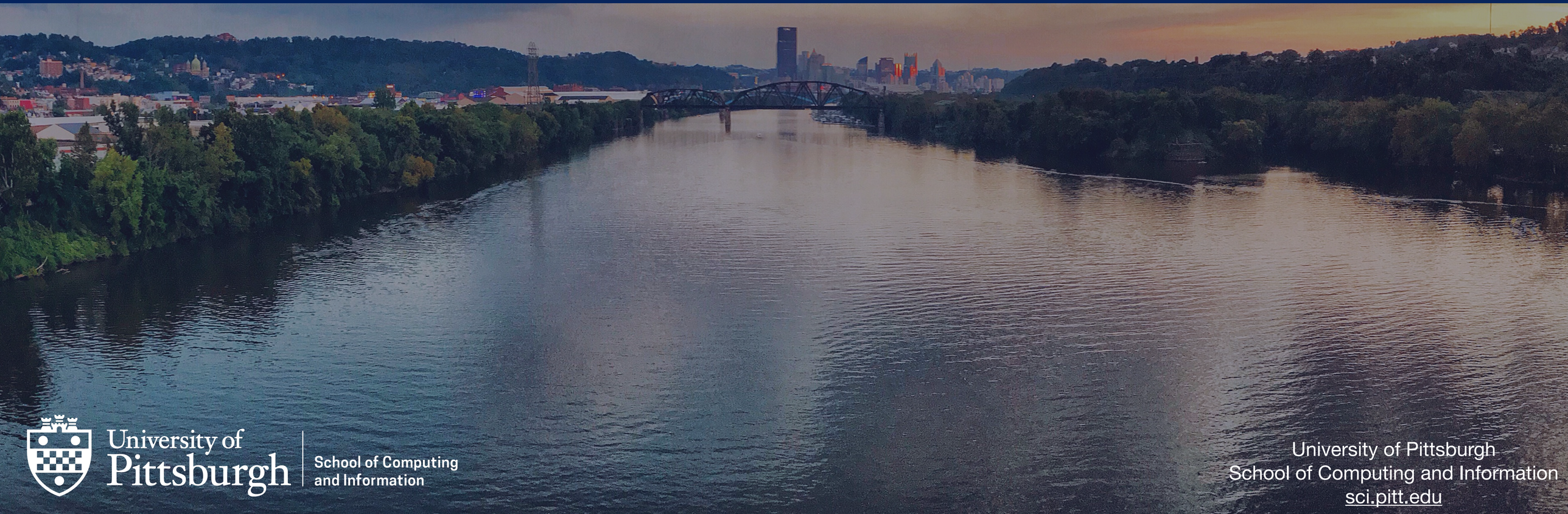
Fall 2025

L14 Dynamic Modeling

CS 1530 Software Engineering

Nadine von Frankenberg



University of Pittsburgh
School of Computing and Information

University of Pittsburgh
School of Computing and Information
sci.pitt.edu

Copyright

- These slides are intended for use by students in CS 1530 at the University of Pittsburgh only and no one else. They are offered free of charge and must not be sold or shared in any manner. Distribution to individuals other than registered students is strictly prohibited, as is their publication on the internet.
  - All materials presented in this course are protected by copyright and have been duplicated solely for the educational purposes of the university in accordance with the granted license. Selling, modifying, reproducing, or sharing any portion of this material with others is prohibited. If you receive these materials in electronic format, you are permitted to print them solely for personal study and research purposes.
  - Please be aware that failure to adhere to these guidelines could result in legal action for copyright infringement and/or disciplinary measures imposed by the university. Your compliance is greatly appreciated.
- Material from these notes is obtained from various sources, including, but not limited to, the following:
  - Bruegge, & Dutoit. Object–oriented software engineering. using UML, patterns, and Java. Pearson, 2009.
  - Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides. Design Patterns. Pearson, 1994.
  - Sommerville, Ian. "Software Engineering" Pearson. 2011.
  - http://scrum.org/

Learning goals

- You can model an UML activity diagram
- You can model an UML state chart diagram
- You understand the purpose of a UML communication diagram

[Example] Battery

- Design Goal 1: Battery Capacity (Range)
  - Maximize the battery capacity to provide an extended driving range for electric vehicles, allowing users to travel longer distances on a single charge.
- Design Goal 2: Battery Weight and Size (Efficiency)
  - Minimize the weight and size of the battery pack to improve vehicle efficiency and reduce energy consumption.
- Design Goal 3: Battery Lifespan (Longevity)
  - Design the battery to have a long lifespan, reducing the frequency and cost of battery replacements for EV owners.

