Object Oriented Analysis & Design

Engr. Abdul-Rahman Mahmood

DPM, MCP, QMR(ISO9001:2000)

armahmood786@yahoo.com

alphapeeler.sf.net/pubkeys/pkey.htm

pk.linkedin.com/in/armahmood

www.twitter.com/alphapeeler

www.facebook.com/alphapeeler

abdulmahmood-sss alphasecure

armahmood786@hotmail.com

http://alphapeeler.sf.net/me

alphasecure@gmail.com

http://alphapeeler.sourceforge.net

http://alphapeeler.tumblr.com

armahmood786@jabber.org

alphapeeler@aim.com

mahmood_cubix 48660186

alphapeeler@icloud.com

http://alphapeeler.sf.net/acms/

Activity Diagrams

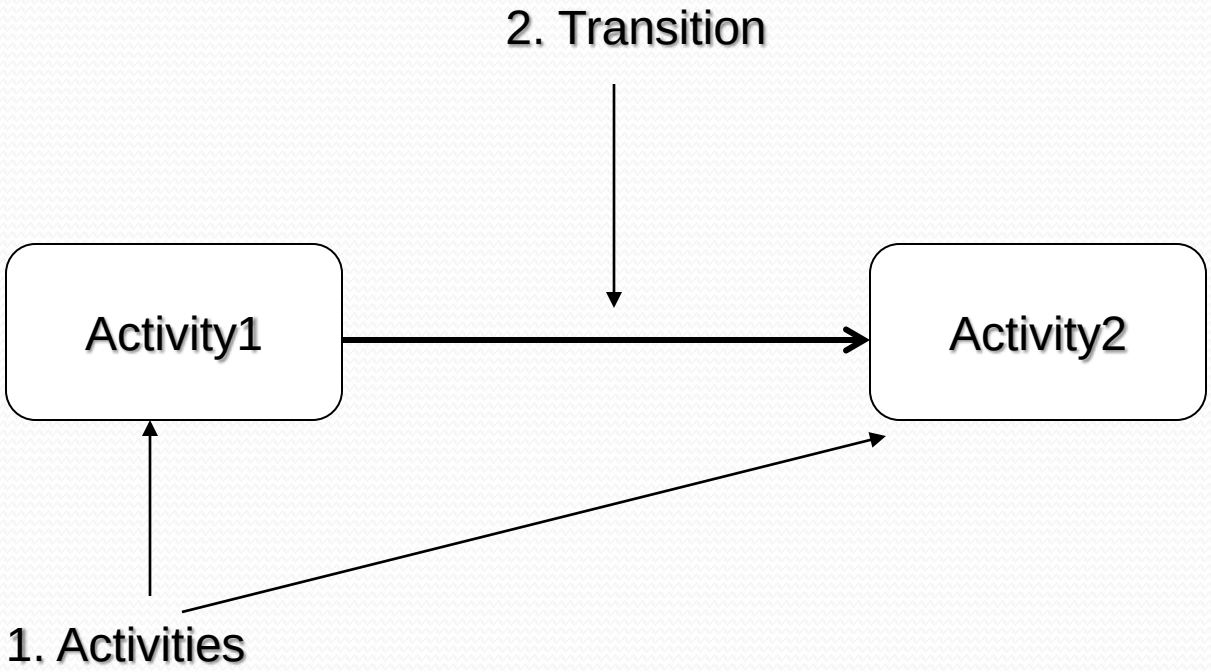
Objectives

- Discuss and understand activity diagrams
- Understand the elements of activity diagrams
  - Activity
  - Transition
  - Synch. Bar
  - Decision Diamond
  - Start & Stop Markers
- What is an activity diagram?
- Example: Student Enrollment in IIT (SEIIT)
- Activity diagram for a use case in SEIIT
- Basic components in an activity diagram and their notations
- Managing the large activity diagram: Swim Lane
- Activity diagram vs. Flow chart

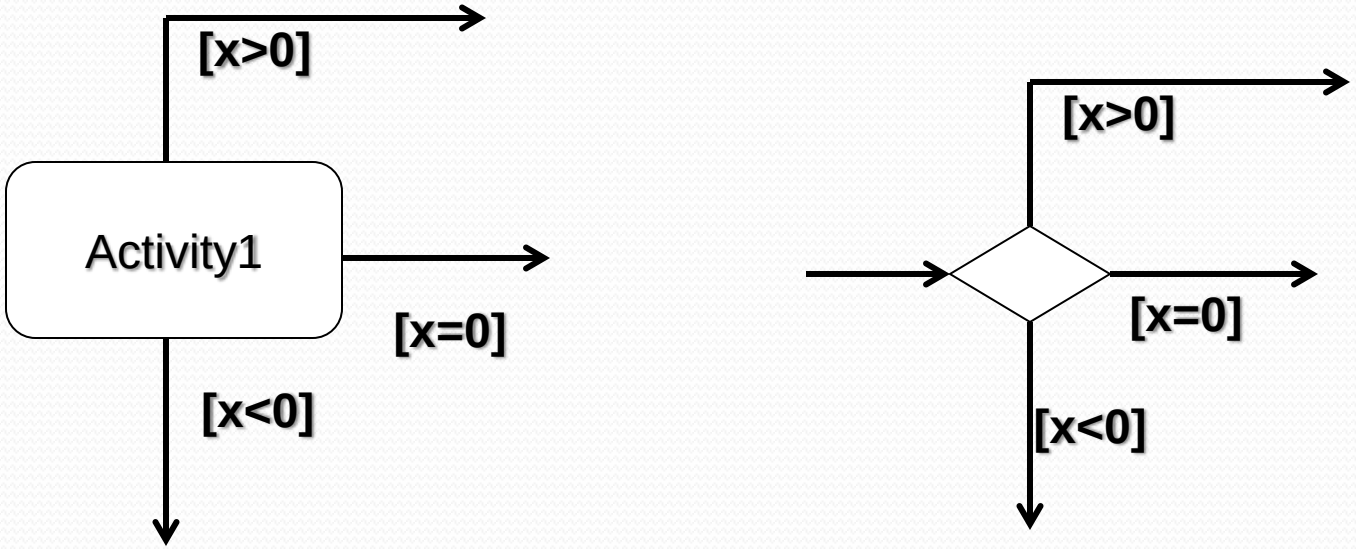
What is an Activity Diagram?

- Represent the dynamic (behavioral) view of a system
- Used for business (transaction) process modeling and modeling the logic captured by a single use-case or usage scenario
- Used to represent flow across use cases or within a use case
- UML activity diagrams are the object oriented equivalent of flow chart and DFDs in function-oriented design approach
- Describes how activities are coordinated.
- Records the dependencies between activities, such as which things can happen in parallel and what must be finished before something else can start.
- Represents the workflow of the process.
- Activity diagram contains activities, transitions between activities, decision points, synchronization bars, swim lanes and many more.

