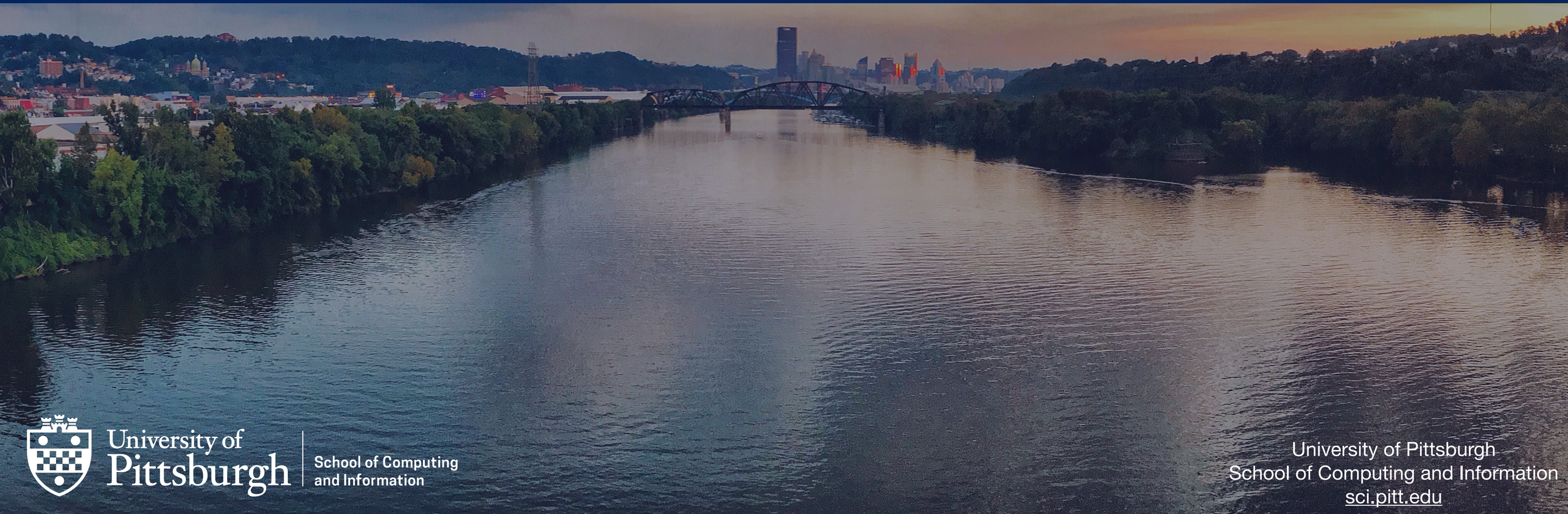
Fall 2025

L09 Scenarios & Use Cases

CS 1530 Software Engineering

Nadine von Frankenberg



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 understand the different means of representing requirements
  - Scenarios
  - Use case
- You can model a use case diagram

Today's roadmap

- Scenarios
- Use cases
- User stories

So far …

- We have defined functional, non-functional requirements, and constraints
- They form the foundation of the system we want to build

Problem

How do we ensure every stakeholder understands the system?

How can we validate the requirements with stakeholders?

-> We need techniques that make requirements concrete and communicable!

Common techniques

- **Interviews**
  - Structured vs. unstructured, open-ended questions
- **Surveys**/questionnaires (useful for large groups)
- **Workshops**/brainstorming
- **Observation** (shadowing users, contextual inquiry)
- **Document Analysis:** Legacy system specs, regulations
- **Prototyping** – Low-fidelity mockups to elicit feedback
- **Use Case / Scenario Writing:** Stories of user interaction
- **User Stories** (agile format)
- …

Validation techniques align requirements with user needs

- **Scenarios:** A concrete story of a user interacting with the system
- **Use Cases**: A structured description involving generic end-users (actors) and the system, abstraction that covers all possible instances
- **User Stories**: A short statement of the need from the end-user's perspective, making it easier to understand and prioritize

These techniques help validate requirements with stakeholders and identify missing FRs, NFRs, or constraints.

Identifying scenarios

- Informal **short** descriptions of specific user interactions in **natural language**
- They provide a **narrative** of how the system should function through a **sequence of events, actions, or steps**
  - Used for exploring detailed interactions and edge cases
- Basis for test cases & validation
  - Best case scenario
  - Worst case scenario
  - Average case scenario

Describe a flow of events with multiple potential outcomes

[Example]

FR3 View Sinkhole: Residents and city officials should be able to view reported sinkholes on a map.

