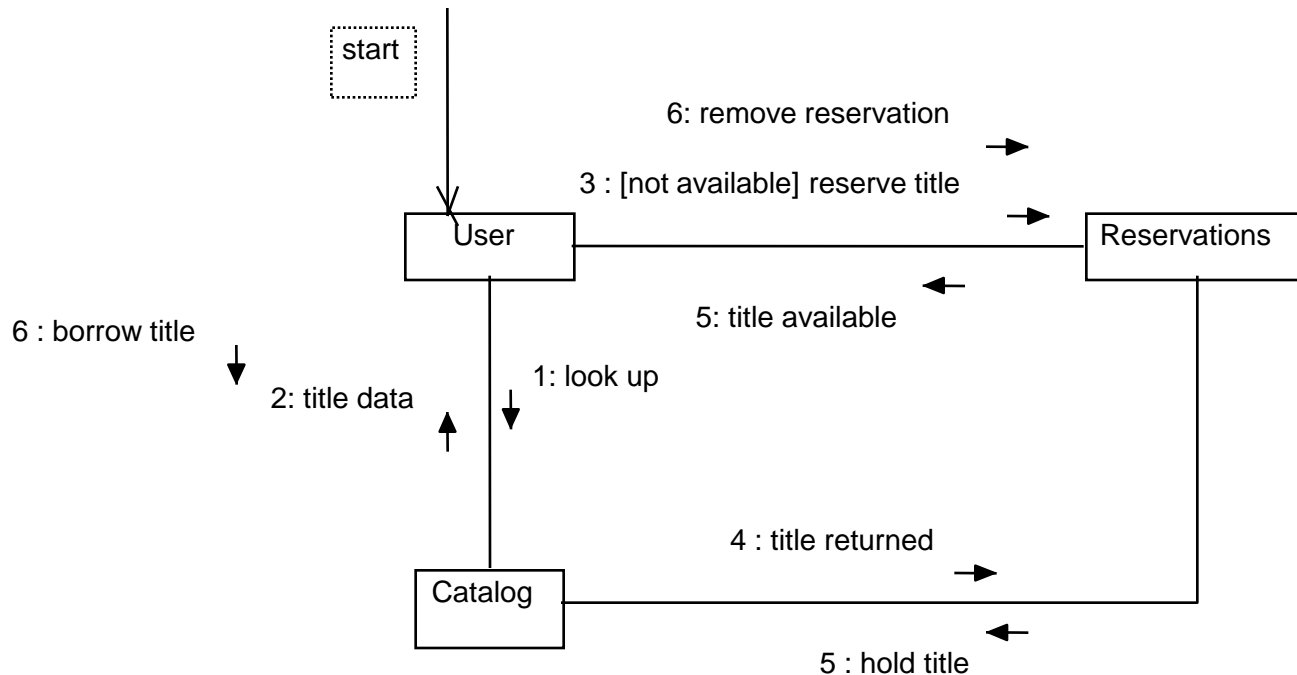


A Sequence Diagram



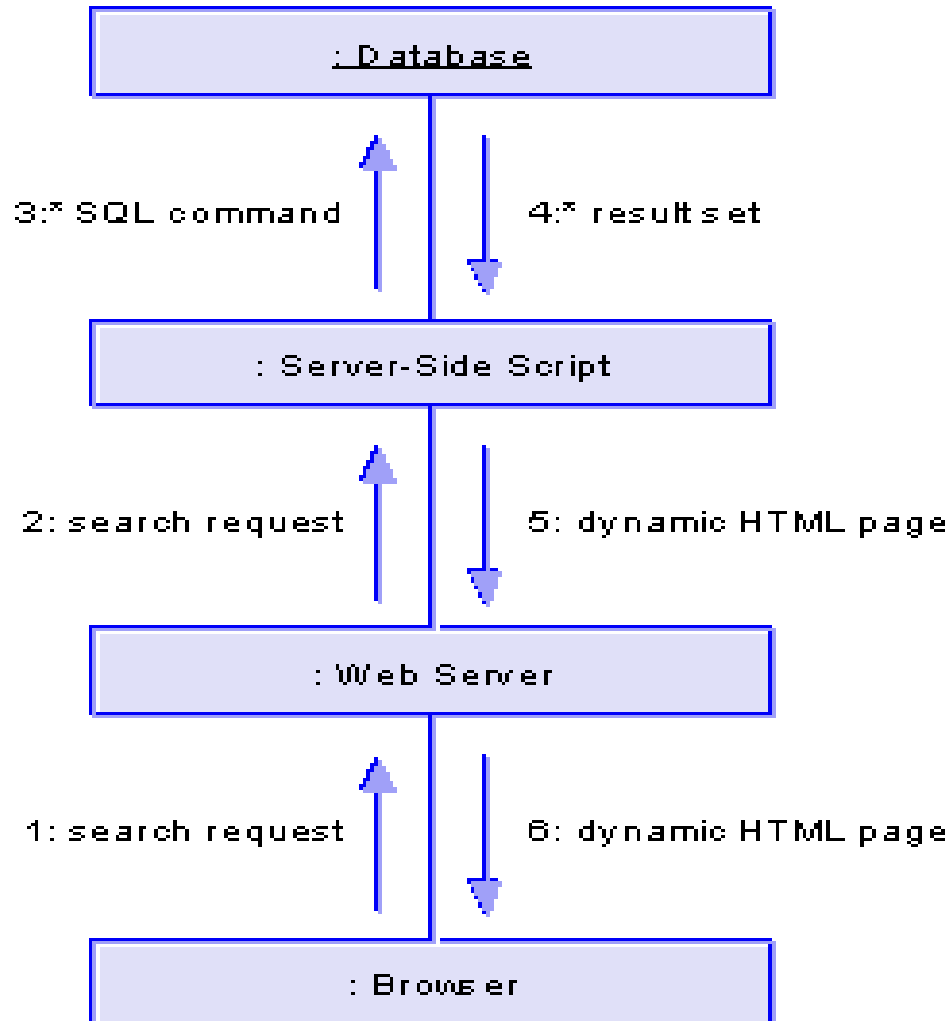
Collaboration diagram

Collaboration diagrams

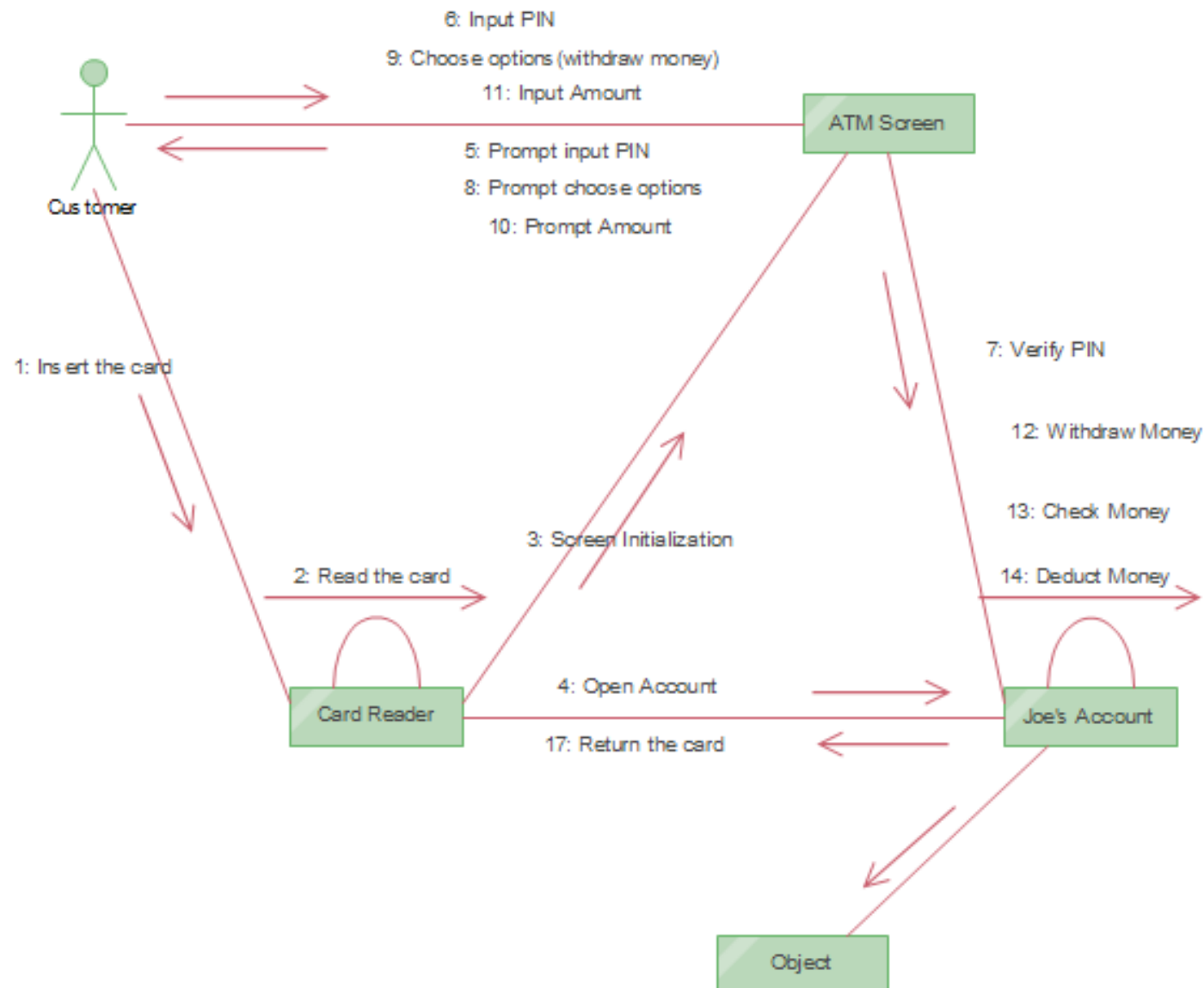


- Collaboration diagrams are equivalent to sequence diagrams. All the features of sequence diagrams are equally applicable to collaboration diagrams
- Use a sequence diagram when the transfer of information is the focus of attention
- Use a collaboration diagram when concentrating on the classes

