Activity Diagram

Instructor: Mehroze Khan

Activity Diagram

- Activity diagrams provide **visual depictions** of the **flow of activities**, whether in a system, business, workflow, or other process.
- These diagrams focus on the **activities that are performed** and **who (or what)** is responsible for the performance of those activities.
- The elements of an activity diagram are **action nodes, control nodes, and object nodes**.
- There are three **types of control nodes**:
  - Initial and final (final nodes have two types, activity final and flow final)
  - Decision and merge
  - Fork and join

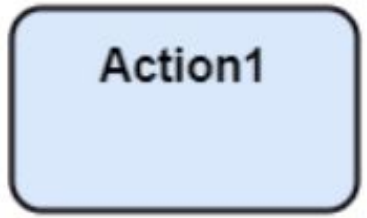
Activity

- The purpose of an activity diagram is to **show the steps within a complex process** and the **sequencing constraints** among them.
- Some activities run forever until an outside event interrupts them, but most activities eventually complete their work and terminate by themselves.
- The completion of an activity is a **completion event** and usually indicates that the next activity can be started.

Syntax of Activity Diagram

Action Nodes

- Actions are the **elemental unit of behavior** in an activity diagram. Action nodes represent a specific task or action that needs to be performed within the activity.
- **Example**: "Send email", "Calculate total", "Submit form."



Action box

Control Nodes

- Control nodes are used to **manage the flow of control between different activities**. They guide how activities are executed, such as sequentially or concurrently.

Starting and Stopping (Control Nodes)

- Activity diagram shows a process flow, that flow must **start and stop** somewhere.
- The **starting point (the initial node)** for an activity flow is shown as a **solid dot.**
- The **stopping point (activity final node)** is shown as a **bull's-eye**.

Initial State

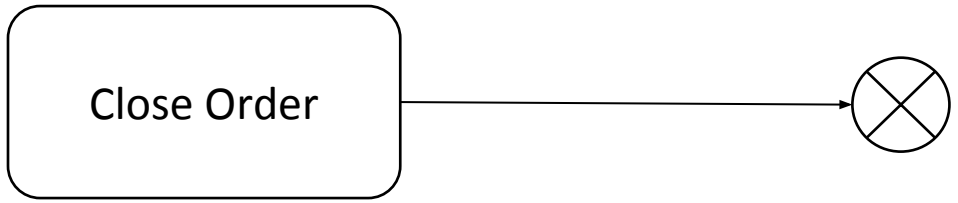
Final State

Starting and Stopping (Control Nodes)

- When an activity diagram is activated, control starts at the solid circle and proceeds via the outgoing arrow toward the first activities.
- A bull's-eye symbol only has incoming arrows. When control reaches a bull's-eye, the overall activity is complete and execution of the activity diagram ends.

Starting and Stopping (Control Nodes)

- Another type of final node is the **flow final node**, which is denoted by **a circle with a nested "X" symbol**.
- The flow final node, used to stop a single flow without stopping the entire activity.

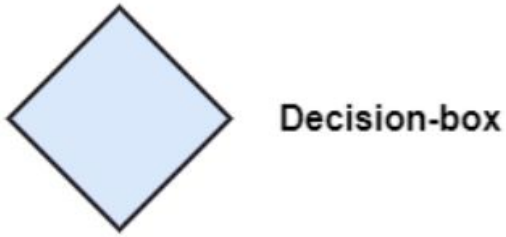


Difference between Activity Final Node and Flow Final Node

- The difference between the two node types is that the flow final node denotes the **end of a single control flow**; the activity final node denotes the **end of all control flows** within the activity.

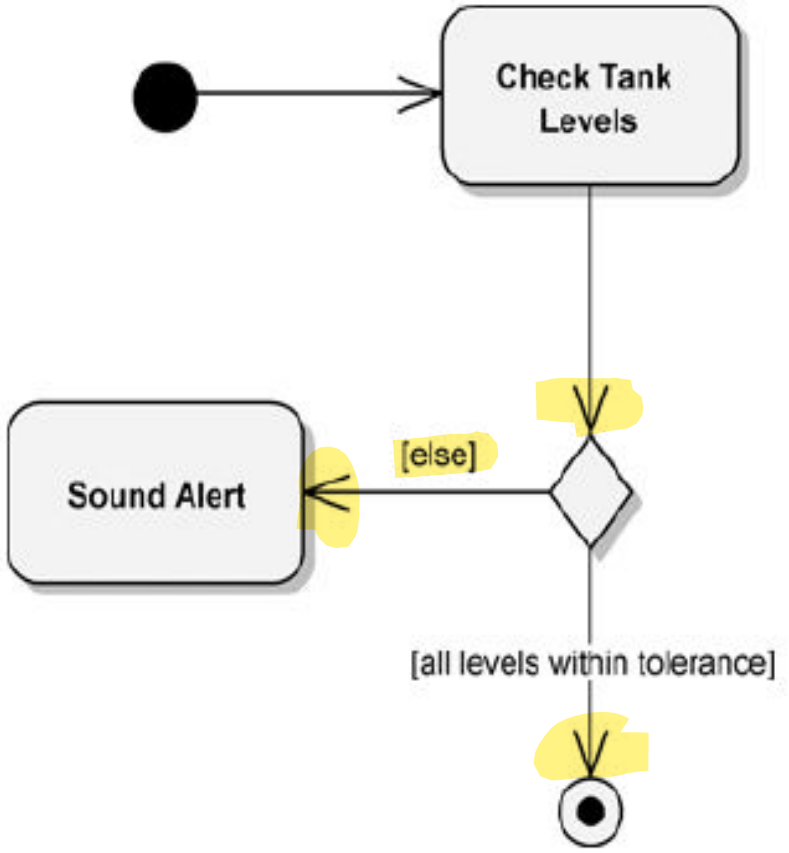
Decision and Merge Nodes (Control Nodes)

- Decision and merge nodes **control the flow** in an activity diagram.
- Each node is represented by a **diamond shape** with incoming and outgoing arrows.



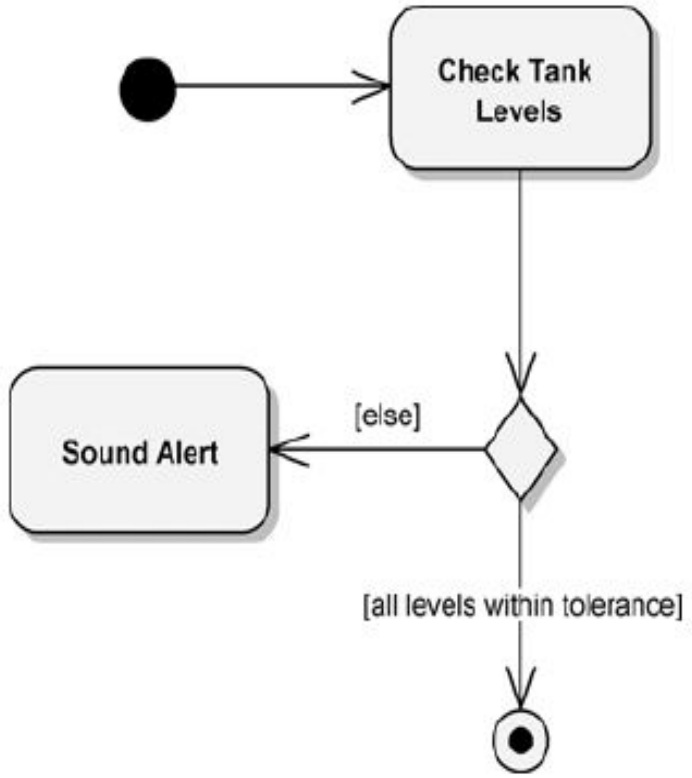
Decision Nodes (Control Nodes)

- A decision node has **one incoming flow and multiple outgoing flows**.
- Its purpose is to direct the one incoming flow into **one (and only one)** of the node's outgoing flows.
- The outgoing flows usually have **guard conditions** that determine which outgoing path is selected.



