Activity Diagram

Lecture B2 — Dynamic Modeling: Workflows and Processes

Prof. Ing. Lelio Campanile

Dipartimento di Matematica e Fisica
Università degli Studi della Campania Luigi Vanvitelli
Corso di Laurea Magistrale in Data Science

A.A. 2025/2026

Based on: Seidl et al. — *UML @ Classroom*, Ch. 5 (Springer, 2015)

Topics covered

Core elements

- Actions and control flow
- Initial and final nodes
- Decision and merge nodes
- Fork and join nodes (parallelism)
- Object nodes (data flow)

Advanced elements

- Swimlanes (partitions)
- Activity Diagram vs. Sequence Diagram
- Activity Diagram for data pipelines
- Best practices and pitfalls

Running example: the **"Enroll in course"** workflow of the Student Administration System — same scenario modeled in B1.

Core Elements

Actions, decisions, forks, and joins

What is an Activity Diagram?

Definition

An **Activity Diagram** describes the *flow of control* (and optionally data) through a system. It models:

- The steps of a **business process** or workflow
- The flow of an **algorithm** at a high level
- The behavior of a **use case** from a process perspective

Activity Diagram vs. Sequence Diagram

Sequence	Activity
Focus: *who* sends *what* to *whom*	Focus: *flow of steps*
One scenario	Process / algorithm
Time on y-axis	Control flow arrows

When to use it

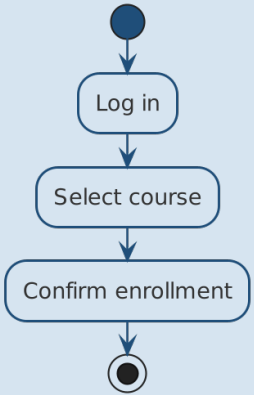
- Documenting business processes
- Describing complex UC flows with branches and parallel paths
- Modeling **data pipelines** (very natural for Data Scientists)

Initial Node, Final Node, Action, Control Flow

Four fundamental elements

Initial node	Filled black circle. One per diagr Starting point.
Action	Rounded rectangle. One step atomic unit of behavior.
Control flow	Arrow between actions. Shows exe tion order.
Activity final	Circle with inner circle (bullse Ends the entire activity.
Flow final	Circle with X. Ends one flow only.

Minimal example



Decision Node and Merge Node

Decision node

A **diamond** with one incoming and **multiple outgoing** edges.

Each outgoing edge has a *guard condition* in brackets `[condition]`.

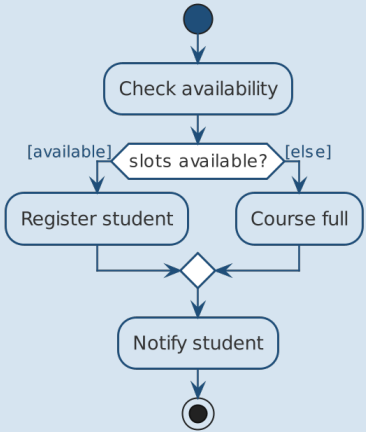
Exactly **one** guard must be true at runtime. Use `[else]` for the default branch.

Merge node

A **diamond** with multiple incoming and **one outgoing** edge.

Brings alternative branches back together. No guard conditions on merge.

Example: availability check



Fork Node and Join Node — Parallelism

Fork node

A thick horizontal (or vertical) bar with **one incoming** and **multiple outgoing** edges.

All outgoing flows are **started simultaneously**.

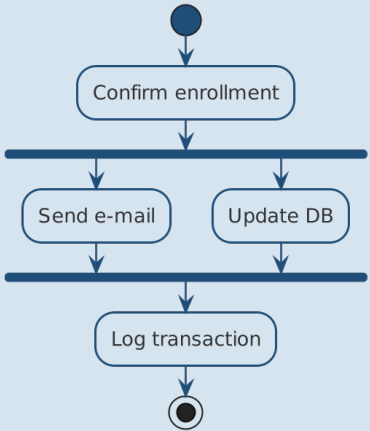
Represents the beginning of *concurrent* execution.

Join node

A thick bar with **multiple incoming** and **one outgoing** edge.

Waits until **all** incoming flows complete (synchronization point).

Example: parallel notification



Object Nodes — Data Flow

Definition

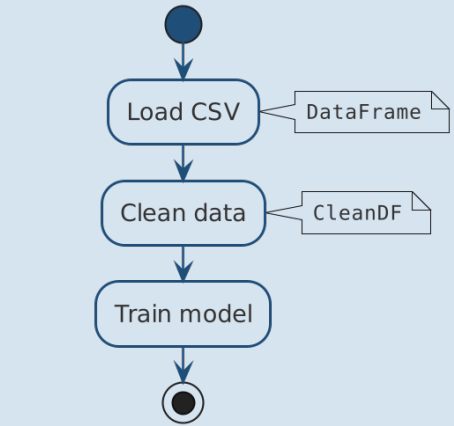
An **object node** represents data (an object) that flows through the activity.

Shown as a *rectangle* (like a class box head, but without compartments).

Pins are a compact notation: small squares attached directly to the action's border.

Object nodes are especially useful when modeling **data pipelines**: each action transforms data from one format to another.