Collaboration Diagram : Database to Browser



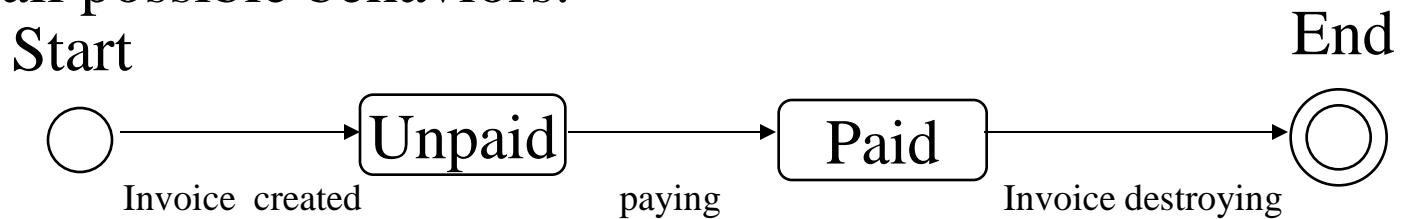
ATM UML Collaboration Diagram



Statechart diagram

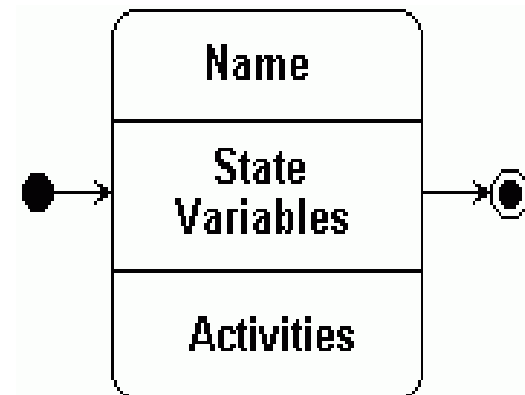
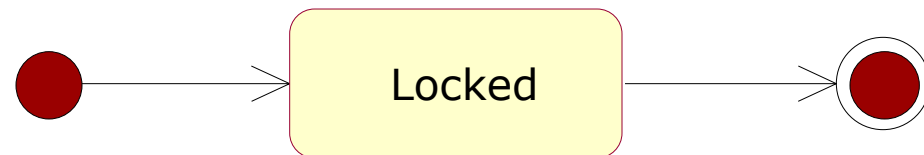
State Diagrams (Billing Example)

State Diagrams show the sequences of states an object goes through during its life cycle in response to stimuli, together with its responses and actions; an abstraction of all possible behaviors.



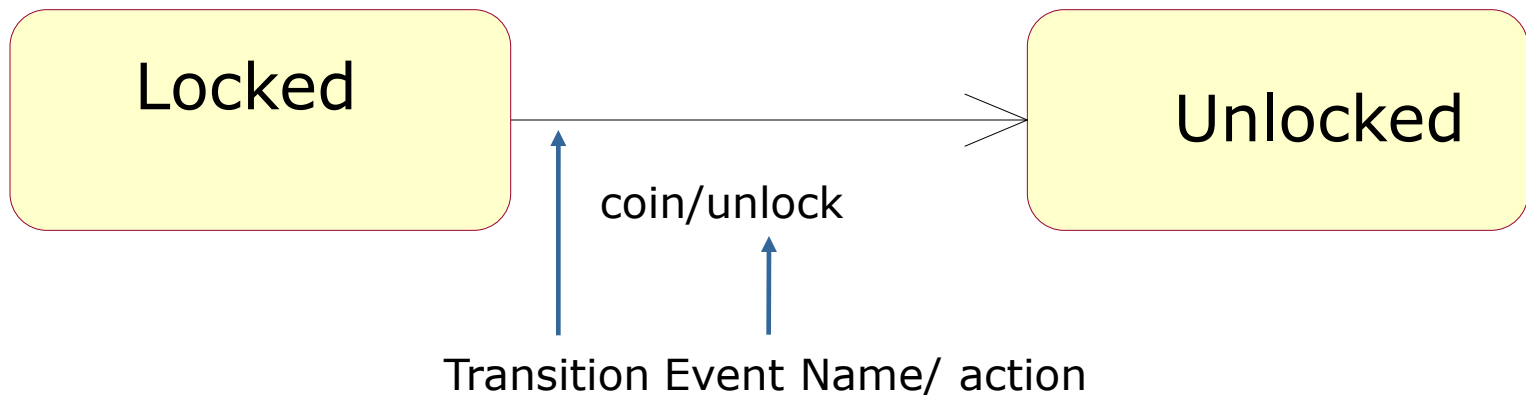
Special States

- The initial state is the state entered when an object is created.
 - An initial state is mandatory.
 - Only one initial state is permitted.
 - The initial state is represented as a solid circle.
- A final state indicates the end of life
 - A final state is optional.
 - A final state is indicated by a bull's eye.
 - More than one final state may exist.



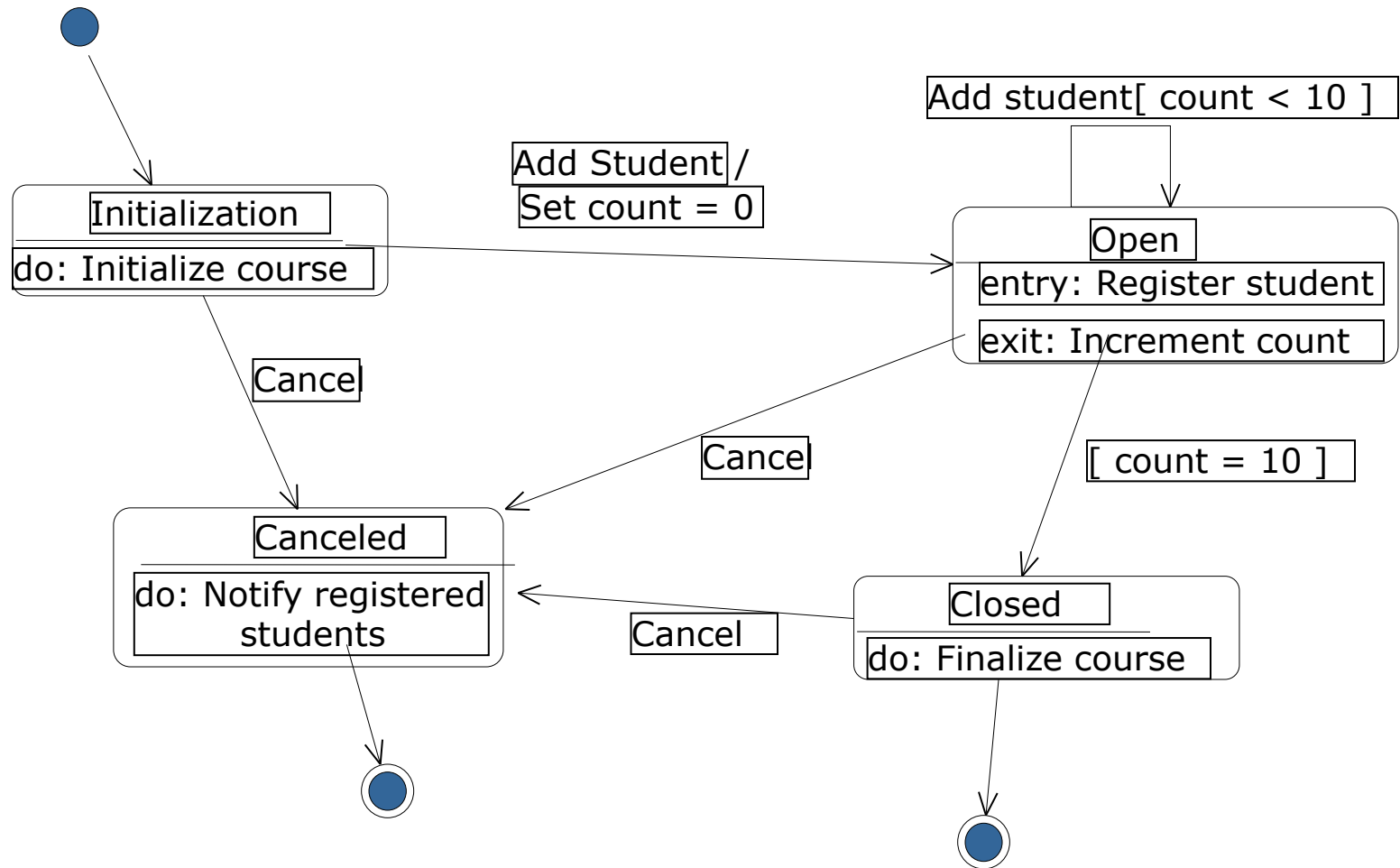
Events, Actions & Transitions

- An event: stimulus that can trigger a state transition.
- A transition: is a change from an originating state to a successor state as a result of some stimulus.
 - The successor state could possibly be the originating state.
- A transition may take place in response to an event.
- Transitions can be labeled with event names.