Notation

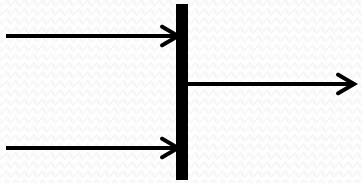


Notation - 2

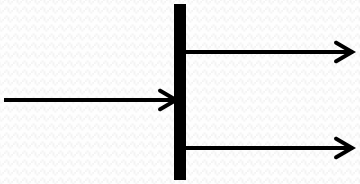


3. Decision Diamond

Notation - 3



4.1 Synch. Bar (Join)



4.2 Splitting Bar (Fork)

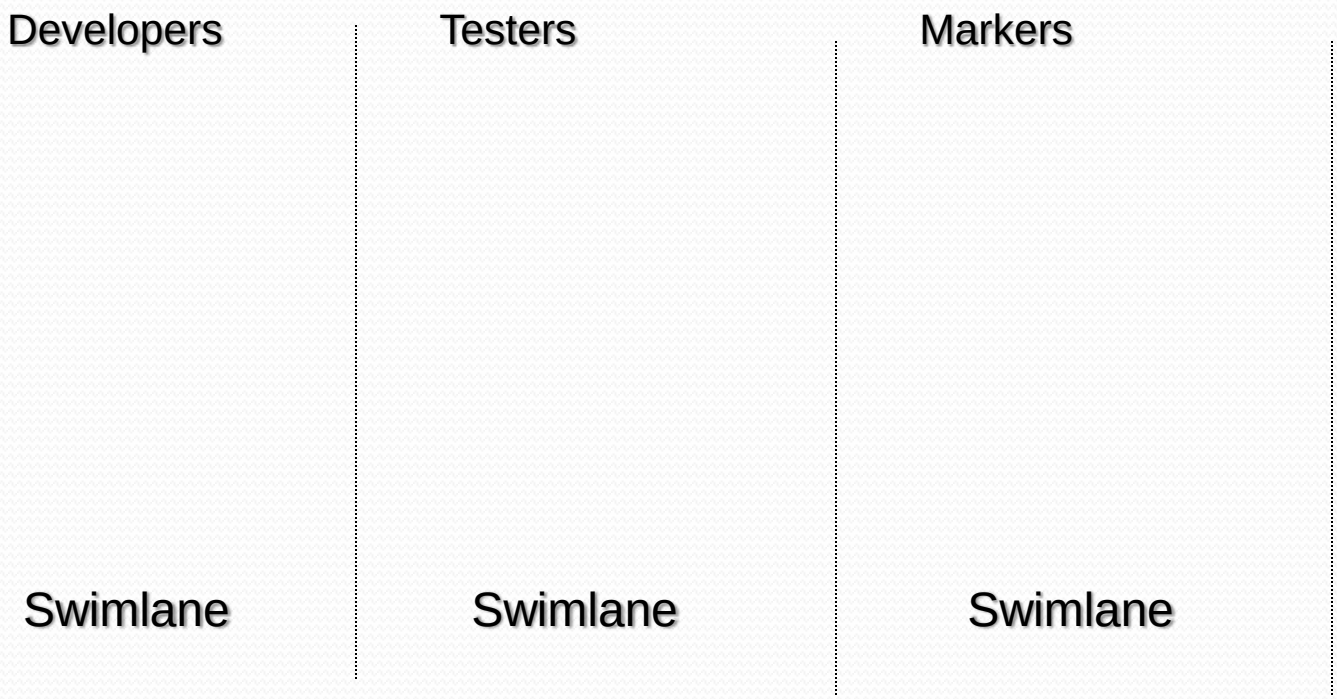
Notation - 3

Start Marker

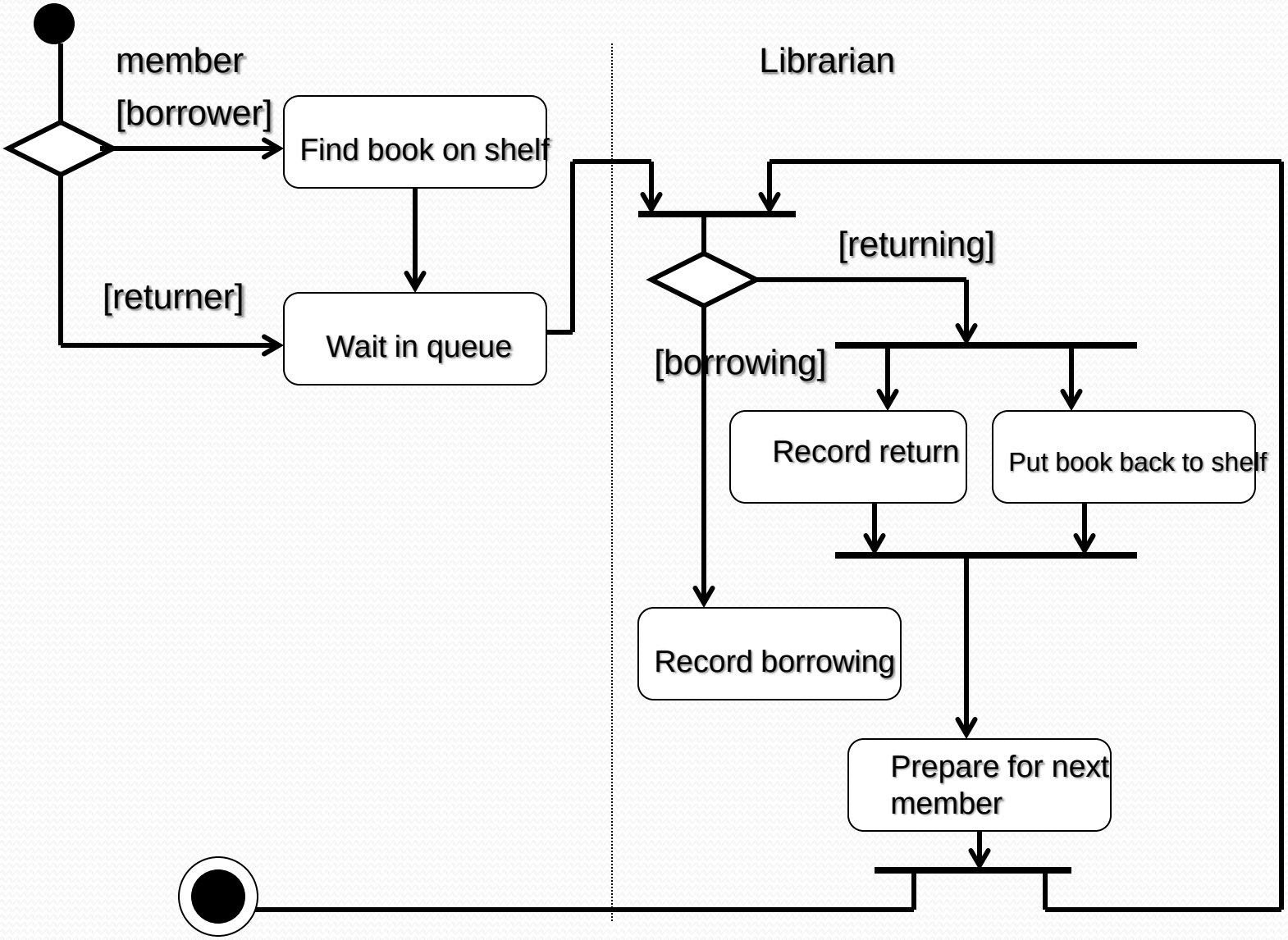
Stop Marker

5. Start & Stop Markers

Notation - 4


Developers
Testers
Markers
Swimlane
Swimlane
Swimlane
Application/Department/Group/Role Boundaries

Example: Business Level Activity Diagram of Library



Activity Diagrams (1)

- To model the ***dynamic*** aspects of a system
- It is essentially a ***flowchart***
  - Showing ***flow of control*** from activity to activity
- Purpose
  - Model business workflows
  - Model operations

Activity Diagrams (2)

- Activity diagrams commonly contain
  - Activity states and action states
  - Transitions
  - Objects

Action States and Activity States

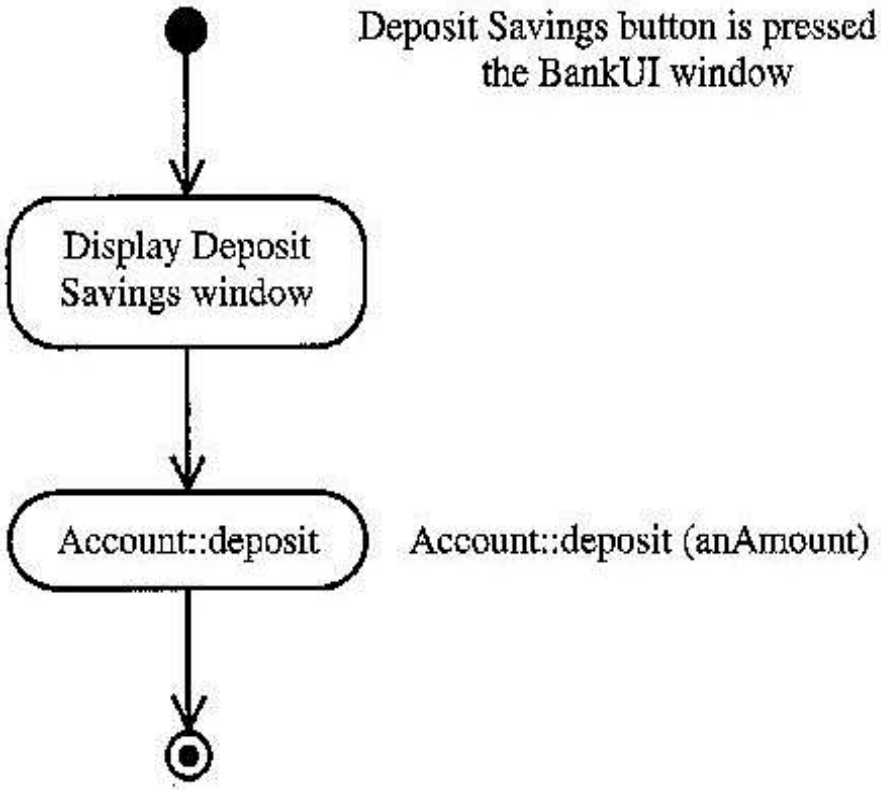
- **Action** states
  - Represents the execution of an atomic action, typically the invocation of an operation.
  - Work of the action state is not interrupted
- **Activity** states can be further decomposed
  - Their activity being represented by other activity diagrams
  - They may be interrupted
- ActionState has is replaced, in UML 2.0, by Action.

Transitions (1)

- When the action or activity of a state completes, flow of control passes immediately to the next action or activity state
- A flow of control has to start and end someplace
  - initial state -- a solid ball
  - stop state -- a solid ball inside a circle

Activity Diagram: Example (1)

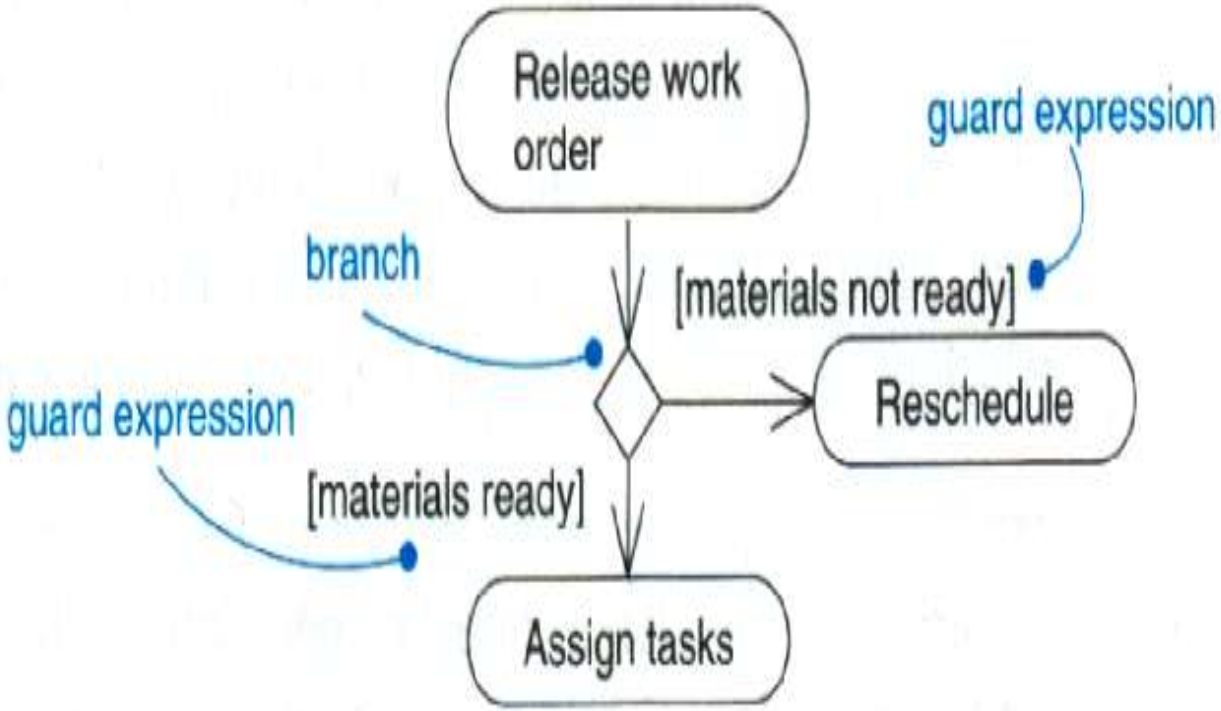
Activity diagram for processing a deposit to a savings account.



Branching (1)

- A branch specifies alternate paths taken based on some Boolean expression
- A branch may have one incoming transition and two or more outgoing ones

Branching (2)


Release work order
guard expression
branch
[materials not ready]
Reschedule
guard expression
[materials ready]
Assign tasks

Activity Diagram: Example (2)

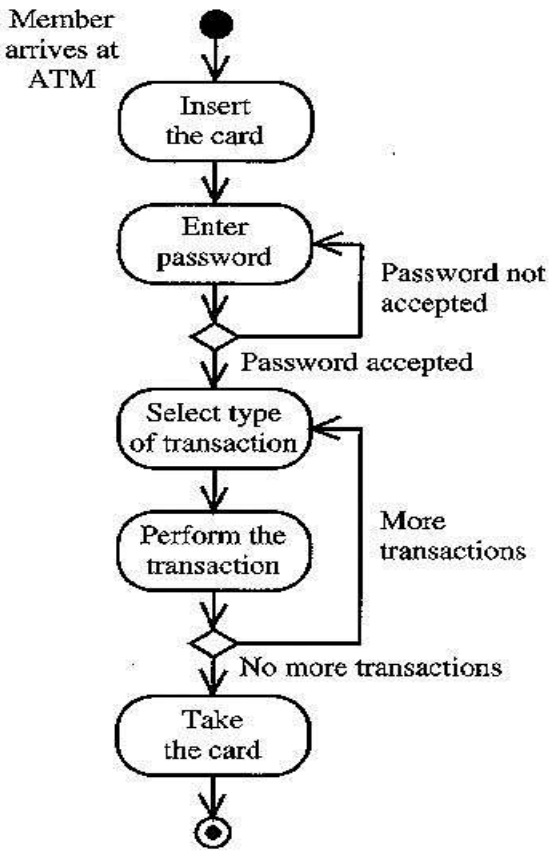


FIGURE 6–8
Activities involved in an ATM transaction.

