

Object Oriented Analysis and Design Using the UML

Analysis: Dynamic Modeling



Dynamic Modeling with UML

- 1) represent how objects behave when you put them to work using the structure already defined in structural diagrams
- 2) model how the objects communicate in order to accomplish tasks within the operation of the system
- 3) describe how the system:
 - a) responds to actions from the users
 - b) maintains internal integrity
 - c) moves data
 - d) creates and manipulates objects, etc.
- 4) describe discrete pieces of the system, such as individual **scenarios** or operations

Note: not all system behaviour have to be specified - simple behaviours may not need a visual explanation of the communication required to accomplish them.

Dynamic Modeling with UML

▶ Diagrams for dynamic modeling

- *Interaction diagrams* describe the dynamic behavior between objects
- *Statecharts* describe the dynamic behavior of a single object

▶ Interaction diagrams

- Sequence Diagram:
 - Dynamic behavior of a set of objects arranged in time sequence.
 - Good for real-time specifications and complex scenarios
- Collaboration Diagram :
 - Shows the relationship among objects. Does not show time

▶ State Chart Diagram:

- A state machine that describes the response of an object of a given class to the receipt of outside stimuli (Events).
- *Activity Diagram*: A special type of statechart diagram, where all states are action states (Moore Automaton)

Sequence Diagram

- ▶ From the flow of events in the use case or scenario proceed to the sequence diagram
- ▶ A sequence diagram is a graphical description of objects participating in a use case or scenario using a DAG (direct acyclic graph) notation
- ▶ Relation to object identification:
 - Objects/classes have already been identified during object modeling
 - Objects are identified as a result of dynamic modeling
- ▶ Heuristic:
 - A event always has a sender and a receiver.
 - The representation of the event is sometimes called a message
 - Find them for each event => These are the objects participating in the use case

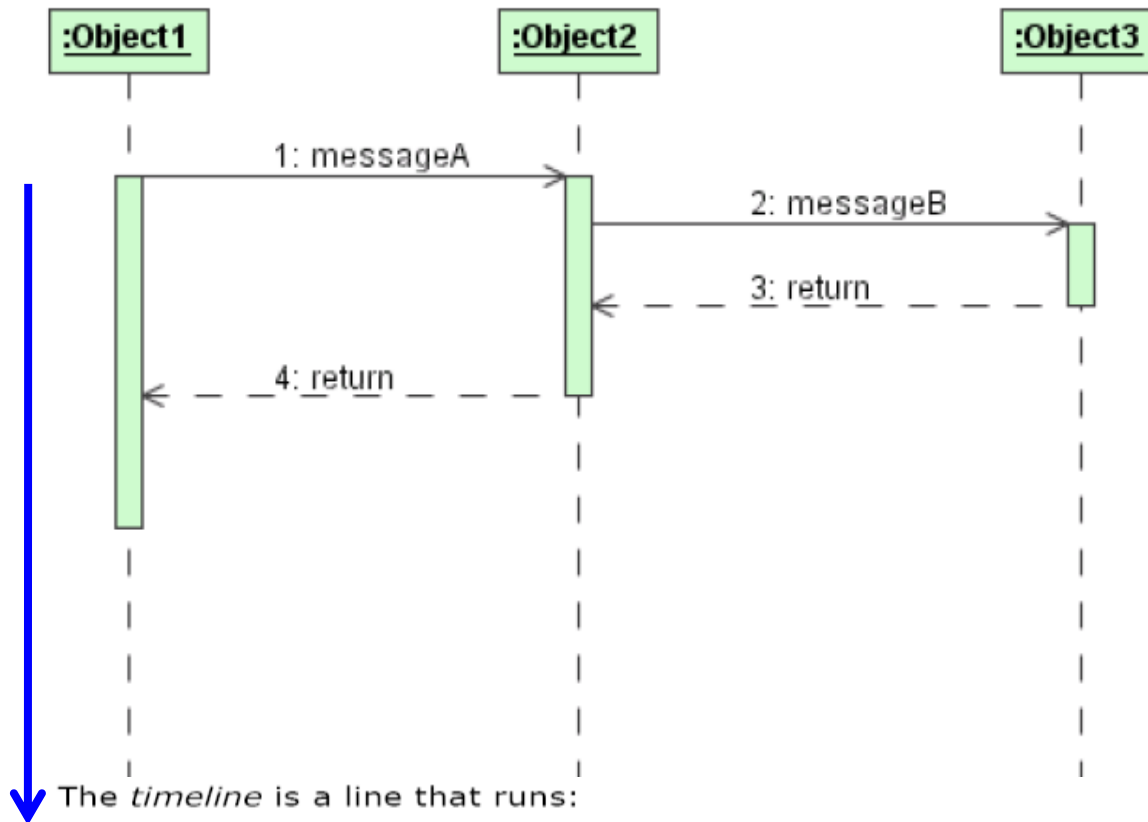
Sequence Diagram

A sequence diagram shows interactions between objects.

Components of sequence diagrams:

- 1) object lifelines
 - a) object
 - b) timeline
- 2) messages
 - a) message, stimulus
 - b) signal, exception
 - c) operations, returns
 - d) identification
- 3) message syntax

Sequence Diagram Example



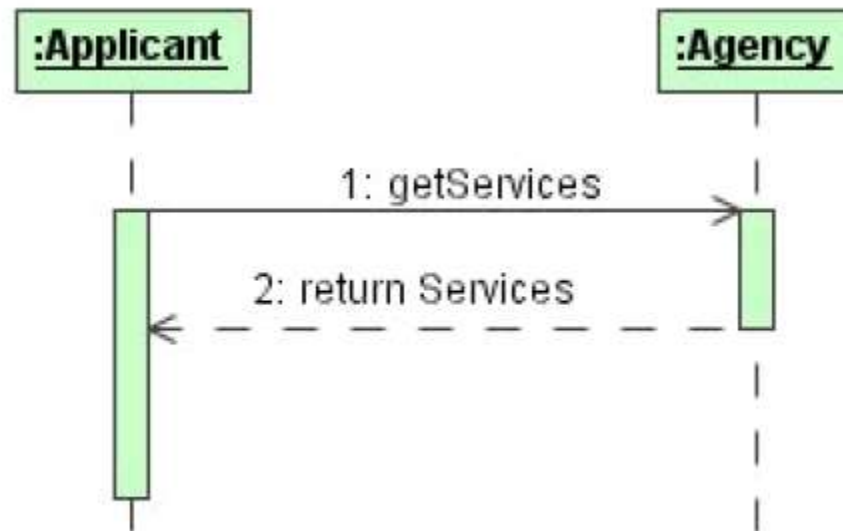
1. from the beginning of a scenario at the top of the diagram
2. to the end of the scenario at the bottom of the diagram.

Layout:

- ❑ 1st column: Should correspond to the actor who initiated the use case
- ❑ 2nd column: Should be a boundary object
- ❑ 3rd column: Should be the control object that manages the rest of the use case

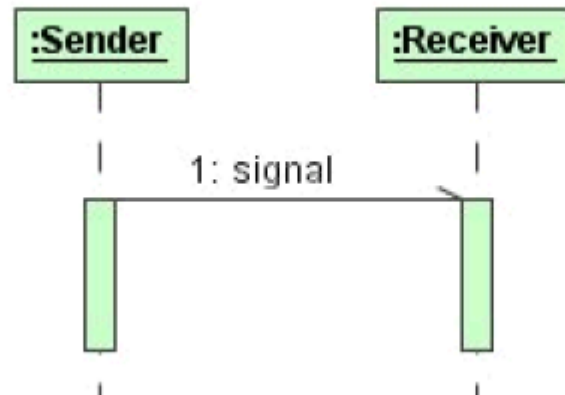
Operations and Returns Example

Example: The *Applicant* object sends a message to the *Agency* object to get the list of services it provides. The *Agency* object returns this information.



Signal

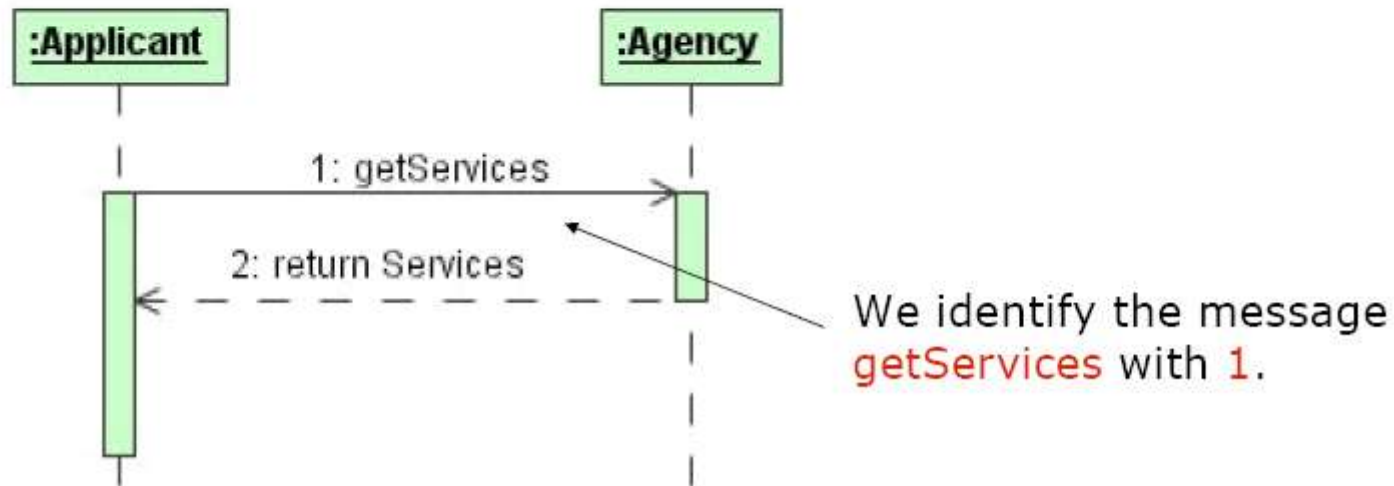
- 1) an object may raise a signal through a message
- 2) a **signal** is a special type of a class associated with an event that can trigger a procedure within the receiving object
- 3) a signal does not require a return from the receiving object



Identification of Messages

A message number or name is used to identify messages.