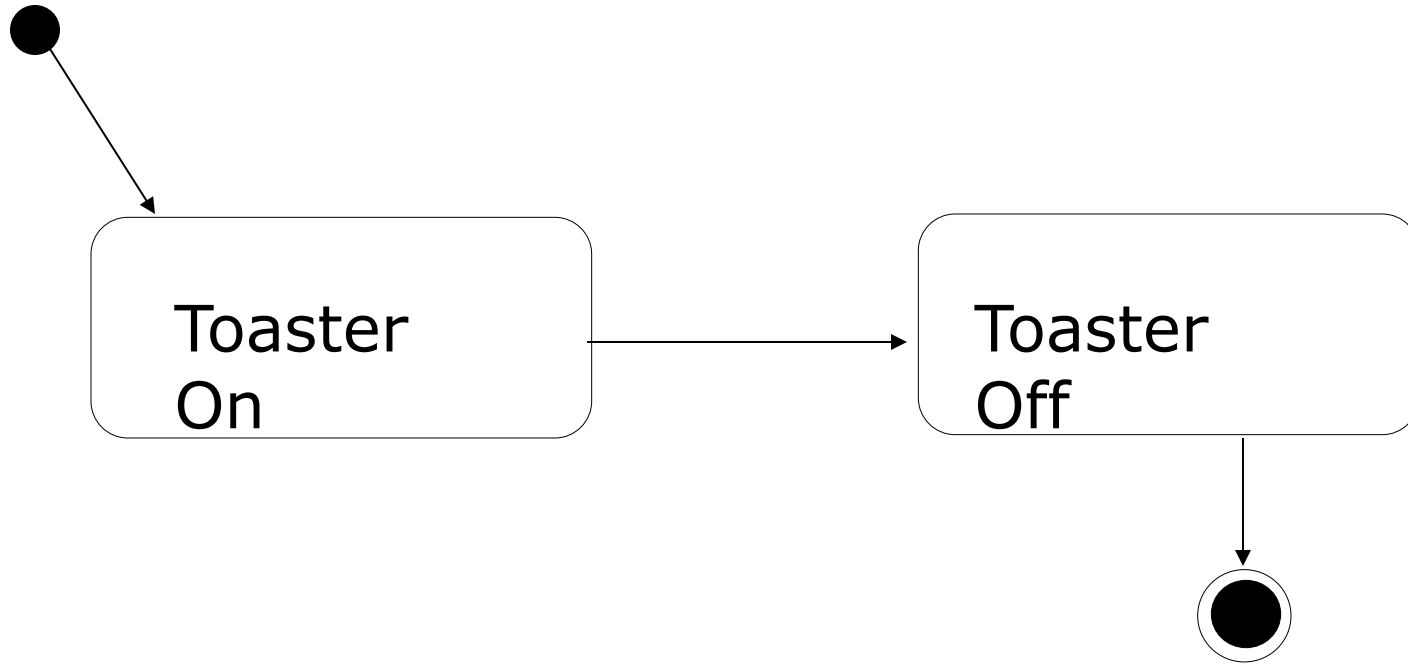


Statechart Diagram

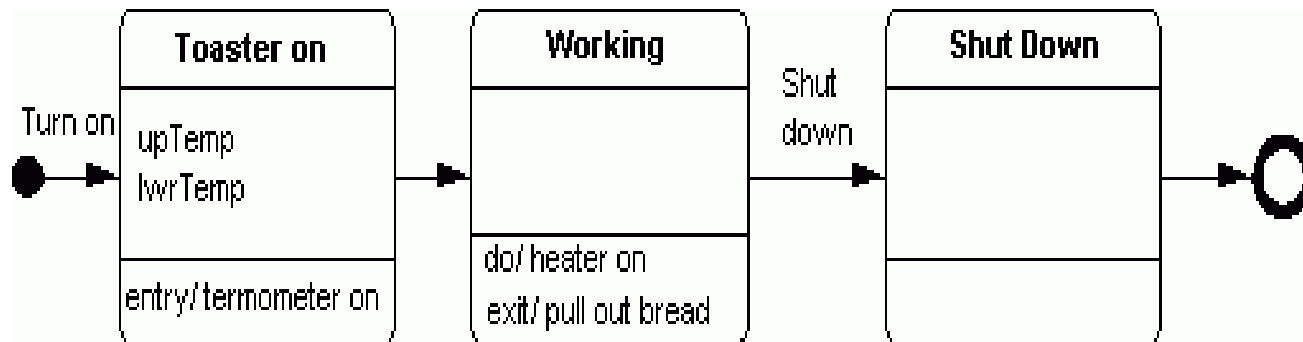
- A statechart diagram shows the lifecycle of a single class/course.



