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."

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

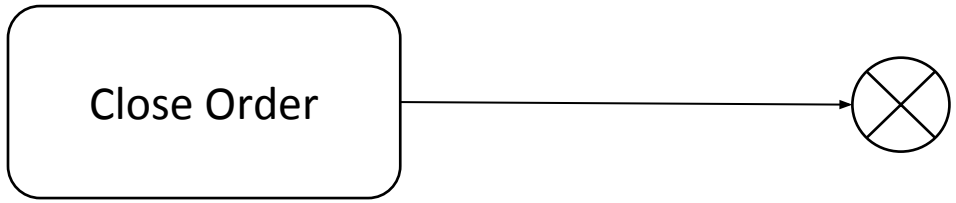
Final State

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.

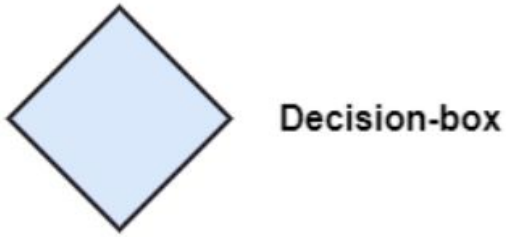


Difference between Activity Final Node and Flow Final Node

- 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.

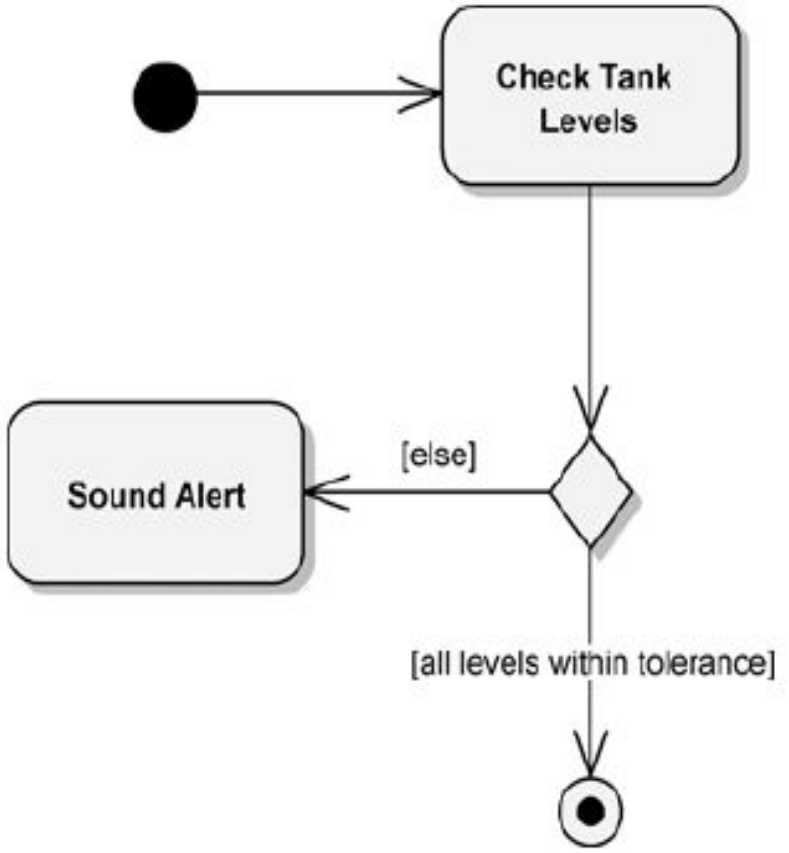
Decision and Merge Nodes (Control Nodes)