Example:



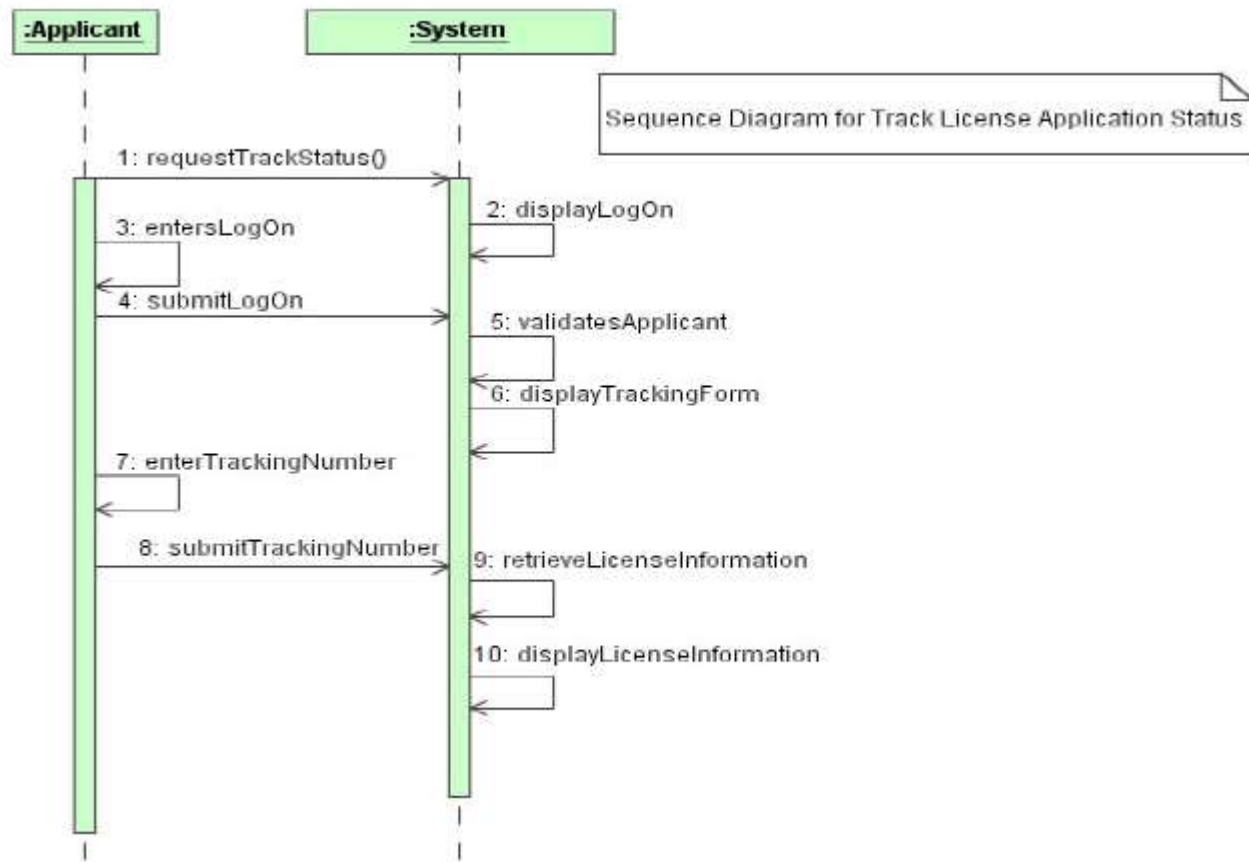
Example Scenario

Recall the scenario: an applicant tracks the status of a license application and the system displays the license information.

Procedure:

1. Applicant requests to track the status of a license application
2. System displays the logon form
3. Applicant enters the logon information
4. Applicant submits the logon information
5. System validates the applicant
6. System displays the form to enter the tracking number
7. Applicant enters the tracking number
8. Applicant submits the tracking number
9. System retrieves the license information
10. System displays the license information

Example Sequence Diagram

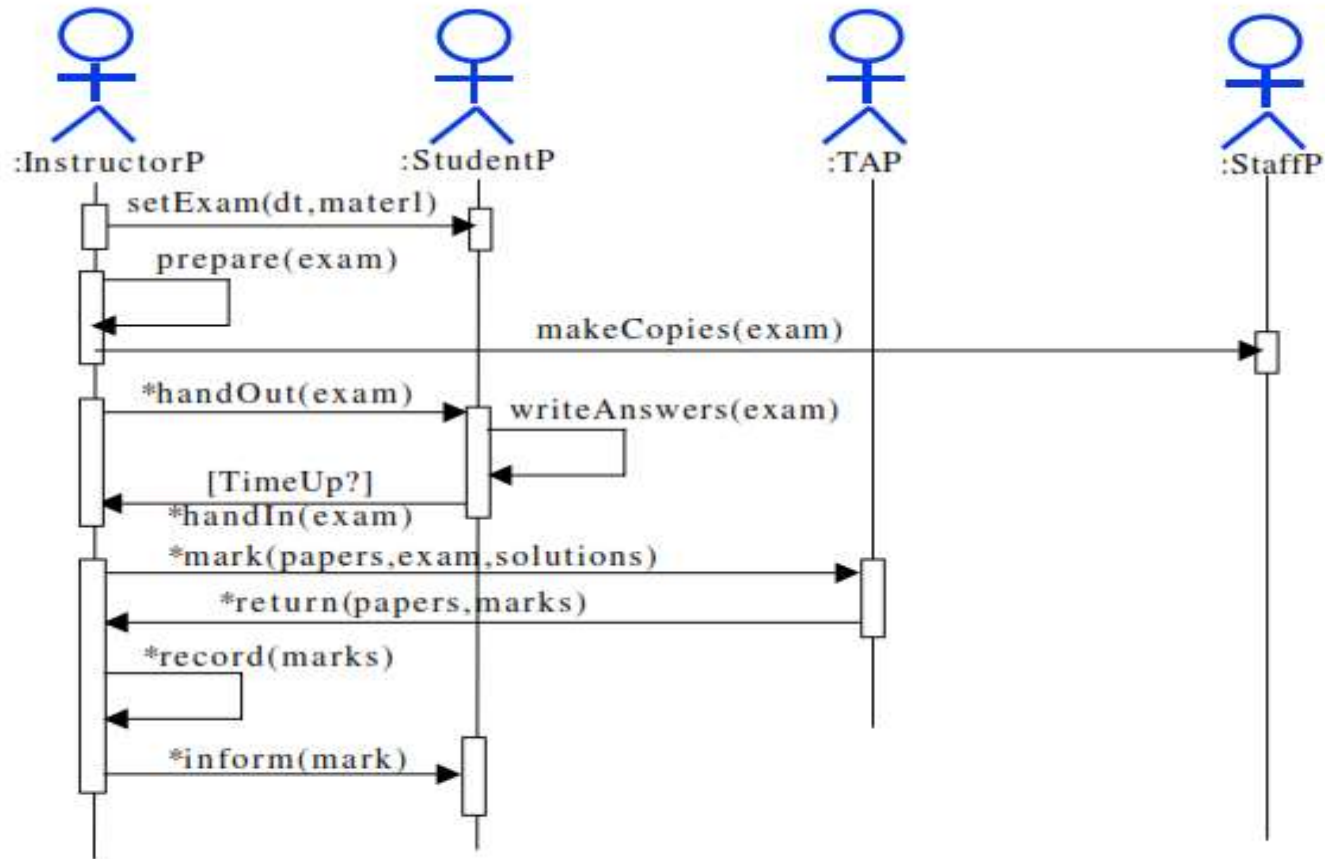


Example Scenario

Exercise

- ▶ To give an exam, an instructor first notifies the students of the exam date and the material to be covered. She then prepares the exam paper (with sample solutions), gets it copied to produce enough copies for the class, and hands it out to students on the designated time and location. The students write their answers to exam questions and hand in their papers to the instructor. The instructor then gives the exam papers to the TAs, along with sample solutions to each question, and gets them to mark it. She then records all marks and returns the papers to the students.
- ▶ Draw a sequence diagram that represents this process. Make sure to show when each actor is participating in the process. Also, show the operation that is carried out during each interaction, and what its arguments are.