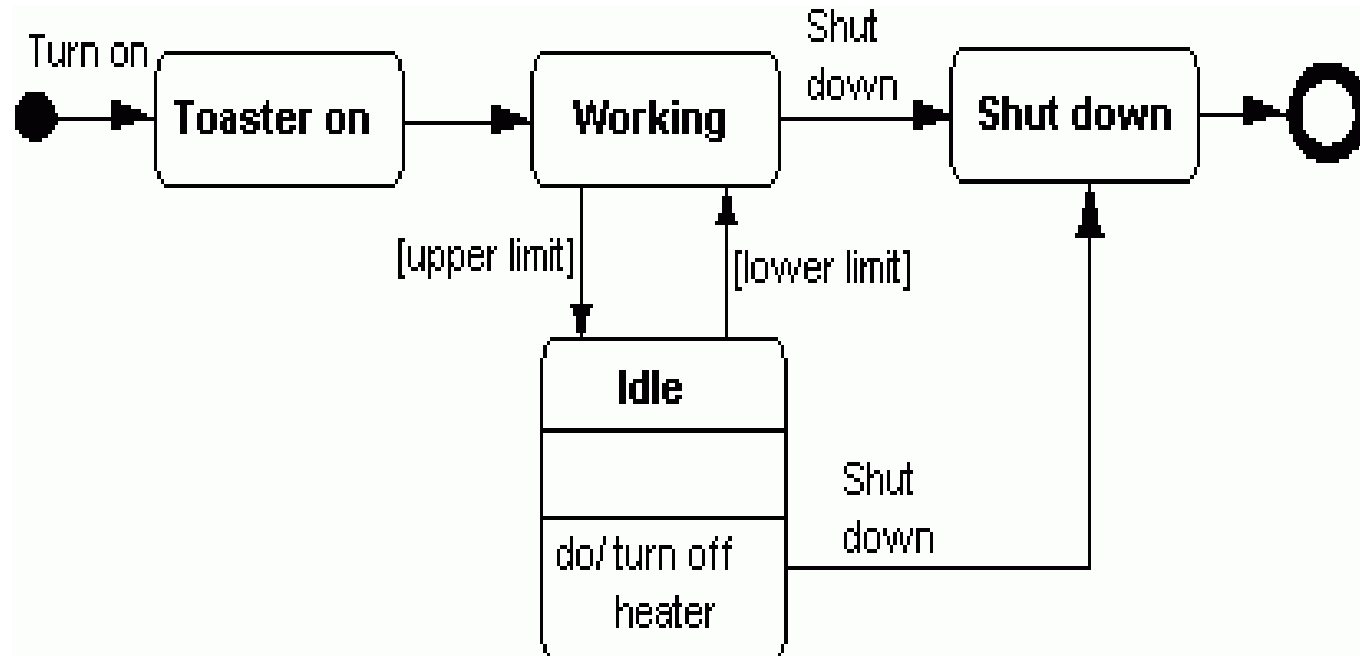
Example: Toaster



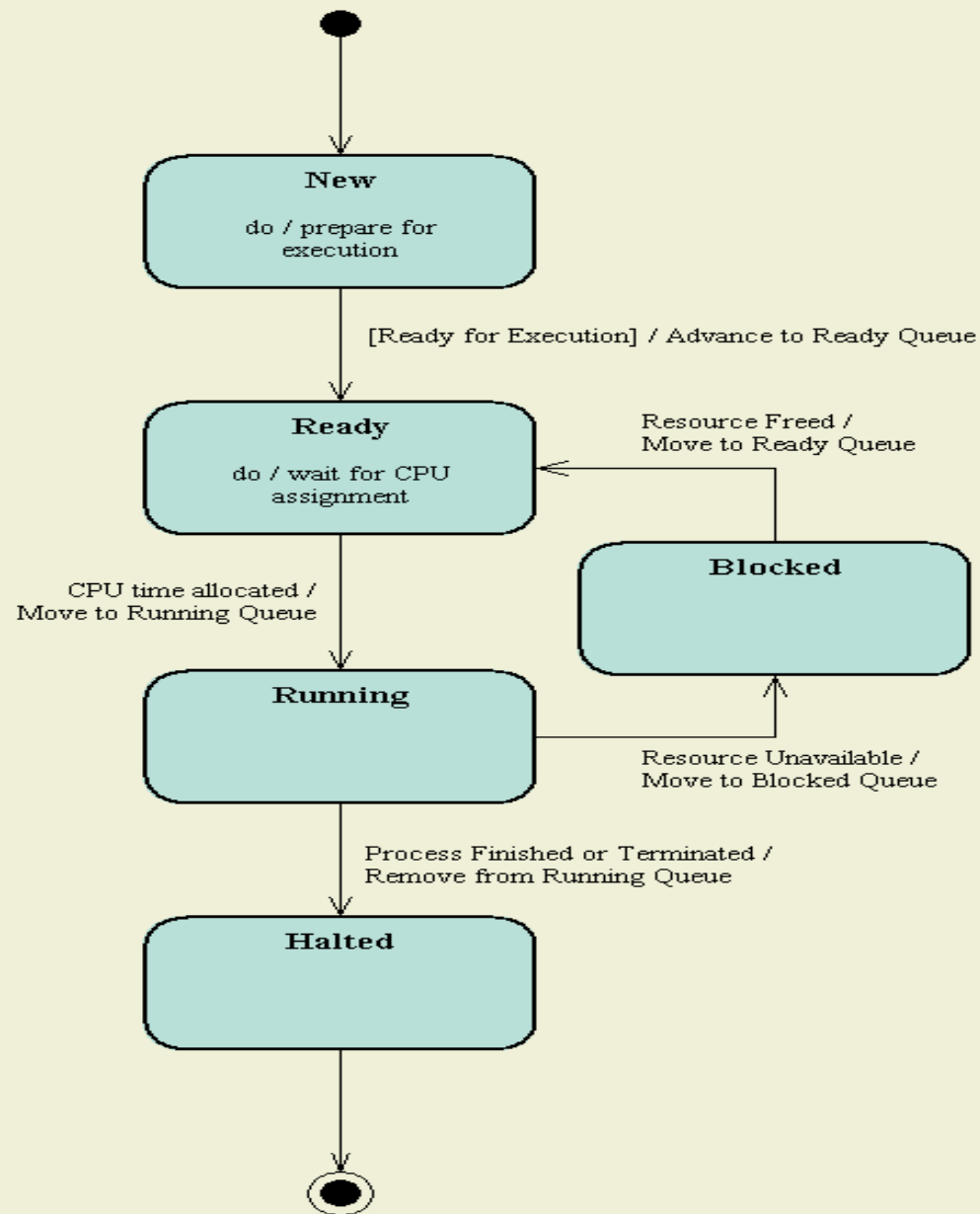
Example: Toaster



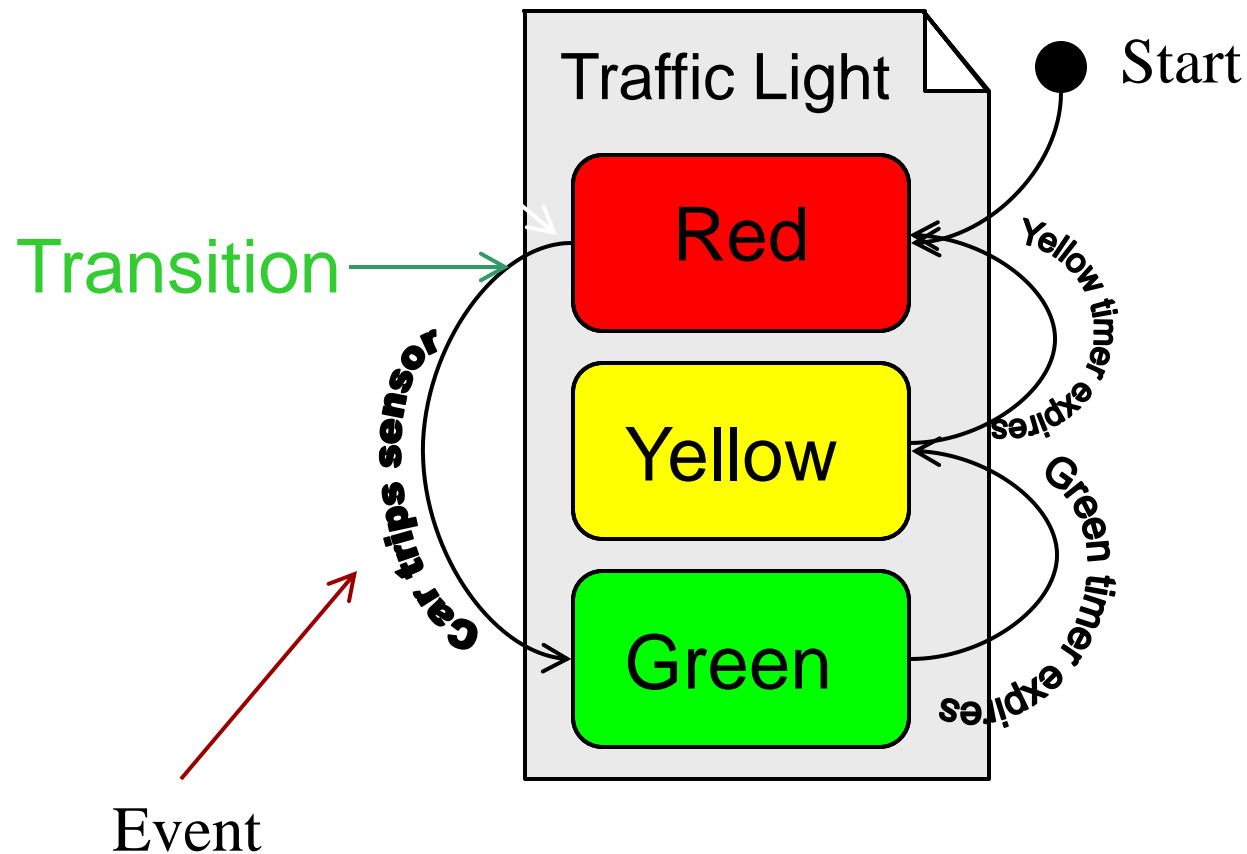
Example: Toaster



STATE DIAGRAM : CPU EXECUTION



State Diagrams (Traffic light example)

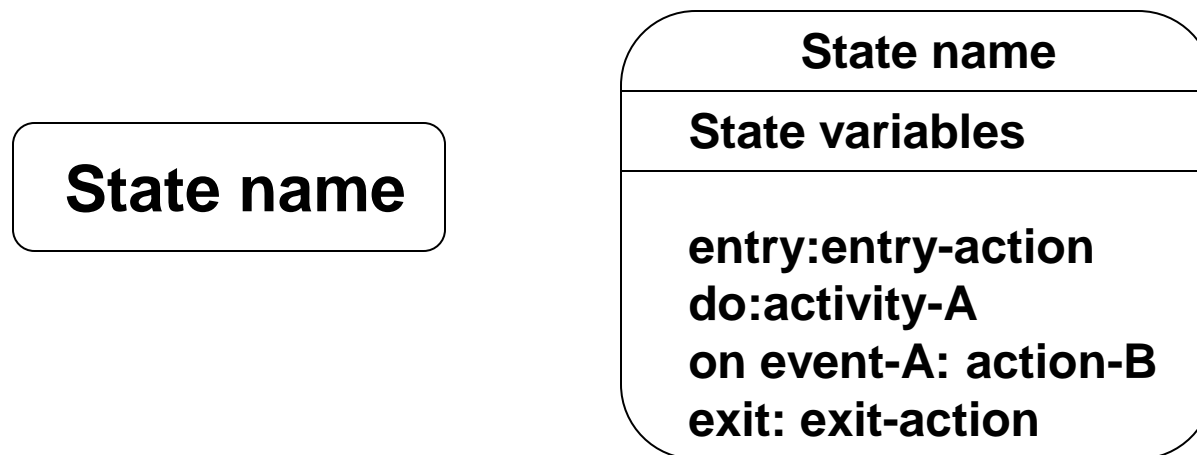


State Diagrams

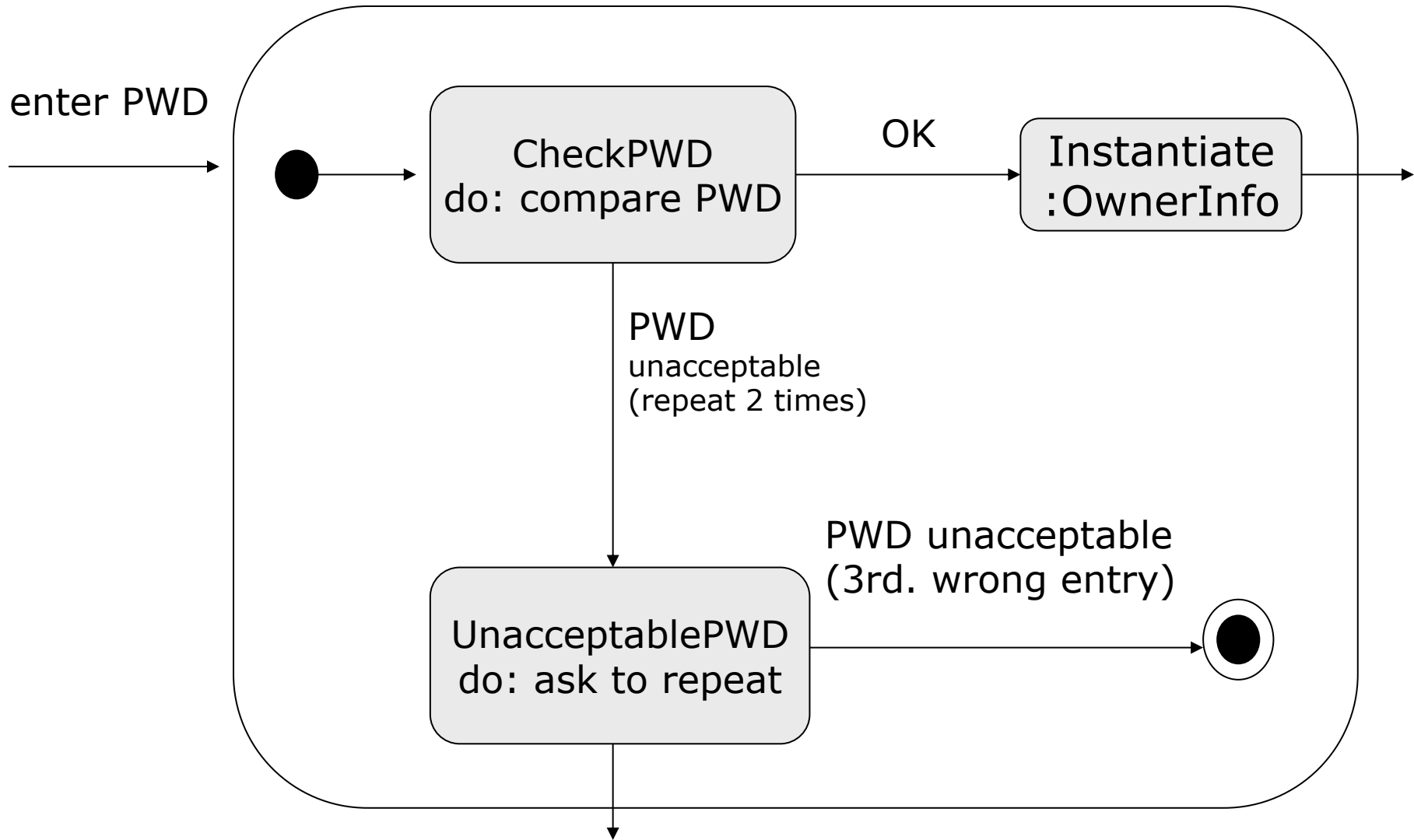


Reserved actions:

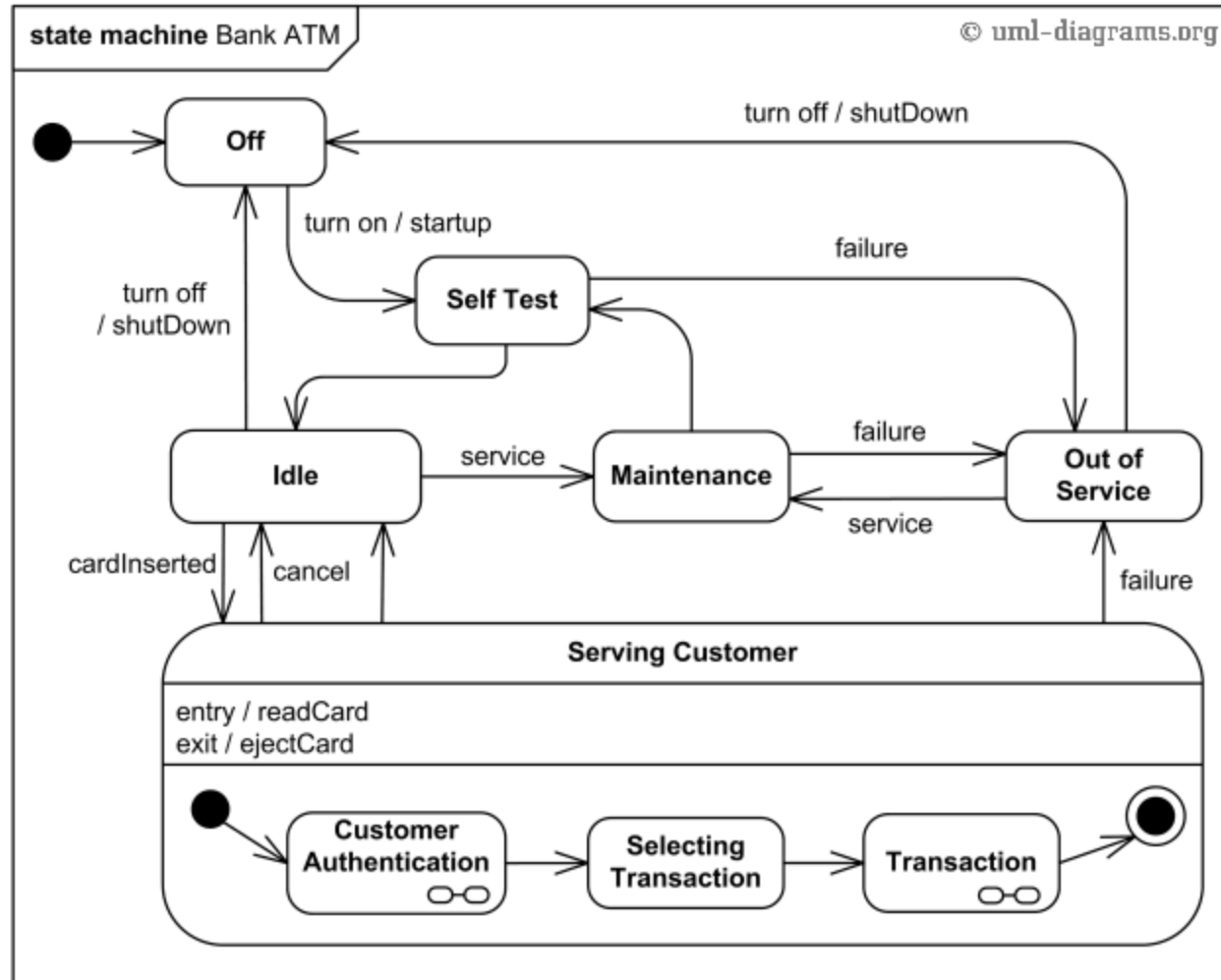
- **entry:** a specific action performed on the entry to the state
- **do:** an ongoing action performed while in the state
- **on:** a specific action performed as a result of a specific event
- **Exit:** a specific action performed on exiting the state