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



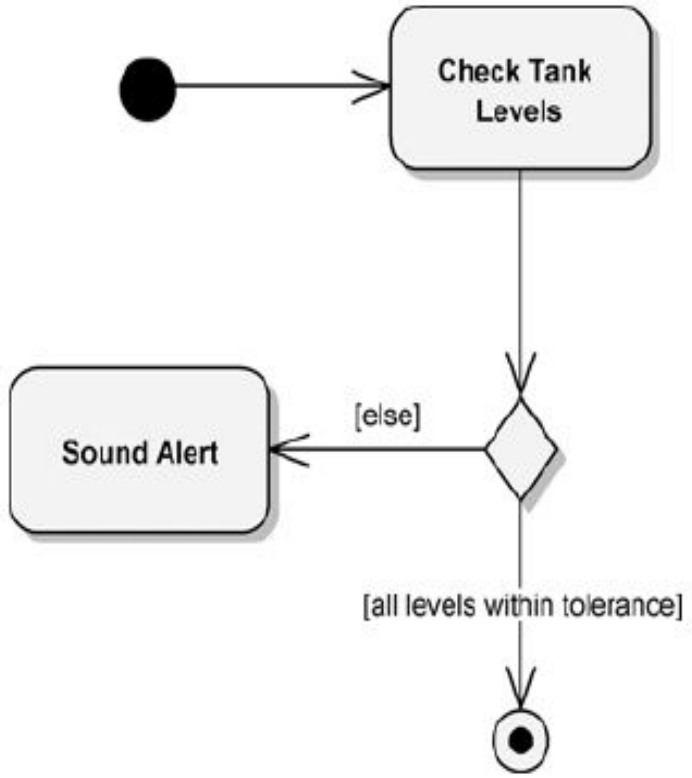
Decision Nodes (Control Nodes)

- 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.



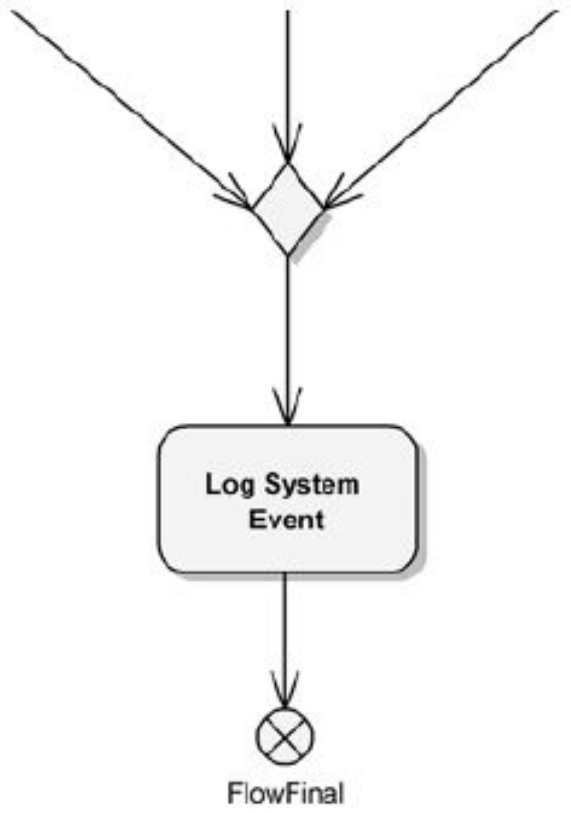
Decision Nodes (Control Nodes)

- 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.

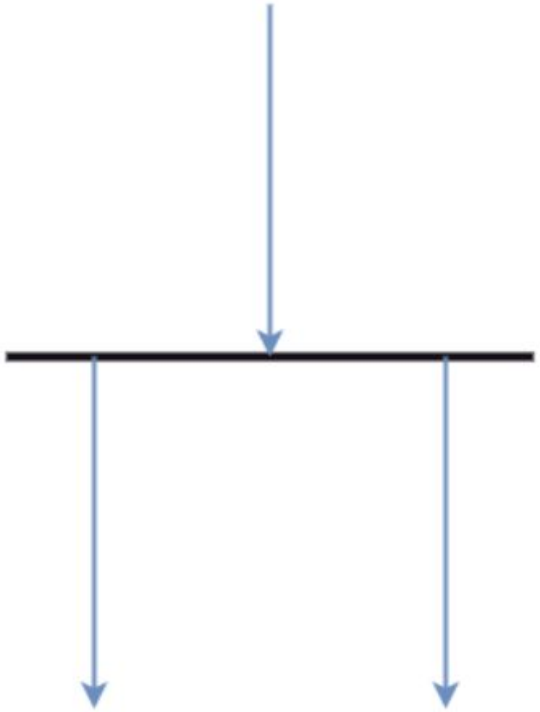


Forks, Joins and Concurrency (Control Nodes)