Jamie opens the SiMCity app on their phone while walking home.
They see visual markers for sinkholes on a street map.
When they select a sinkhole marker, the app shows the sinkhole's location, date reported, and verification & repair status.

Yes, I want to see a detailed status!

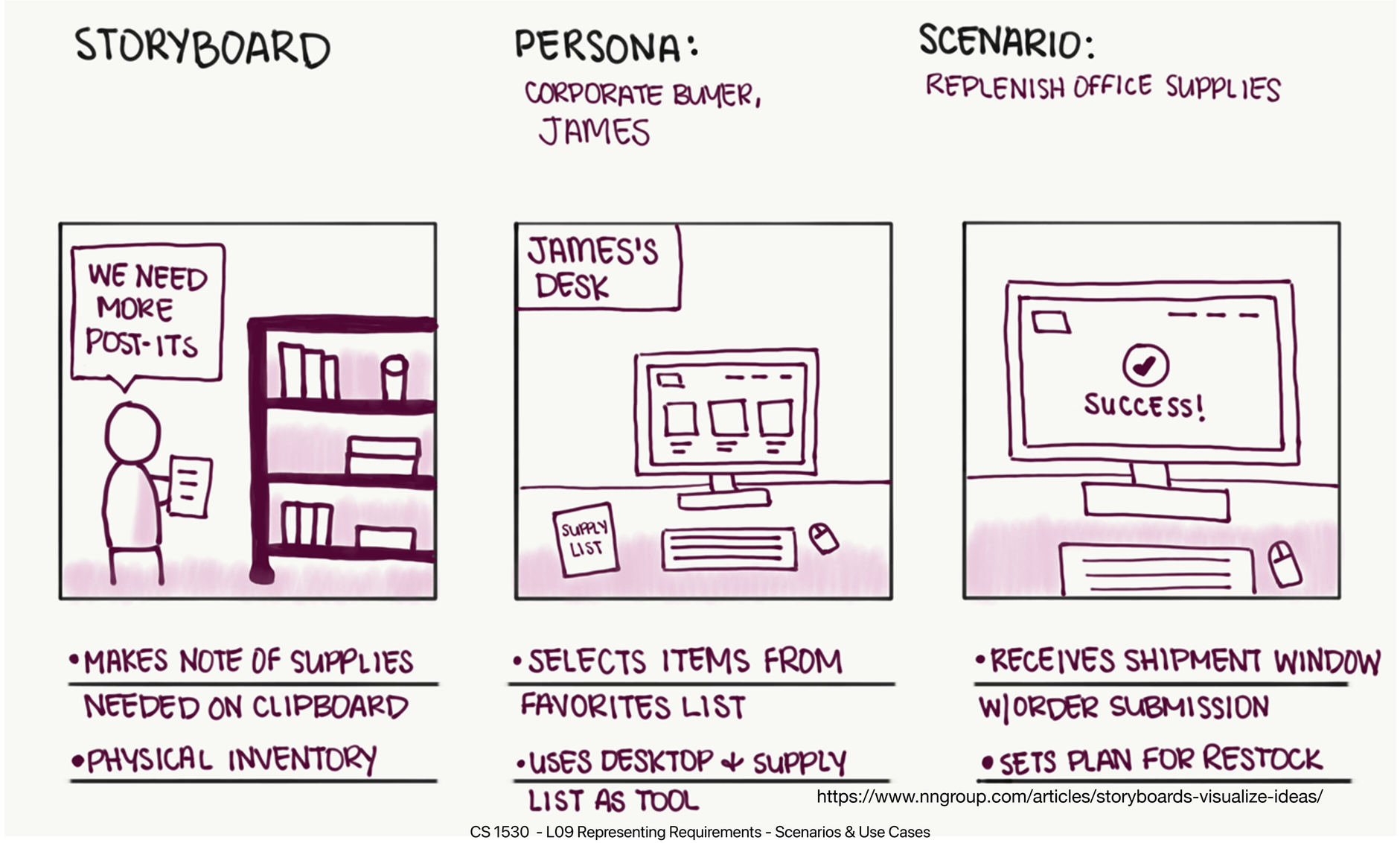
Residents should not see unverifiied reports.