State Diagram Example



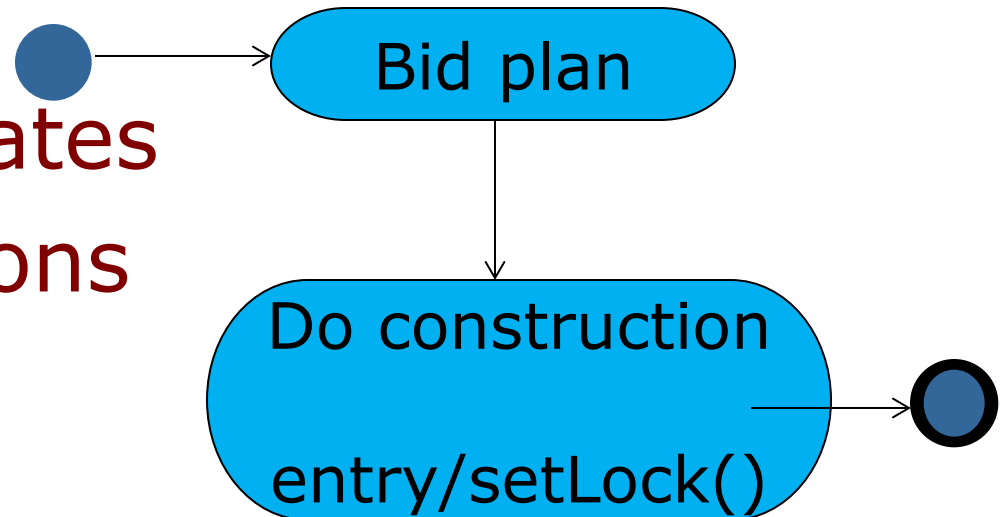
State Diagram Example



State Diagram Carryovers

The following items are common to **state diagrams and activity diagrams**:

- activities
- actions
- transitions
- initial/final states
- guard conditions



Breaking Up Flows

alternate paths:

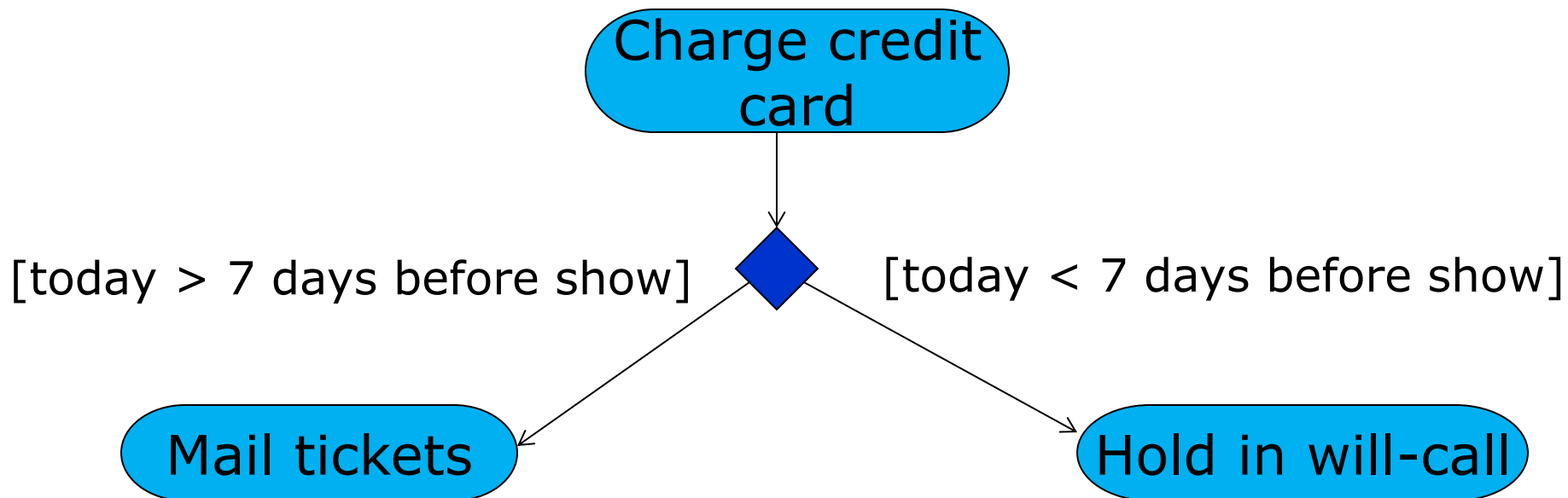
- branch
- merge

parallel flows:

- fork
- join

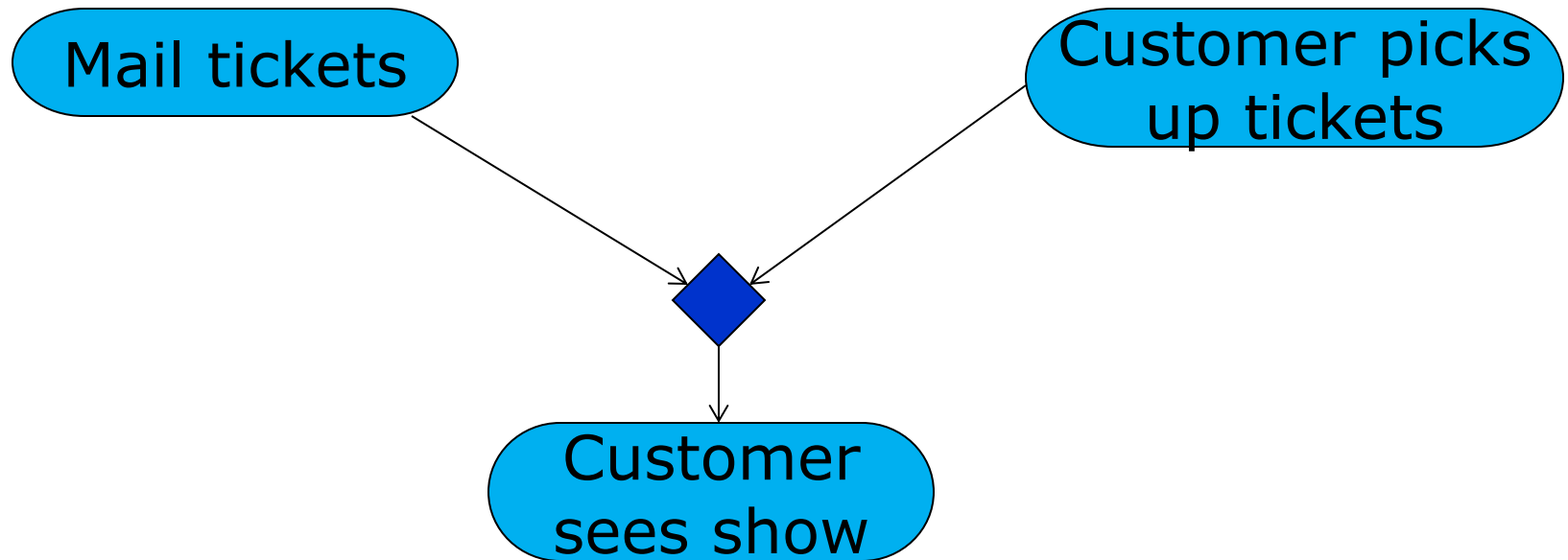
Branching

A branch has one incoming transition and two or more outgoing transitions:



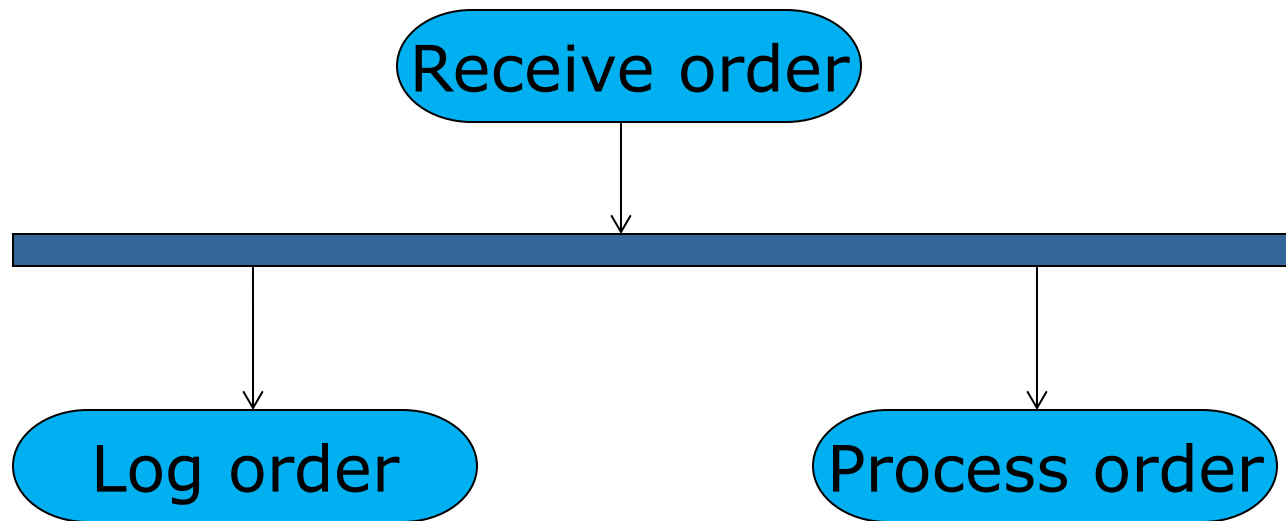
Merging

A merge has two or more incoming transitions and one outgoing transition:



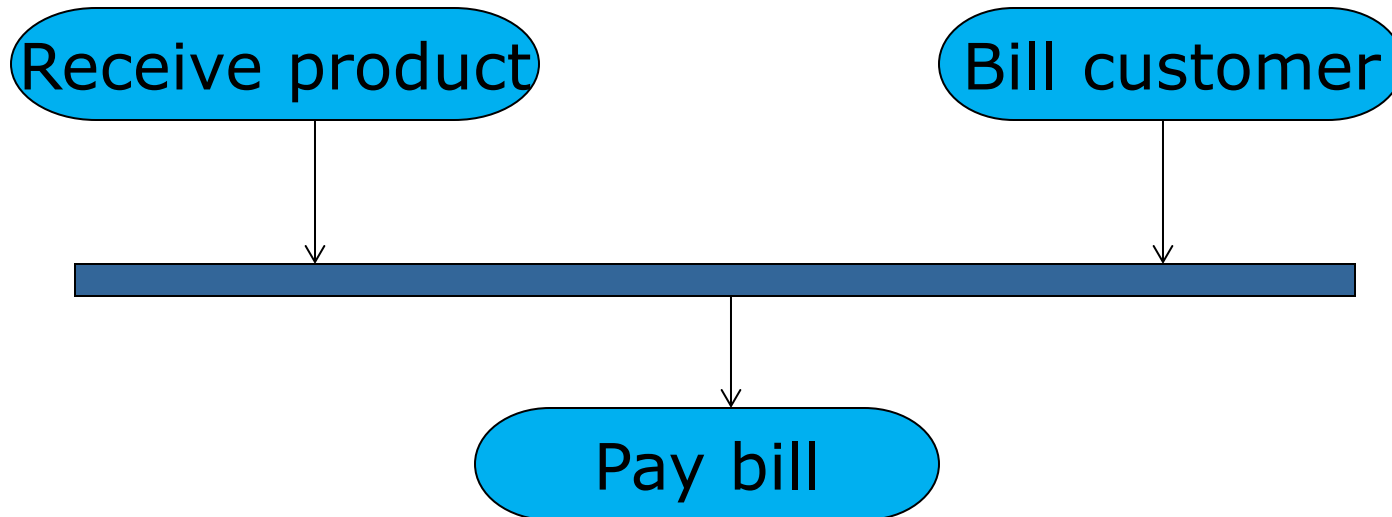
Forking

A fork represents the splitting of a single flow of control into two or more concurrent flows of control:



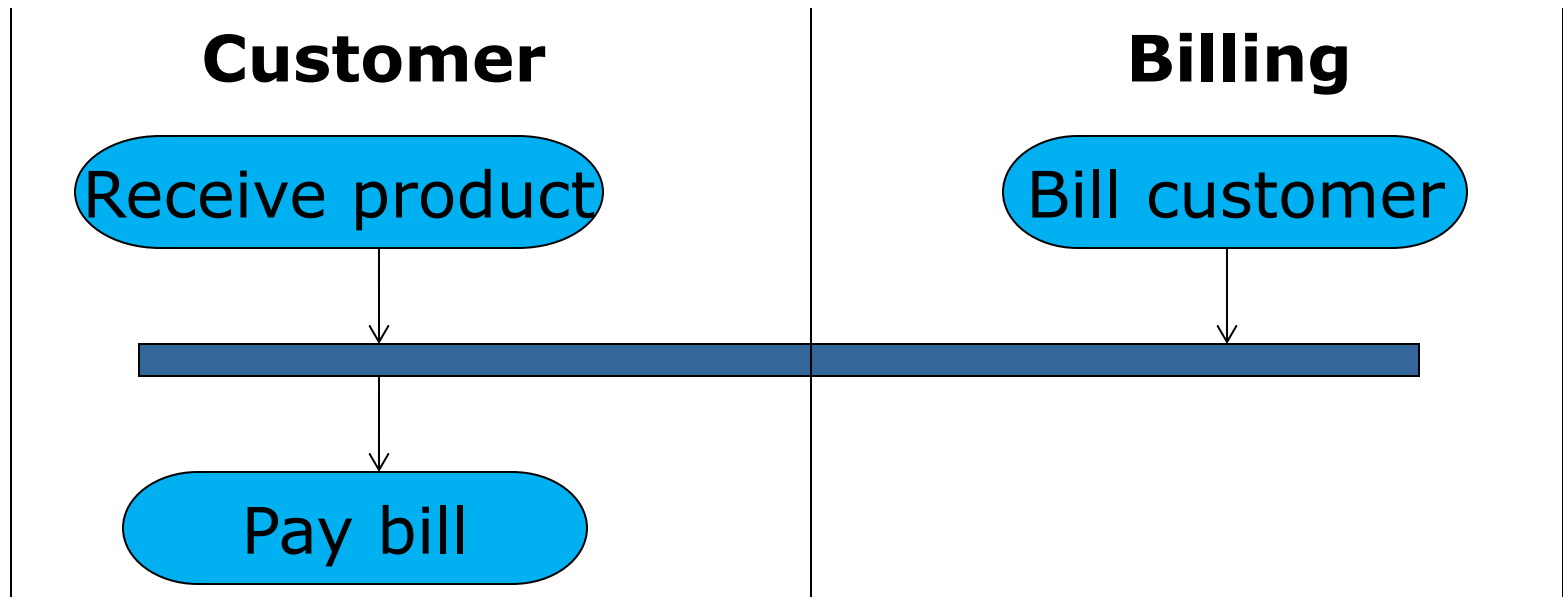
Joining

A join represents the synchronization of two or more flows of control into one sequential flow of control:



Swimlanes

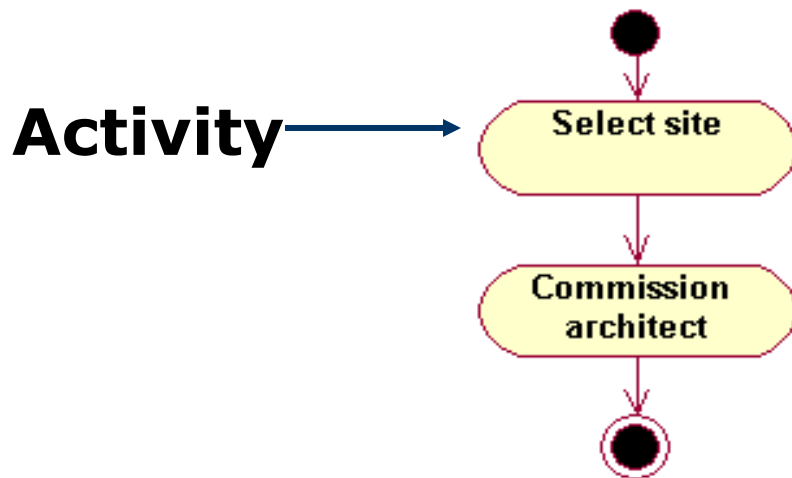
Swimlanes partition groups of activities based on, for instance, business organizations:



Activity Diagrams

Activity

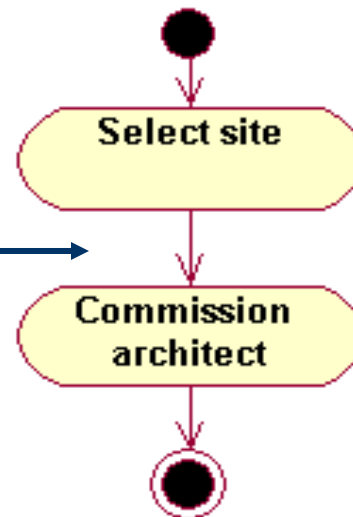
- An **activity** represents the performance of a task within the workflow.
- In the UML, an activity is represented by a lozenge (horizontal top and bottom with convex sides).



State Transitions

- A **state transition** shows what activity follows after another.
- In the UML, a state transition is represented by a solid line with an arrow.