[Example] SIMCity

- DG Usability: After completing a 10-minute tutorial, ≥ 80% of users can independently navigate the main menu and recall the core functions of the system.
- DG2 Reliability: The system ensures end-to-end data integrity by synchronizing verified sinkhole status changes across all clients within 5 minutes.
- NFRs
  - NFR1 Usability (Learnability): The system should provide a 10min tutorial to help users understand its core features (…).
  - NFR2 Usability (Navigation): Users should be able to access the menu within two steps.
  - NFR3 Usability (Accessibility): The system should support text-to-speech for improved accessibility.
  - NFR4 Robustness: The system should handle increased traffic (+ 60%) during rush hours, with higher loads in winter than in summer (+ 70%).
  - NFR5 Performance: The map should reflect new and updated reports within five minutes.
  - NFR6 Reliability: The sinkhole status should be updated within five minutes after verifying a sinkhole and after removing a sinkhole.
  - NFR7 Portability: The system should be available as a mobile app.
  - NFR8 Availability: The system should require and verify Internet connectivity for (…) functionality.

System design concepts — Overview

- ✅ Design goals
- → Control flow
- Subsystem decomposition
- HW/SW mapping
- Data management
- Access control
- Boundary conditions

University of Pittsburgh

Please complete the

Midterm Survey

Thank you!

CS 1530 - SOFTWARE ENGINEERING - 1000 - Lecture

Students

https://go.blueja.io/XXDCWA7Ltkii1npCrtobcA

Today's roadmap

- Dynamic modeling
  - Overview
  - Modeling workflows (UML Activity Diagram)

Dynamic modeling

- Describes the **dynamic behavior** of parts (components) of the system
- Typically, we focus on **interesting behavior**
  - e.g., specific workflows involving the user/system with many actions
- Dynamic model can be described in many ways
  - An entity's state
  - An entity's internal behavior / behavior with the system / behavior with other entities

Dynamic modeling tools

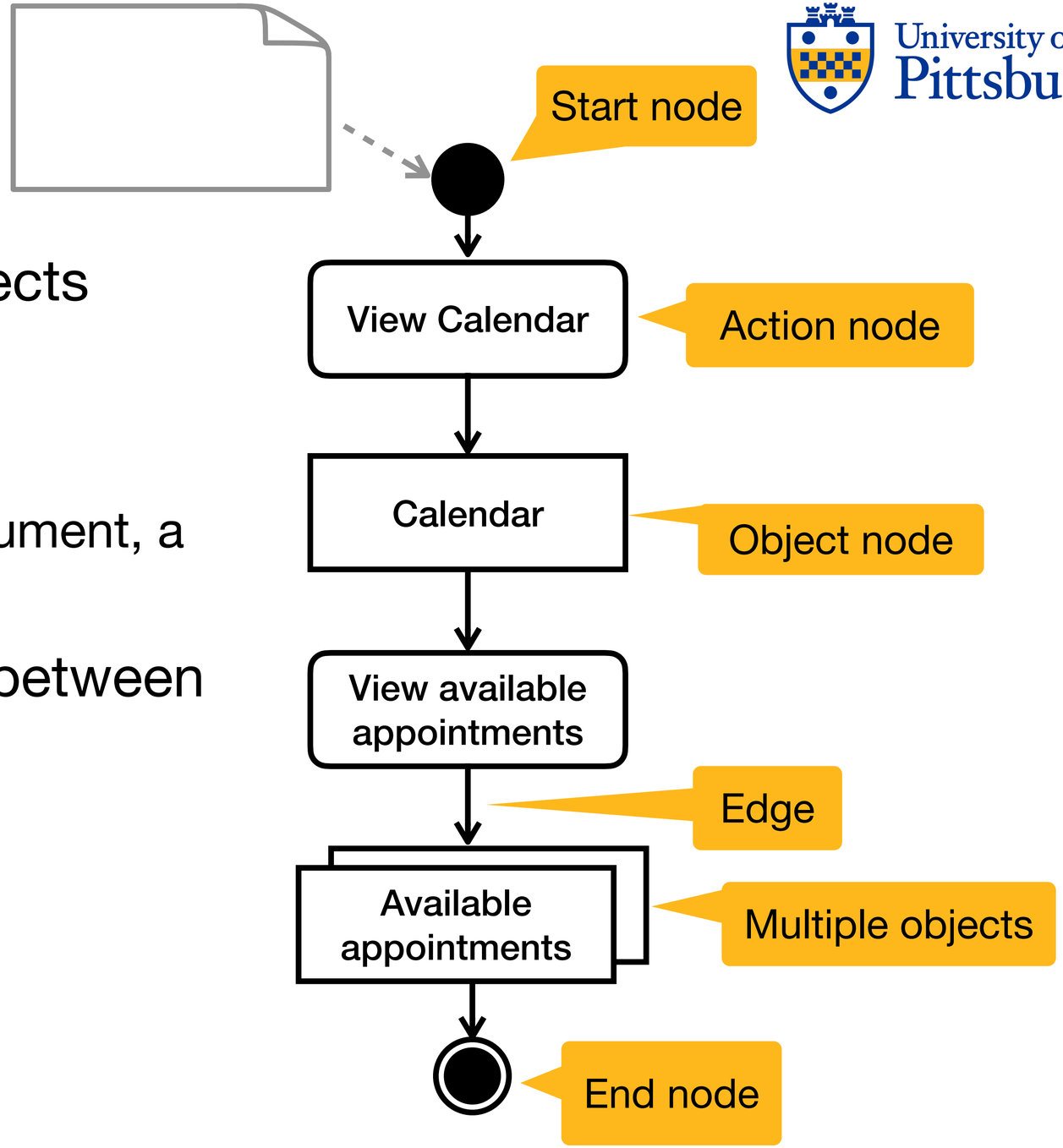
- **Activity Diagram**
  - Purpose: Represent workflows within use cases
  - Usage: Visualize complex user interactions with multiple actions/decision points
- **State Chart Diagram**
  - Purpose: Model the dynamic states and transitions of a single entity or object
  - Usage: Effective for illustrating how an object changes states in response to events
- **Communication Diagrams**
  - Purpose: Illustrate the interactions and messages exchanged among multiple entities
  - Usage: Useful for modeling collaboration and communication patterns in a system
- **Sequence Diagrams**
  - Purpose: Visualize the interactions between objects in a system over time
  - Usage: Show message sequence exchanged between objects to perform a specific function