- 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).**

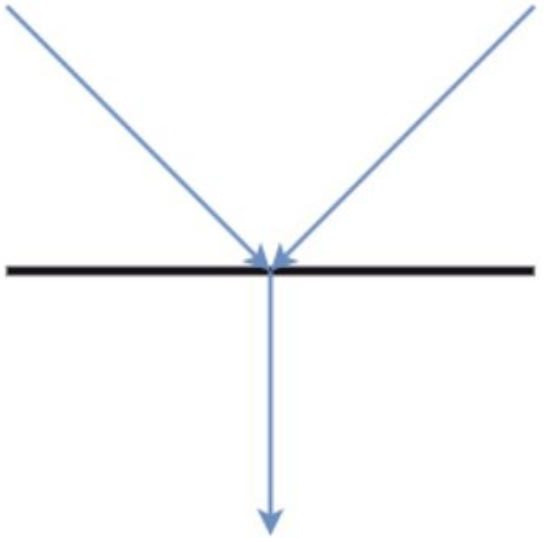
Forks (Control Nodes)

- 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**.

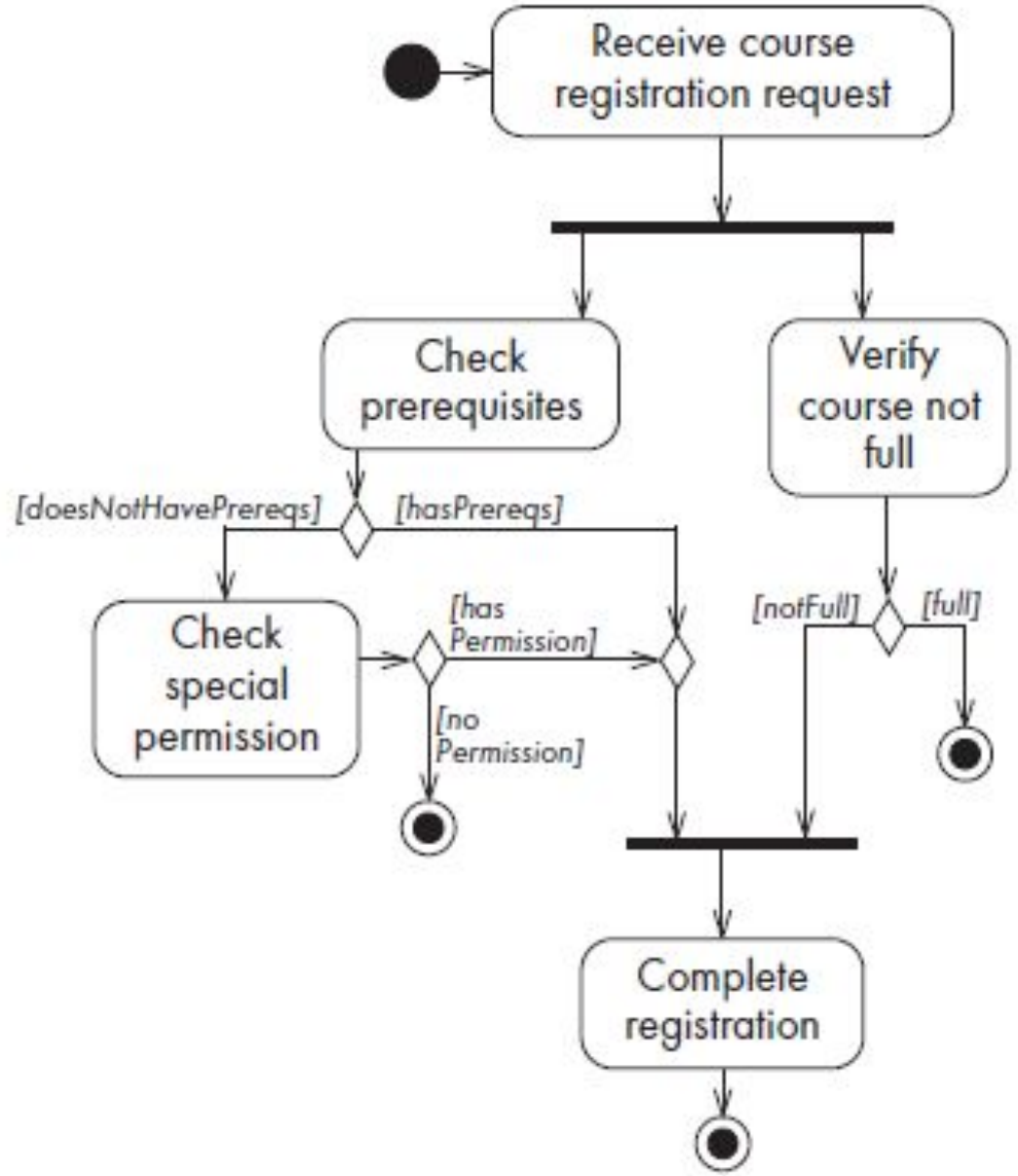