State Transition



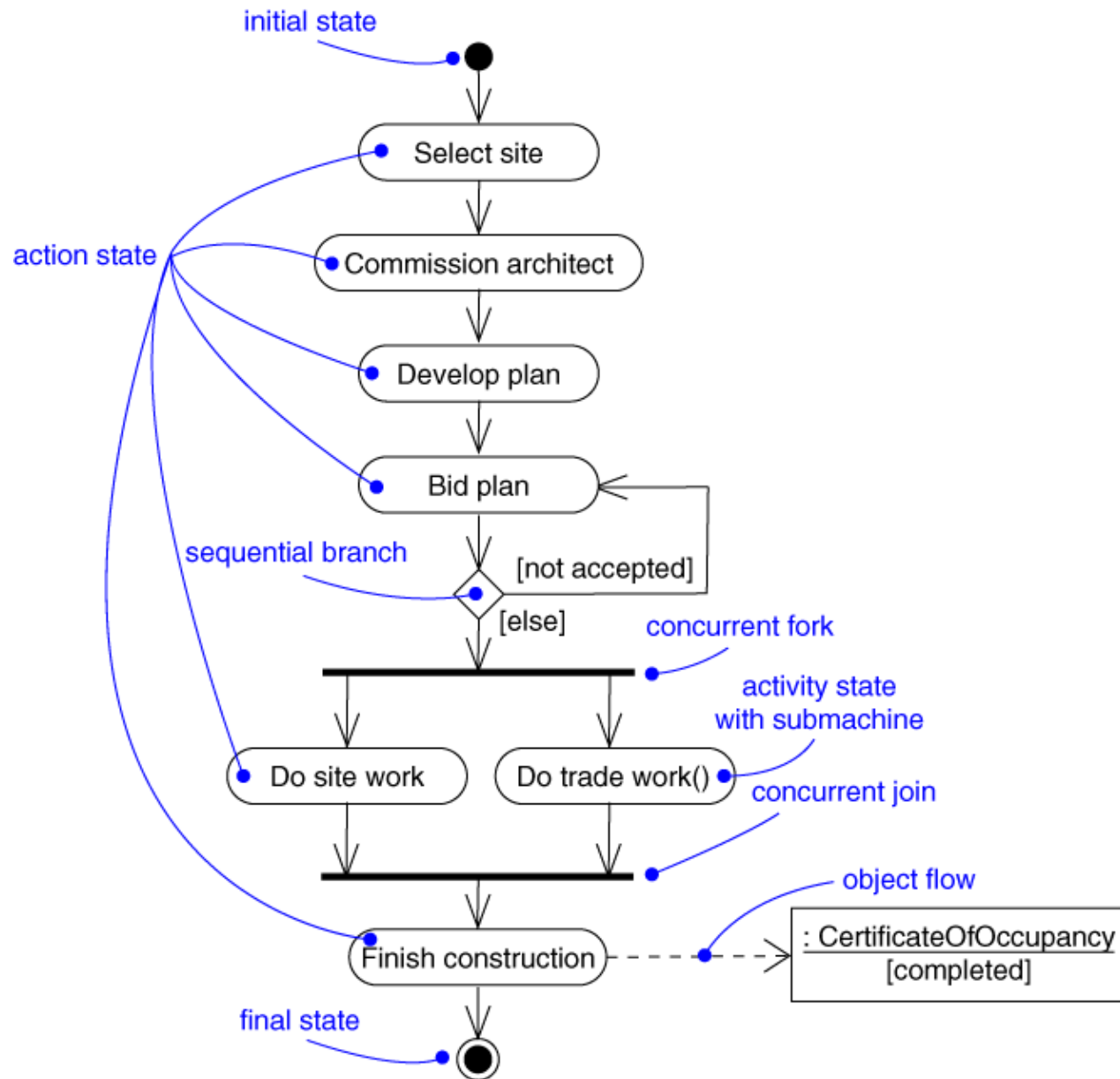
Decisions

- A **decision** is a point in an activity diagram where guard conditions are used to indicate different possible transitions.
- In the UML, a decision is represented by a diamond.

Synchronization Bars

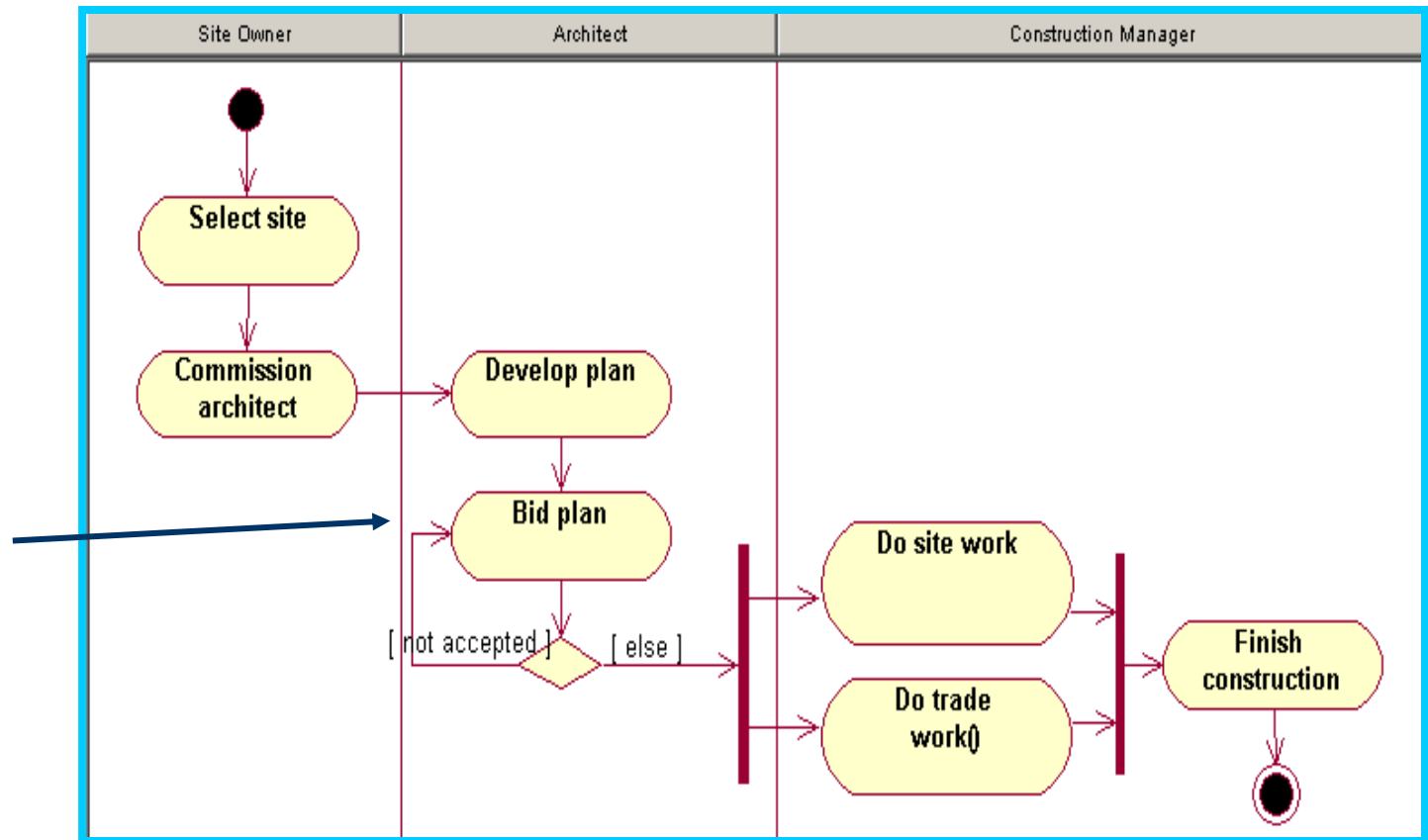
- A **synchronization bar** allows you to show concurrent threads in a workflow of a use case.
- In the UML, a synchronization bar is represented by a thick horizontal or vertical line.

Activity Diagram

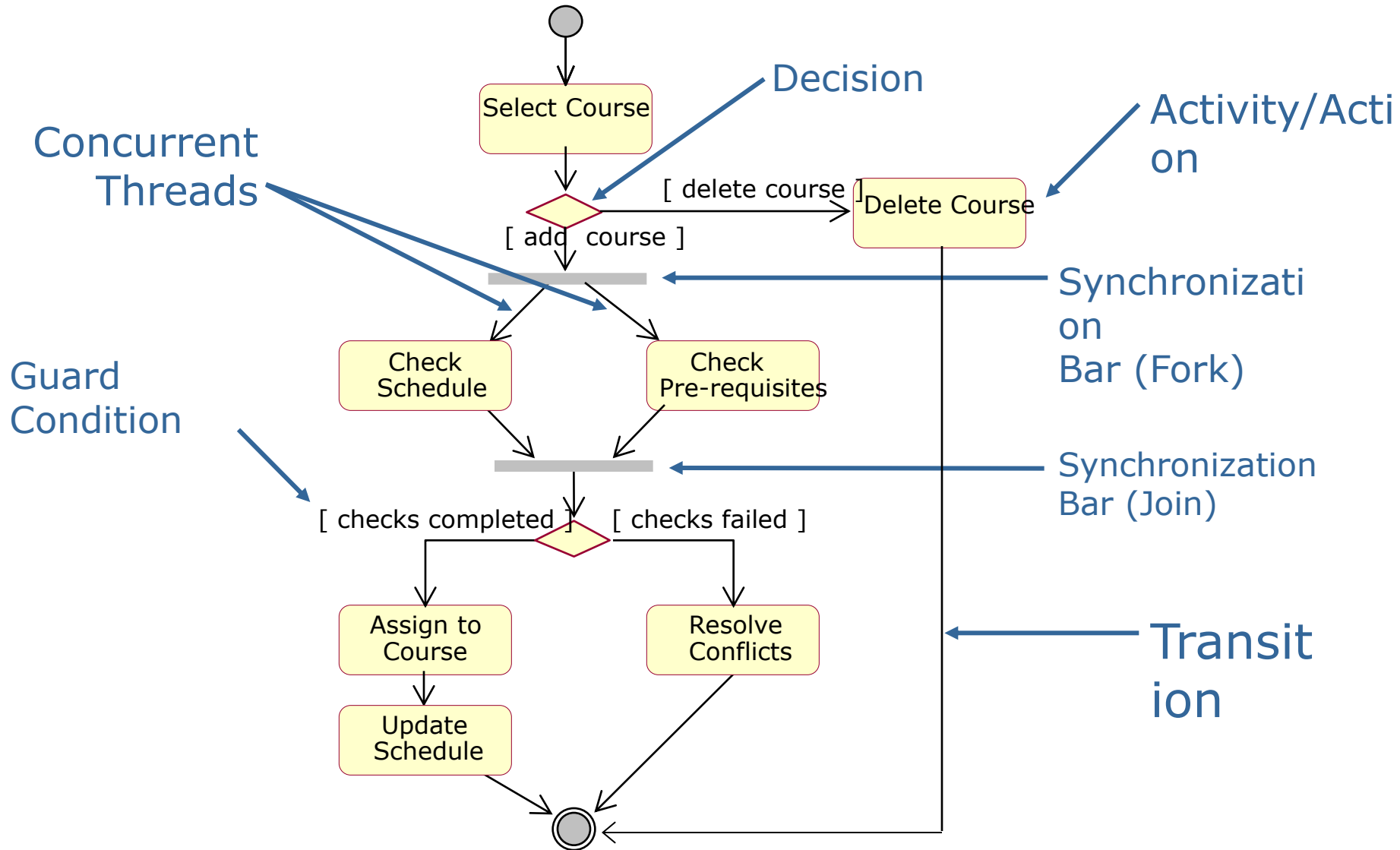


Swimlanes

- A **swimlane** is used to partition an activity diagram to help us better understand who or what is initiating the activity.