Scenarios have a concrete story-line

- Scenarios enhance requirements elicitation by providing a tool that is understandable to stakeholders
- Concrete, user-centered descriptions of system features used by actors (end-users), often involving informal descriptions
  - Textual representation of how the system is used **written from the** (end)**user's perspective**
  - Scenarios can incorporate visual elements: e.g., videos, pictures, or storyboards
  - Scenarios can encompass contextual details, e.g., the (physical) environment, social situations, …
- In **scenario-based design**, scenarios serve as the foundation for designing the hypothetical interaction of end-users with a system

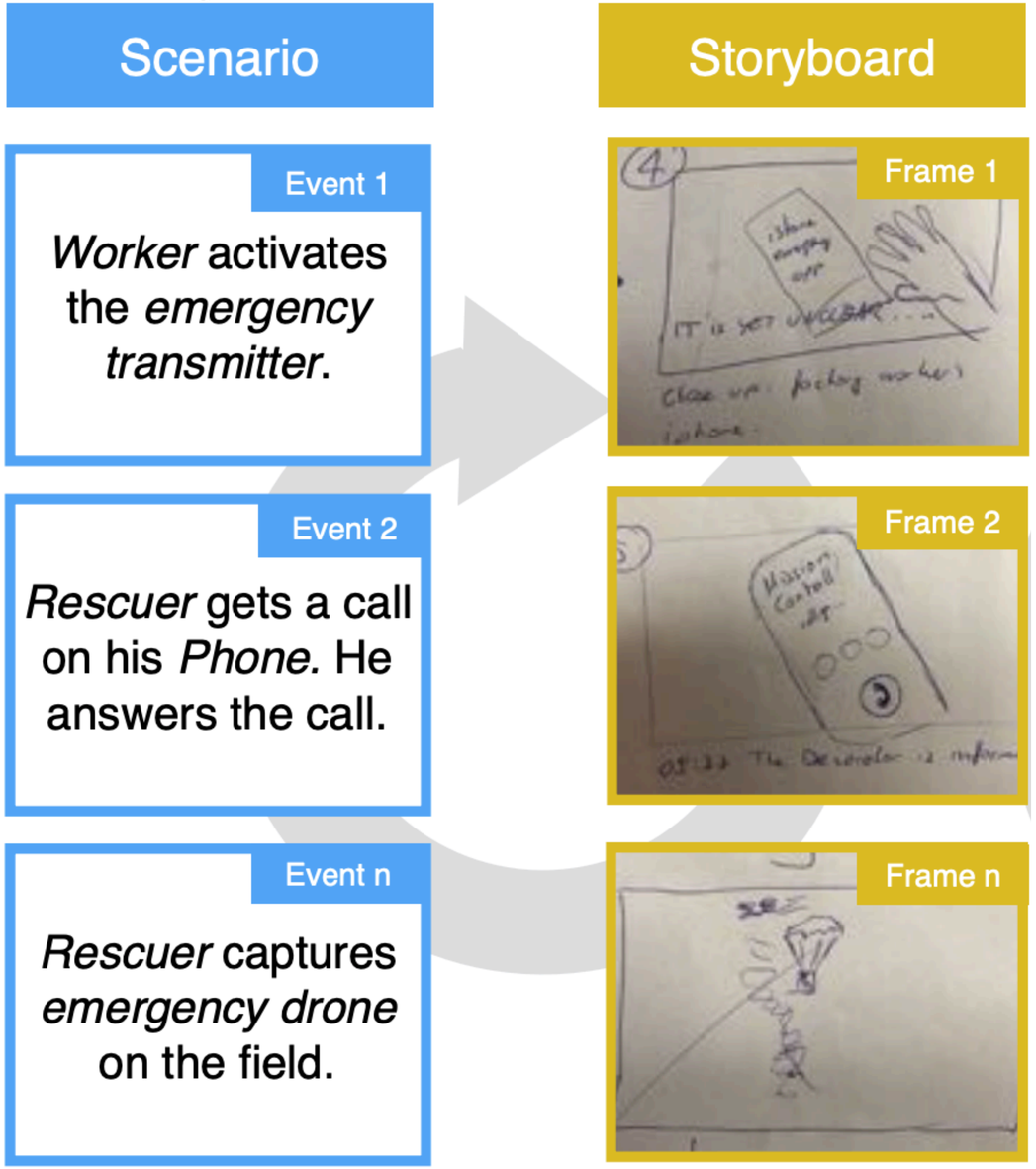
[Example] As-is scenario

FR4 Reorder Supplies: The system allows users to reorder supplies.



https://www.nngroup.com/articles/storyboards-visualize-ideas/

[Example] From Storyboards to Code



Lukas Alperowitz, Constantin Scheuermann, and Nadine von Frankenberg. "From Storyboards to Code: Visual Product Backlogs in Agile Project Courses." *SEUH*. 2017.