UML activity diagrams

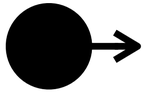
- Organize and represent the flow of activities and actions, grouping them by the interacting system or actor
- Help to more easily understand complex workflows
- Show the order of execution, enabling modeling of both simple and complex behaviors

UML activity diagrams

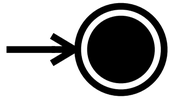
- Consist of nodes and edges
- **Nodes** describe activities and objects
  - Control nodes
  - Executable nodes
  - Object nodes (an object, e.g., a document, a SLF)
- An **edge** is a directed connection between nodes



UML activity diagram syntax



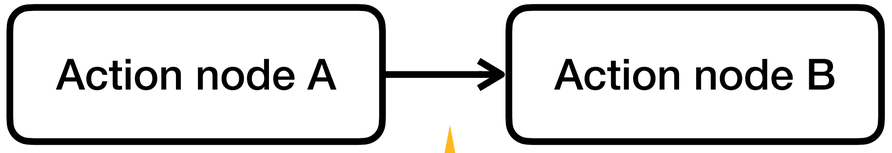
Start or initial node



End or final node

Describes an action

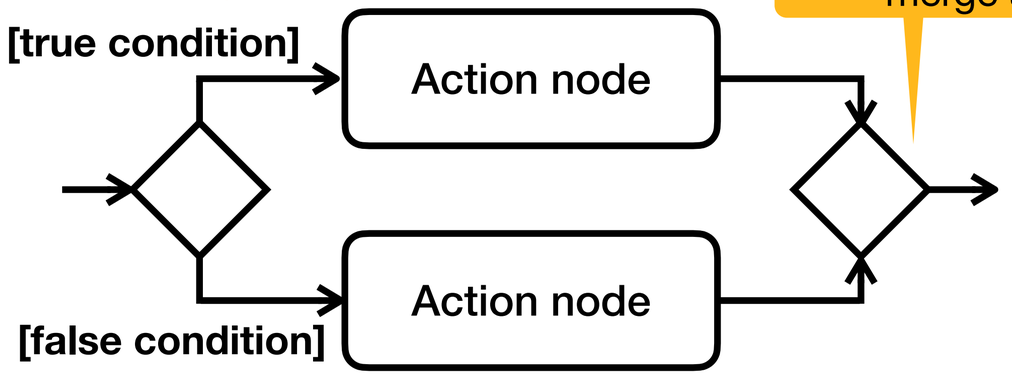
Describes an object
(often hints at data)



Control flow: action A is followed by action B

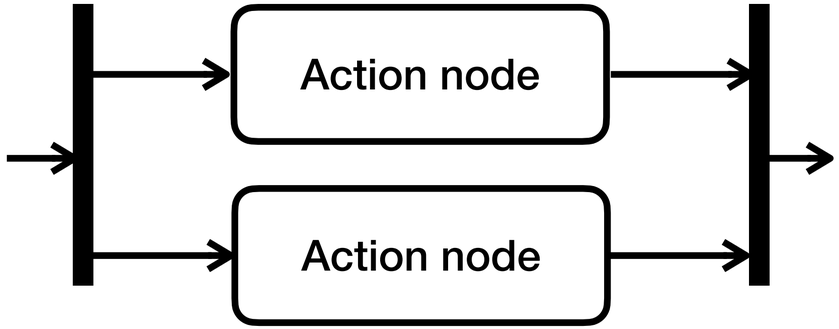
Decision node: Decide on the control flow