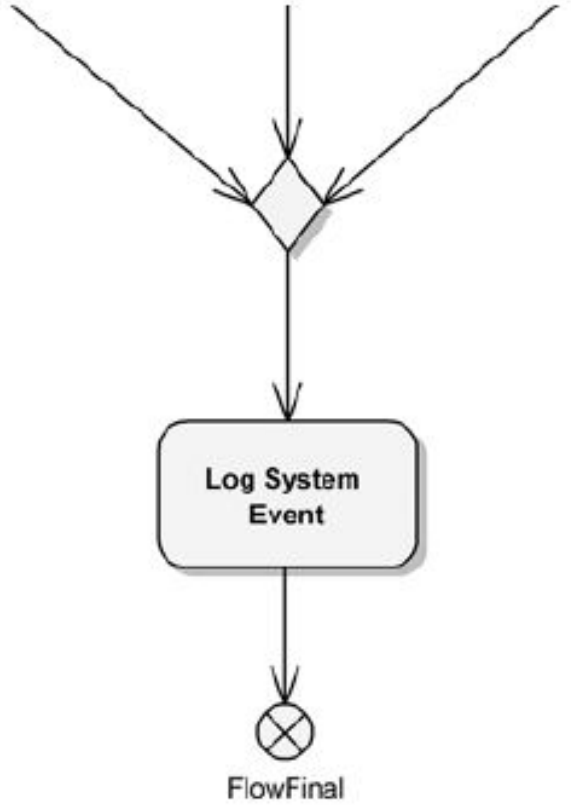
Decision Nodes (Control Nodes)

- If there is more than one successor to an activity, each **arrow may be labeled with a condition in square brackets,** for example, *[failure]*.
- All subsequent conditions are tested when an activity completes. If **one condition is satisfied**, its arrow indicates the next activity to perform.
- If **no condition is satisfied**, the diagram is badly formed and the system will hang unless it is interrupted at some higher level.
- To avoid this danger, you can use the ***else* condition**; it is satisfied in case no other condition is satisfied.
- If **multiple conditions are satisfied**, only one successor activity executes, but there is no guarantee which one it will be.
- Sometimes this kind of nondeterminism is desirable, but often it indicates an error, so the modeler should determine whether any overlap of conditions can occur and whether it is correct.



Merge Nodes (Control Nodes)

- Merge nodes **take multiple input flows and direct any and all of them to one outgoing flow**.
- There is **no waiting or synchronization** at a merge node.
- Whenever any of the three incoming flows reach the merge point (shown as a diamond), each will be routed through it to the Log System Event action. Thus, multiple events will be logged.

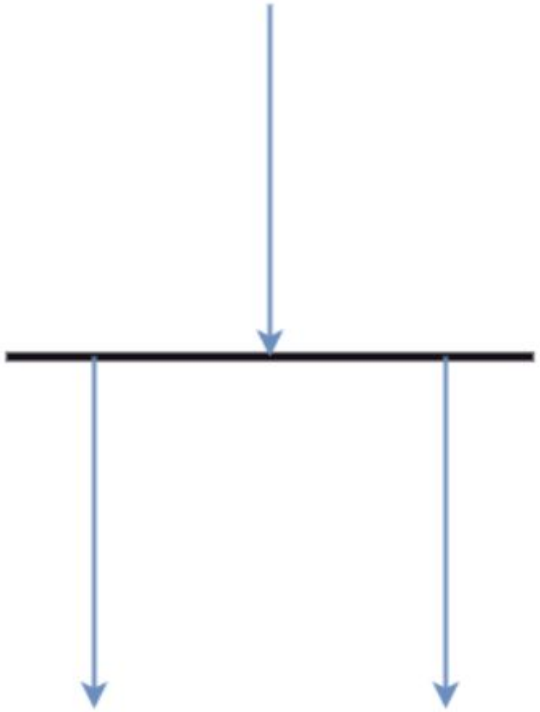


Forks, Joins and Concurrency (Control Nodes)

- Fork and join nodes are analogous to decision and merge nodes.
- The critical difference is **concurrency**.
- The pace of activity can also change over time. For example, one activity may be followed by another activity **(sequential control),** then split into several concurrent activities **(a fork of control)**, and finally be combined into a single activity **(a merge of control).**

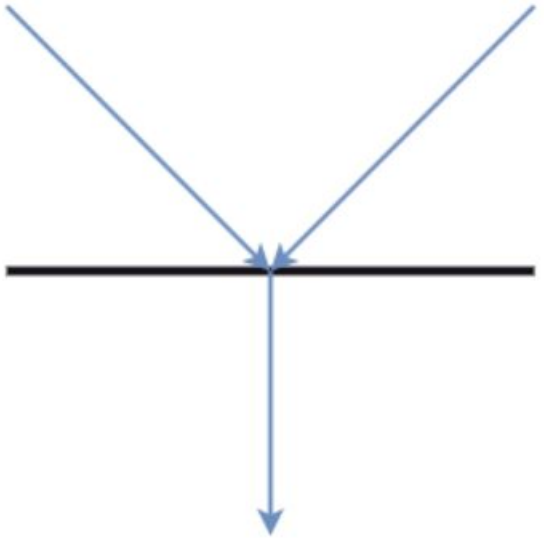
Forks (Control Nodes)

- Forks have **one flow in and multiple flows out**, as do decision nodes.
- The difference is, where a decision node selects a single outbound flow, a **single flow into a fork results in multiple outbound flows**.

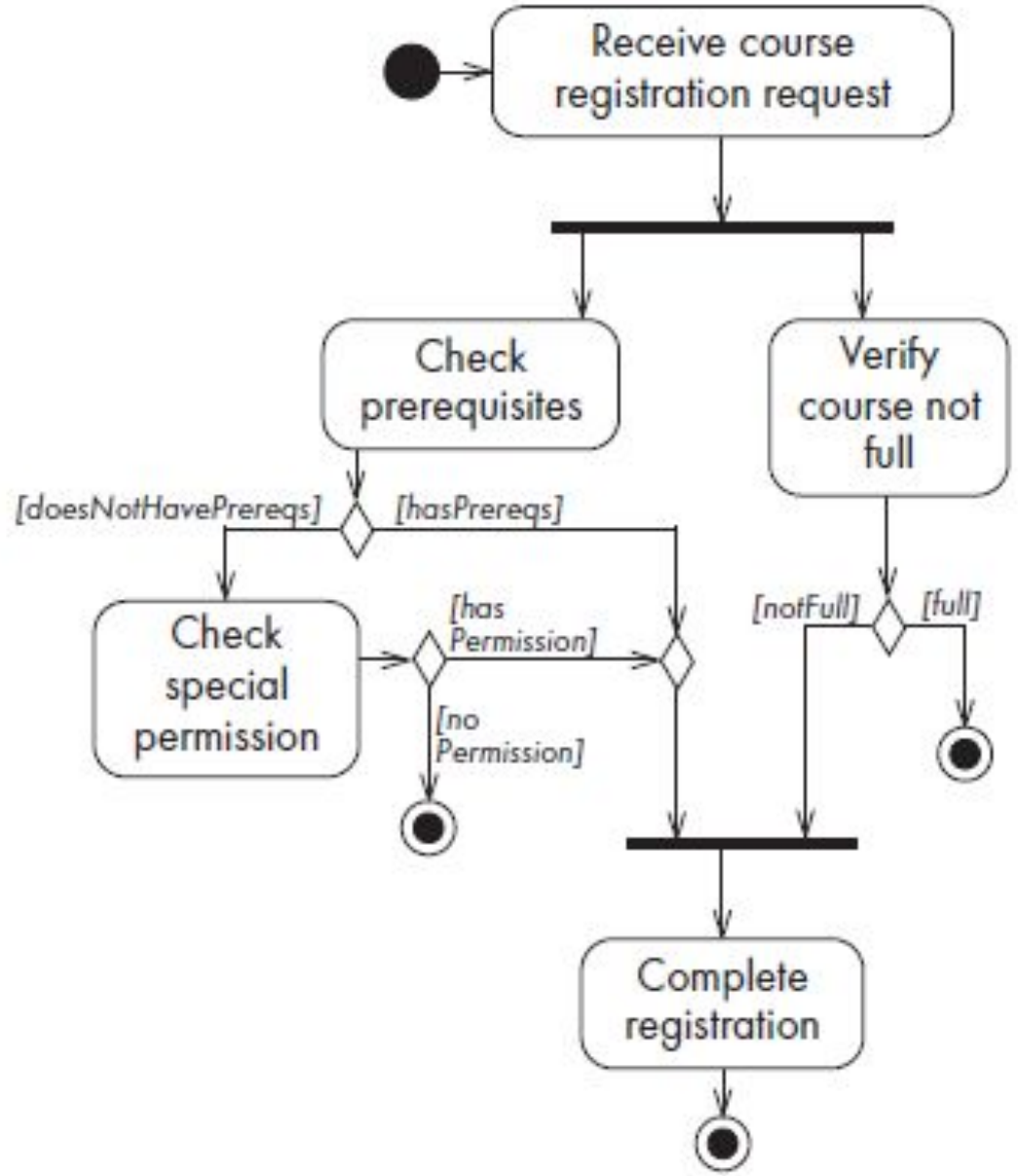


Joins (Control Nodes)