Scenarios

- A scenario is a textual description that should include:
  - The **name** of the scenario
  - The **user** ("instantiated" main actor of the scenario)
  - The **purpose** of the scenario
  - **Steps** of the scenario
  - *Optional:* Assumptions about, e.g., equipment/software

How to write a scenario?

User

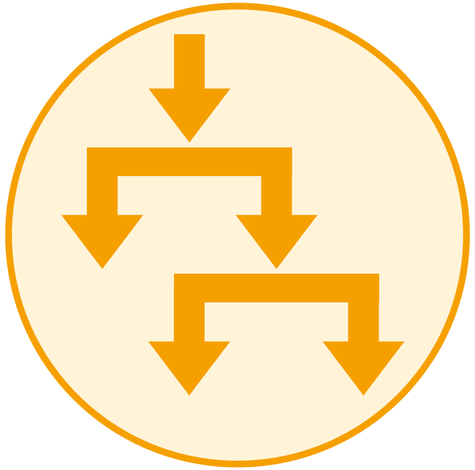
Personas

Define the target user

Name & Purpose

Goal

Define the goal



Steps

Paths

Define the task(s), possible paths, and problems

Icons source

How to write a scenario?

- Format can be **informal** or **formalized**
- The rest is up to the author!
  - E.g., include different formatting to highlight situations or context
  - Names and details should (somewhat) resonate with the audience

Informal scenario
Scenario name
Scenario description *(natural language text)*

Formal scenario
Scenario name
Participating actors
Flow of events: 1. Scenario description in bullet points 2. Typically each point describes one step (i.e., event)

[Example] Visionary Scenario

👓 Read

University of Pittsburgh

Pizza Cravings: Ordering a Pizza

On a Wednesday night, Sarah, a CS senior at Pitt, is fully immersed in her study session for 1530. Around 10pm, she suddenly craves a pizza. She opens the pizza ordering app "CrustRight", logs in with her Pitt student ID, and selects to have her order delivered.

After ordering a small Supreme pizza with extra cheese and a surprise soda, she selects her dorm room as delivery address and pays using her student meal plan.

She receives a confirmation and an info that her order will be delivered in 15 — 20 minutes.

She waits patiently. 16 minutes later, she receives a message that her pizza is outside. She buzzes the delivery person in and receives her order at her doorstep. She enjoys her late-night meal, reenergizing her study session.

[Example] As-is scenario

🚨 **[Notation alert]**
An instantiated actor is denoted as
Concrete Name : ActorType
(both **:** and underlining the actor indicate that we are dealing with an instance)

Scenario	**Depositing money**
Participating actors	Robin: BankClient
Flow of events	1. Robin celebrated their birthday last Wednesday and received birthday money ($300) from their relatives and wants to deposit it to their bank account. 2. Robin visits a bank branch of their bank "HelloSavings", a 5 minute walk from their apartment. 3. They queue up at the teller's counter, behind 10 other customers and wait patiently. 4. Finally, when it's Robin's turn to talk to an agent, they need to provide physical identification. Robin uses their driver's license and the agent confirms their identity. 5. Robin hands over the $300. The agent counts it, confirms the amount, and uses the bank's internal system to schedule the move to Robin's account. 6. Robin receives a printed receipt for the transaction. 7. Two weeks later, Robin receives their bank statement by mail and checks their balance to confirm that the birthday money transaction has made its wait to their account.

[Example] Visionary scenario

Scenario	**Depositing money**
Participating actors	Robin: BankClient
Flow of events	1. Robin celebrated their birthday last Wednesday and received birthday money ($300) from their relatives and wants to deposit it to their bank account. 2. Robin opens the "Coinnect" app. 3. The "Coinnect" app shows a welcome message and asks Robin to log in. 4. Robin logs in with a biometric login. 5. The "Coinnect" app shows an overview of Robin's favorite banks. 6. Robin selects to file a deposit at a physical location with "HelloSavings" bank. 7. The "Coinnect" app shows all physical locations of Robin's bank within a 5 mile radius and the current wait times. 8. Robin's closest bank location (a 5 minute walk away) shows a current wait time of 32 minutes. 9. Robin selects to virtually stand in line. 10. Robin receives a notification 8 minutes before it is their turn in line. 11. Robin leaves their apartment and walks over to the physical location of the "HelloSavings" bank. 12. When Robin arrives, the "Coinnect" app registers their arrival and confirms their spot in the current queue. 13. Robin receives a number, which the bank teller calls out when it is their turn 2 minutes later.