Joins (Control Nodes)

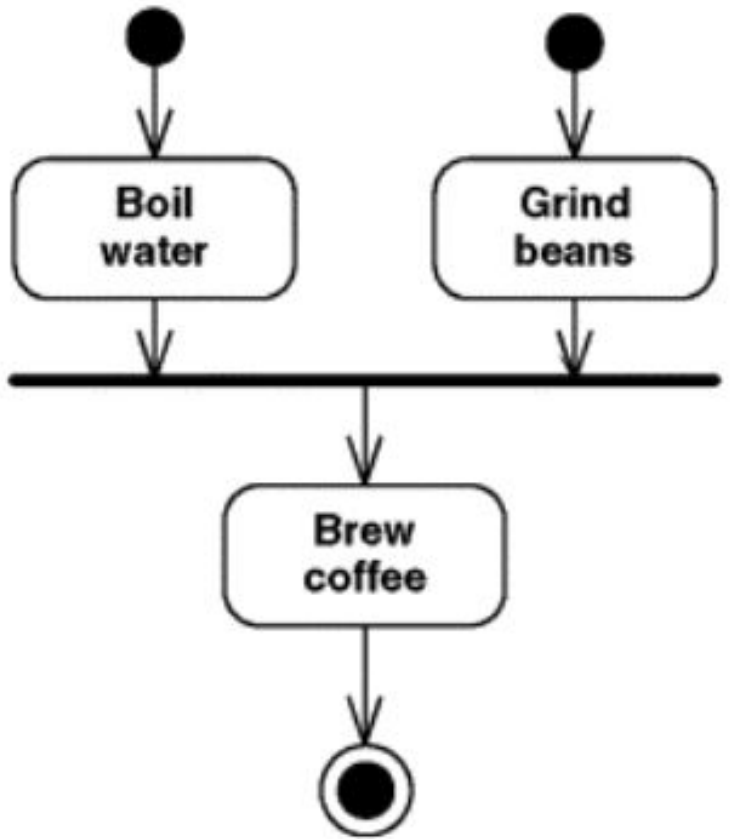
- A join has **multiple incoming flows and a single outbound flow**, like merge nodes.
- With a join, all the **incoming flows must be completed before the outbound flow commences**.