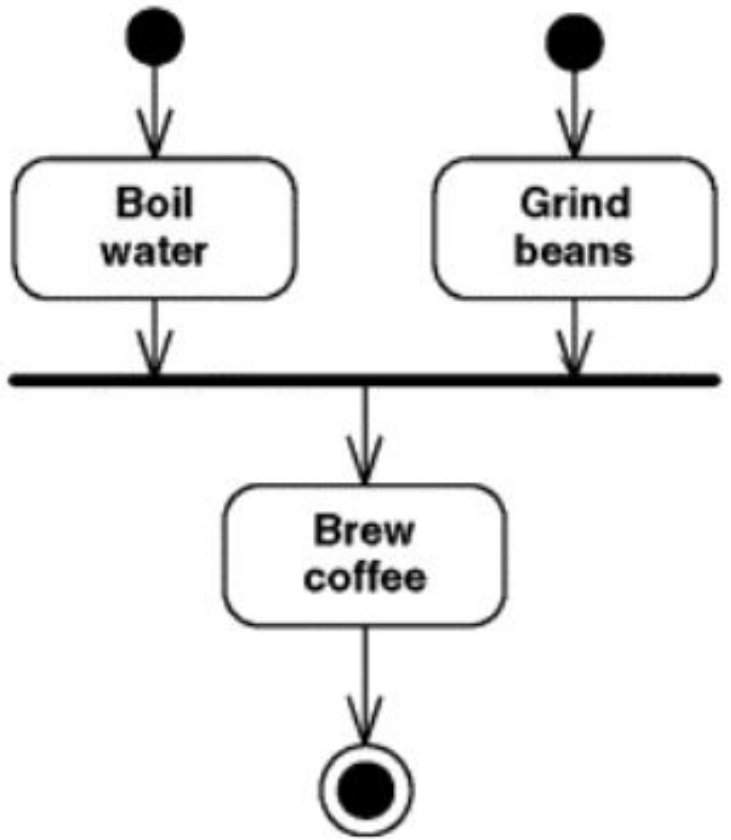
- A join has **multiple incoming flows and a single outbound flow**, like merge nodes.
- With a join, all the **incoming flows must be completed before the outbound flow commences**.



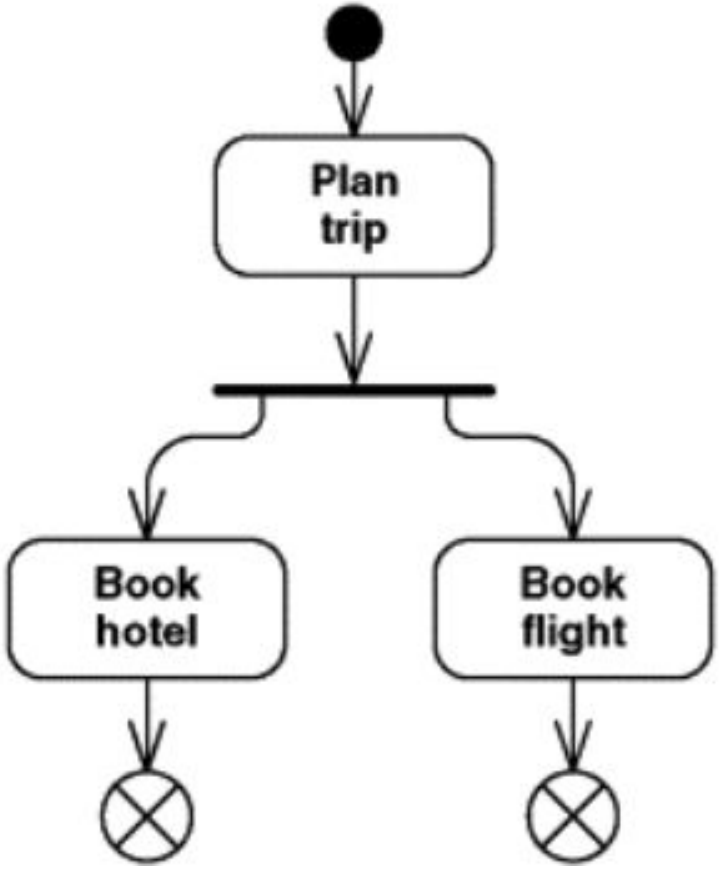
Activity Diagram for Course Registration



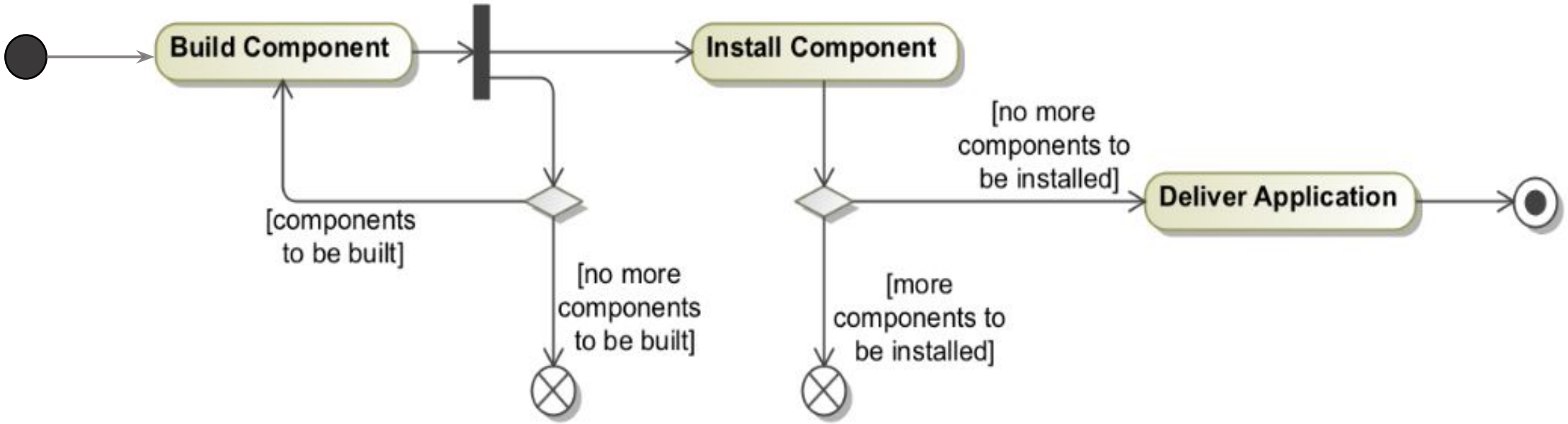
Activity Final Node



Flow Final Node

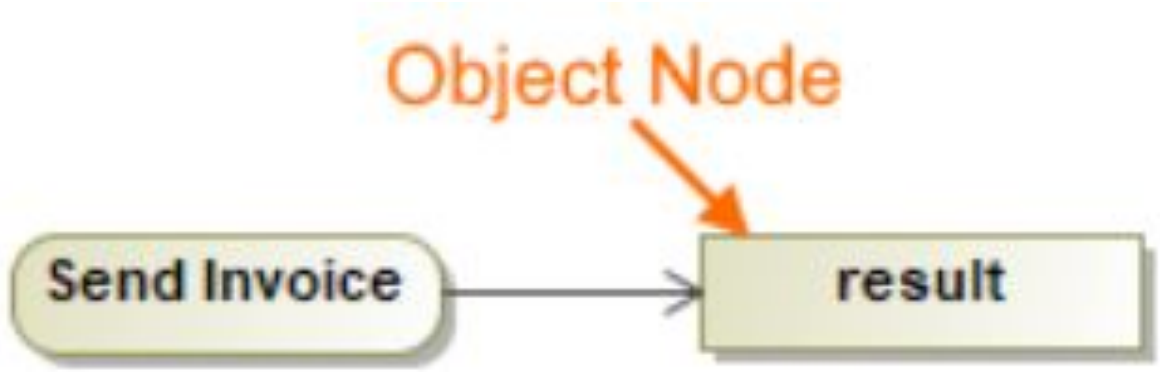


Activity Diagram (Development of Application)