Merge node: If the different control flows merge again



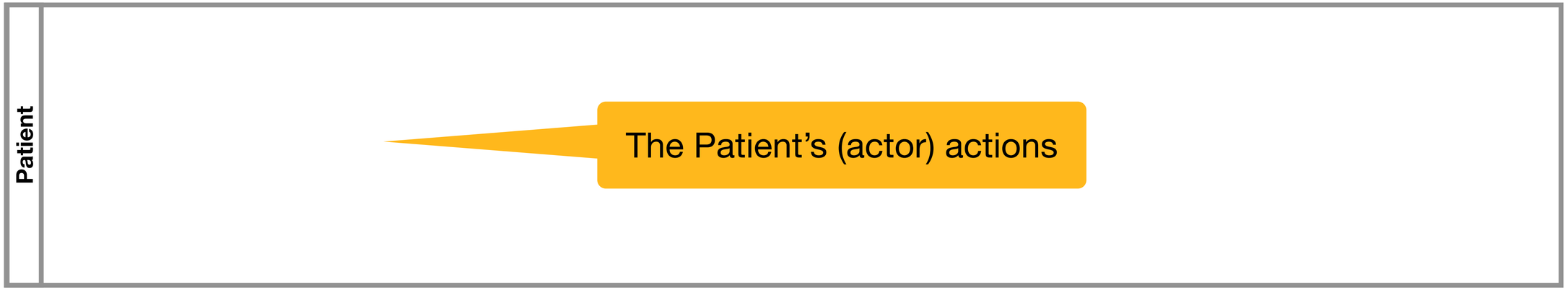
Fork node: Concurrent actions (order is not important)

Join node: Indicates the end of the concurrent actions



Activity diagrams can use **swimlanes**

- Purpose: **Group actions** by viewpoint/role to indicate **responsibilities within a process or workflow**
- A swimlane within an activity diagram represents a **boundary**
  - Helps in separating responsibilities and interactions
- Swimlanes combine the system's high-level logic with responsibilities and interactions



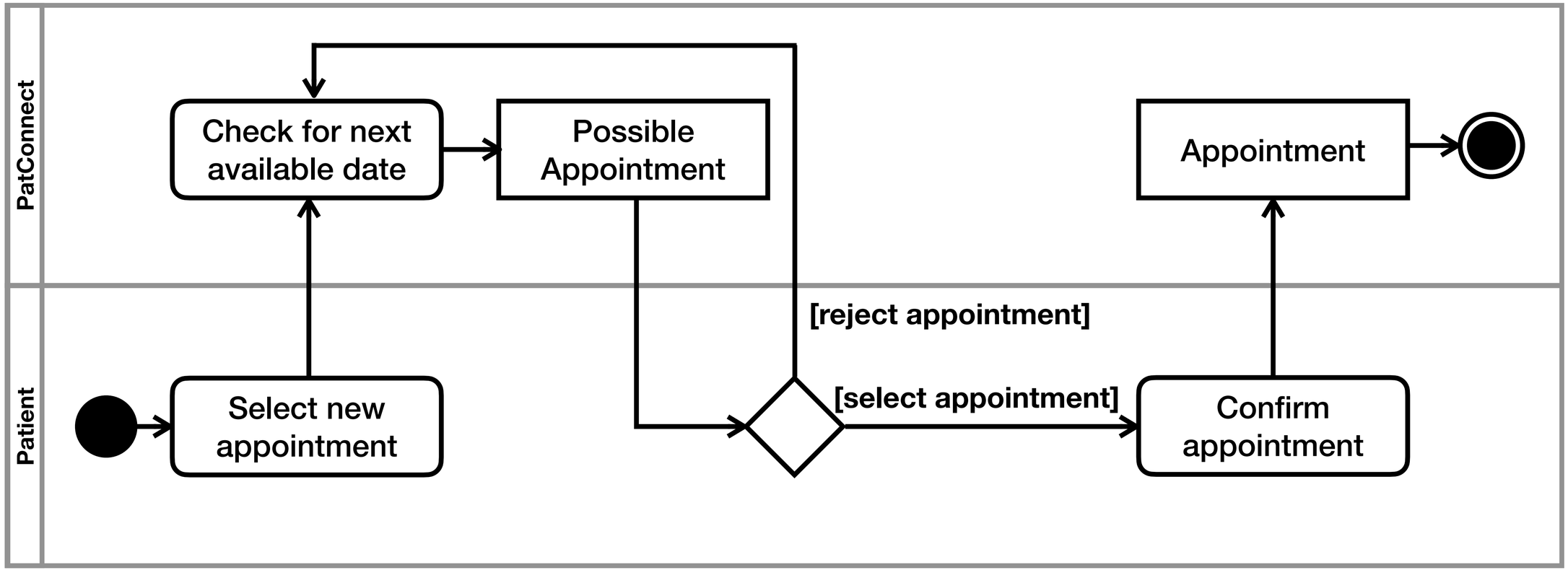
[Example] Swimlanes

"Depth" of the model depends on the context!