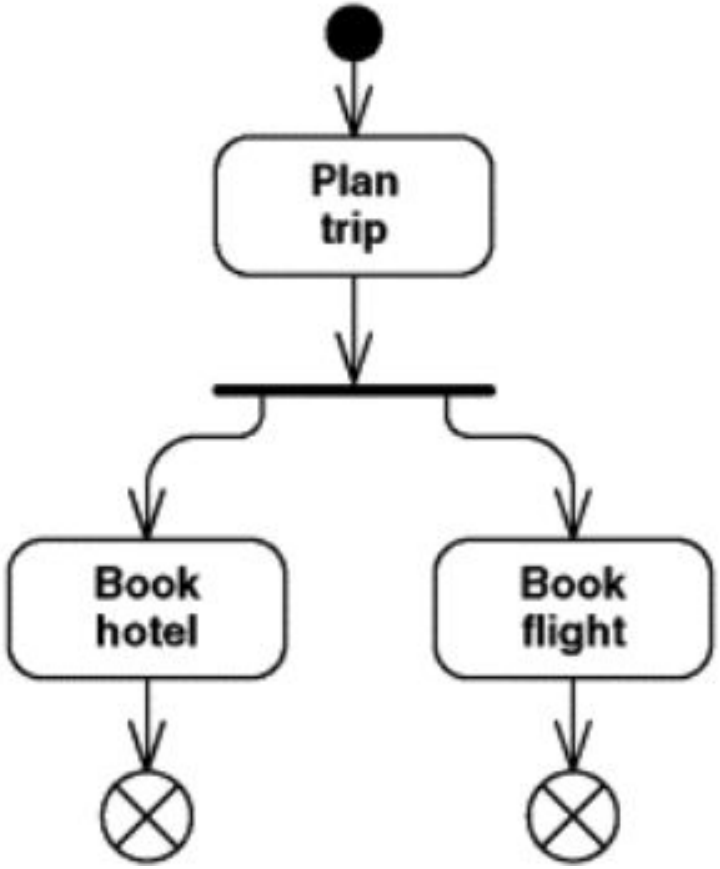
Activity Diagram for Course Registration