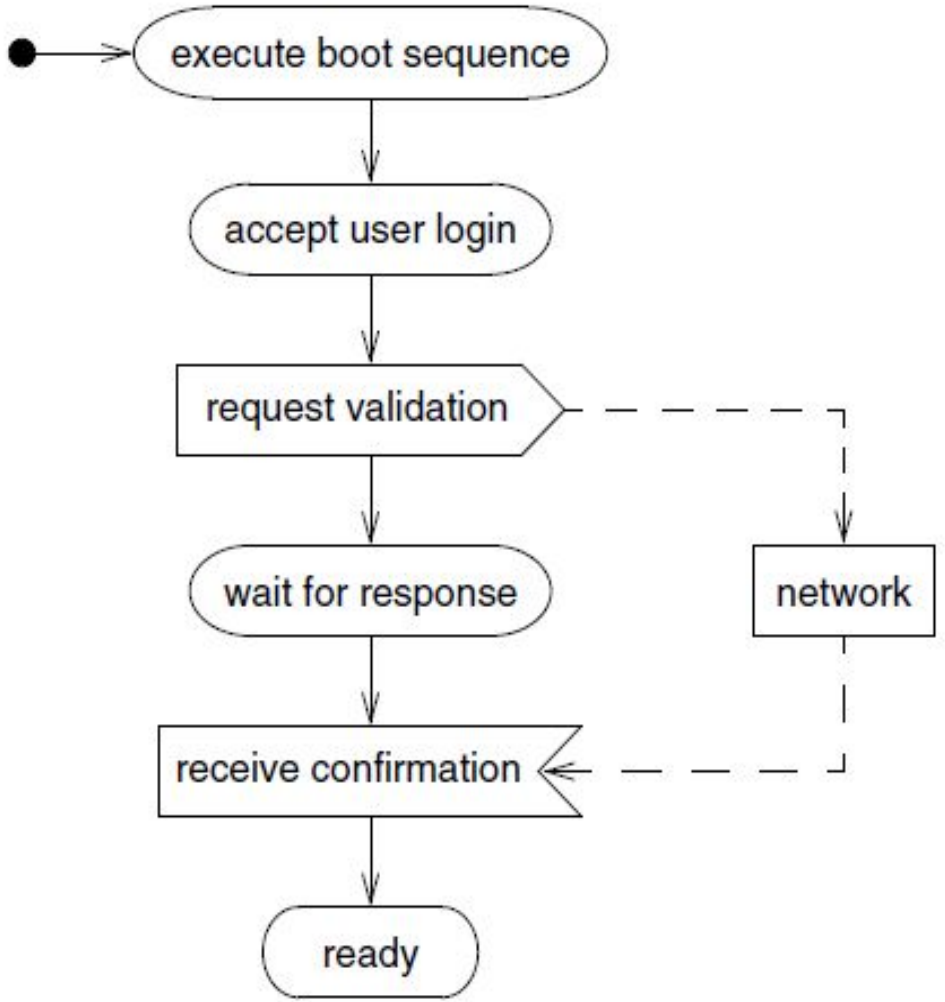
Object Nodes

- Object nodes are used to represent **objects (data or information)** that are involved in the activity flow. They capture the state of data at different points in the process.
- Notation: A rectangle.



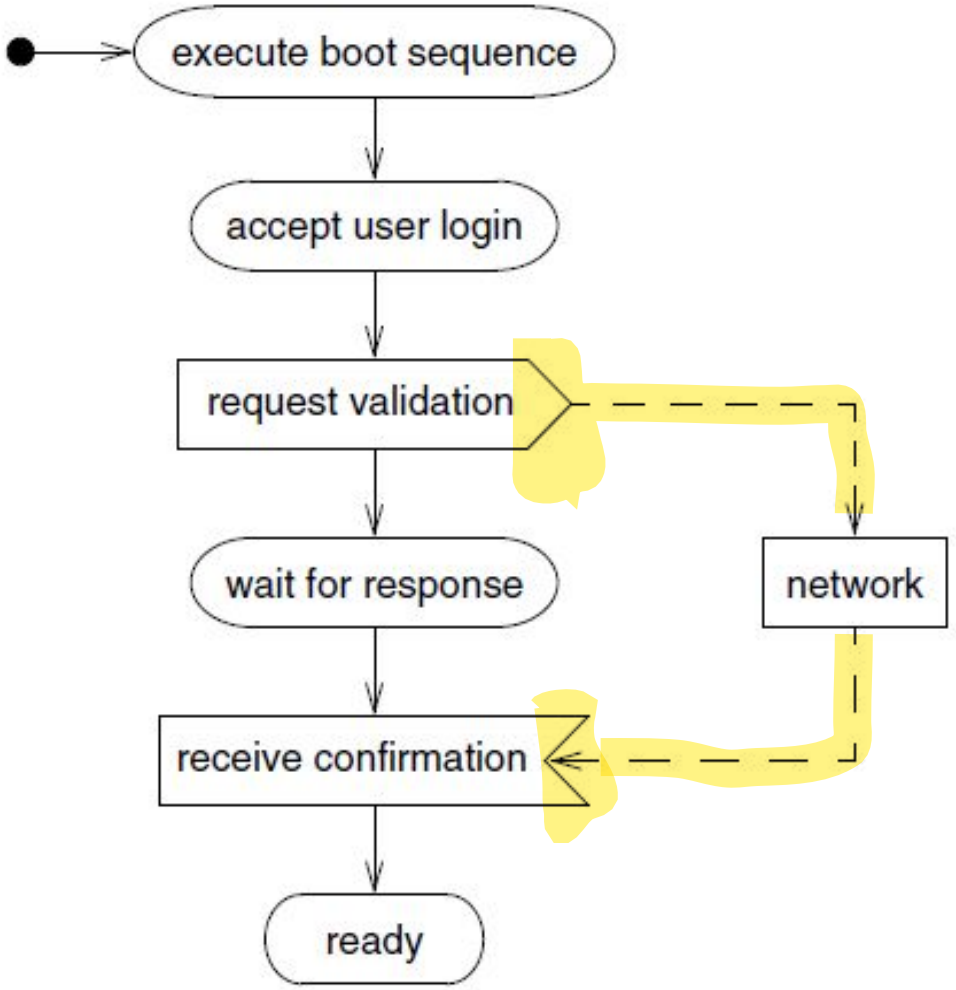
Sending and Receiving Signals

- Consider a workstation that is turned on.
- It goes through a boot sequence and then requests that the user log in.
- After entry of a name and password, the workstation queries the network to validate the user.
- Upon validation, the workstation then finishes its startup process.



Sending and Receiving Signals

- The UML shows the sending of a signal as a **convex pentagon**.
- When the preceding activity completes, the signal is sent, then the next activity is started.
- The UML shows the receiving of a signal as a **concave pentagon**.
- When the preceding activity completes, the receipt construct waits until the signal is received, then the next activity starts.



Guidelines for Activity Diagram

- **Don’t misuse activity diagrams**. Activity diagrams are intended to elaborate use case and sequence models so that a developer can study algorithms and workflow. Activity diagrams supplement the object-oriented focus of UML models and should not be used as an excuse to develop software via flowcharts.
- **Level diagrams**. Activities on a diagram should be at a consistent level of detail. Place additional detail for an activity in a separate diagram.
- **Be careful with branches and conditions**. If there are conditions, at least one must be satisfied when an activity completes—consider using an *else* condition. In undeterministic models, it is possible for multiple conditions to be satisfied—otherwise this is an error condition.
- **Be careful with concurrent activities**. Concurrency means that the activities can complete in any order and still yield an acceptable result. Before a merge can happen, all inputs must first complete.
- **Consider executable activity diagrams**. Executable activity diagrams can help developers understand their systems better. Sometimes they can even be helpful for end users who want to follow the progression of a process.