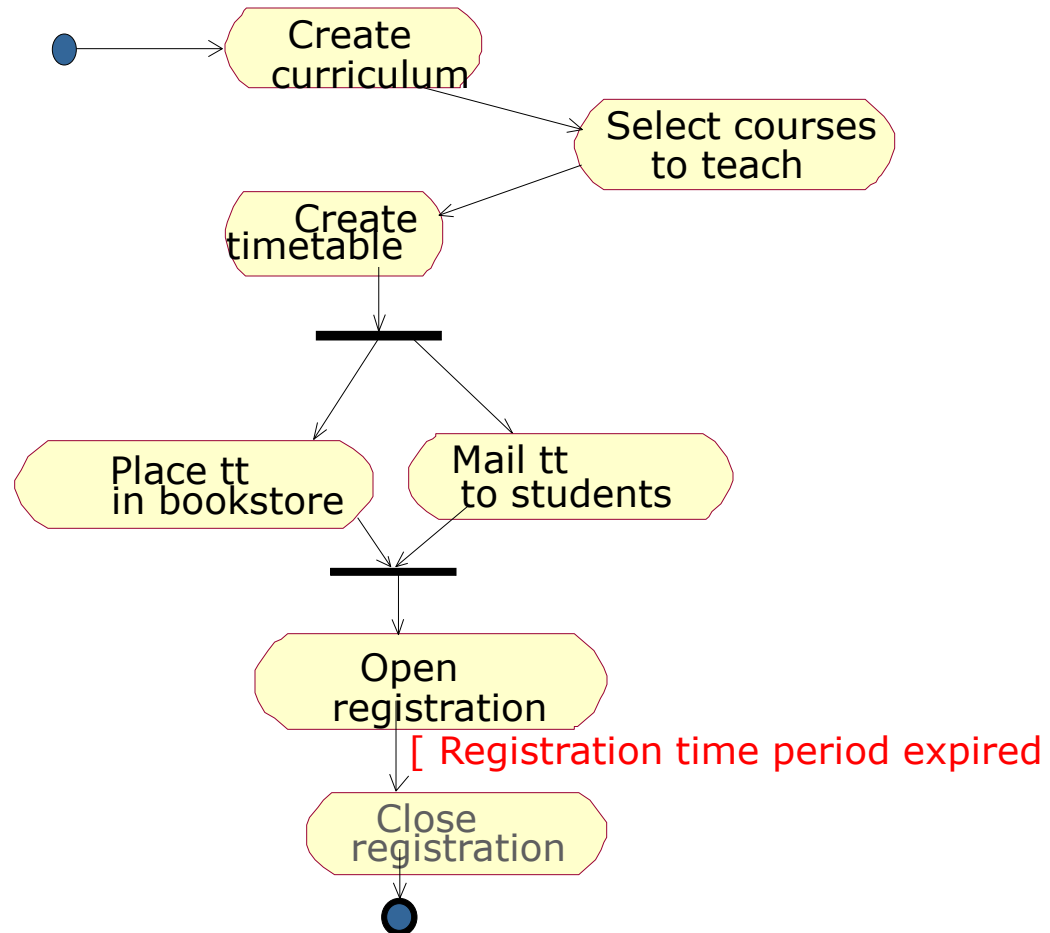


Example: Activity Diagram

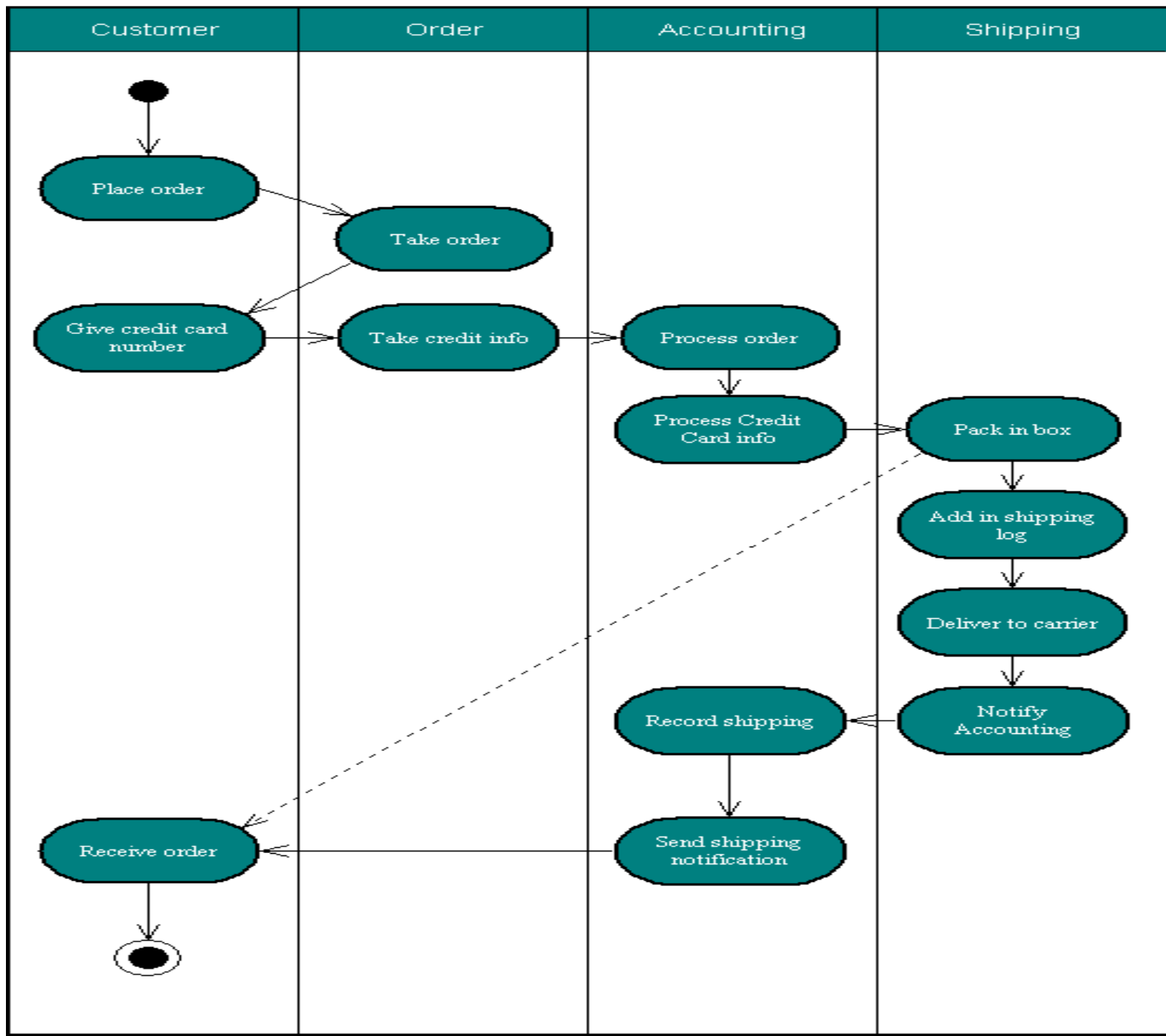


Activity Diagram

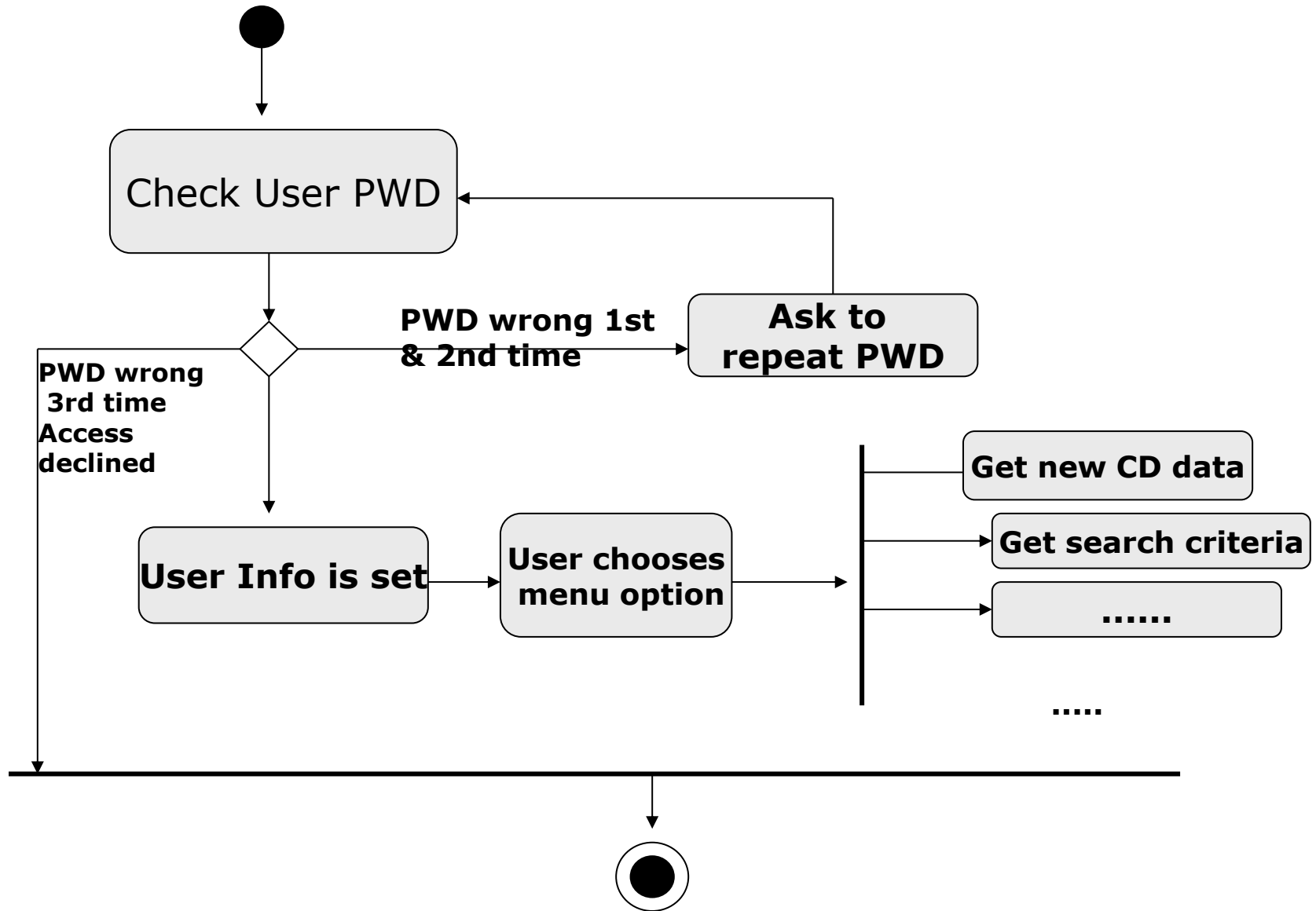
- An **activity diagram** shows the flow of events within our system.



UML Activity Diagram: Order Processing



Activity Diagrams Example



Activity Diagram Example