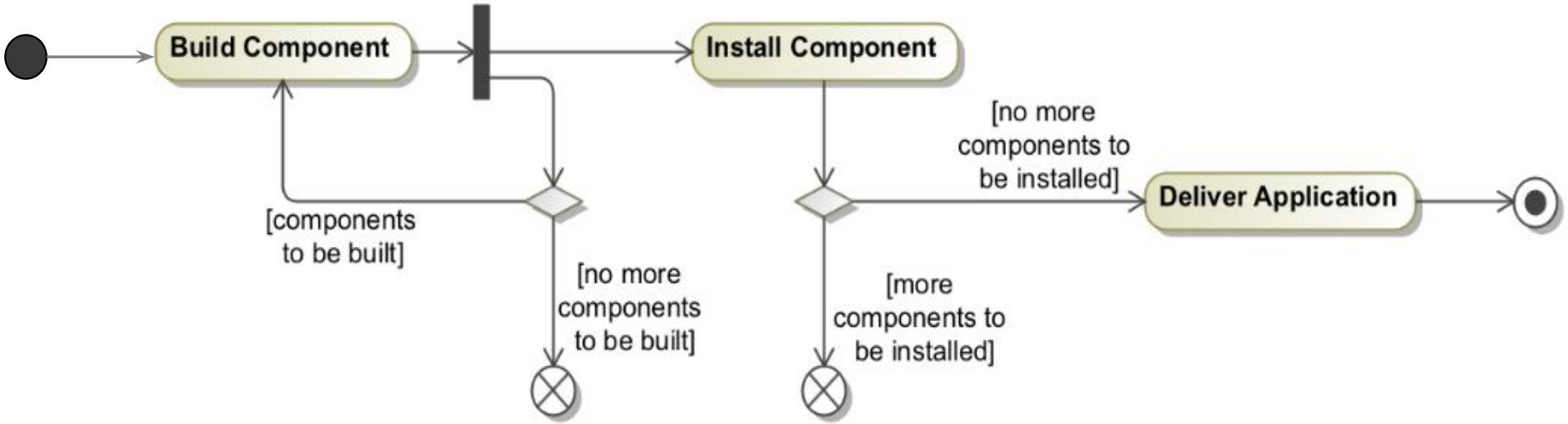
Activity Final Node



Flow Final Node

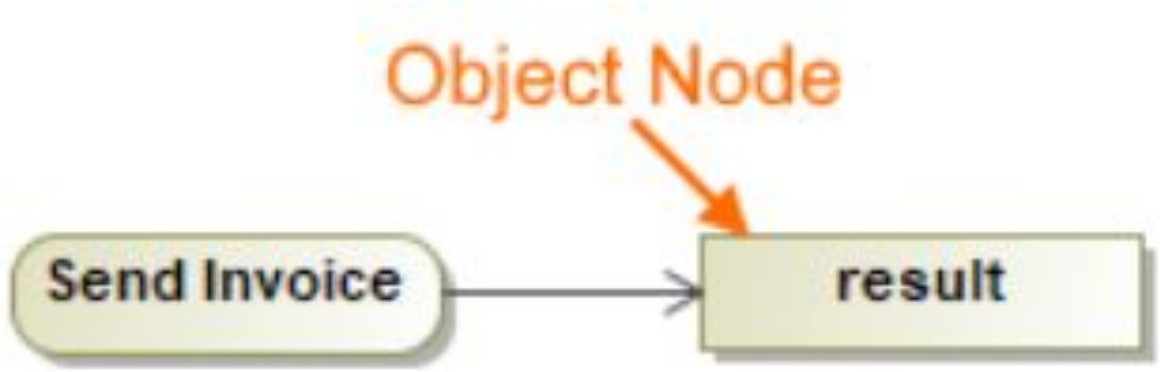


Activity Diagram (Development of Application)