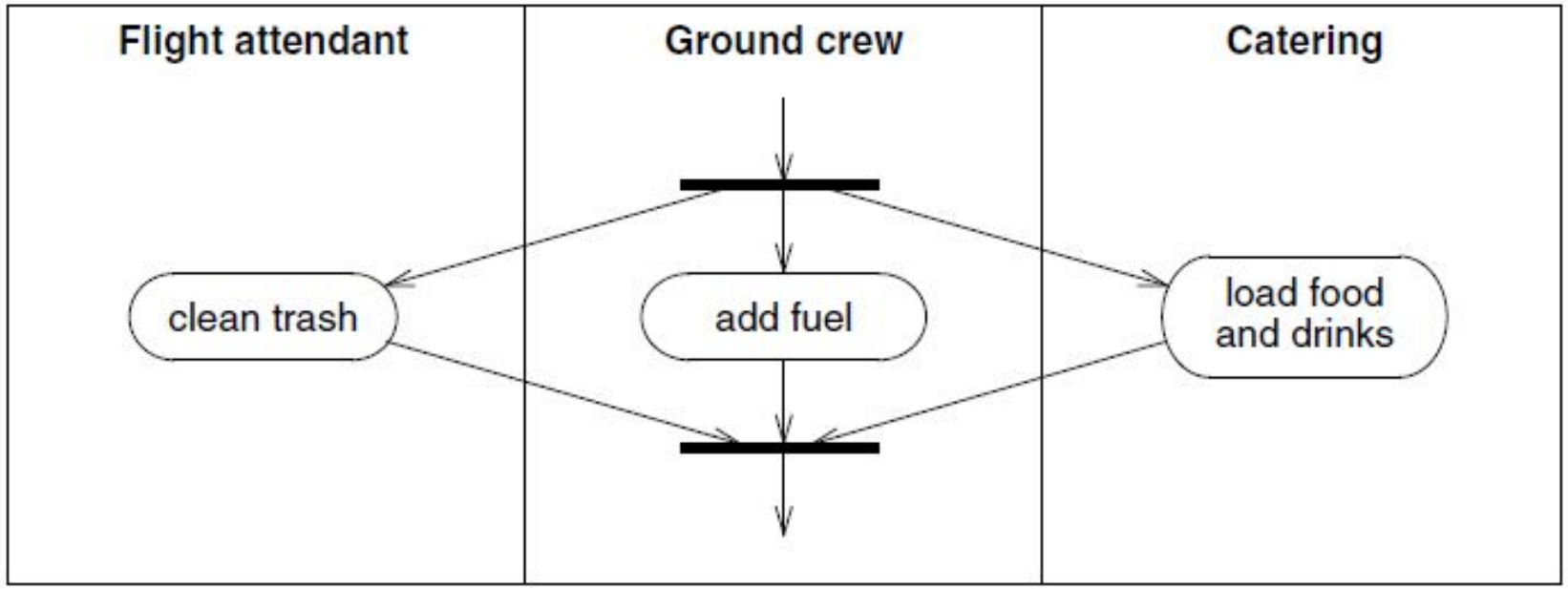
Swimlanes

- In a business model, it is often useful to know **which human organization is responsible for an activity**.
- Sales, finance, marketing, and purchasing are examples of organizations.
- When the design of the system is complete, the activity will be assigned to a person, but at a high level it is sufficient to partition the activities among organizations.
- Partitions may be business units, divisions, or organizations.
- For systems, the partitions may be other systems or subsystems.
- In application modeling, the partitions may be objects in the application.

Swimlanes

- You can show such a partitioning with an activity diagram by dividing it into **columns and lines**.
- Each column is called a ***swimlane*** by analogy to a swimming pool.
- Placing an activity within a particular swimlane indicates that it is performed by a person or persons within the organization.
- Lines across swimlane boundaries indicate **interactions among different organizations**, which must usually be treated with more care than interactions within an organization.

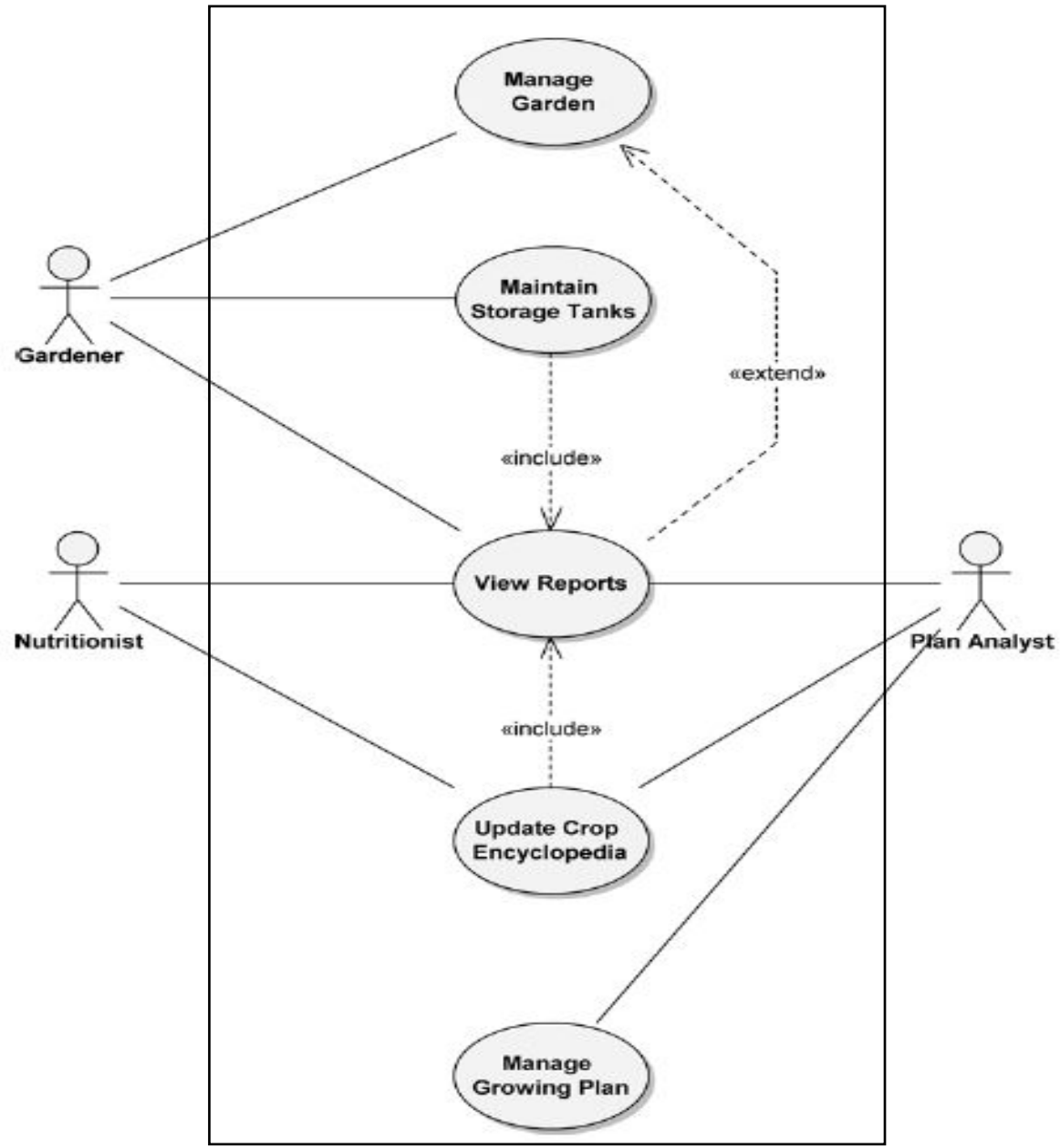
Activity Diagram with Swimlanes (Servicing an Airplane)



Hydroponics Gardening System (Example)

- On a hydroponics farm, plants are grown in a nutrient solution, without sand, gravel, or other soils.
- Maintaining the proper greenhouse environment is a delicate job and depends on the kind of plant being grown and its age.
- One must control diverse factors such as temperature, humidity, light, pH, and nutrient concentrations.
- On a large farm, it is not unusual to have an automated system that constantly monitors and adjusts these elements.
- Simply stated, the purpose of an automated gardener is to efficiently carry out, with minimal human intervention, growing plans for the healthy production of multiple crops.

Hydroponics Gardening System (Use Case Diagram)



Hydroponics Gardening System (Use Case Description)

Use case name: `Maintain Storage Tanks`

Use case purpose: This use case provides the ability to maintain the fill levels of the contents of the storage tanks. This use case allows the actor to maintain specific sets of water and nutrient tanks.

Optimistic flow:

A. Gardener examines the levels of the storage tanks' contents.

B. Gardener determines that tanks need to be refilled.

C. Normal hydroponics system operation of storage tanks is suspended by the Gardener.

D. Gardener selects tanks and sets fill levels.

For each selected tank, steps E through G are performed.