Example Scenario

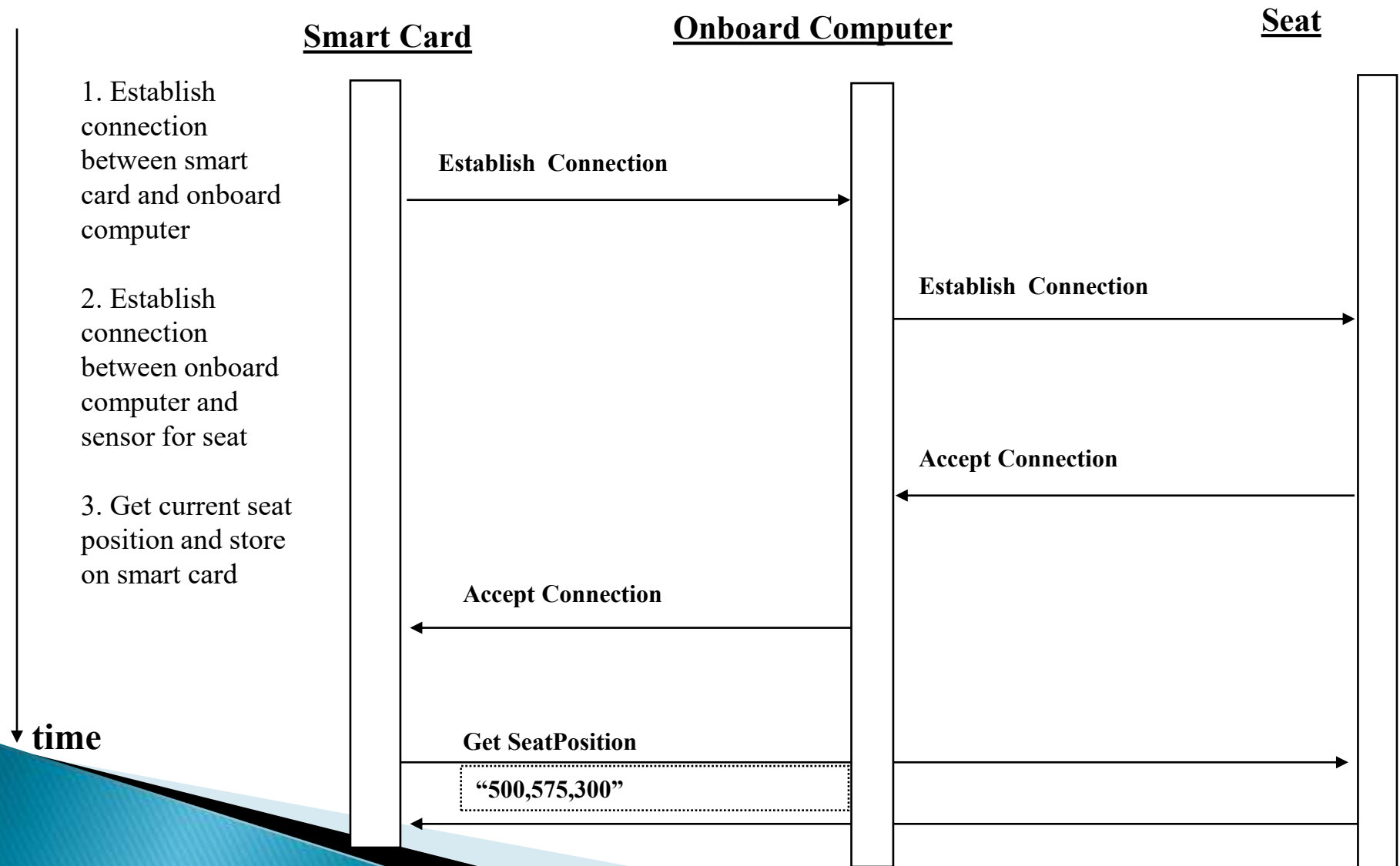


Example Scenario

GetSeatPosition: Passenger tries to find an empty seat in a train using an onboard computer connected to seat sensors and a smart card.

- ▶ Flow of events in a “Get SeatPosition” use case :
 1. Establish connection between smart card and onboard computer
 2. Establish connection between onboard computer and sensor for seat
 3. Get current seat position and store on smart card
- ▶ Which are the objects?

Sequence Diagram for “Get SeatPosition”



Normal Flow of Events: For withdrawal of cash

- ▶ 1.(SR) The ATM asks the user to insert a card.
- ▶ 2.(AA) The user inserts a cash card.
- ▶ 3.(SR) The ATM accepts the card and reads its serial number.
- ▶ 4.(SR) The ATM requests the password.
- ▶ 5.(AA) The user enters 1234.
- ▶ 6.(SR) The ATM verifies the serial number and password with the bank and gets the notification accordingly.
- ▶ 7.(SA)The ATM asks the user to select the kind of transaction.
- ▶ 8.(AA)User selects the withdrawal.
- ▶ 9.(SR)The ATM asks for the amount of cash; user enters Rs. 2500/-
- ▶ 10.(SR)The ATM verifies that the amount of cash is within predefined policy limits and asks the bank, to process the transaction which eventually confirms success and returns the new account balance.

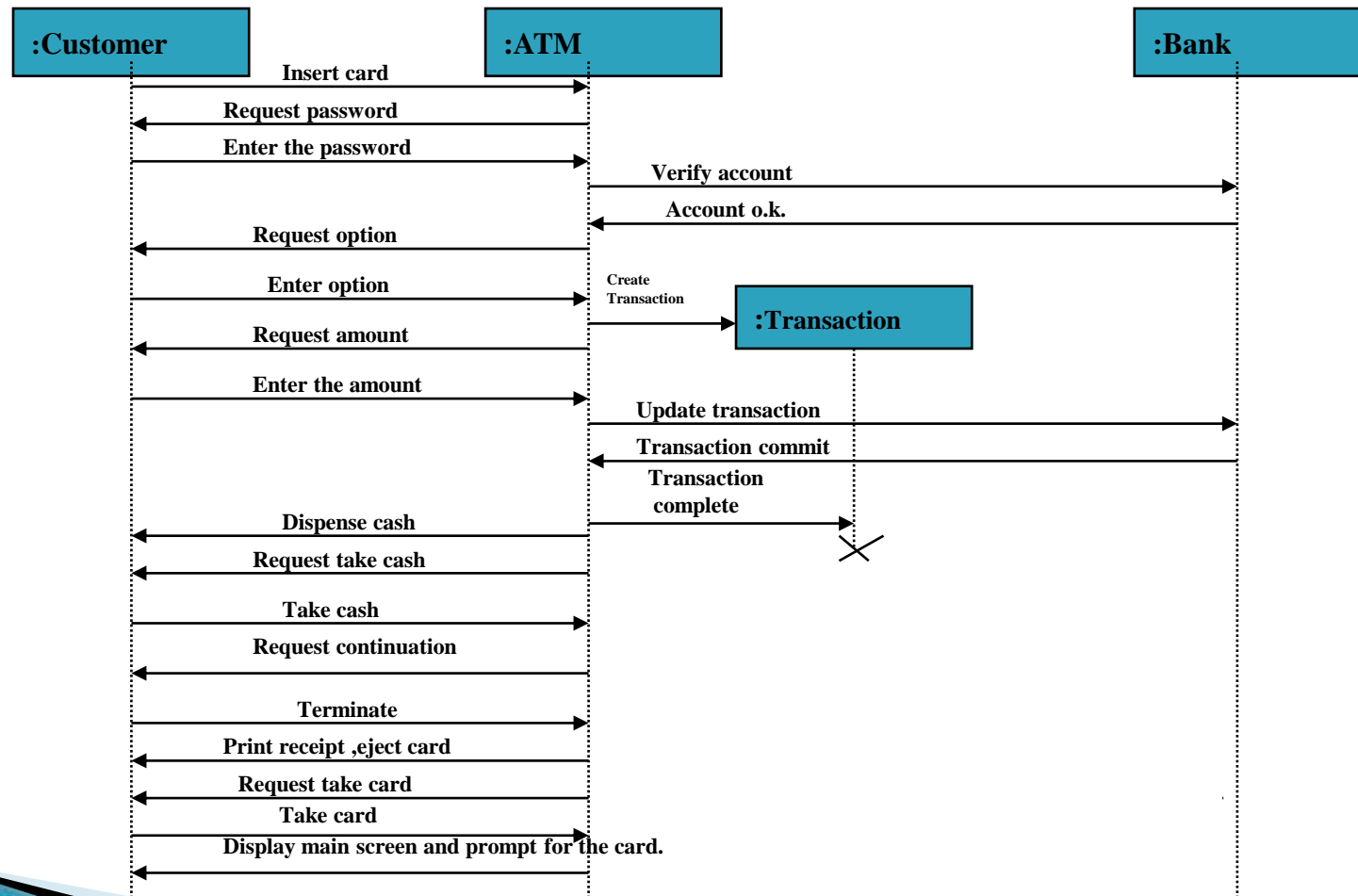
Normal Flow of Events: For withdrawal of cash

- ▶ 11.(SR) The ATM dispenses cash and asks the user to take it.
- ▶ 12.(AA) The user takes the cash.
- ▶ 13.(SR) The ATM asks whether the user wants to continue.
- ▶ 14.(AA) The user indicates no.
- ▶ 15.(SR) The ATM prints a receipt, ejects the card and asks the user to take them
- ▶ 16.(AA) The user takes the receipt and the card.
- ▶ 17.(SR) The ATM asks a user to insert a card.