Forking and Joining

- Use a synchronization bar to specify the forking and joining of parallel flows of control
- A synchronization bar is rendered as a thick horizontal or vertical line

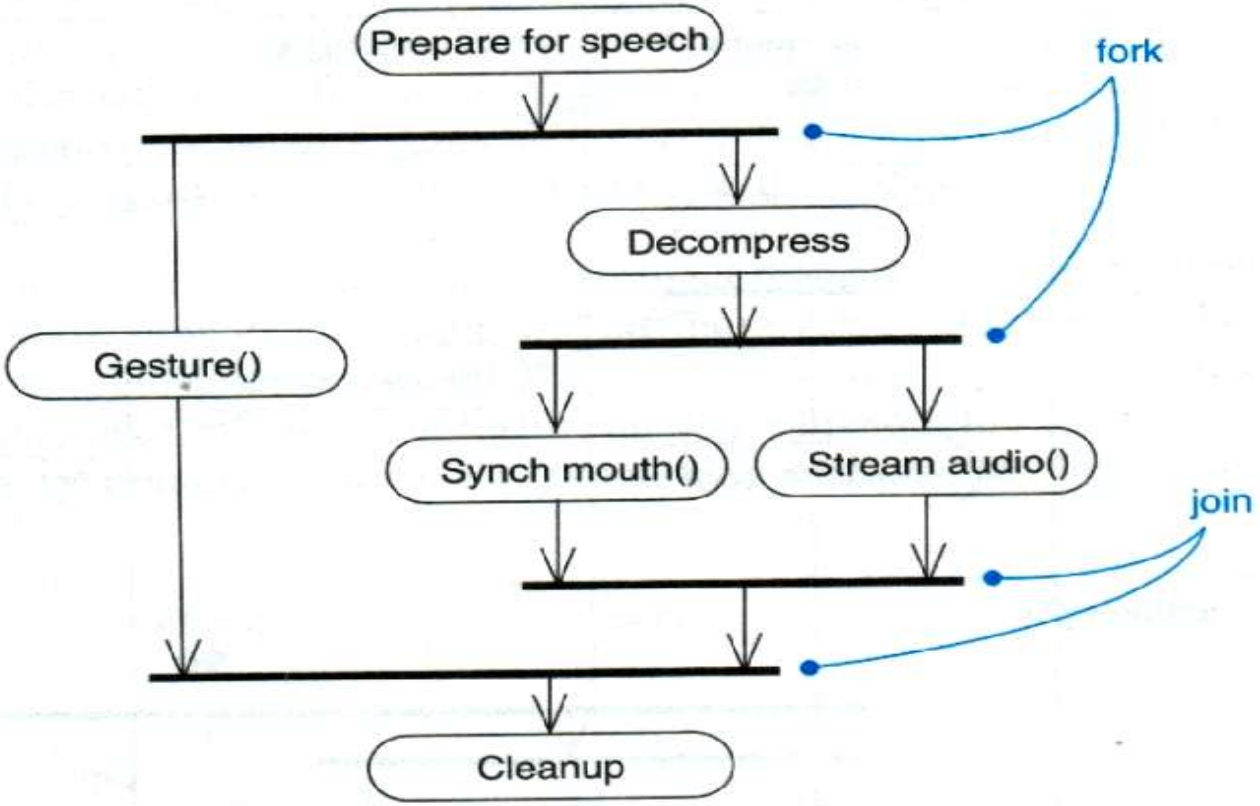
Fork

- A fork may have one incoming transitions and two or more outgoing transitions
  - each transition represents an independent flow of control
  - conceptually, the activities of each of outgoing transitions are concurrent
    - either truly concurrent (multiple nodes)
    - or sequential yet interleaved (one node)

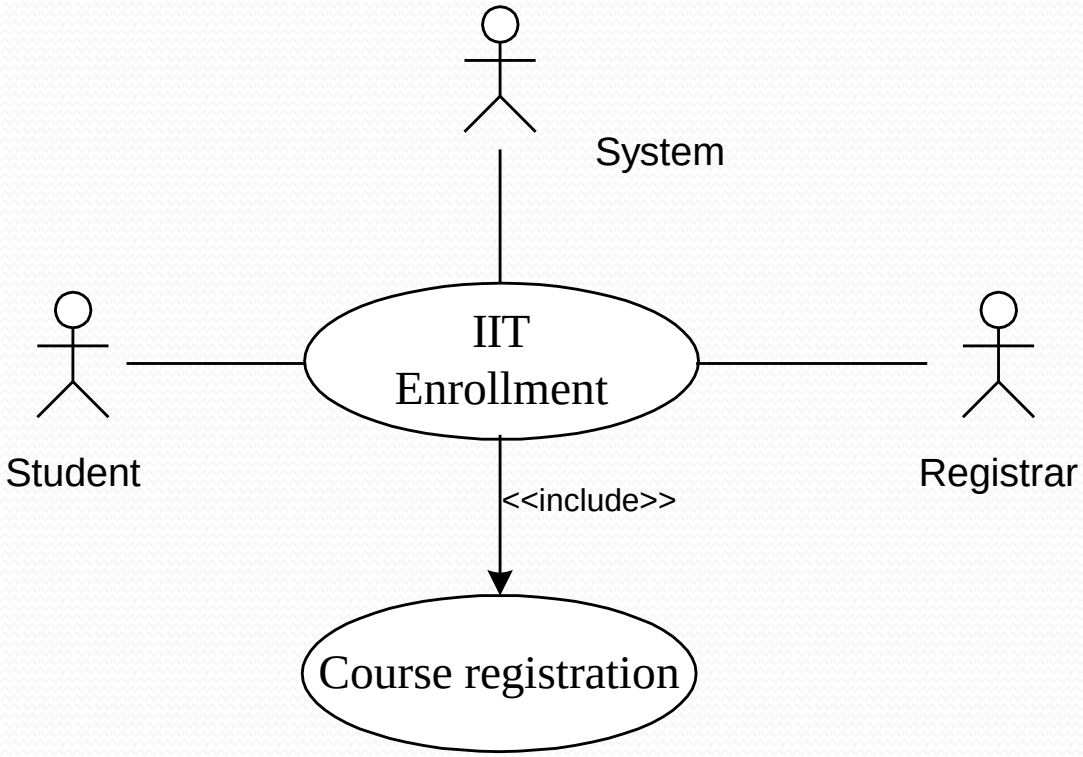
Join

- A join may have two or more incoming transitions and one outgoing transition
  - above the join, the activities associated with each of these paths continues in parallel
  - at the join, the concurrent flows synchronize
    - each waits until all incoming flows have reached the join, at which point one flow of control continues on below the join

Fork


Prepare for speech
fork
Decompress
Gesture()
Synch mouth()
Stream audio()
join
Cleanup

Case Study

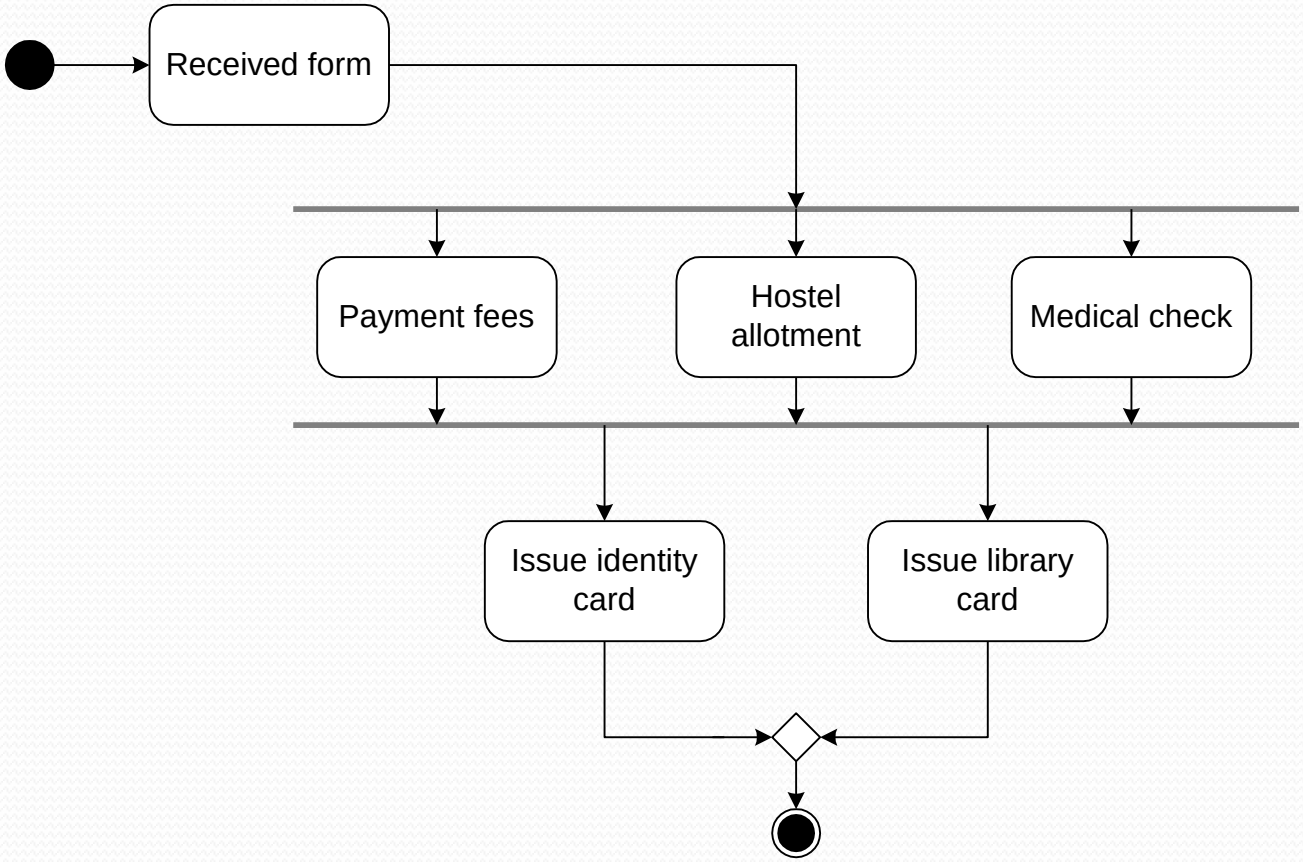


Student Enrollment in IIT (SEIIT)

SEIIT System

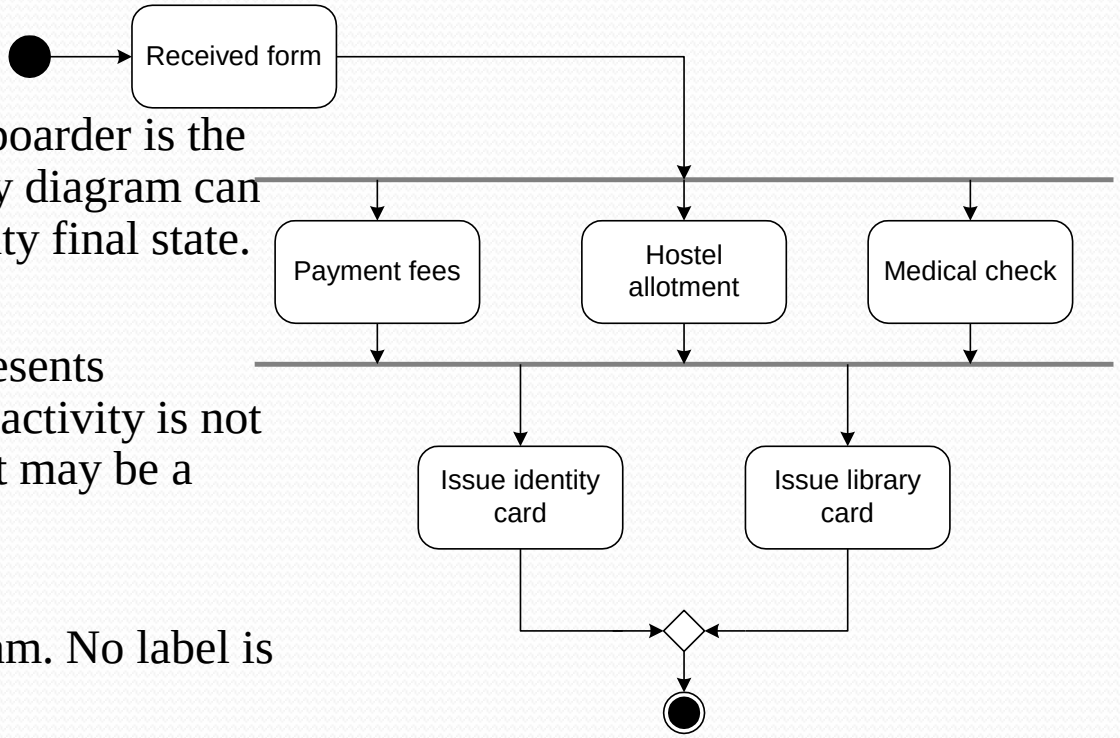
- Here different activities are:
  - Received enrollment form filled by the student
    - Registrar checks the form
    - Input data to the system
    - System authenticate the environment
  - Pay fees by the student
    - Registrar checks the amount to be remitted and prepare a bill
    - System acknowledge fee receipts and print receipt
  - Hostel allotment
    - Allot hostel
    - Receive hostel charge
    - Allot room
  - Medical check up
    - Create hostel record
    - Conduct medical bill
    - Enter record
  - Issue library card
  - Issue identity card