E. If tank is heated, the system disables heaters.

1. Heaters reach safe temperature.

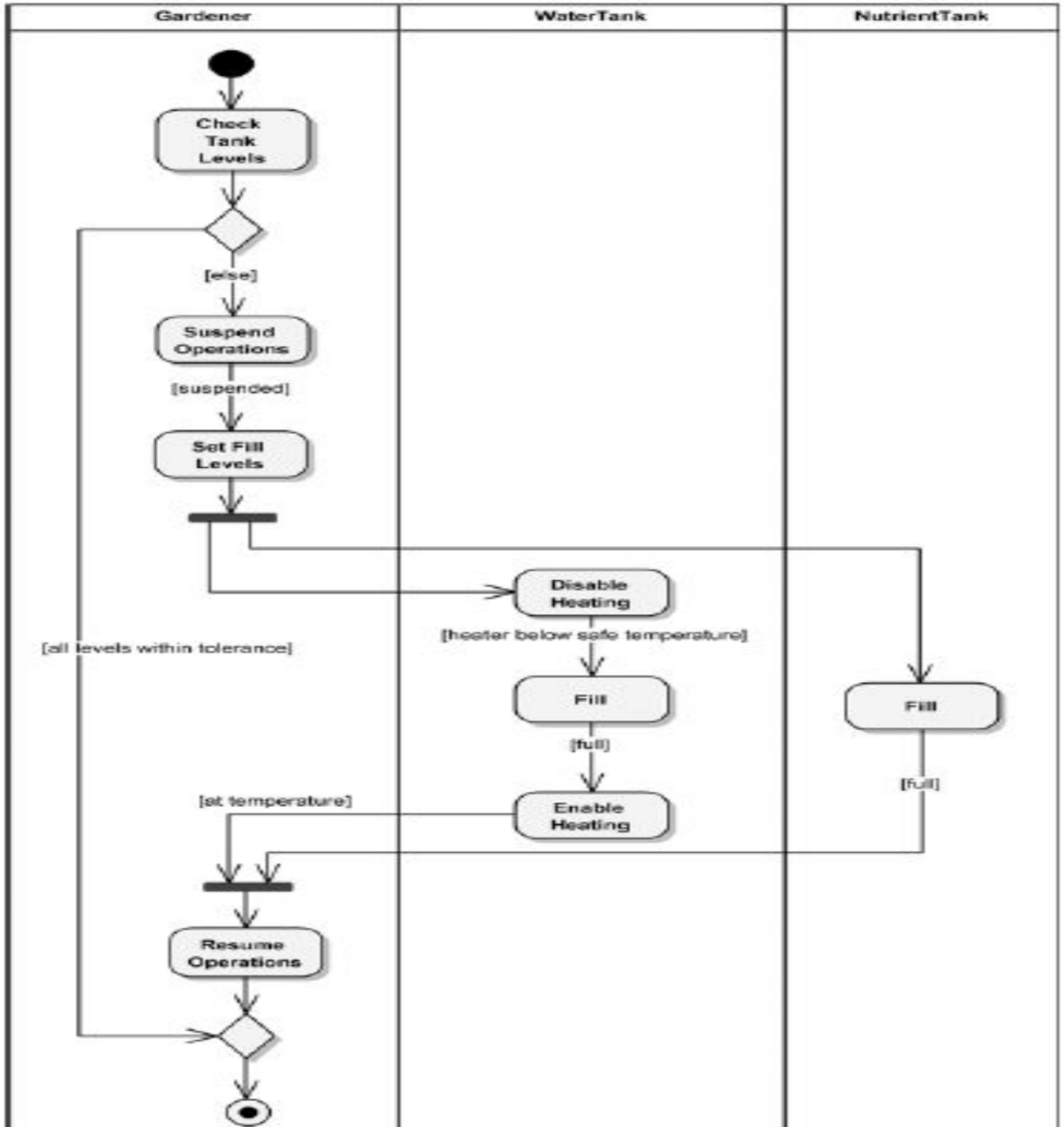
F. The system fills tank.

G. When tank is filled, if tank is heated, the system enables heaters.

1. Tank contents reach operating temperature.

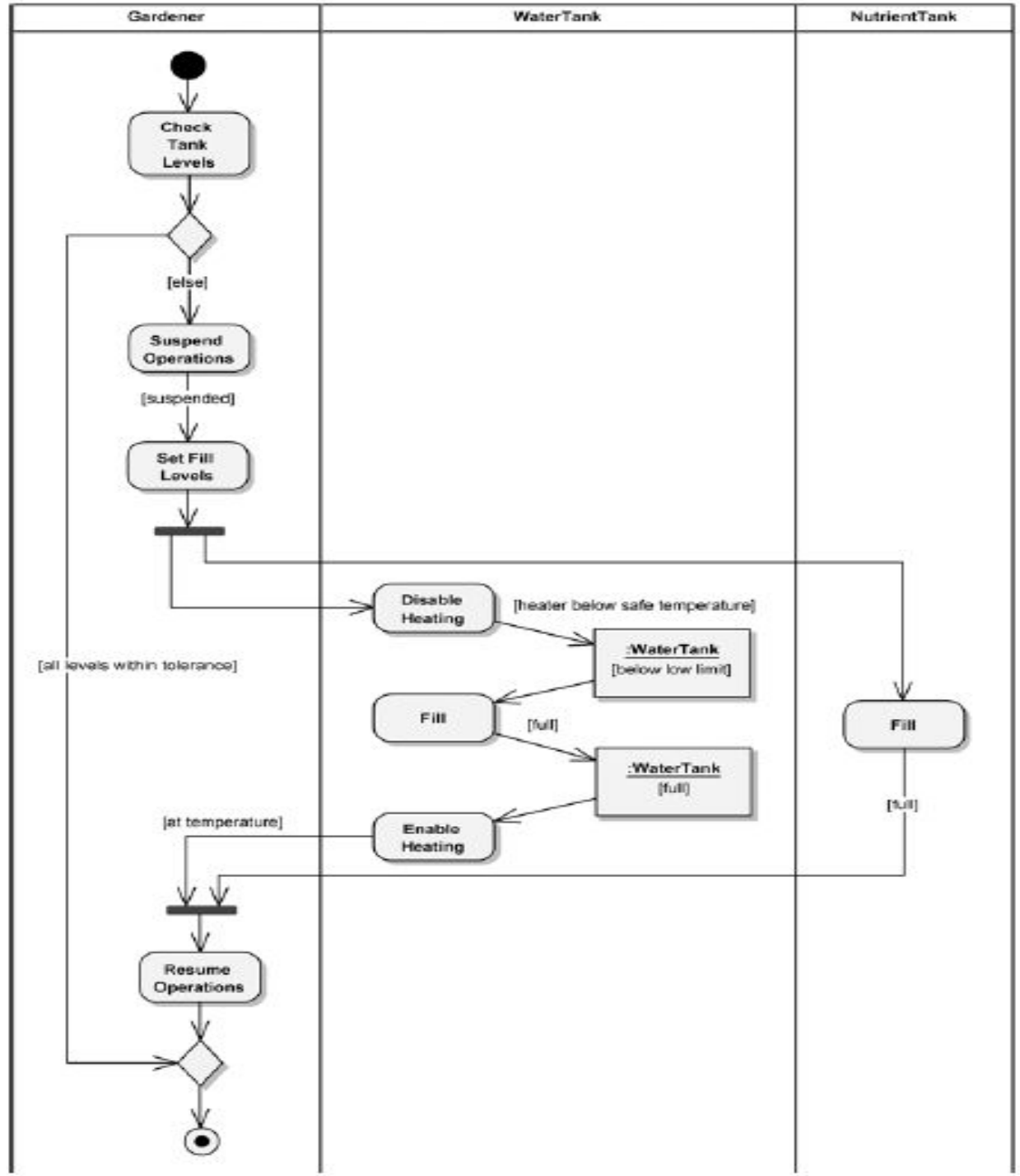
H. Gardener resumes normal hydroponics system operation.