University of Pittsburgh

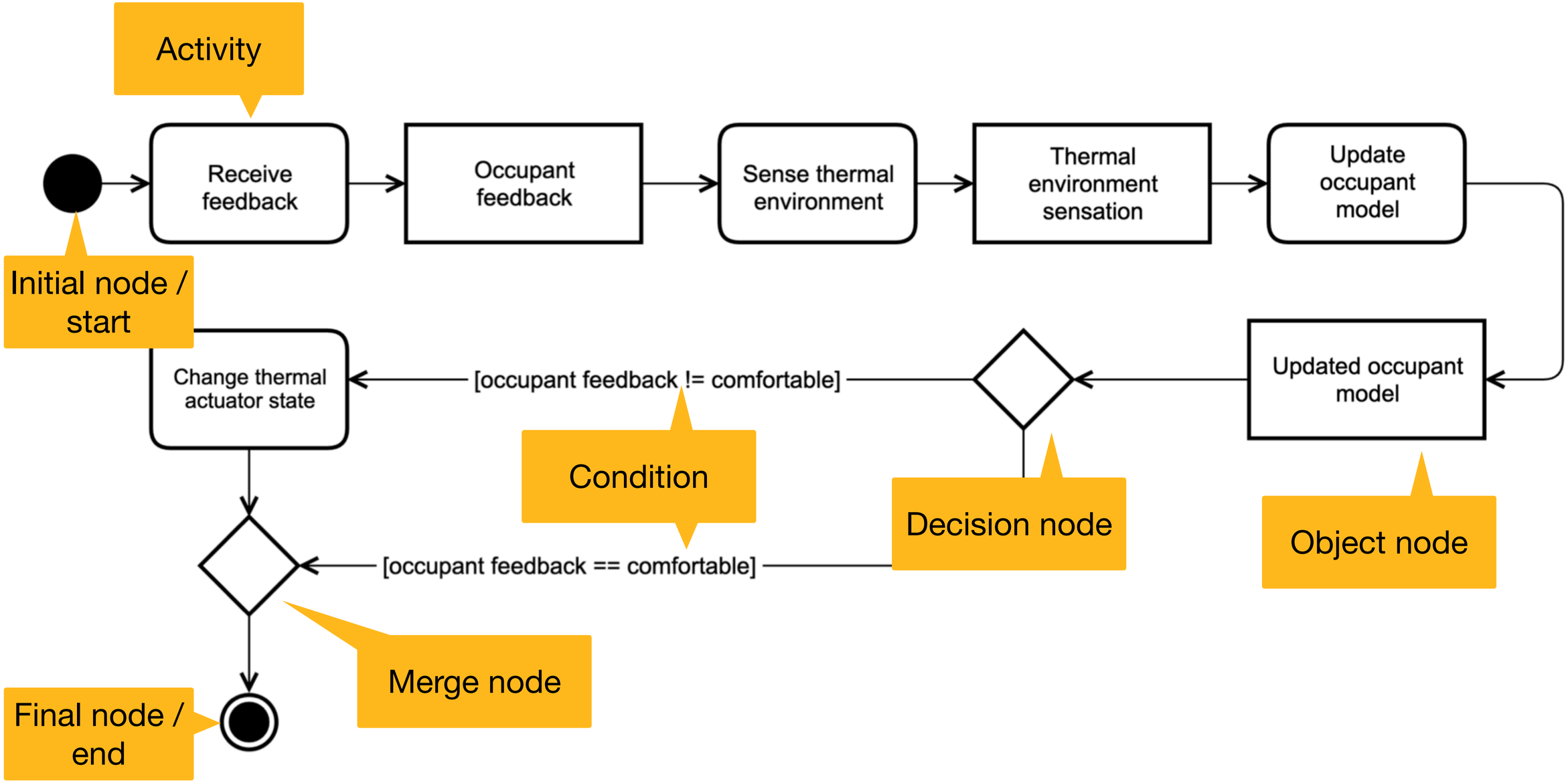
System's responsibility

Actor's responsibility



Swimlane: Describe the activities from one actor's or system's point of view

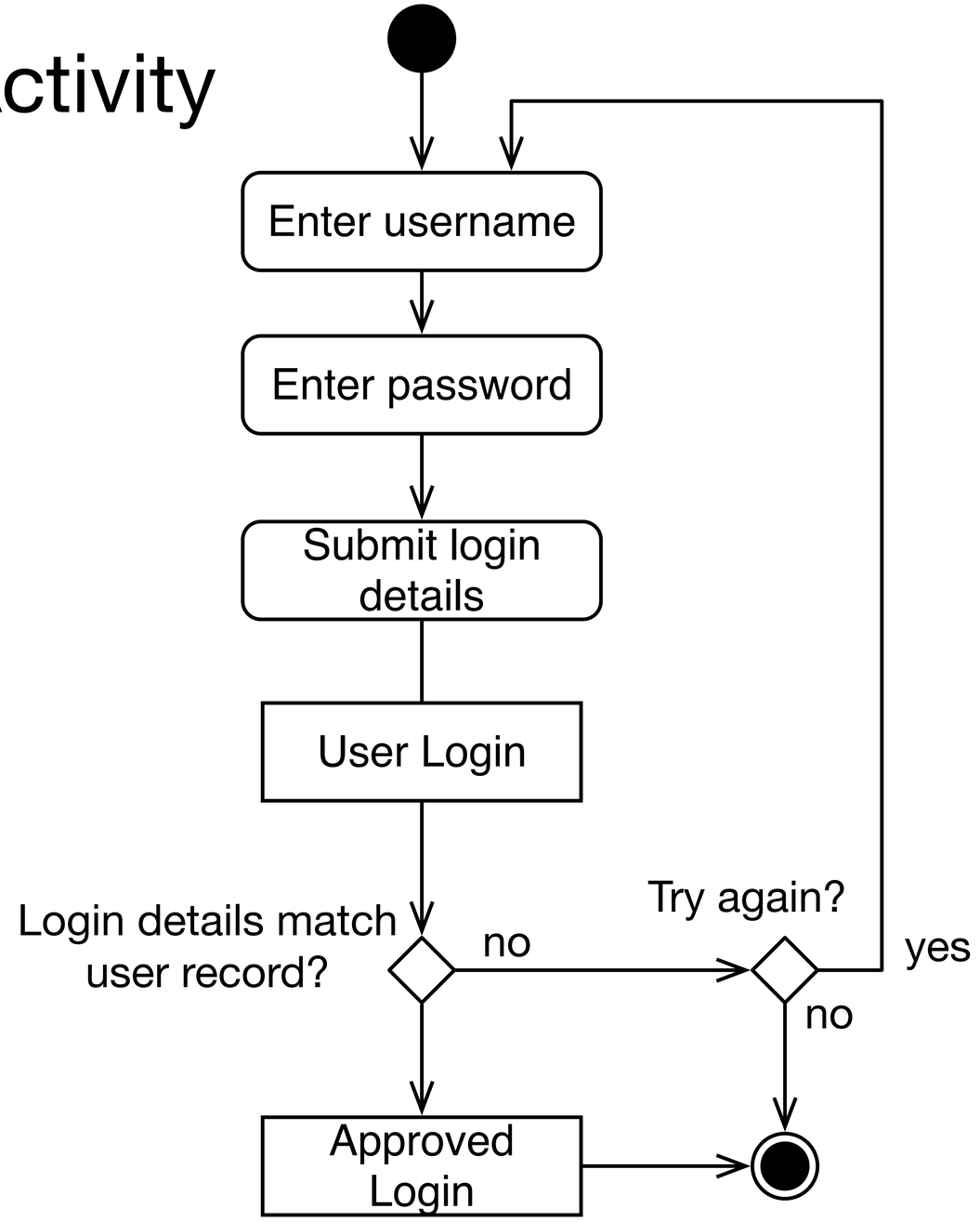
[Example] Dynamic model - activity diagram