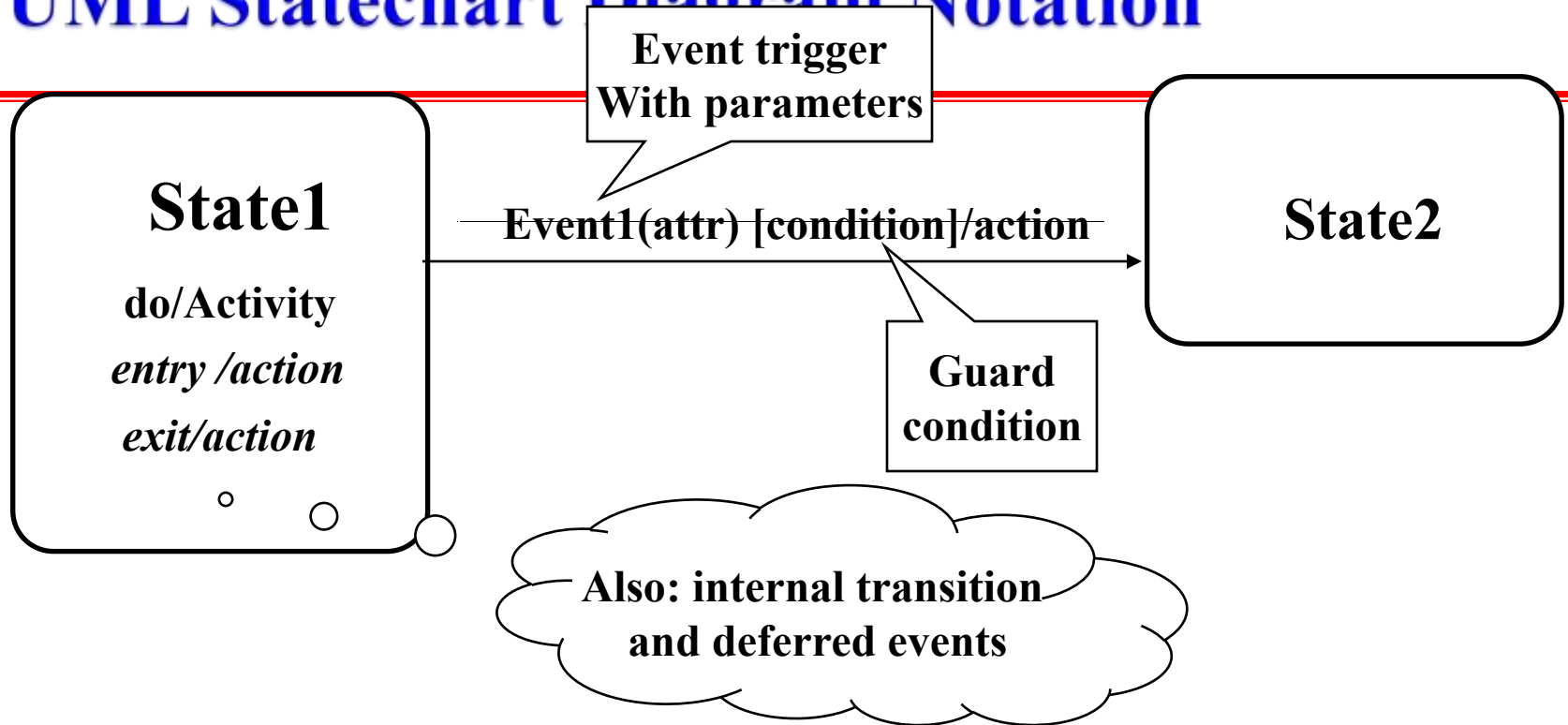
Alternative Flow of Events: For withdrawal of cash

- ▶ 9. The ATM asks for the amount of cash; the user has change of mind and hits the “cancel”.

Sequence diagram [for withdrawal of cash, normal flow]



UML Statechart Diagram Notation



- ▶ Notation based on work by Harel
 - Added are a few object-oriented modifications
- ▶ A UML statechart diagram can be mapped into a finite state machine

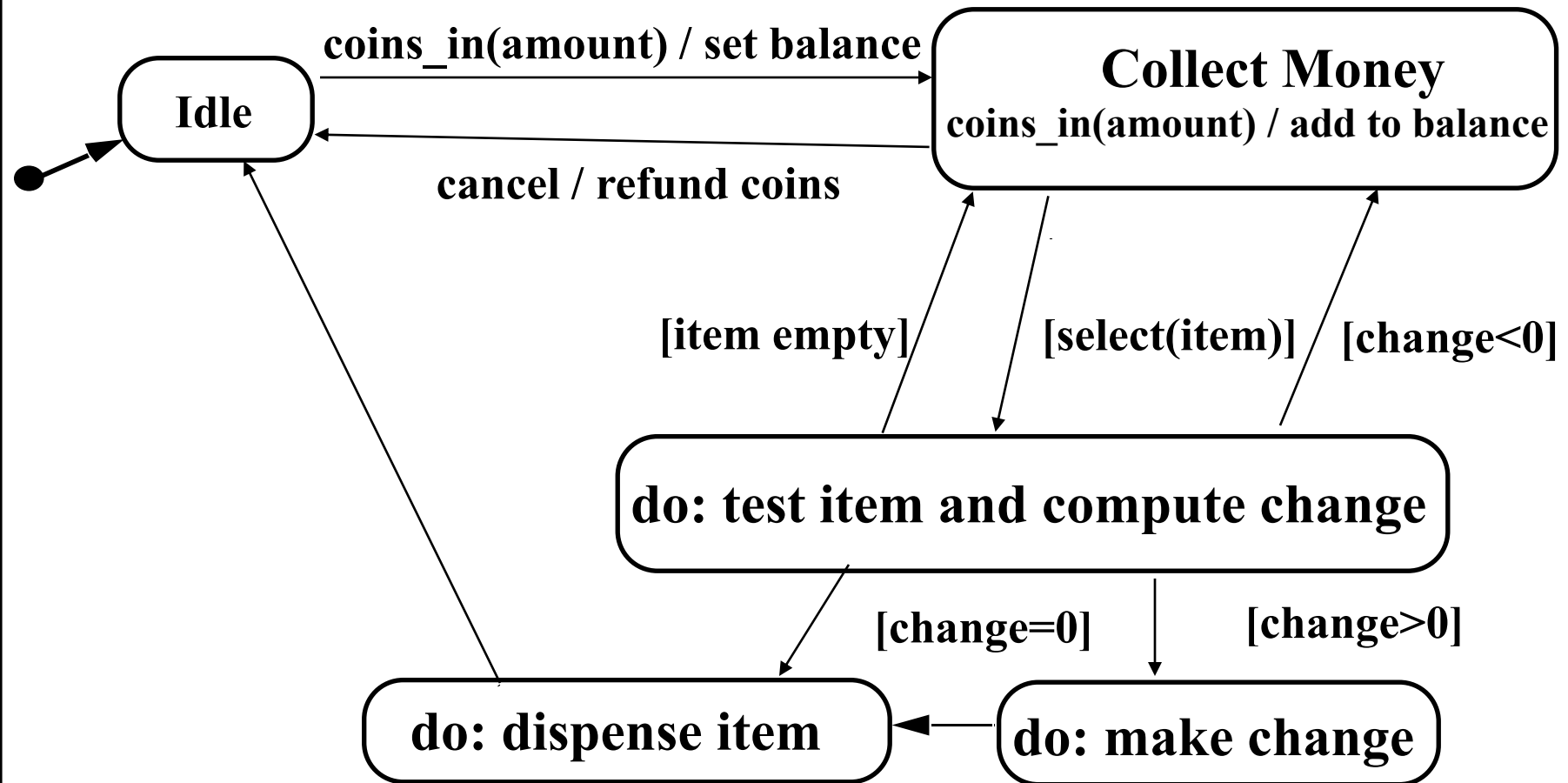
Statechart Diagrams

- ▶ Graph whose nodes are states and whose directed arcs are transitions labeled by event names.
- ▶ We distinguish between two types of operations in statecharts:
 - Activity: Operation that takes time to complete
 - associated with states
 - Action: Instantaneous operation
 - associated with events
 - associated with states (reduces drawing complexity): Entry, Exit, Internal Action
- ▶ A statechart diagram relates events and states for *one class*
 - An object model with a set of objects has a set of state diagrams