Activity Diagram for SEIIT



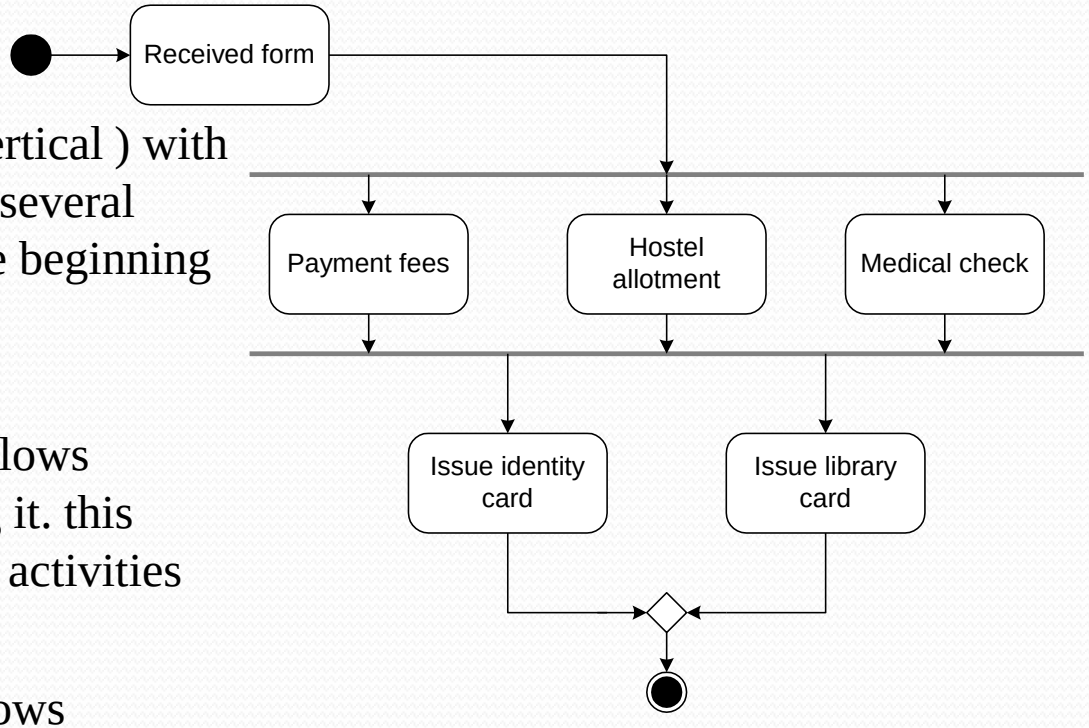
Basic Components in an Activity Diagram

- **Initial node**
  - The filled circle is the starting point of the diagram
- **Final node**
  - The filled circle with a boarder is the ending point. An activity diagram can have zero or more activity final state.
- **Activity**
  - The rounded circle represents activities that occur. An activity is not necessarily a program, it may be a manual thing also
- **Flow/ edge**
  - The arrows in the diagram. No label is necessary



Basic Components in an Activity Diagram

- **Fork**
  - A black bar (horizontal/vertical) with one flow going into it and several leaving it. This denotes the beginning of parallel activities
- **Join**
  - A block bar with several flows entering it and one leaving it. this denotes the end of parallel activities
- **Merge**
  - A diamond with several flows entering and one leaving. The implication is that all incoming flow to reach this point until processing continues

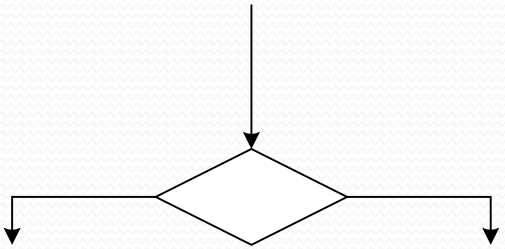


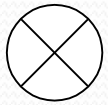
Basic Components in an Activity Diagram

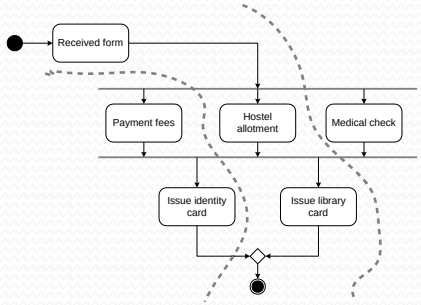
- Difference between Join and Merge
  - A join is different from a merge in that the join synchronizes two inflows and produces a single outflow. The outflow from a join cannot execute until all inflows have been received
  - A merge passes any control flows straight through it. When all incoming flow reach this point then processing continues

Basic Components in an Activity Diagram

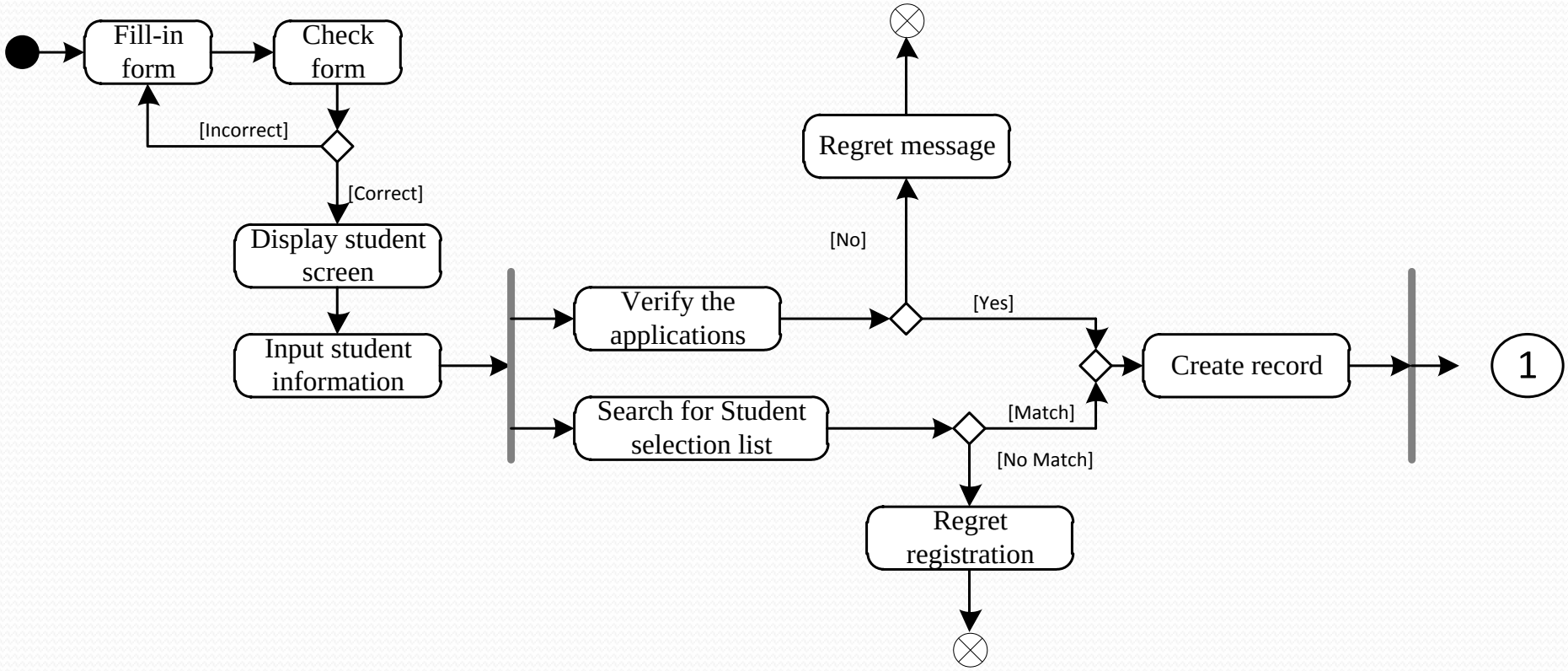
- **Decision**
  - A diamond with one flow entering and several leaving. The flow leaving includes conditions as yes/ no state
- **Flow final**
  - The circle with X though it. This indicates that Process stop at this point
- **Swim lane**
  - A partition in activity diagram by means of dashed line, called swim lane. This swim lane may be horizontal or vertical