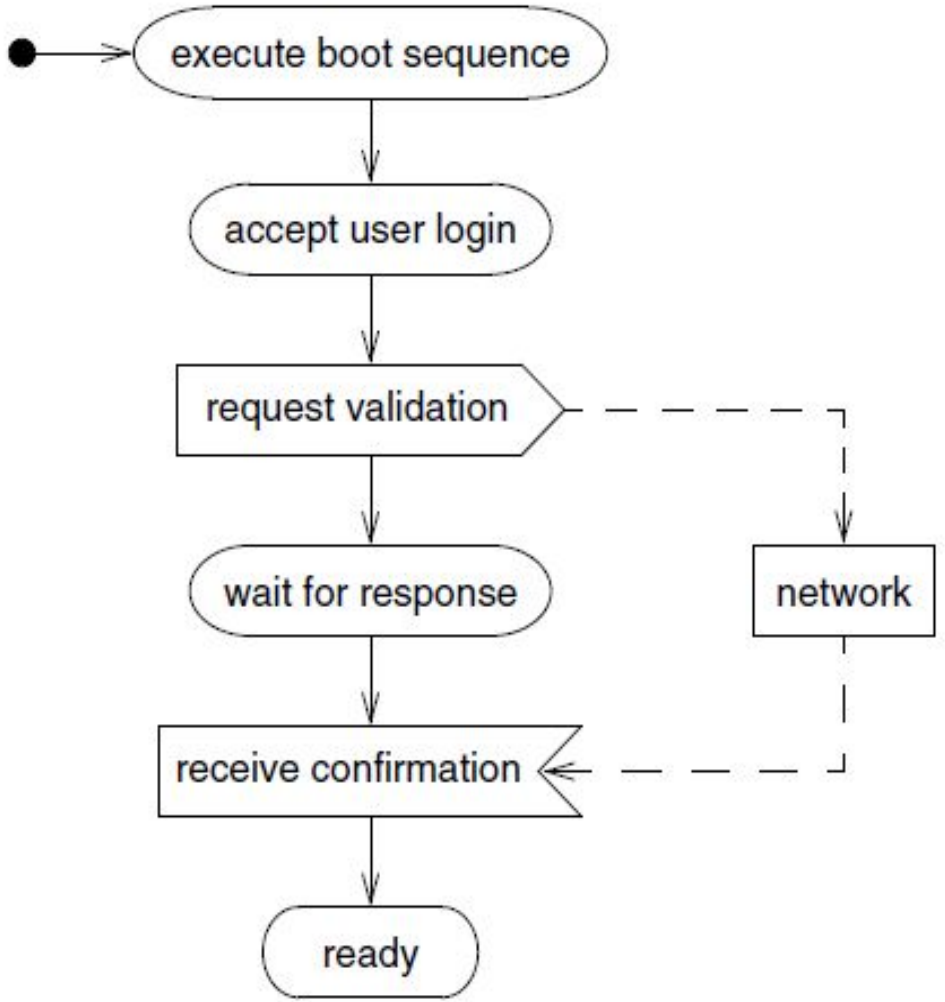
Object Nodes

- 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.



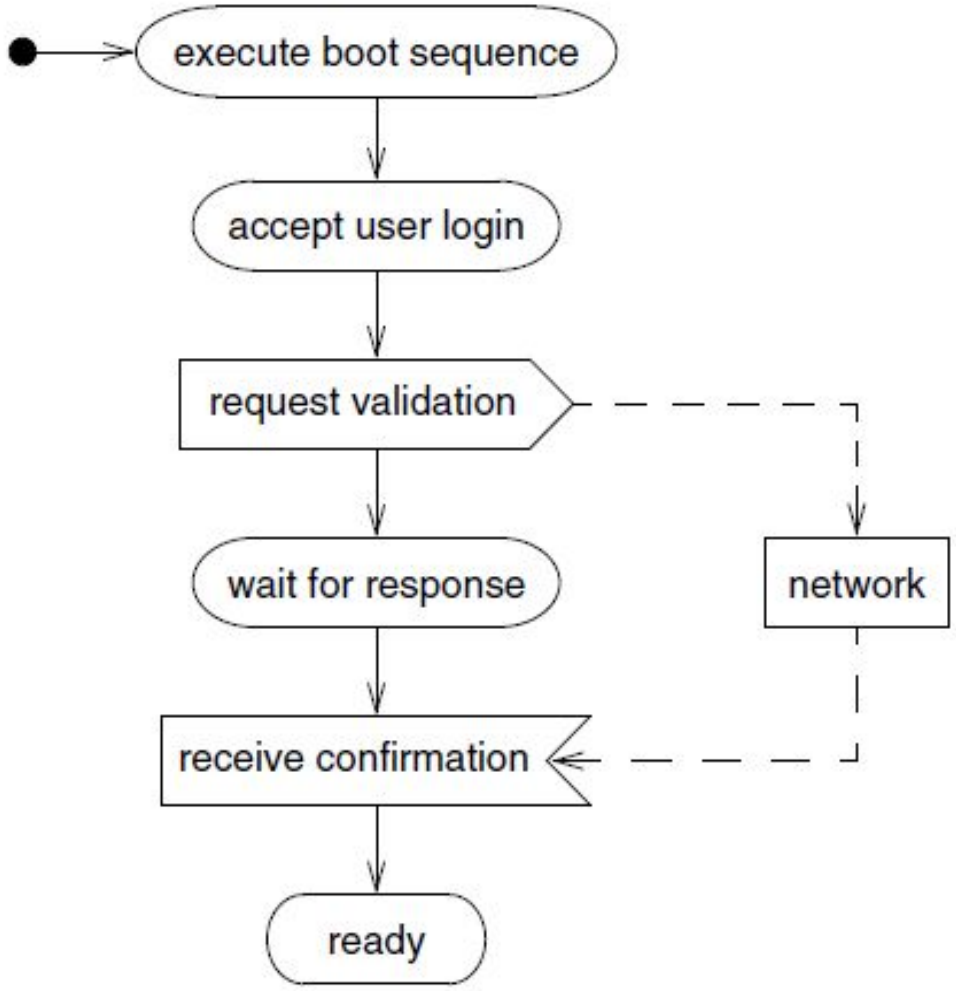
Sending and Receiving Signals

- 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.

Executable Activity Diagrams

Tools that Support Executable Activity Diagrams:

- IBM Rational Rhapsody
- Enterprise Architect
- MagicDraw
- Modelio

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.