State

- ▶ An abstraction of the attributes of a class
 - State is the aggregation of several attributes a class
- ▶ Basically an equivalence class of all those attribute values and links that do no need to be distinguished as far as the control structure of the system is concerned
 - Example: State of a bank
 - A bank is either solvent or insolvent
- ▶ State has duration

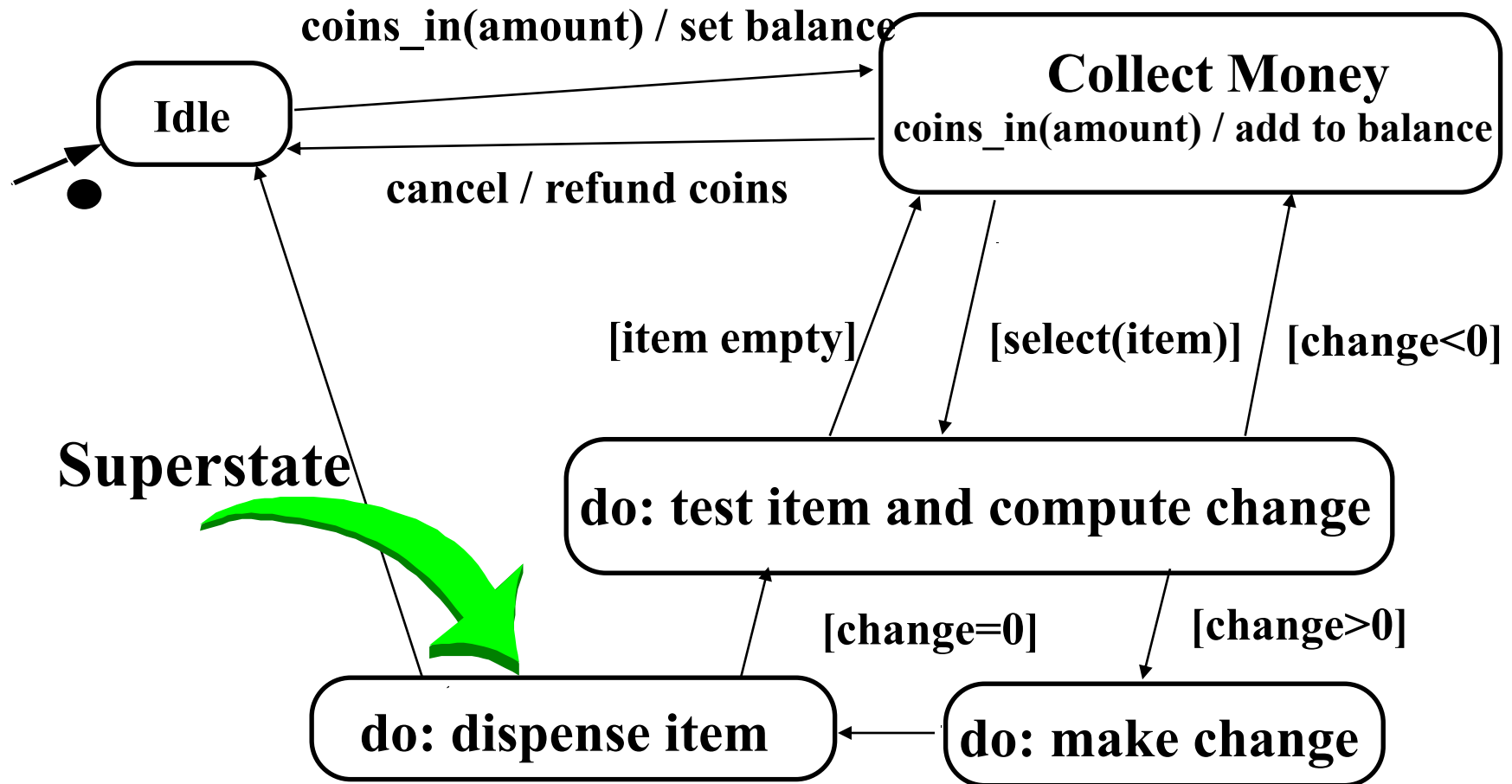
Example of a StateChart Diagram



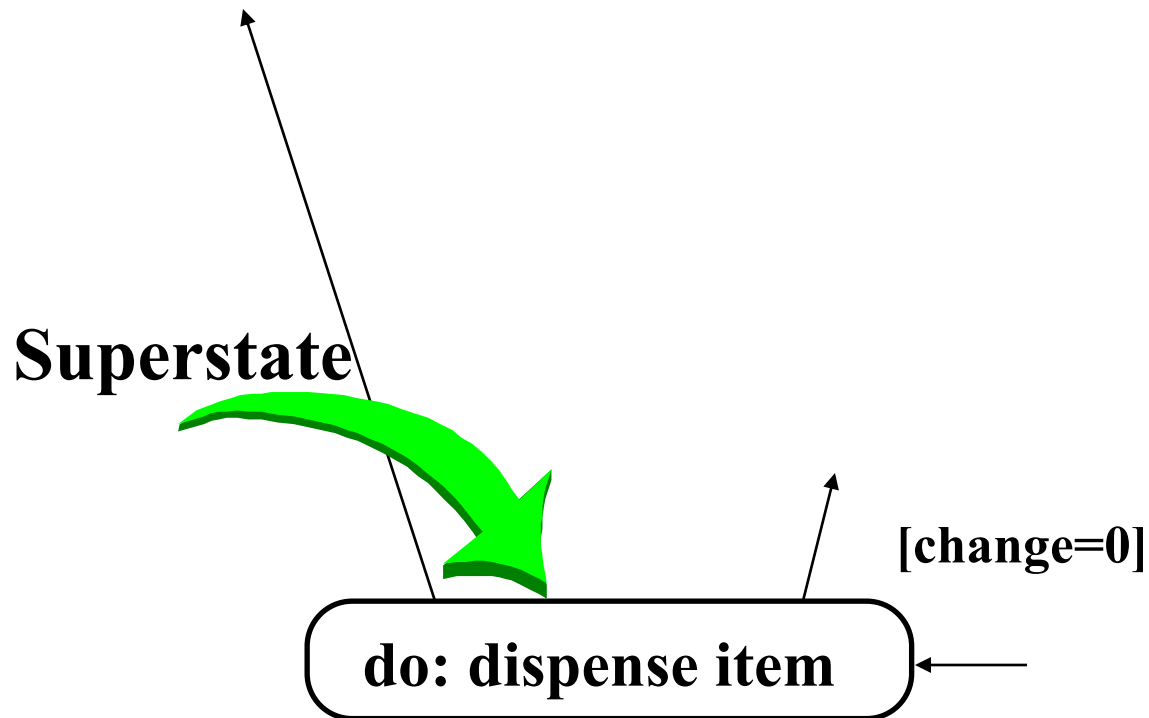
Nested State Diagram

- ▶ Activities in states are composite items denoting other lower-level state diagrams
- ▶ A lower-level state diagram corresponds to a sequence of lower-level states and events that are invisible in the higher-level diagram.
- ▶ Sets of substates in a nested state diagram denote a **superstate** are enclosed by a large rounded box, also called contour.