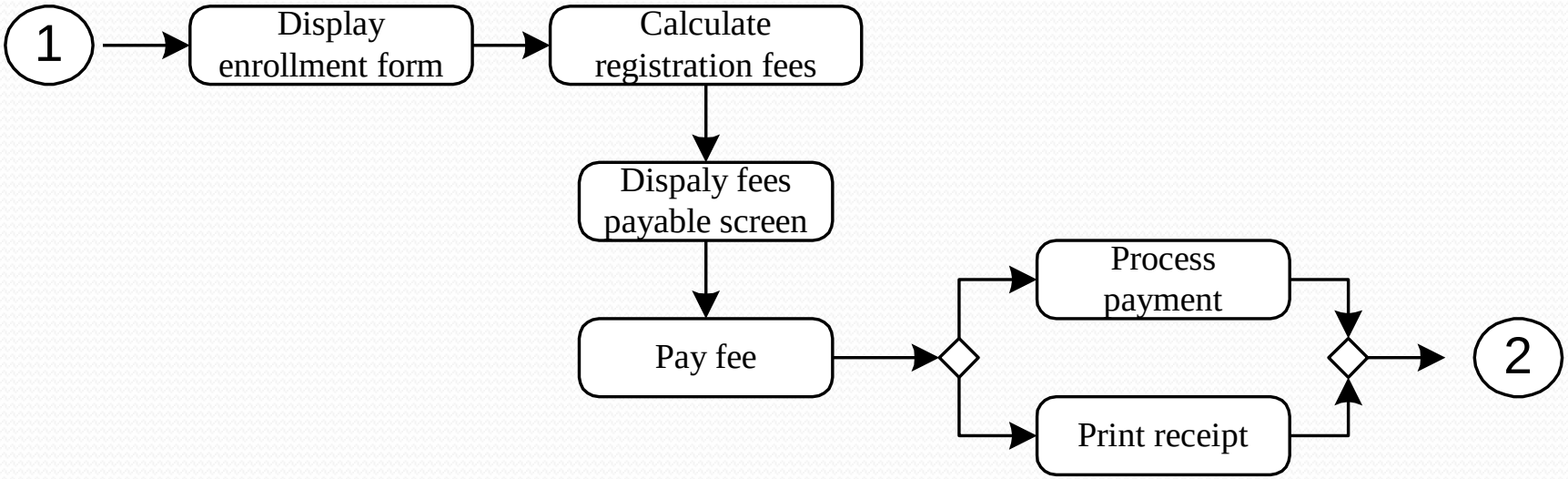




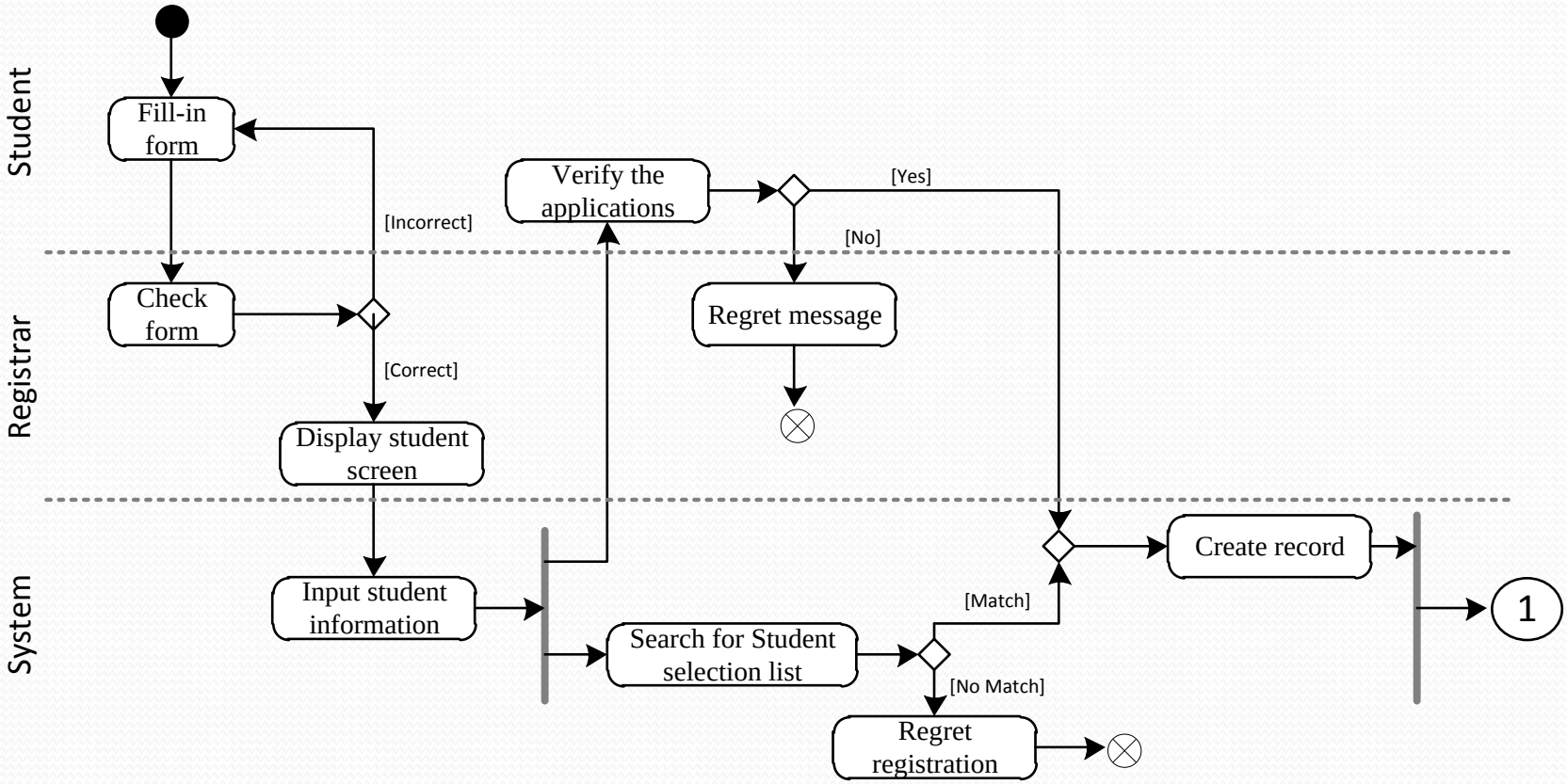
Detailed Activity Diagram of SEIIT



Detailed Activity Diagram of SEIIT



Activity Diagram of SEIIT with Swim Lane



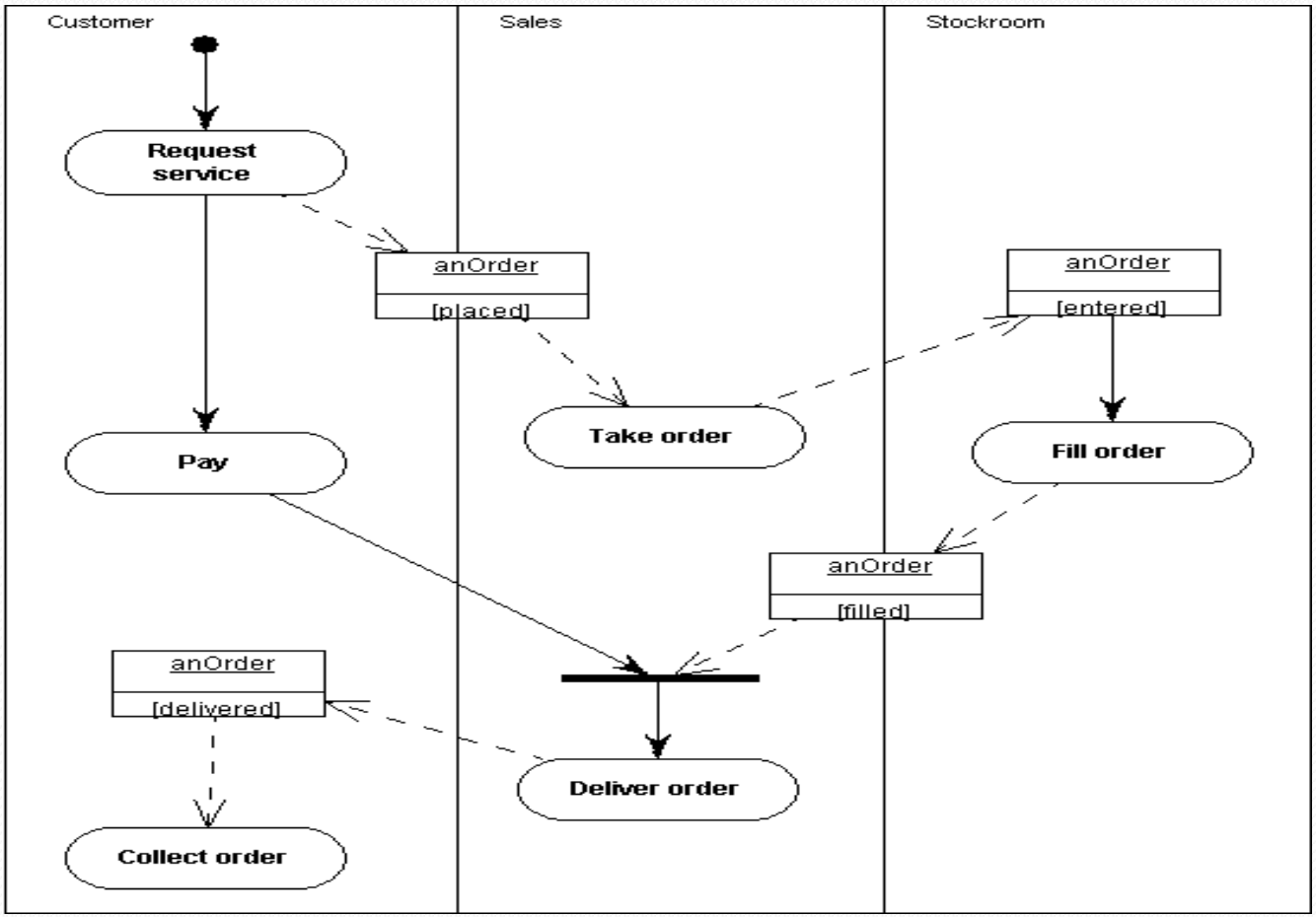
Swimlanes

- A swimlane specifies a locus of activities
- To partition the activity states on an activity diagram into groups
  - each group representing the business organization responsible for those activities
  - each group is called a swimlane
- Each swimlane is divided from its neighbor by a vertical solid line

Swimlanes

- Each swimlane has a name unique within its diagram
- Each swimlane may represent some real-world entity
- Each swimlane may be implemented by one or more classes
- Every activity belongs to exactly one swimlane, but transitions may cross lanes

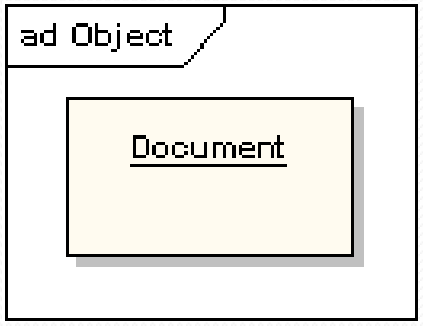
Activity Diagram: Example (3)

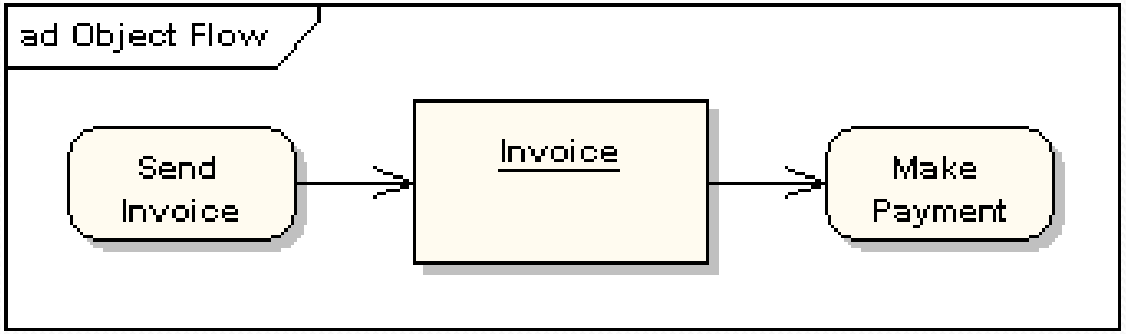


Some more features in Activity Diagrams

Object and Object Flow

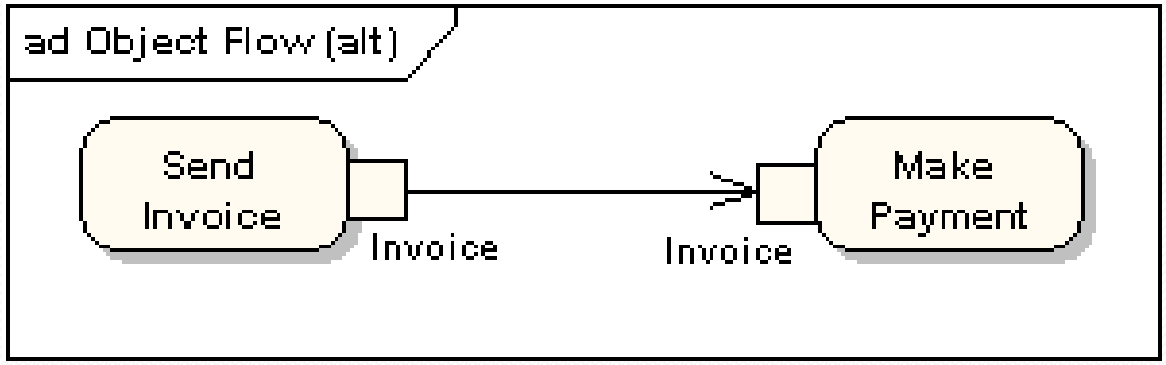
- There is no data flow in activity diagram. Object flow plays role of data flow as well.
- An object flow is a path along which objects can pass. An object is shown as a rectangle
- An object flow is shown as a connector with an arrowhead denoting the direction the object is being passed.