Types of scenarios

- **As-is:** Describes the current situation, often focusing on an existing system
- **Visionary:** Envisions the desired future state, outlining system goals
- **Demo:** Showcases system functionality to stakeholders
- **Test:** Specifies steps for quality assurance and validation
- **Evaluation:** Assesses system effectiveness against criteria
- **Training:** Guides users in effective system use

Greenfield
&
Re-engineering Projects

Demos

Demos
&
Acceptance Tests

Post delivery

Deriving scenarios

- **Client conciseness**: Customers may not provide extensive details for non-existent systems
- **Domain expertise**: Customers understand the problem domain, not the solution domain
- **Avoid assumptions**: What is obvious to you may not be apparent to the customer
- **Dialectic engagement**: Engage in dialogue to help clients formulate scenarios
- **Dynamic requirements**: Requirements often evolve during scenario formulation
- **Observation**: For existing systems, insist on task observation, interface engineering, or re-engineering
- (End-)**user engagement**: Speak to (end-)users for valuable insights

How to write a scenario? (cont.)

- **Starting point**: Begin with the problem statement (if available)
- Answer the following **questions**:
  - Which actors are involved?
  - What are the core functions that the system must perform? (From the actor's perspective)
  - What information can the actors access, generate, store, modify, delete, or introduce into the system?
  - Which external changes does the actor need to inform the system about? How often? When?
  - What events should trigger notifications to the actor using the system?
- Decide on names, contextual circumstances if needed, etc.
  - **Know your audience** — familiarity with a situation may be relevant

Today's roadmap

- Scenarios
- Use cases
- User stories

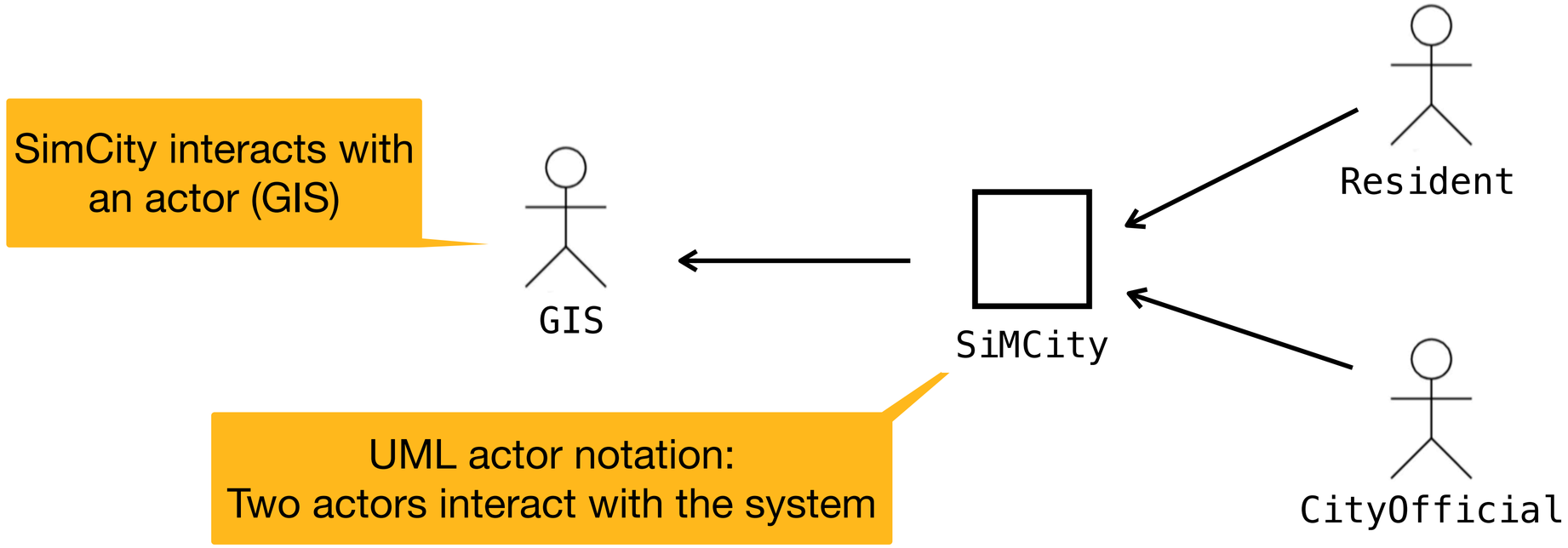
Deriving functionality

- Scenarios focus on actor identification & the application domain
  - Shortcoming: static, does not adapt to different processes that need to be supported by a (new) feature
- Next step: Based on the scenario(s), identify functionality where an actor interacts with the system → **use cases**
  - Describe the different ways how a user can interact with the system
  - Are there any conditions tied to a functionality?
  - Any specific qualities that need to be considered?