Example of a Nested Statechart Diagram



Example of a Nested Statechart Diagram

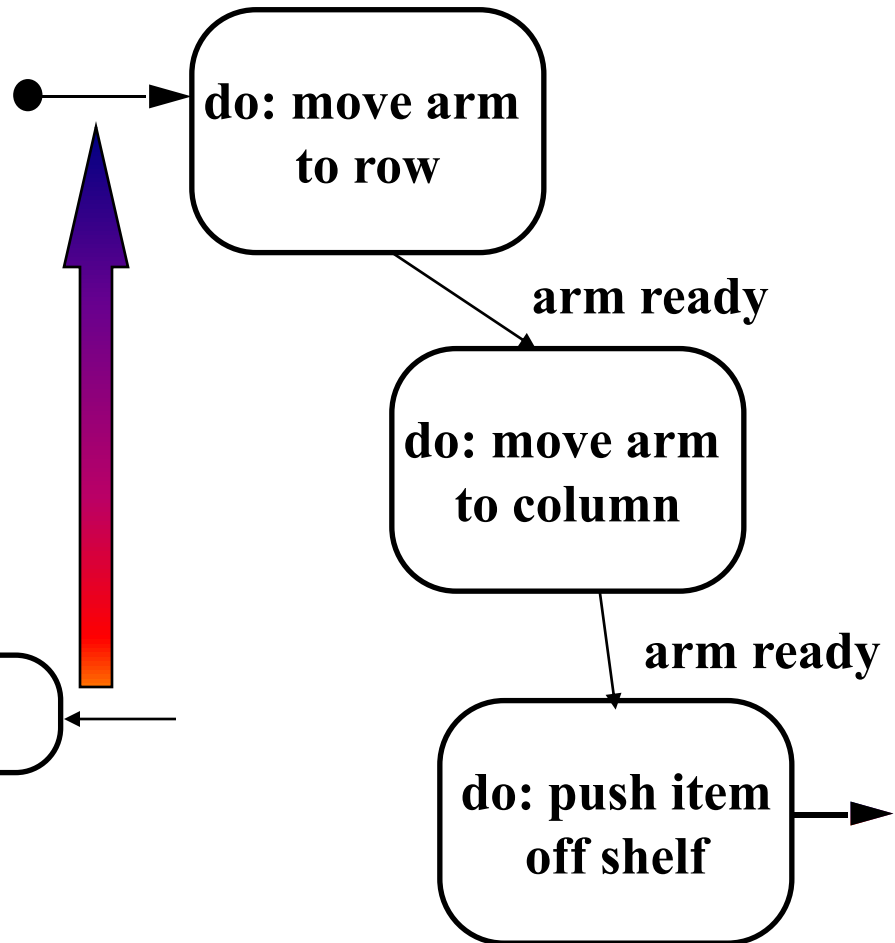


Example of a Nested Statechart Diagram

**‘Dispense item’ as
an atomic activity:**

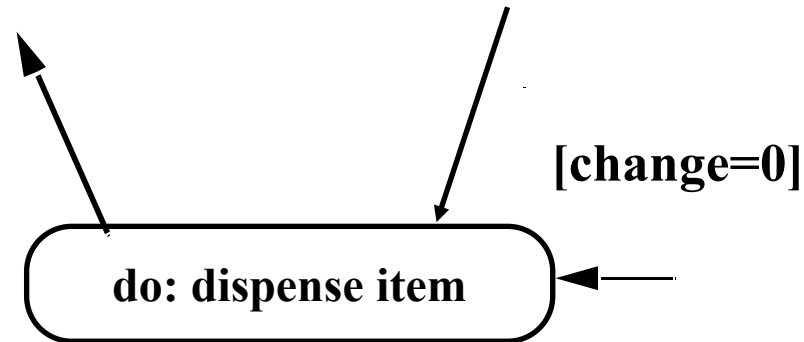


**‘Dispense item’ as
a composite activity:**

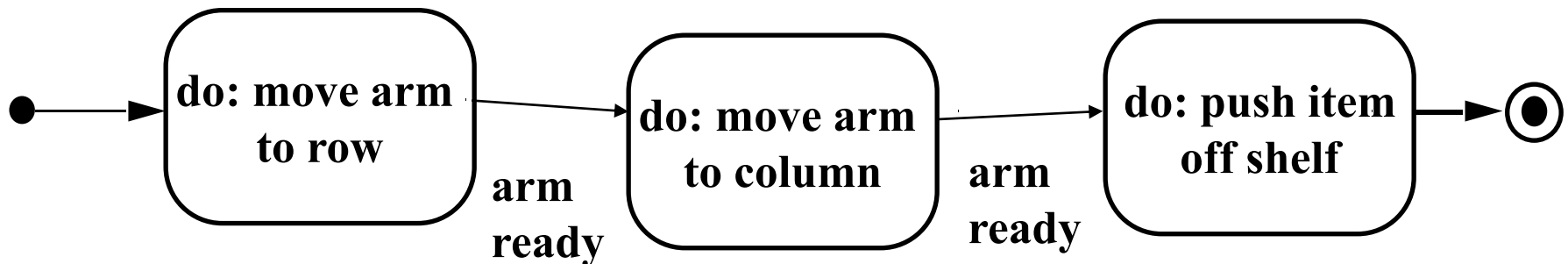


Expanding activity “do:dispense item”

‘Dispense item’ as
an atomic activity:



‘Dispense item’ as a composite activity:



Superstates

- ▶ Goal:
 - Avoid spaghetti models
 - Reduce the number of lines in a state diagram
- ▶ Transitions from other states to the superstate enter the first substate of the superstate.
- ▶ Transitions to other states from a superstate are inherited by all the substates (state inheritance)

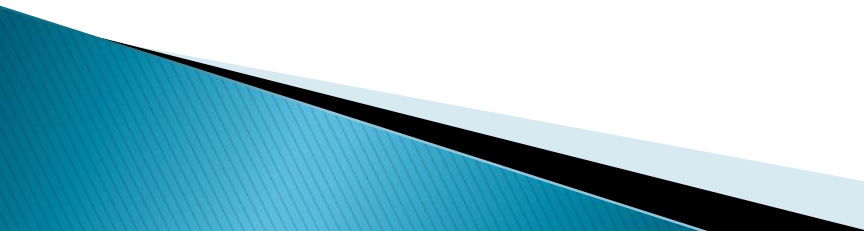
State Chart Diagram vs Sequence Diagram

- ▶ State chart diagrams help to identify:
 - *Changes* to an individual object over time
- ▶ Sequence diagrams help to identify
 - The *temporal relationship* of between objects over time
 - *Sequence of operations* as a response to one ore more events

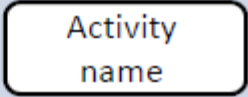




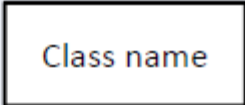
UML : ACTIVITY DIAGRAM



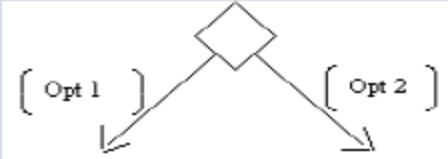
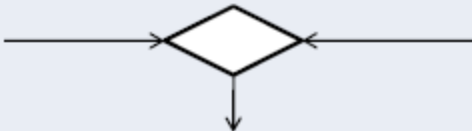



UML : ACTIVITY DIAGRAM

1. Activity diagrams are the object-oriented equivalent of flow charts and data-flow diagrams from structured development.
 2. Activity diagrams describe the workflow behavior of a system.
 3. The process flows in the system are captured in the activity diagram.
 4. Activity diagram illustrates the dynamic nature of a system by modeling the flow of control from activity to activity.
- 

Elements of Activity Diagram

Description	Symbol
Activity : Is used to represent a set of actions	
A Control Flow : Shows the sequence of execution	
An Object Flow : Shows the flow of an object from one activity (or action) to another activity (or action).	
An Initial Node : Portrays the beginning of a set of actions or activities	
A Final-Activity Node : Is used to stop all control flows and object flows in an activity (or action)	
An Object Node : Is used to represent an object that is connected to a set of Object Flows.	

Elements of Activity Digram

Description	symbol
A Decision Node: Is used to represent a test condition to ensure that the control flow or object flow only goes down one path	
A Merge Node: Is used to bring back together different decision paths that were created using a decision-node.	
A Fork Node: Is used to split behavior into a set of parallel or concurrent flows of activities (or actions)	
A Join Node: Is used to bring back together a set of parallel or concurrent flows of activities (or actions).	
A Swimlane :A swimlane is a way to group activities performed by the same actor on an activity diagram or to group activities in a single thread	

Example 1: Creating document

1. **Open** the word processing package.
2. **Create** a file.
3. **Save** the file under a unique name within its directory.
4. **Type** the document.
5. **If** graphics are necessary, **open** the graphics package, create the graphics, and paste the graphics into the document.
6. **If** a spreadsheet is necessary, **open** the spreadsheet package, create the spreadsheet, and paste the spreadsheet into the document.
7. **Save** the file.
8. **Print** a hard copy of the document.
9. Exit the word processing package.