Object node notation



Swimlanes

Partitions: assigning responsibility

Swimlanes (Partitions)

Definition

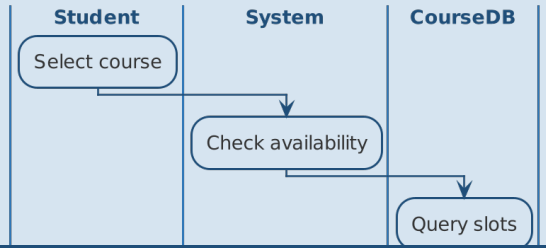
Swimlanes (also called *partitions*) divide the activity diagram into columns or rows, each assigned to a **specific actor, system, or component**.

Actions placed in a lane are the **responsibility** of that lane's owner. Control flow *can cross* lane boundaries.

Why use swimlanes?

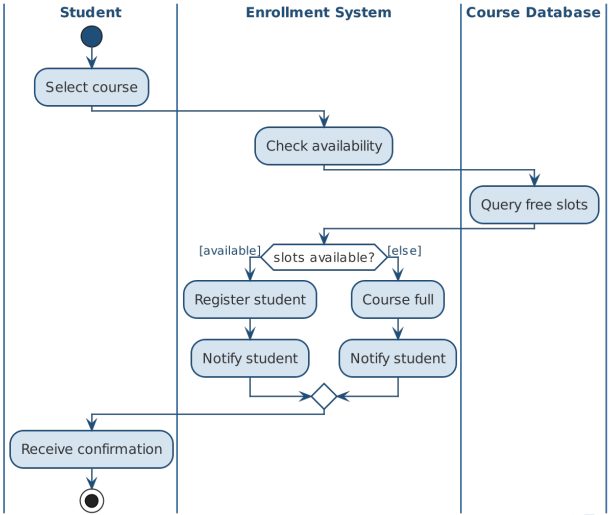
- Makes **responsibility** explicit at a glance
- Highlights **handoffs** between actors or systems (where delays and errors often occur)
- Useful for documenting business processes where multiple departments

Swimlane layout



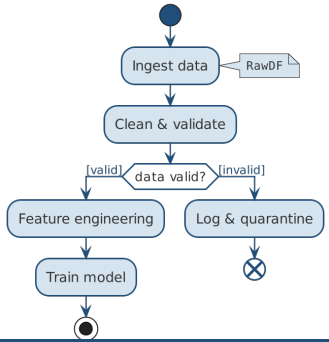
Example: “Enroll in course” with swimlanes

Student Administration System — activity view with responsibility assignment



Why Activity Diagrams suit data pipelines

A data pipeline *is* an activity: a sequence of **transformations**, **decisions** (quality checks, branching on data type), and **parallel** loading steps. Object nodes explicitly represent the **data artefacts** flowing between actions.



Best Practices & Notation Reference

Best Practices and Pitfalls

Best practices

- **One diagram per process**: keep diagrams focused; use sub-activities for nested processes
- **Name actions as verb phrases**: "Check availability", not "Availability" or "Check"
- **Always have an initial node and at least one final node**
- **Balance forks and joins**: every fork must have a corresponding join
- **Use swimlanes** when responsibility assignment matters to the audience

Common pitfalls

- **Confusing activity with sequence**: Activity diagrams do not show *who sends what*; use Sequence Diagrams for that
- **Unbalanced decisions**: each outgoing guard must cover all cases; use `[else]` as safety
- **Missing synchronization**: a fork without a join implies flows never synchronize
- **Over-detailing**: keep actions at a consistent granularity level

Notation Elements

Element	Notation	Description
Initial node	Filled circle	Start of the activity
Activity final	Bullseye (circle in circle)	End of the entire activity
Flow final	Circle with X	End of one flow path only
Action	Rounded rectangle	One step / atomic behavior
Control flow	Arrow	Sequence between actions
Decision	Diamond, 1 in / n out	Conditional branch (guards required)
Merge	Diamond, n in / 1 out	Rejoins alternative branches
Fork	Thick bar, 1 in / n out	Starts parallel flows
Join	Thick bar, n in / 1 out	Synchronizes parallel flows
Object node	Rectangle (no compartments)	Data artefact flowing through
Swimlane	Vertical/horizontal partition	Assigns responsibility

Key points

Activity Diagrams model flow

They describe the *sequence of steps* in a process or algorithm, including branches, loops, and parallel paths. They do not show which object sends which message.

Swimlanes add responsibility

Swimlanes make it explicit which actor or system is responsible for each action. Highly recommended for multi-stakeholder processes.

Decision vs. fork

Decision: one branch is chosen based on a condition (mutually exclusive). **Fork**: all branches start simultaneously (concurrent). Never confuse the two.

Natural fit for data pipelines

For Data Scientists: an Activity Diagram with object nodes is the natural way to document a **data pipeline** — from ingestion to model training.

What's next

Lecture B3 — State Machine Diagram

Modeling the *state-driven lifecycle* of a single object.

Where Activity Diagrams model a *process*, State Machine Diagrams model the *states and transitions* of one object in response to events.

Seidl et al., Ch. 7

Module B summary so far

✔	B1	Sequence Diagram Object interactions, one scenario
✔	B2	Activity Diagram Process flow, parallel paths, swim-lanes
	B3	State Machine Diagram Object lifecycle, event-driven

Thank you

Prof. Ing. Lelio Campanile

lelio.campanile@unicampania.it

Dipartimento di Matematica e Fisica
Università degli Studi della Campania Luigi Vanvitelli

Elements of Software Engineering and Information Systems

Corso di Laurea Magistrale in Data Science