Use cases

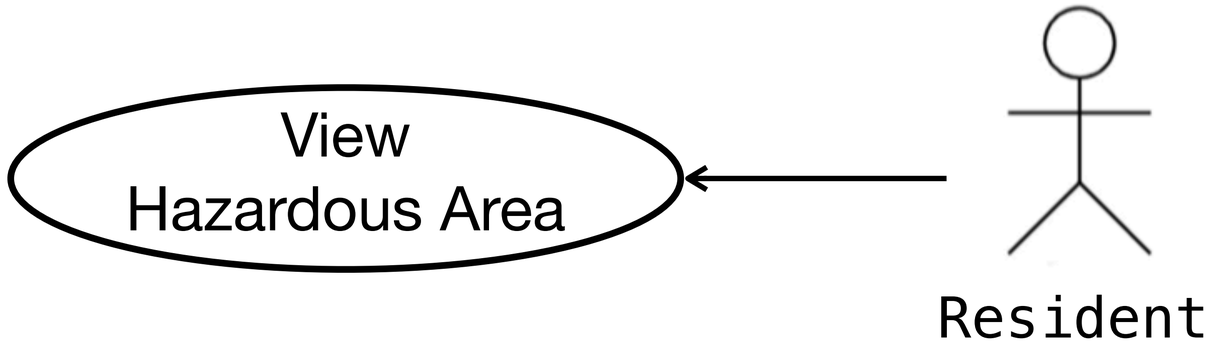
Goal: Define system functions through a series of events, producing observable outcomes for actors

- Use cases are descriptions of interactions between a system and actors (= its users or other systems)
- Actors initiate use cases to access system functionality



Use cases

- Use cases describe the **system's behavior** from the **actor's point of view**
- Typically, functional requirements should be phrased in a way that their respective short title can be directly derived into a use case
  - Example:
    FR1 View Hazardous Area: Pittsburgh's residents can view hazardous areas where sinkholes are likely to occur.
  - Use case - UML representation:



Use cases - textual description

- If a use cases has a flow of events that is non-trivial, a textual description helps to identify and document these events
- Use cases are typically written from the **point of view of the user**

Use case name	Track location of SLFs
Participating actors	Citizens of Pittsburgh (enduser)
Flow of events	1. The citizen select a specific area or region on the map where they suspect or want to track Spotted Lantern Flies (SLFs). 2. The system shows the location of spotted lantern flies in the selected area

An event can also link to another use case. E.g.,
3. The system initiates use case "View location"

System steps (i.e., events that the system initiates) should be highlighted.