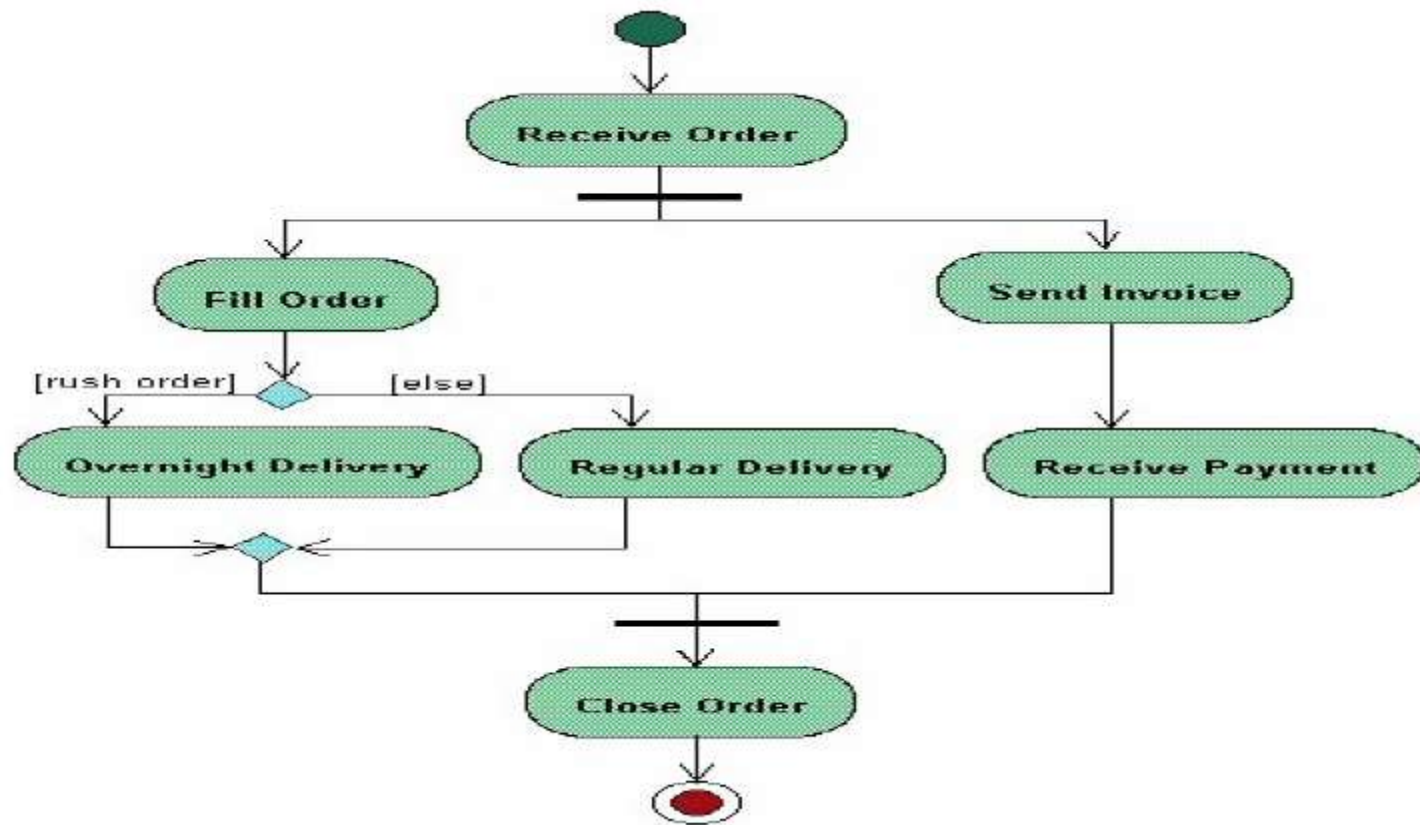
Example 1: Creating document



Example 2: Process Order

Once the order is **received** the activities **split** into two **parallel** sets of activities. **One** side **fills** and **sends** the order while the **other** handles the **billing**. On the **Fill Order** side, the **method** of **delivery** is decided conditionally. Depending on the condition either the **Overnight Delivery** activity or the **Regular Delivery** activity is performed. Finally the **parallel** activities **combine** to **close** the **order**.

Example 2: Process Order



Example 3: Enrollment in university

1. An applicant wants to enroll in the university.
2. The applicant hands a filled out copy of form *U113 University Application Form* to the registrar.
3. The registrar inspects the forms.
4. The registrar determines that the forms have been filled out properly.
5. The registrar informs student to attend in university overview presentation.
6. The registrar helps the student to enroll in seminars
7. The registrar asks the student to pay the initial.

Guards

A guard is a condition that must be true in order to traverse a transition.

1. **Each Transition Leaving a Decision Point Must Have a Guard .** This ensures that you have thought through all possibilities for that decision point.
2. **Guards Should Not Overlap.** For example guards such as $x < 0$, $x = 0$, and $x > 0$ are consistent whereas guard such as $x \leq 0$ and $x \geq 0$ are not consistent because they overlap – it isn't clear what should happen when x is 0.
3. **Guards on Decision Points Must Form a Complete Set.** For example, guards such as $x < 0$ and $x > 0$ are not complete because it isn't clear what happens when x is 0.

Parallel Activities guidelines

1. It is possible to show that activities can occur in parallel, as you see in example3 depicted using two parallel bars. The first bar is called a **fork**, it has **one transition entering** it and **two or more transitions leaving** it. The second bar is a **join**, with **two or more transitions entering** it and **only one leaving** it.
2. A Fork Should Have a Corresponding Join. In general, for every start (fork) there is an end (join). In UML 2 it is not required to have a join, but it usually makes sense.
3. Forks Have One Entry Transition.
4. Joins Have One Exit Transition
5. Avoid Superfluous Forks.

Swimlane Guidelines

A swimlane is a way to group activities performed by the same actor on an activity diagram or to group activities in a single thread.

Actions may be grouped into swimlanes to denote the object or subsystem that implements the actions.

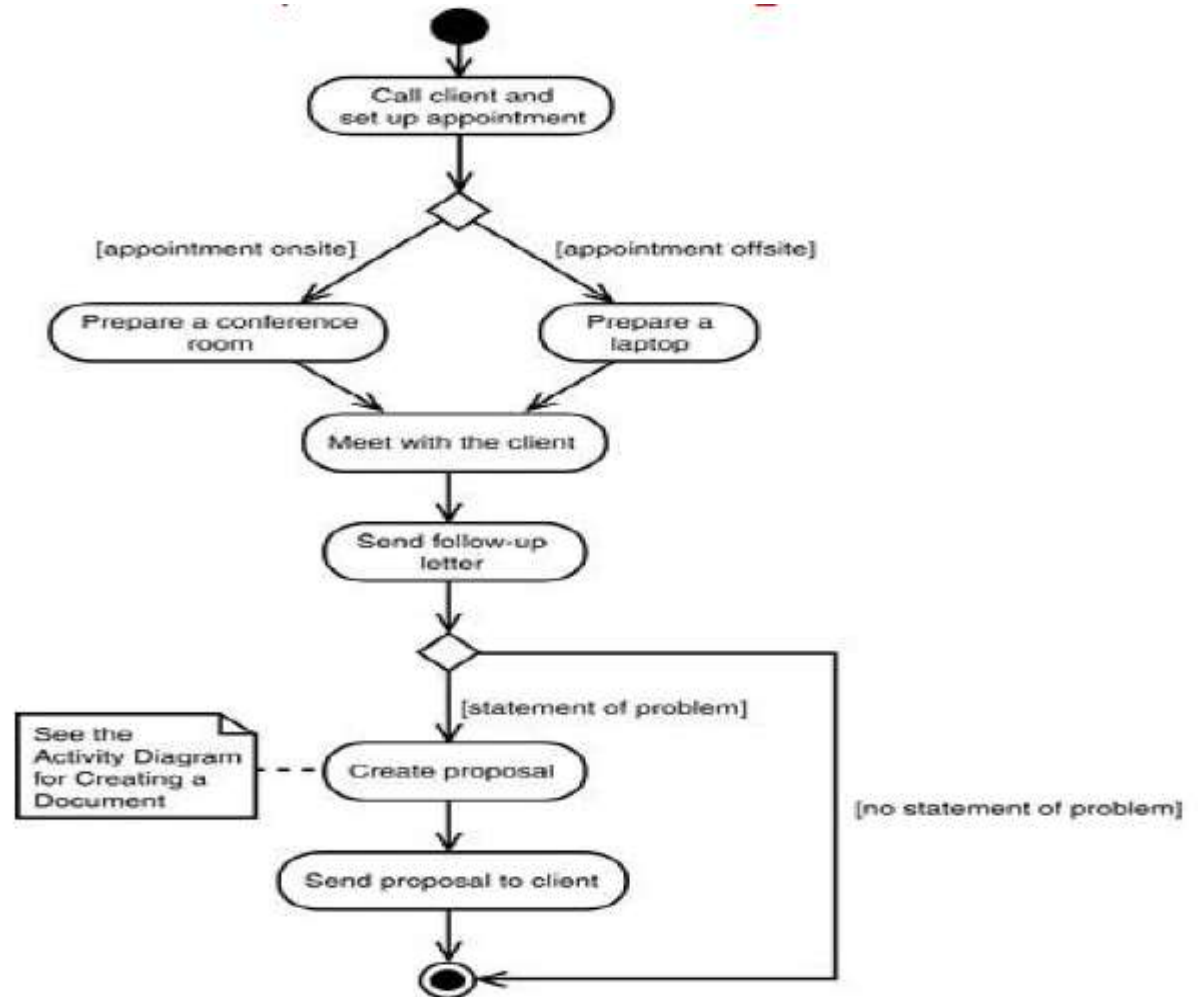
- ▶ Order Swimlanes in a Logical Manner
- ▶ Apply Swimlanes To Linear (sequential)
- ▶ Processes A good rule of thumb is that swimlanes are best applied to linear processes
- ▶ Have Less Than Five Swimlanes
- ▶ Consider Swimlanes For Complex Diagrams
- ▶ Swimlane Suggest The Need to Reorganize Into Smaller Activity Diagrams

Example 5: business process of meeting a new client

1. A salesperson calls the client and sets up an **appointment**.
2. If the appointment is **onsite** (in the consulting firm's office), corporate technicians **prepare conference room** for a presentation
3. If the appointment is **offsite** (at the client's office), a consultant **prepares** a presentation on a **laptop**.
4. The consultant and the salesperson **meet** with the client at the agreed-upon location and time.
5. The salesperson follows up with a **letter**
6. If the meeting has **resulted** in a **statement** of a **problem**, the consultant **create** a **proposal** and **sends** it to the client.