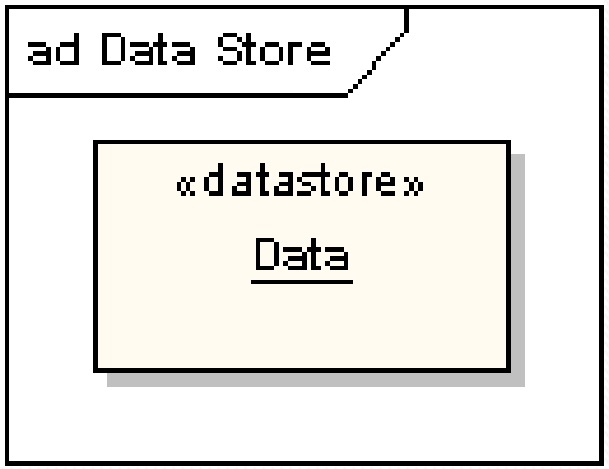
Input and Output Pin

- An object flow must have an object on at least one of its ends. A shorthand notation for the above diagram would be to use input and output pins



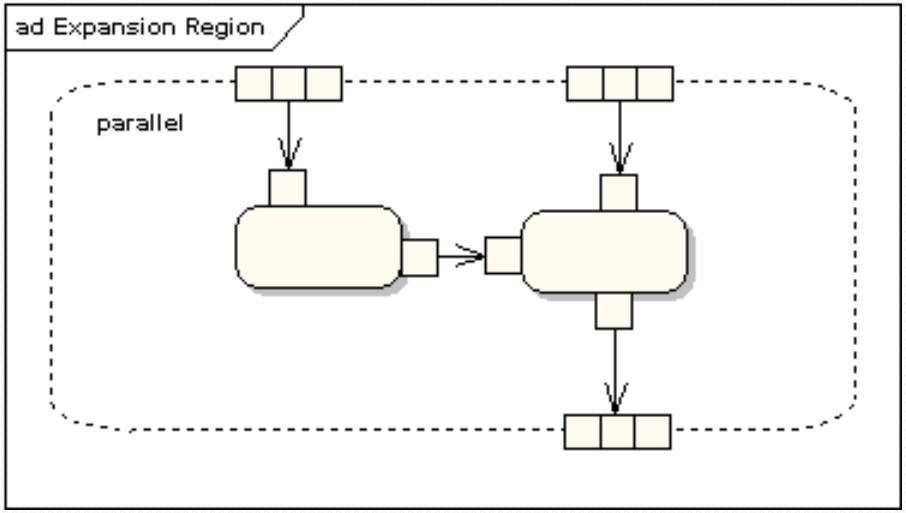
Data Store

- A data store is shown as an object with the «datastore» keyword



Expansion Region

- Expansion regions and structured activities provide mechanisms to show the strategy for managing the synchronization issues in a real-time system.
- An expansion region is a structured activity region that executes multiple times for collection items.
- Input and output expansion nodes are drawn as a group of three / four boxes representing a multiple selection of items. The keyword iterative, parallel or stream is shown in the top left corner of the region



Expansion Region: application-processing

The organization must accept an electronic application, review the eligibility of the application, store the application, and then confirm receipt of a valid application. The top section on the diagram shows an expansion region, a type of structured activity that handles the input and output of collections with multiple executions.

- The collection as two sets of four boxes on the top and one at the bottom showing the application, the personal information, and the verified applications.
- The italicized word in the upper-left corner shows that this expansion
- region allows multiple executions to the actions to occur in parallel, or concurrently.
- The action executions do not have to follow any ordering on the entering
- collections. You can also use *iterative to show the region only allows one set* of actions to execute at a time or *streaming to show execution in the order of the* collection.
- The system also relies on information from a database about the person.
- When combined with the eligibility rules for the application, shown on the diagram as a <<decisionInput>>, the organization filters out ineligible applications and sends the application on for storage.

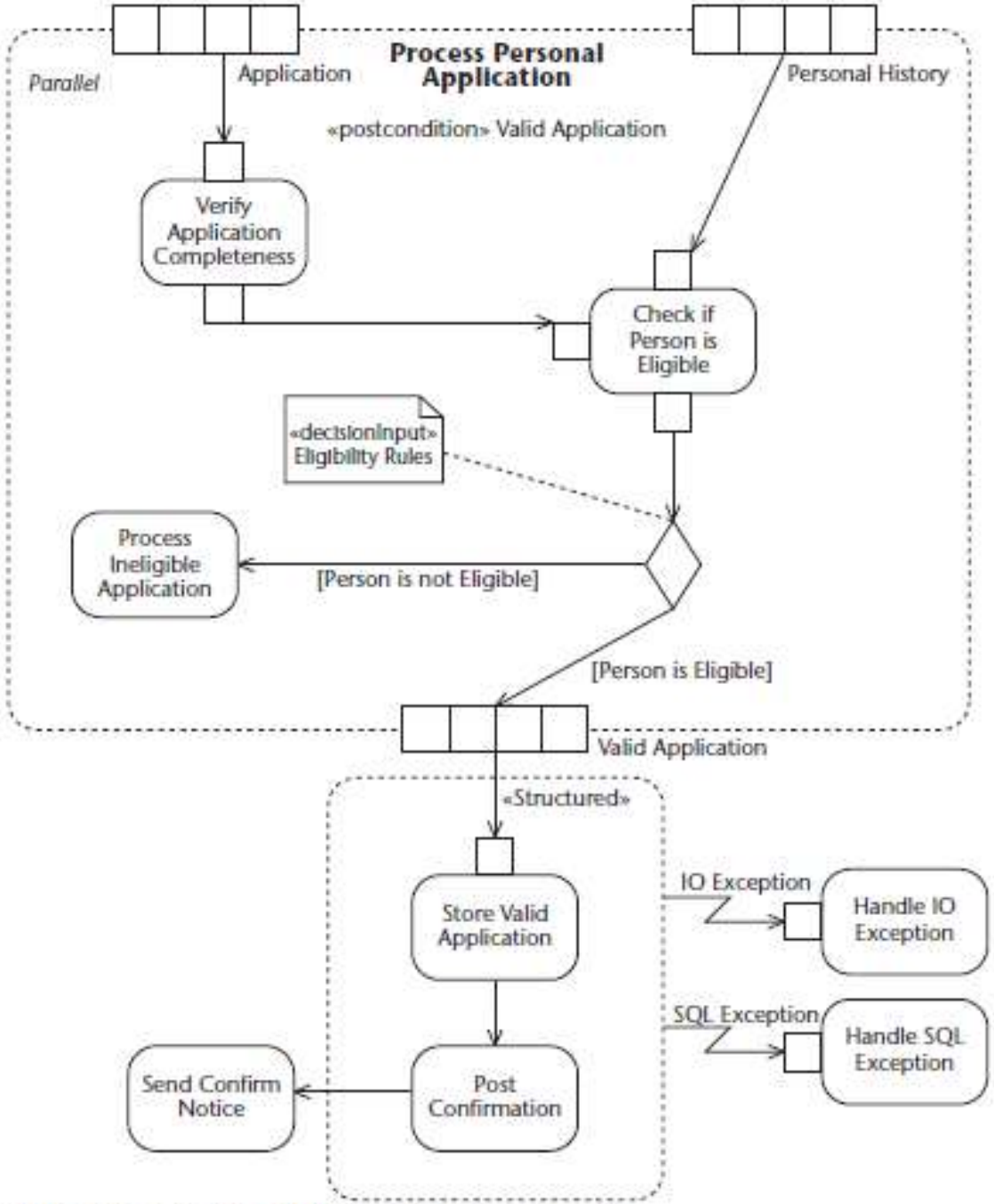
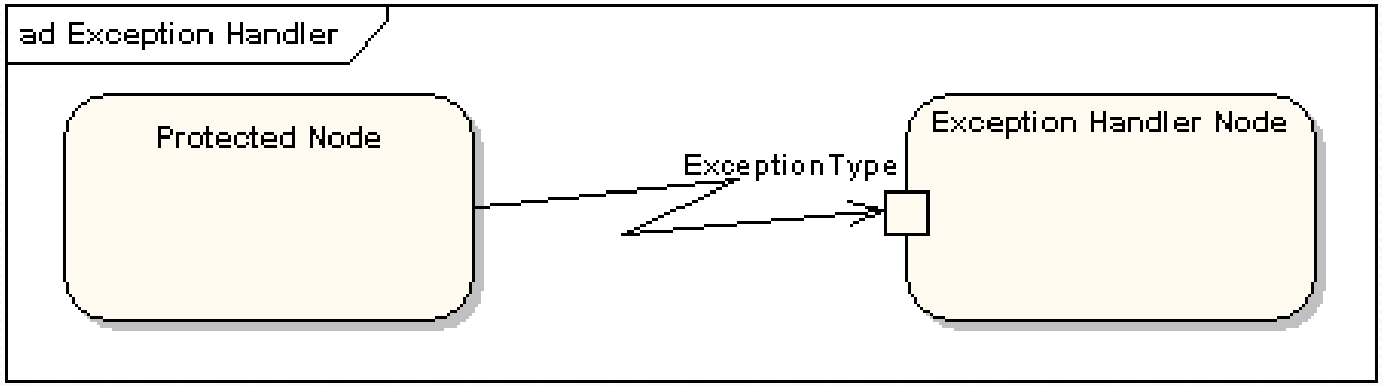


Figure 6.8 Managing applications for monetary grants.