`Maintain Storage Tanks` use case of Hydroponics Gardening System



Object Flows

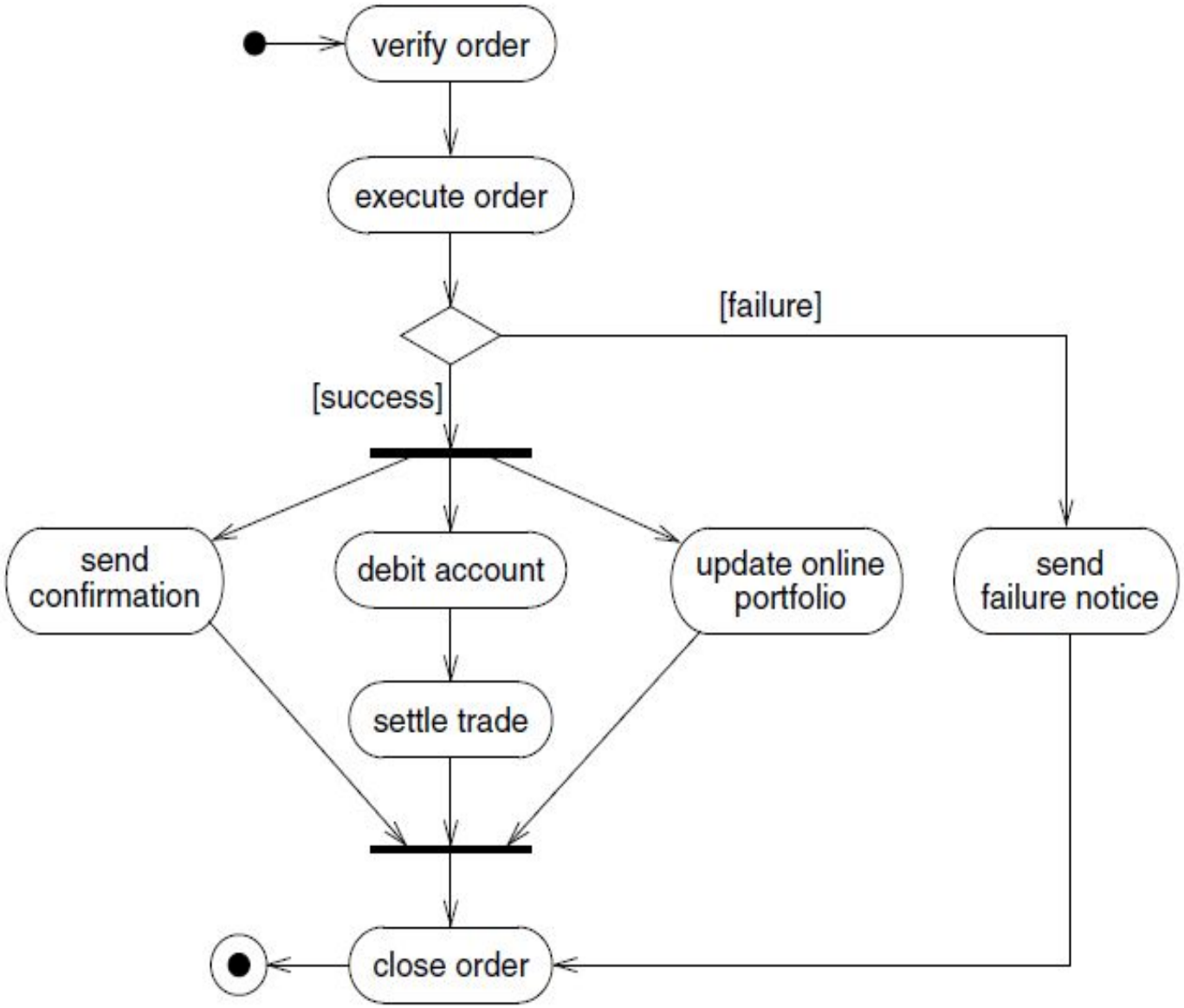
- In some situations, it may be useful to **see the objects that are manipulated** during the execution of an activity.
- You can show the objects in an activity diagram by adding an **object flow**. (Not recommend for all your activity diagrams because adding all the objects would likely make the diagrams too complex)


Gardener
WaterTank
NutrientTank
Check Tank Levels
[else]
Suspend Operations
[suspended]
Set Fill Levels
Disable Heating
[heater below safe temperature]
:WaterTank
[below low limit]
Fill
[full]
:WaterTank
[full]
Enable Heating
[at temperature]
[all levels within tolerance]
Fill
[full]
Resume Operations

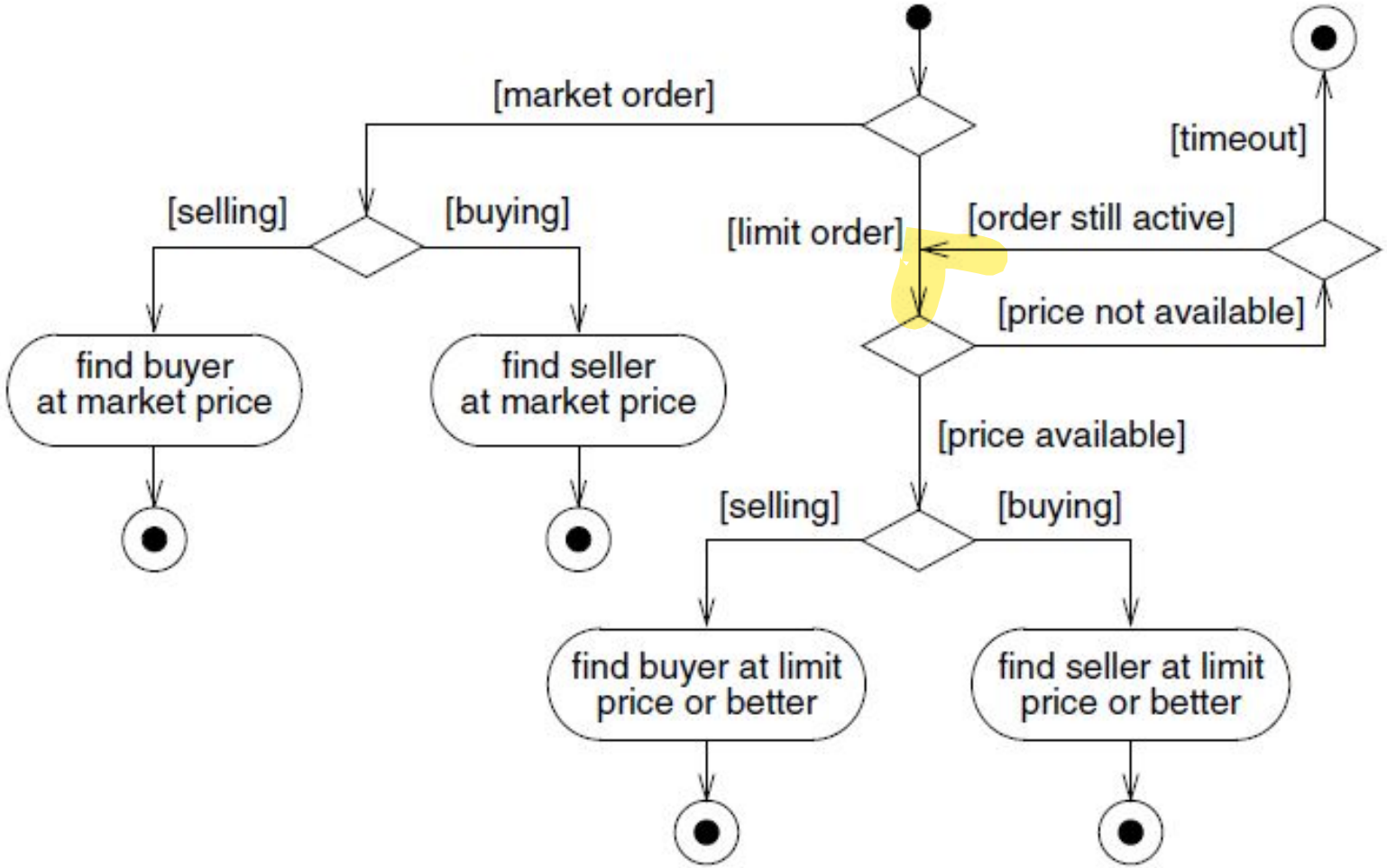
Processing of a stock trade order that has been received by an online stockbroker

- The online stockbroker first verifies the order against the customer's account, then executes it with the stock exchange.
- If the order executes successfully, the system does three things concurrently: mails trade confirmation to the customer, updates the online portfolio to reflect the results of the trade, and settles the trade with the other party by debiting the account.
- When all three things have been completed, the system closes the order.
- If the order execution fails, then the system sends a failure notice to the customer and closes the order.