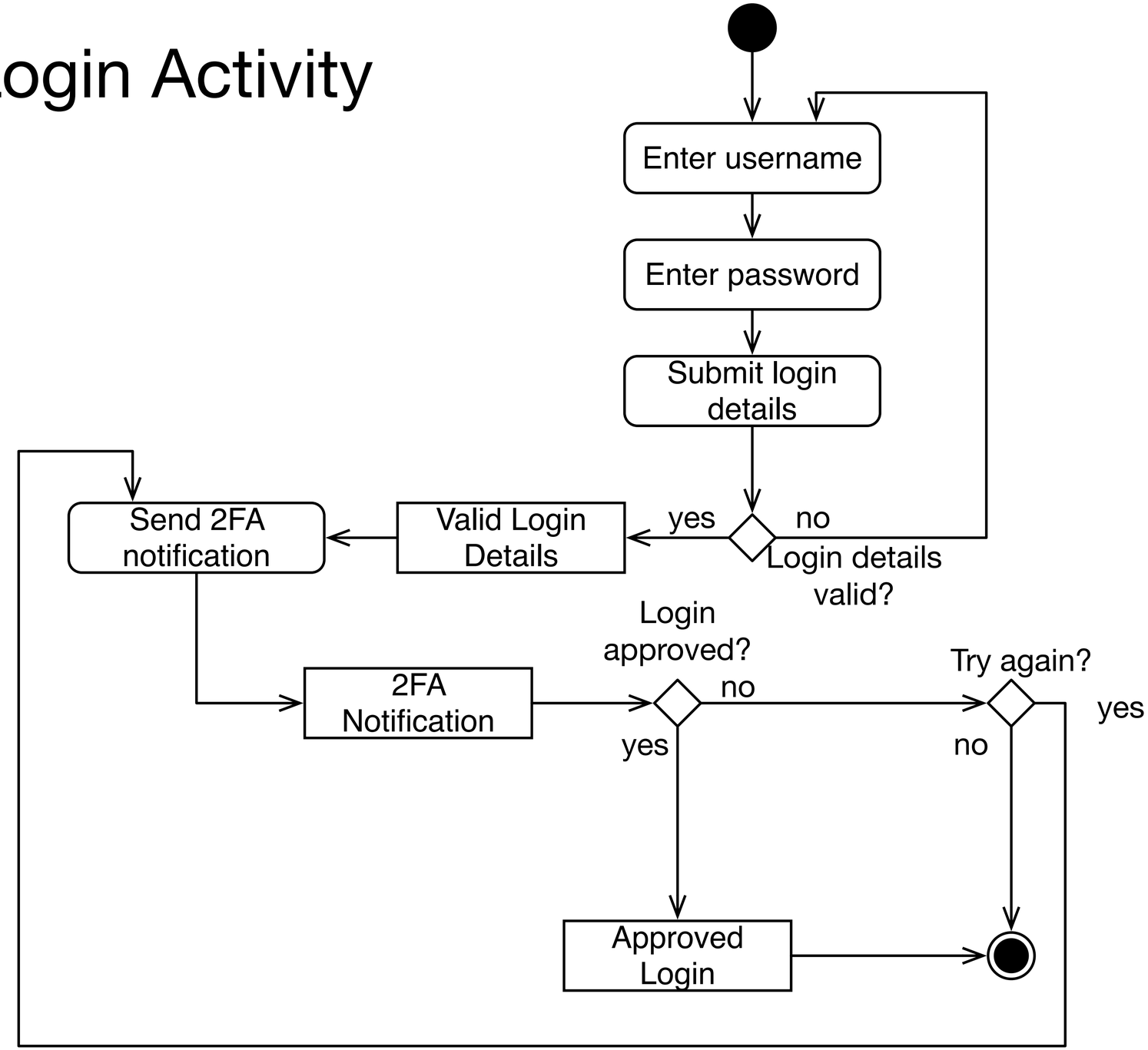
How to model an activity diagram

- Activity diagrams serve to illustrate various workflows, such as:
  - User-specific workflows
  - System workflows
  - Logical workflows
  - …
- These workflows are frequently derived from:
  - The flow of events within use cases
  - Associations found in the analysis object model