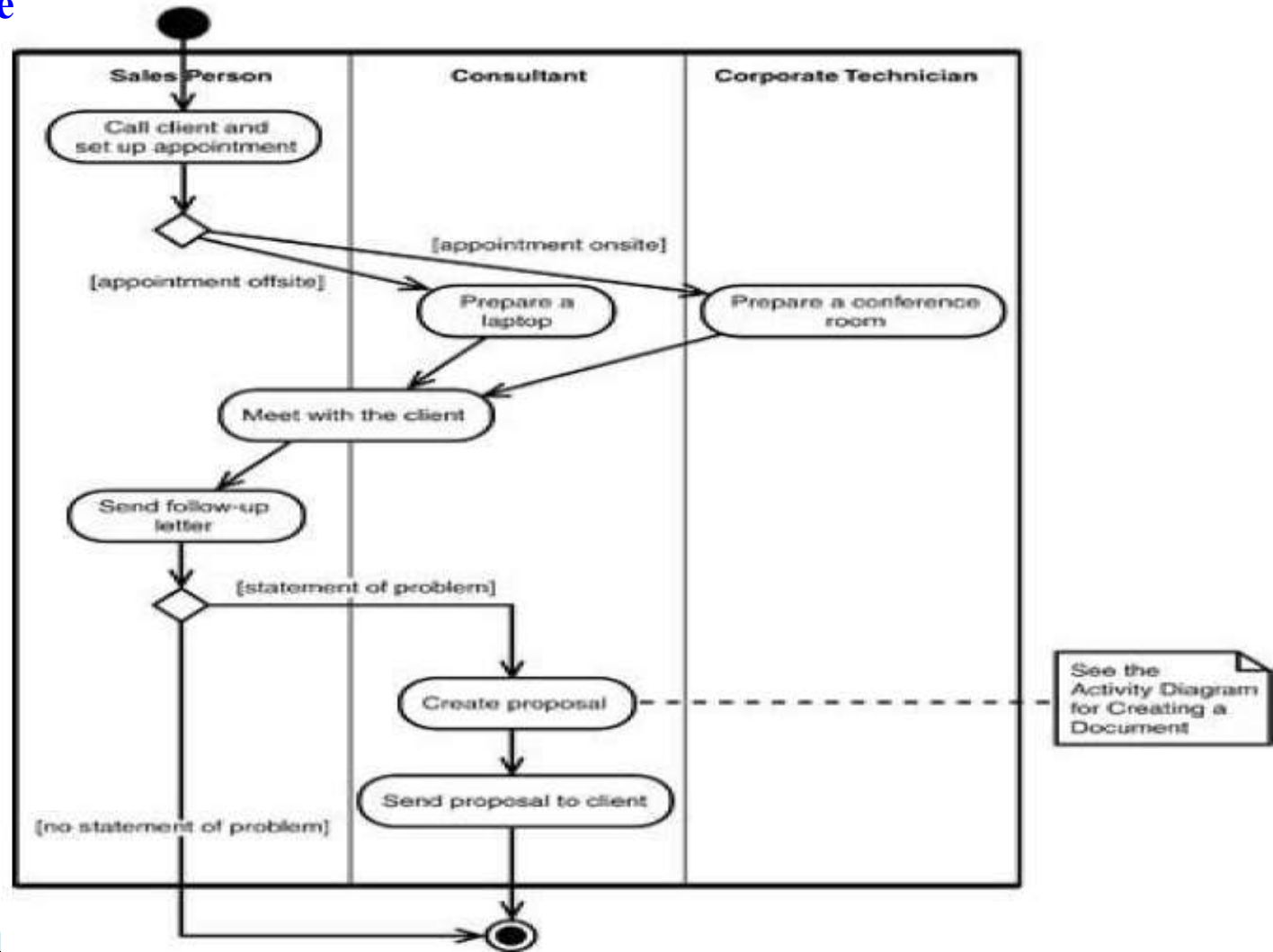
Example 5: business process of meeting a new client

Without Swimlane for
create a proposal



Example 5: business process of meeting a new client

With Swimlane

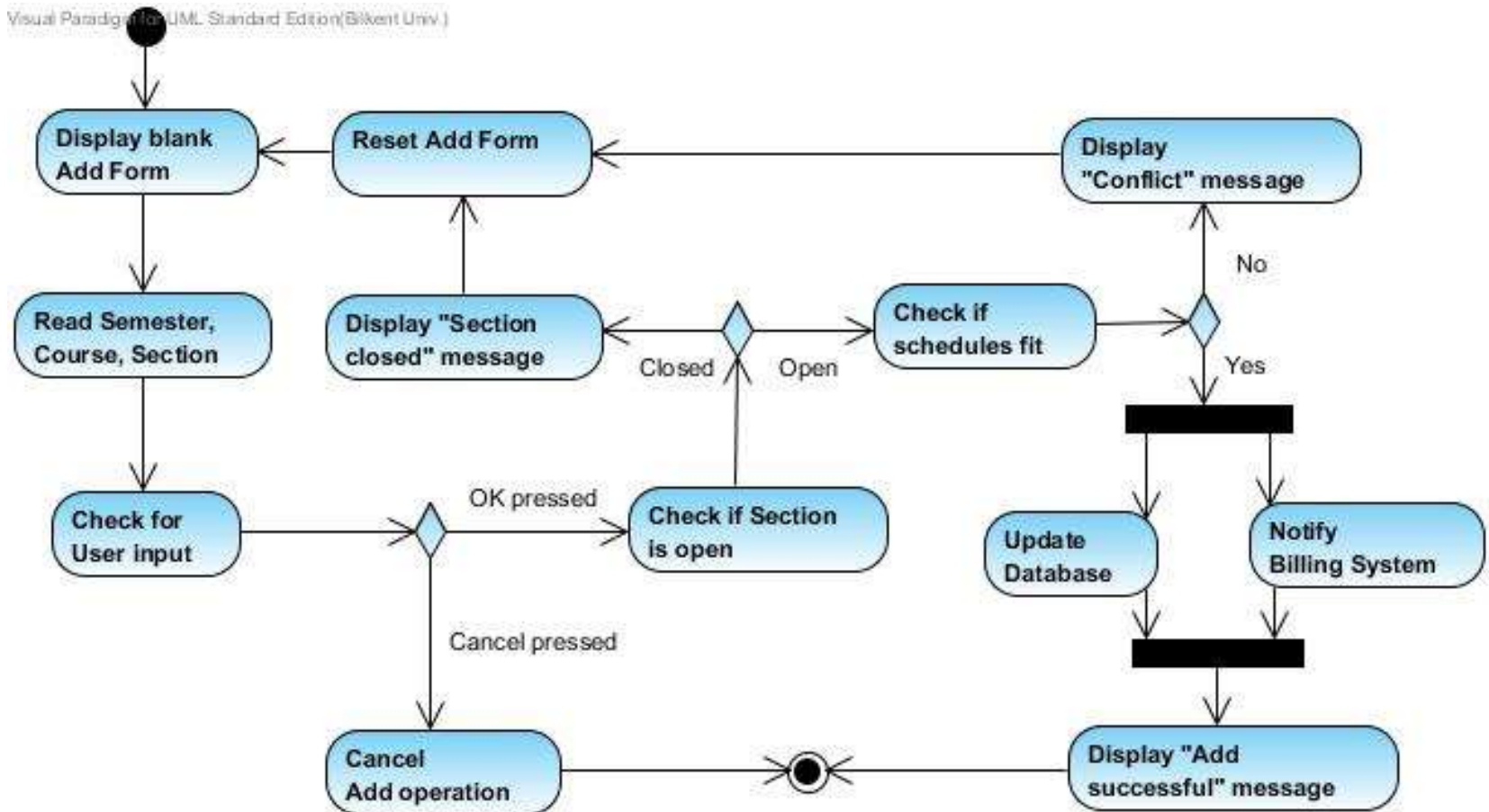


Homework 1

A student wants to add a new course. The student fills out the form by specifying the semester, the course to take (department and course number) and the section, to which the student would like to be added. Then, the student clicks the OK button. The system checks whether the particular section is still open for registration and the maximum count hasn't been reached. If so, the system checks if the particular section of the added course fits the student's schedule. Add operation is not allowed when there are any conflicts in the schedule. **If there is no conflict**, the system updates the database and **simultaneously** notifies the billing system of the change. It **then** displays an appropriate message. The student may, of course, cancel the add operation at any point during this process.

Homework 1

Visual Paradigm for UML Standard Edition (Bilkent Univ.)



Homework 2

Draw an activity diagram for the following problem:

Appointment system for doctor office.

1. A patient came to office, the scheduler get patient info.
2. If the patient is new the scheduler make new patient record.
3. The scheduler display list of possible appointments to patient.
4. Patient choose new appointments , modify appointments or cancel his appointments .
5. Patient make payment.

Hints:

There are about 6 to 8 activities and 2 to 5 objects.



Summary: Requirements Analysis

▶ 1. What are the transformations?



Functional Modeling

- Create *scenarios and use case diagrams*

- Talk to client, observe, get historical records, do thought experiments

2. What is the structure of the system?



Object Modeling

Create *class diagrams*

Identify objects.

What are the associations between them? What is their multiplicity?

What are the attributes of the objects?

What operations are defined on the objects?

3. What is its behavior?



Dynamic Modeling

Create *sequence diagrams*

Identify senders and receivers

Show sequence of events exchanged between objects. Identify event dependencies and event concurrency.

Create *state diagrams*

Only for the dynamically interesting objects.