Formalized use case template

University of Pittsburgh

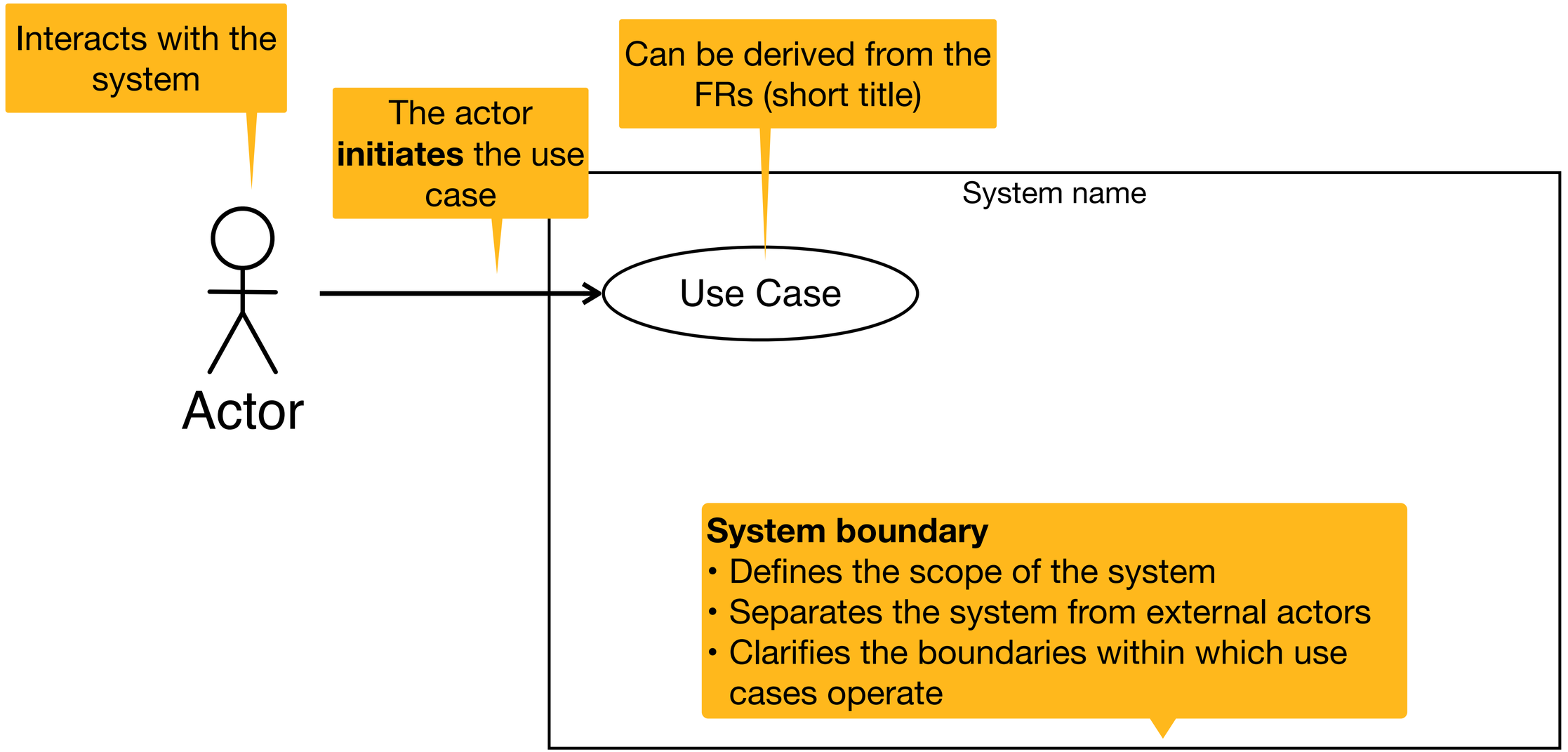
Use case template	Description
Use case name	The name of the use case should indicate what the user is trying to accomplish (See short title of FRs)
Participating actors	Actors should be named with noun phrases (e.g, `CityOfficial`)
Flow of events	The boundary of the system should be made clear. Actor-initiated steps and system-initiated steps should be distinguished Use case steps in the flow of events should be phrased in the active voice
Entry condition	Conditions that need to be satisfied before the use case is initiated
Exit condition	Conditions satisfied after the completion of the use case
Quality requirements	Requirements that are not (directly) related to the functionality of the system