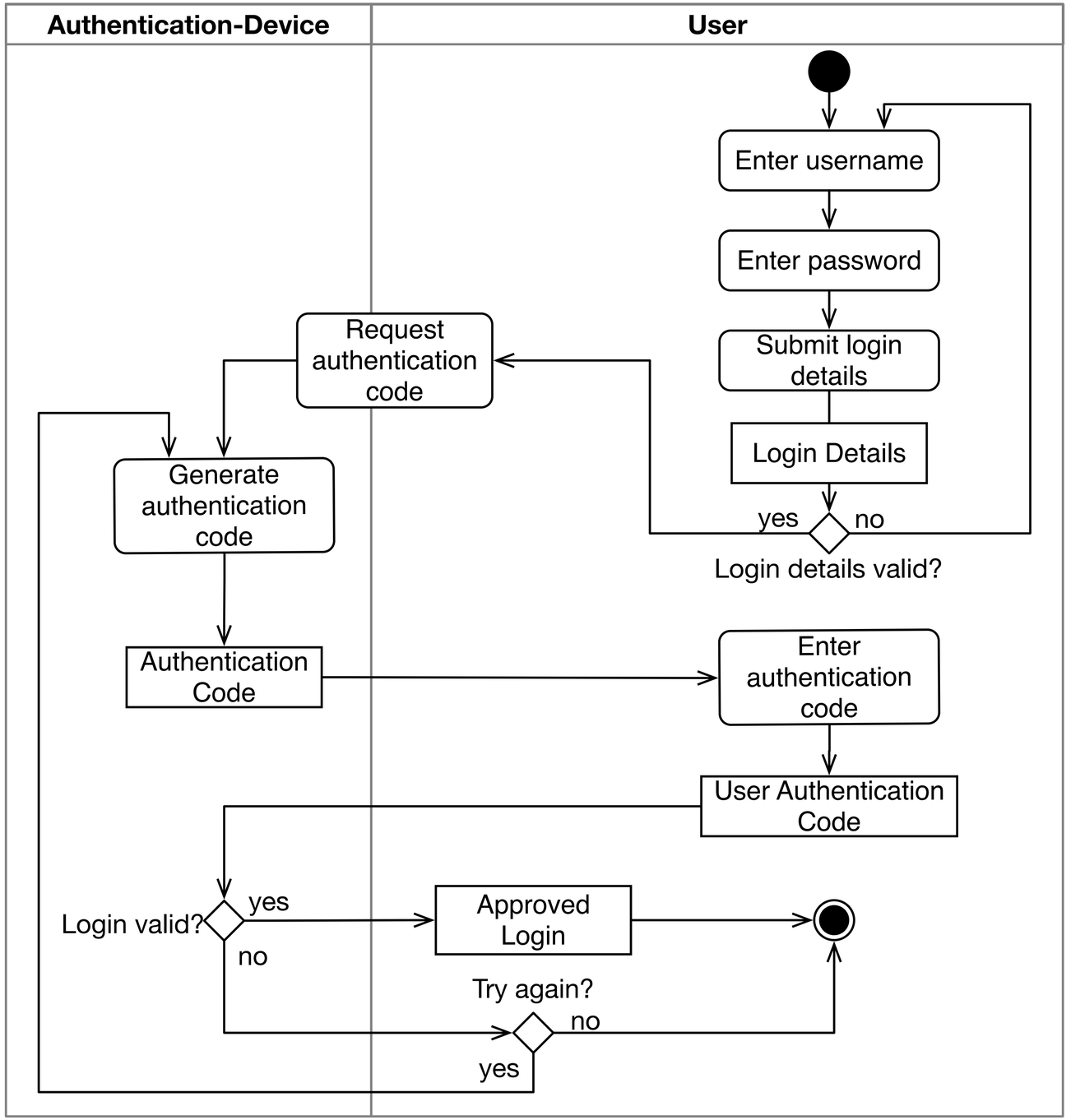
[Example] Login Activity



[Example] Login Activity



[Example] Login Activity (2FA)



L14 — In-class: Activity diagram

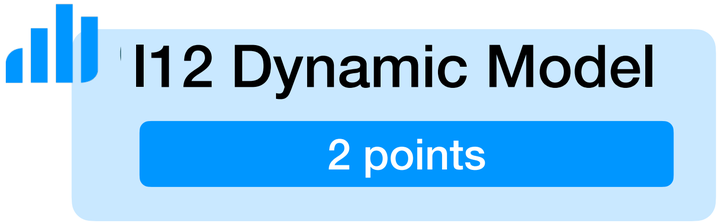
⏳ 20 min

👩🏽‍💻👨🏻‍💻 Pairs

University of Pittsburgh

Model a UML activity diagram to describe an **interesting**, non-trivial workflow, e.g.:

- The enduser's workflow to submit a report
- The city official's workflow to confirm a report
- … (pick your own workflow!)