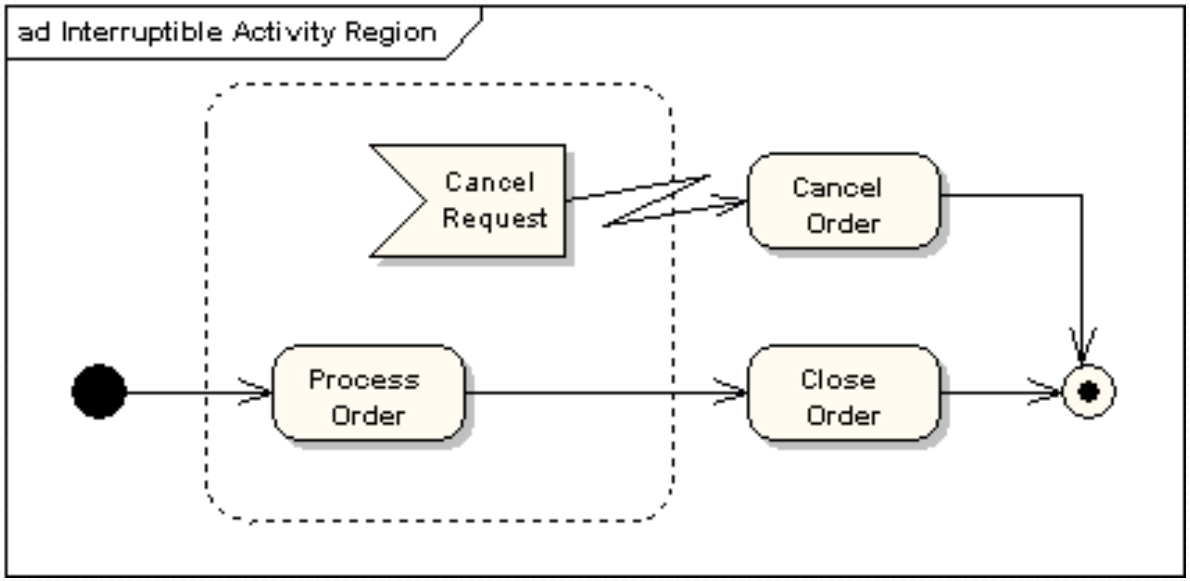
Exception Handling

- Exception Handlers can be modeled on activity diagrams as in the example below

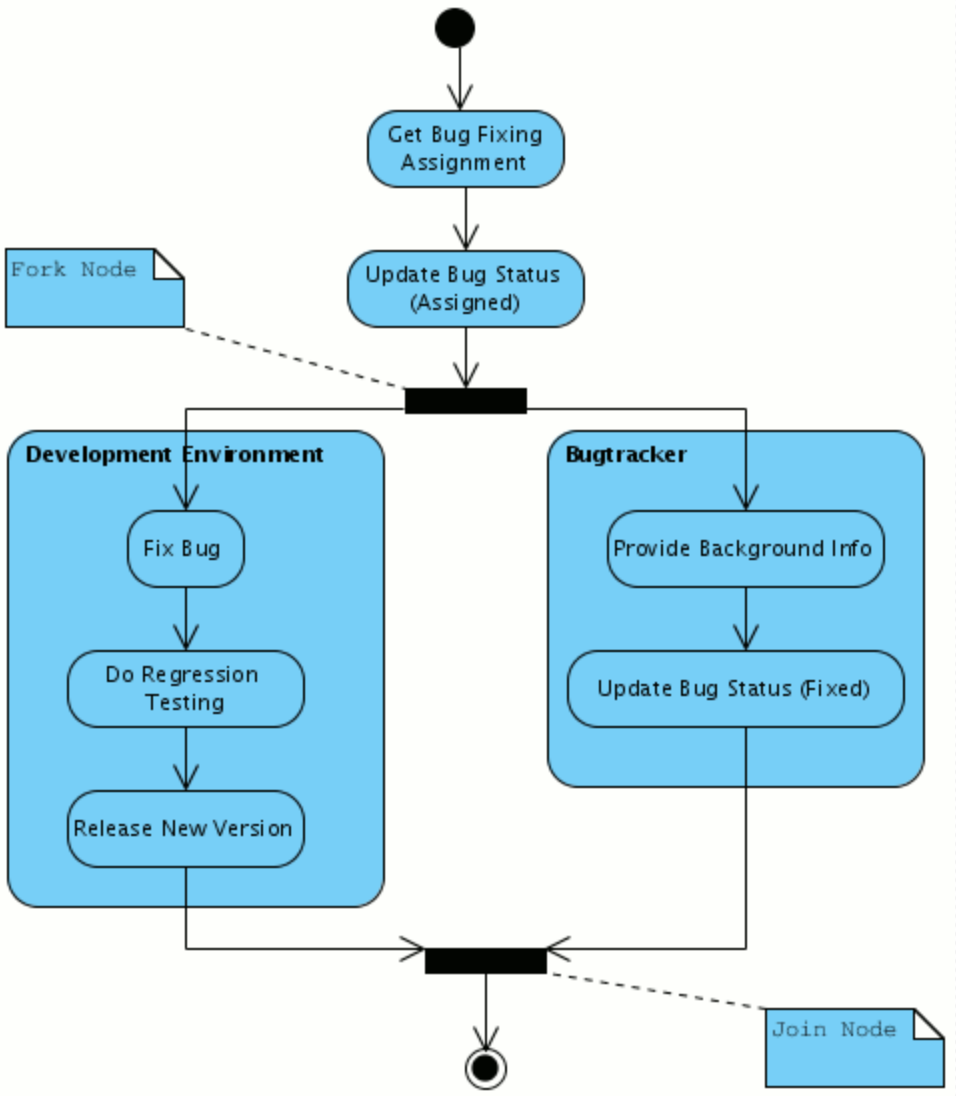


Interruptible Activity Region

- An interruptible activity region surrounds a group of actions that can be interrupted. In the very simple example below, the Process Order action will execute until completion, when it will pass control to the Close Order action, unless a Cancel Request interrupt is received which will pass control to the Cancel Order action

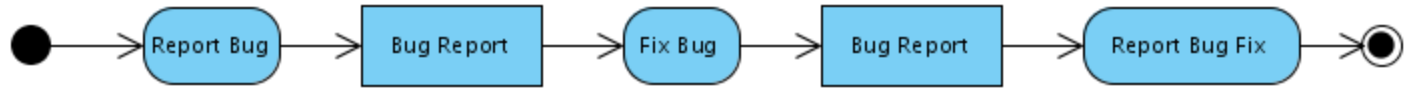


Parallel Activities: Forking and Joining

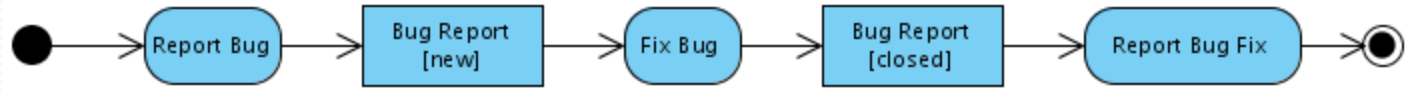


Passing Objects Between Actions

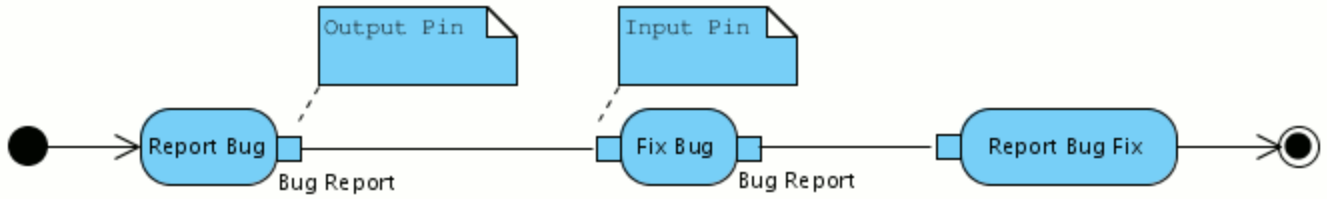
- Objects in the UML language hold information - state - that is passed between actions. Objects may represent class instances:



- Objects usually change state between actions:

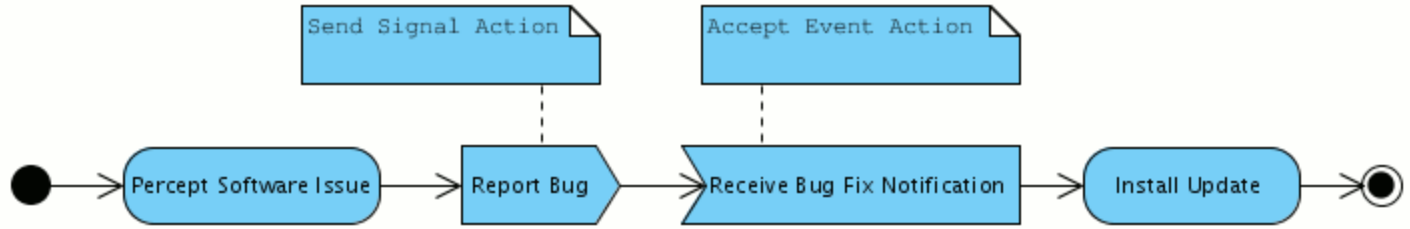


- An alternative notation shows action ***input pins*** and ***output pins***. These emphasize that the corresponding object is *required* while ***object nodes*** rater emphasize the *existance* of that object:



Interacting with External Participants

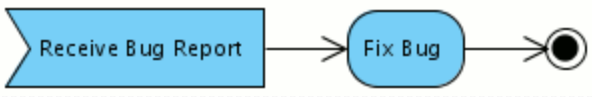
- External participants may be external processes, systems, or people, interacting with an activity.
- A ***receive signal*** "wakes up" an action that is "sleeping".
- A ***send signal*** is sent to an external participant.
- In the following diagram, both a ***send*** and ***receive signal action*** are used.



- Note that the activity flow gets interrupted - gets into a wait state - until the bug fix notification is received. (If there was no ***receive signal action***, however, the flow would just continue after executing the ***send signal action***.)

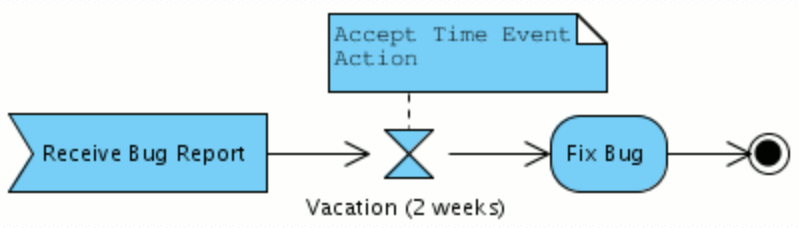
Interacting with External Participants

- An activity may also start with a ***receive signal node***:

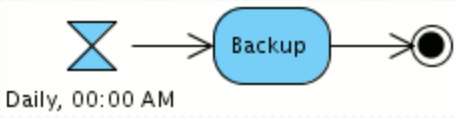


Time Events

- A ***time event*** models a wait period:

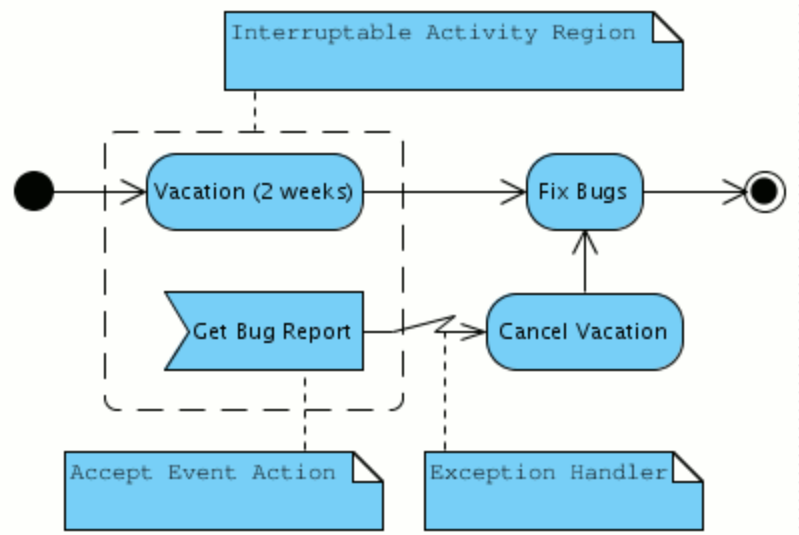


- An activity starting with a time event is launched periodically:



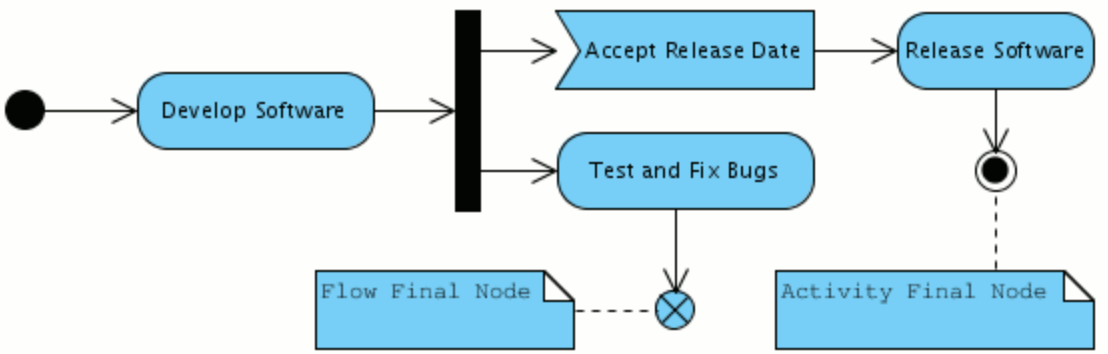
Interrupting an Activity

- Longer running processes can be interrupted by an event - an ***accept event action*** within an ***interruptable activity region:***