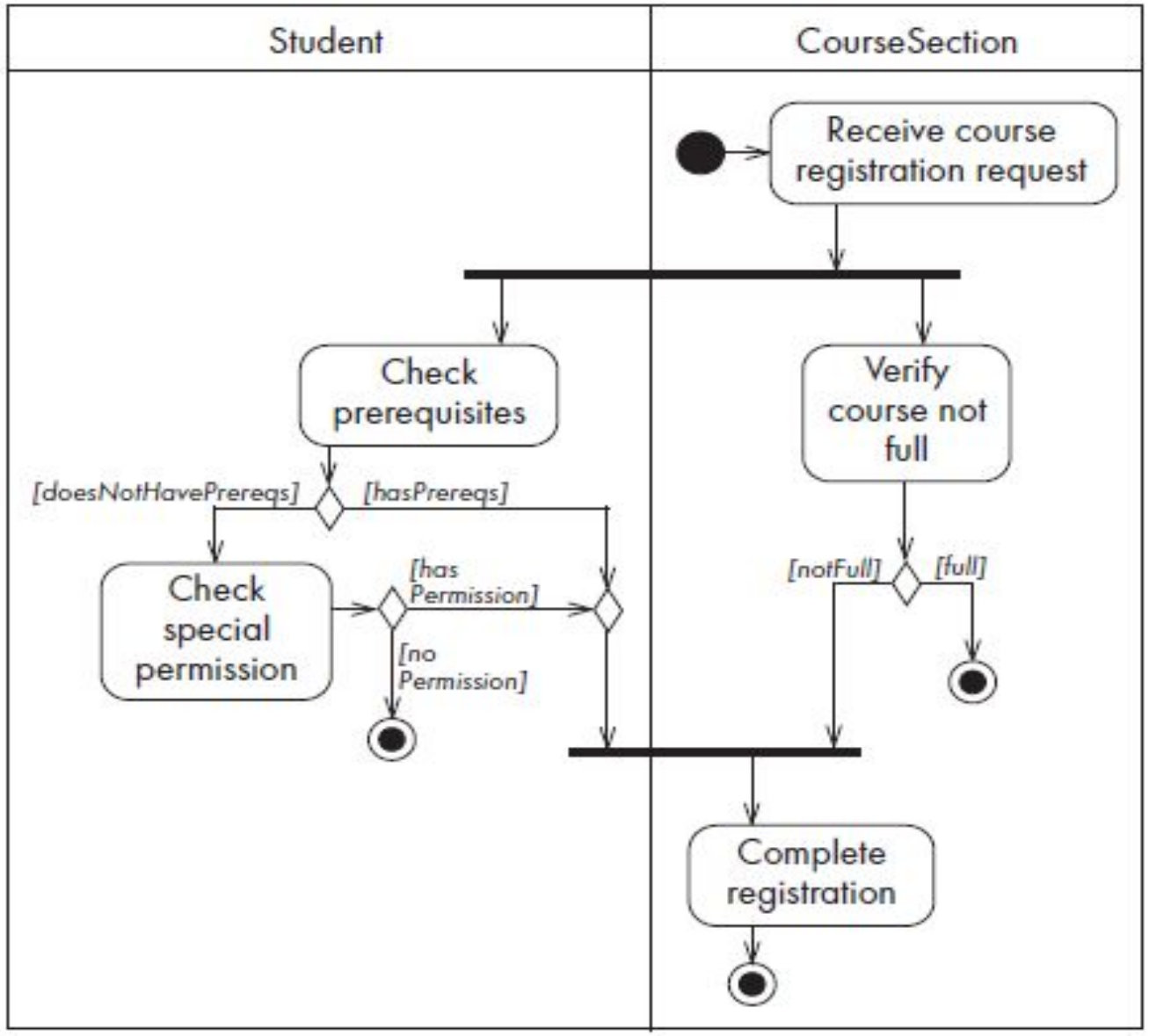
Activity Diagram



Activity Diagram for Execute Order

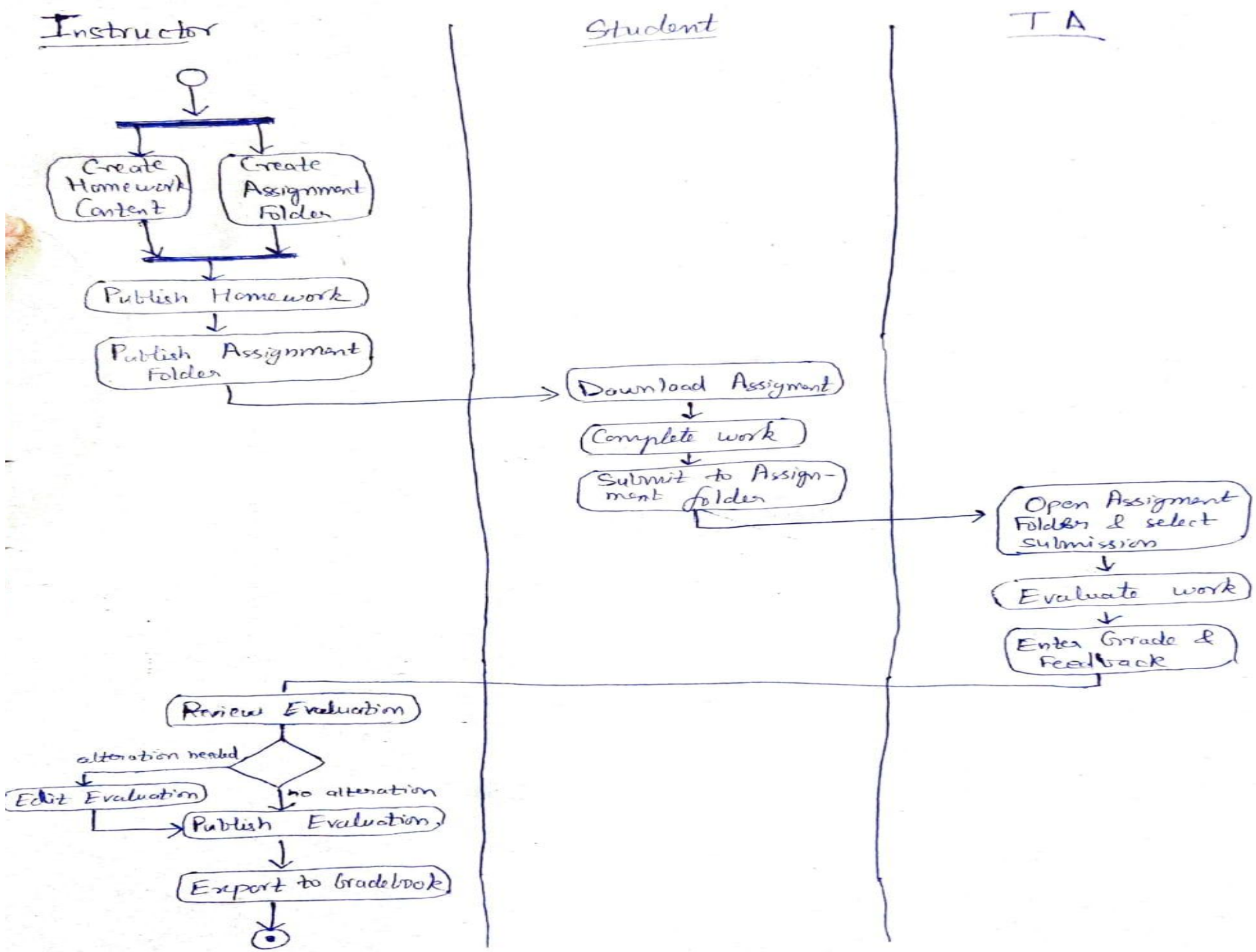


Activity Diagram with Swimlanes
(Course Registration)



Case Study Homework Processing (Swimlane)

- Consider the following process for assigning, submitting, and grading Homework. The process begins when the Instructor creates the Homework instruction content on the course site. Then the Instructor creates an Assignment Folder for the Homework. These two tasks, creating the Homework content and creating the Assignment Folder have no particular order in which they must be completed, but they must both be completed concurrently before the process can continue. The process continues when the Instructor publishes the Homework and then the Assignment Folder. The student downloads the assignment and begins work. When the student has completed the work, they submit it to the Assignment Folder. The TA opens the Assignment Folder and selects the student's submission. The TA evaluates the work, then enters the grade and feedback to the submission. The Instructor reviews the evaluation. If the evaluation needs no alterations, then the Instructor publishes the evaluation. Otherwise, the Instructor edits the evaluation and then publishes it. After the evaluation is complete, the Instructor exports the grade to the Gradebook and the process ends.


Instructor
Student
TA
Create Homework Content
Create Assignment Folder
Publish Homework
Publish Assignment Folder
Download Assignment
Complete work
Submit to Assignment folder
Open Assignment Folder & select submission
Evaluate work
Enter Grade & Feedback
Review Evaluation
alteration needed
no alteration
Edit Evaluation
Publish Evaluation
Export to Gradebook

References

- Object-Oriented Analysis and Design with Applications, Grady Booch et al., 3rd Edition, Pearson, 2007.

- Timothy C. Lethbridge, Robert Laganaiere, Object-Oriented Software Engineering (2nd Edition), McGraw Hill, 2005

- Object-Oriented Modeling and Design with UML, Michael R. Blaha and James R. Rumbaugh, 2nd Edition, Pearson, 2005.