References

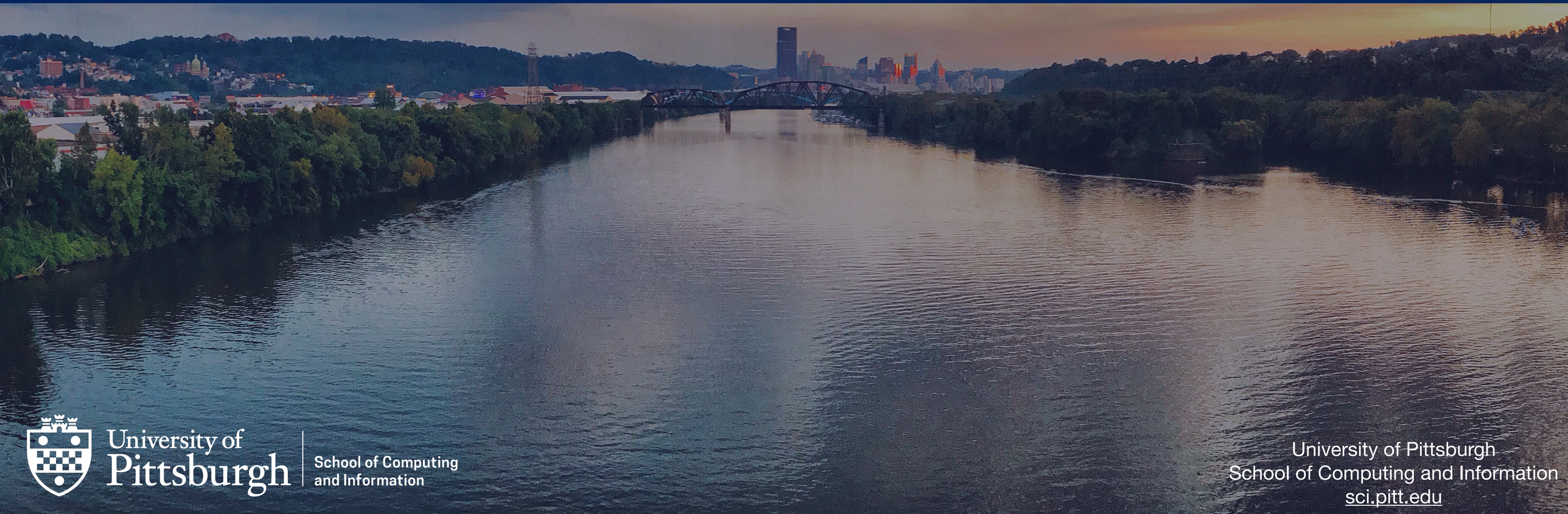
- Bruegge, & Dutoit. Object–oriented software engineering. using UML, patterns, and Java. Pearson, 2009.
- Object Management Group. Unified Modeling Language. Version 2.5.1, 2017

Fall 2025

L14 Dynamic Modeling

CS 1530 Software Engineering

Nadine von Frankenberg



University of Pittsburgh
School of Computing and Information

University of Pittsburgh
School of Computing and Information
sci.pitt.edu