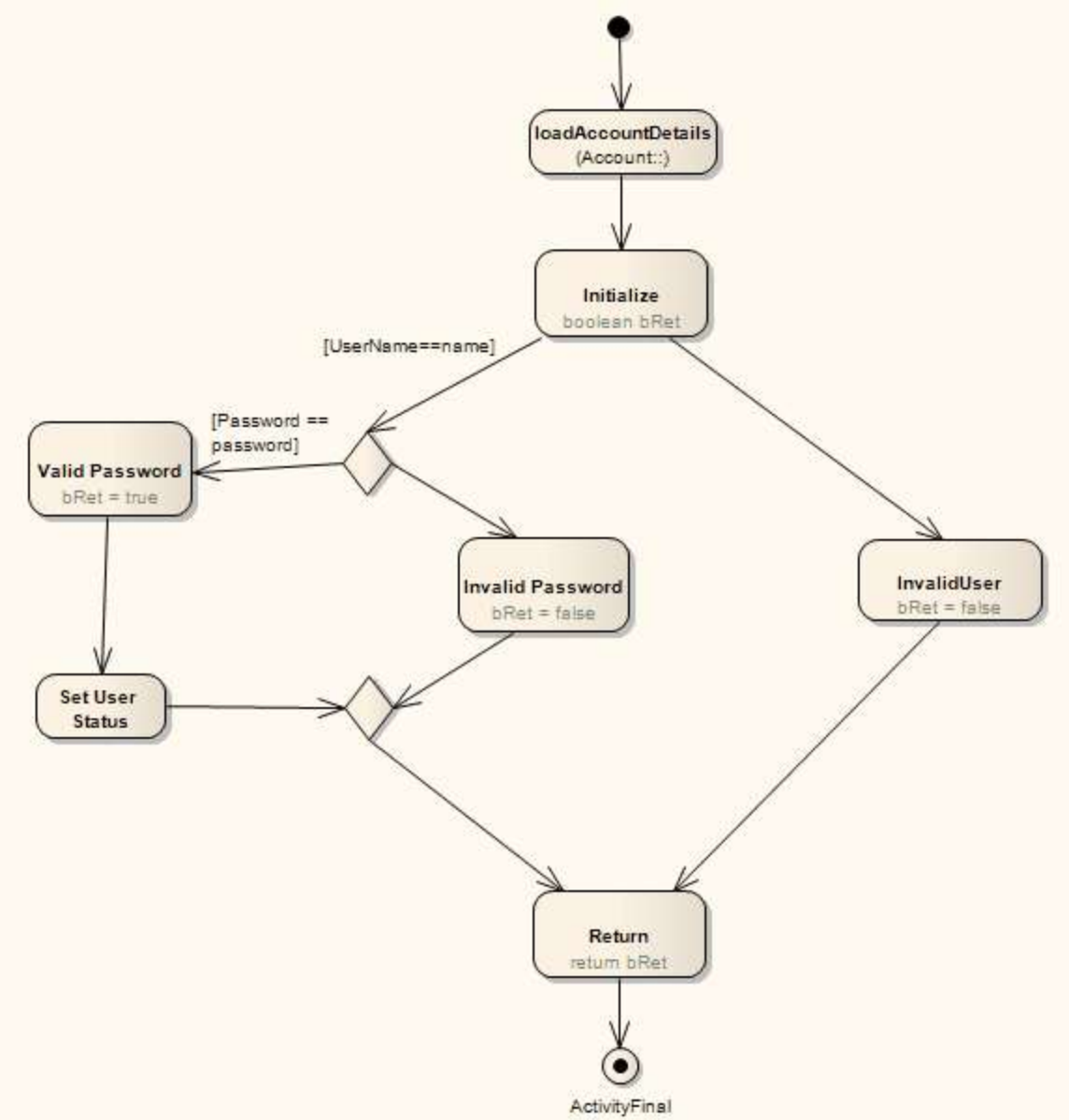


Ending a Flow

- Reaching a ***flow final node*** ends a flow (not the activity). The following diagram models a scenario in which software is tested and bugs are fixed - until the release date is due:



Conditional Statements



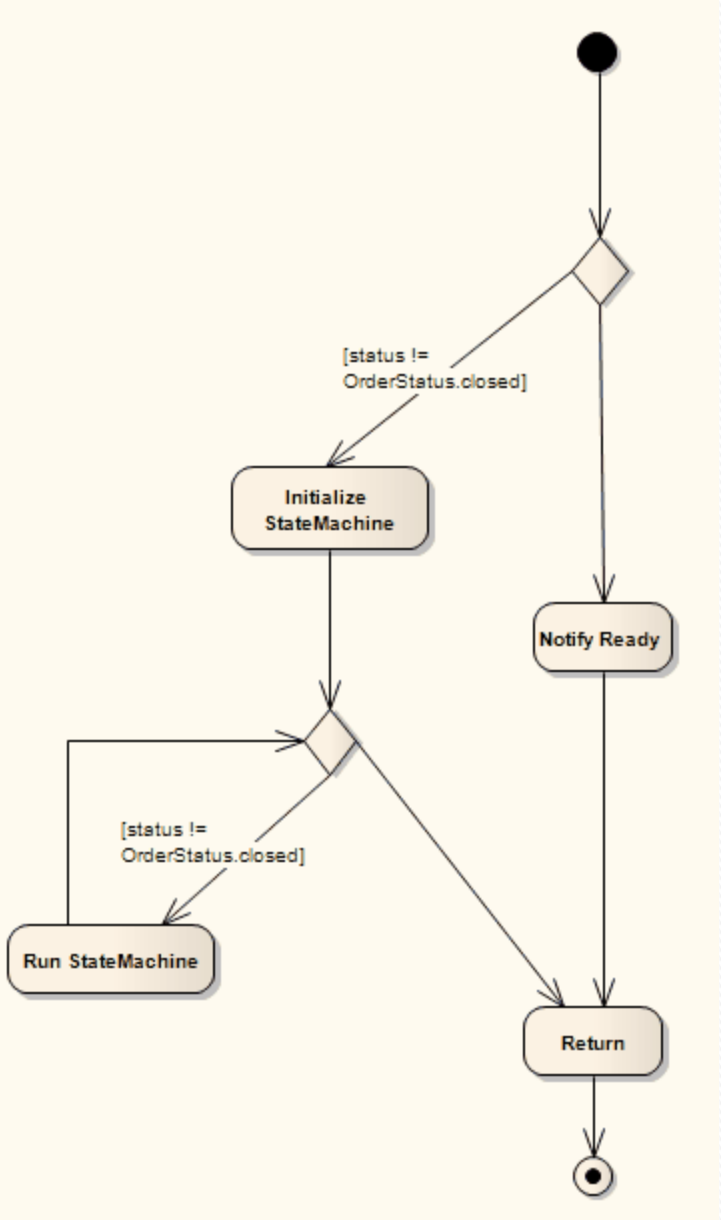
Conditional Statements

```
public boolean doValidateUser(String Password,String UserName)
    {
        loadAccountDetails();
        boolean bRet;
        if (Username==name)
        {
            if (Password == password)
            {
                bRet = true;
                bValidUser = true;
            }
            else
            {
                bRet = false;
            }

        }
        else
        {
            bRet = false;
        }

        return bRet;
    }
```

Loops


[status !=
OrderStatus.closed]
Initialize
StateMachine
Notify Ready
[status !=
OrderStatus.closed]
Run StateMachine
Return

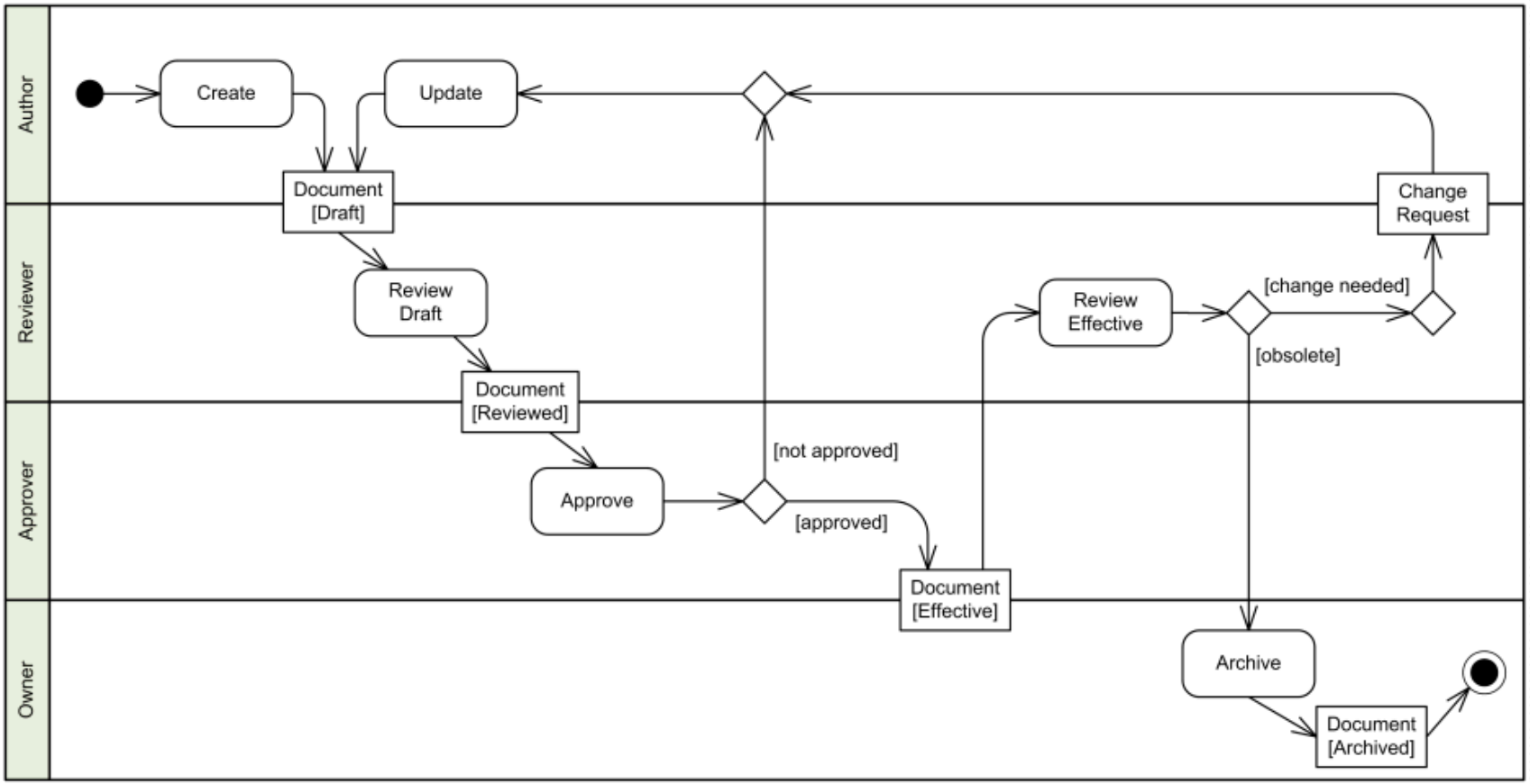
Loops

```
public void doCheckForOutstandingOrders()
    {
        if (status != closed)
        {
            initializeStateMachine();
            while (status != closed)
            {
                runStateMachine();
            }
        }
        else
        {
            //No Outstanding orders;
        }
        return;
    }
```

Class Activity
Draw diagrams

Class Activity Draw diagrams

- **Document Management Process:** A **document** goes through different **state** or stages - it is created, reviewed, updated, approved, and at some point archived. Different roles participating in this process are **Author**, **Reviewer**, **Approver**, and **Owner**.



Ticket Vending Machine

- Activity is started by Commuter **actor** who needs to buy a ticket. Ticket vending machine will request trip information from Commuter. This information will include number and type of tickets, e.g. whether it is a monthly pass, one way or round ticket, route number, destination or zone number, etc.
- Based on the provided trip info ticket vending machine will calculate payment due and request payment options. Those options include payment by cash, or by credit or debit card. If payment by card was selected by Commuter, another actor, Bank will participate in the activity by authorizing the payment.

Ticket Vending Machine