[Example] Textual Use Case

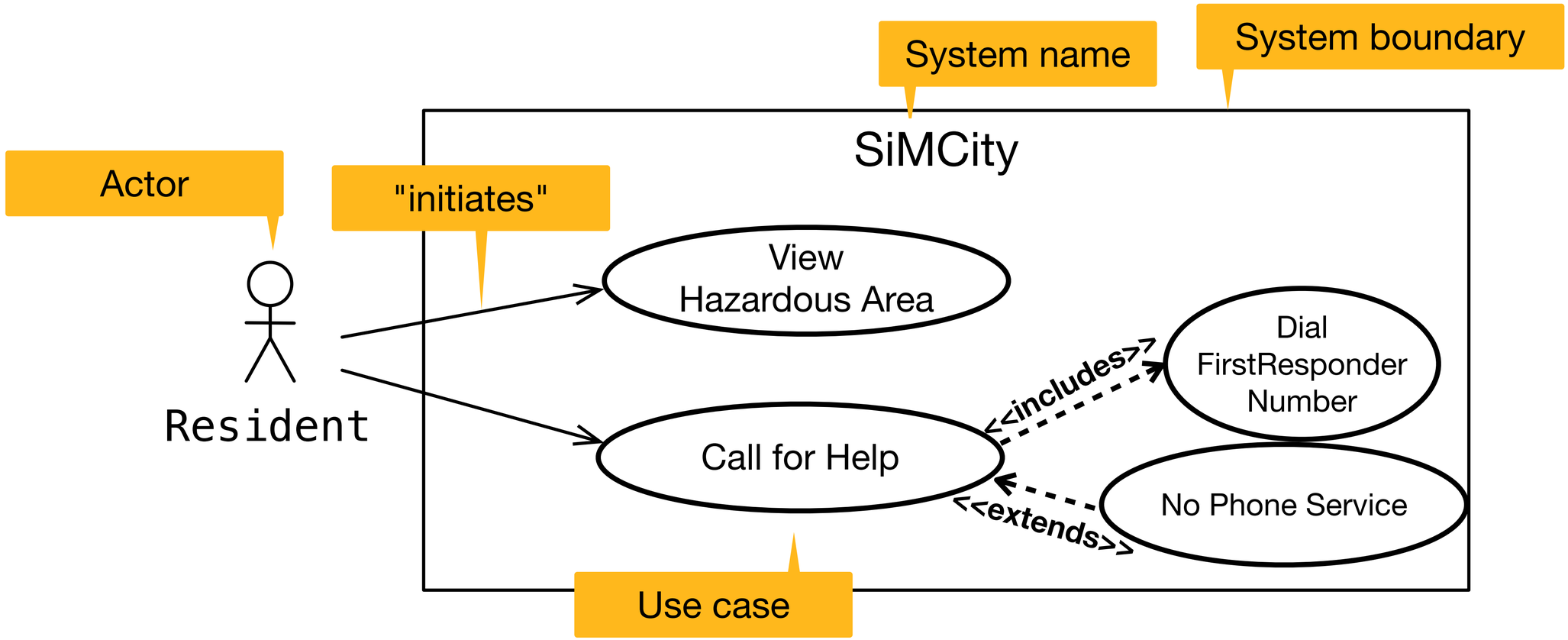
University of Pittsburgh

Use case name	Call for help
Participating actors	Resident, PublicSafety
Flow of events	1. The Resident sees a sinkhole 2. The Resident opens the SiMCity system. 3. The Resident navigates to the sinkhole help overview 4 The System displays available help options. 5a The Resident selects to call for help. 6a The System dials PublicSafety *Alternative path:* 5b The Resident does not have cell phone reception,"No cell phone reception" extension use case is initiated 4a The System shows alternatives to calling for help
Entry condition	Resident has access to SiMCity system
Exit condition	PublicSafecty has been called
Quality requirements	Displayed help options are relevant for resident's area It should take at most 3 clicks to call PublicSafety

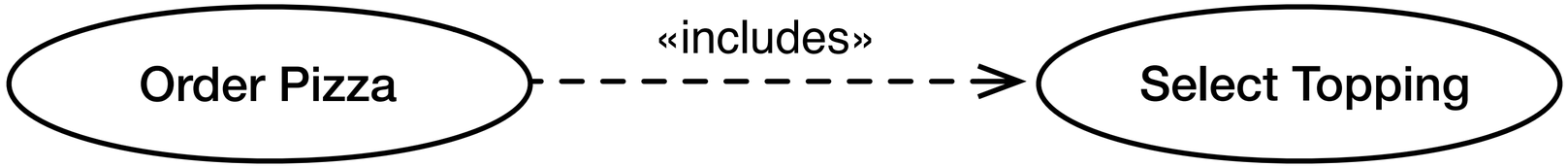
UML Use Case Elements



[Example] UML Use Case Model

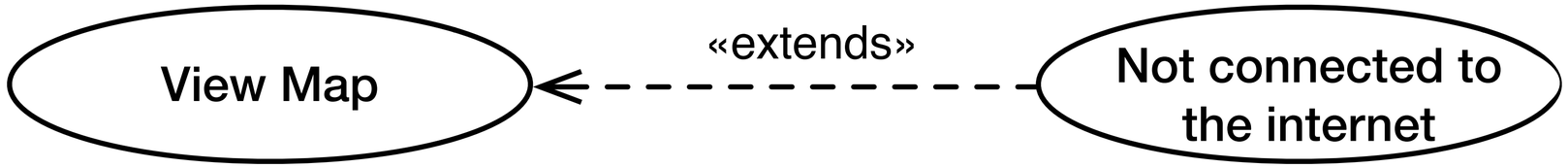


UML use case elements — **<<includes>>**



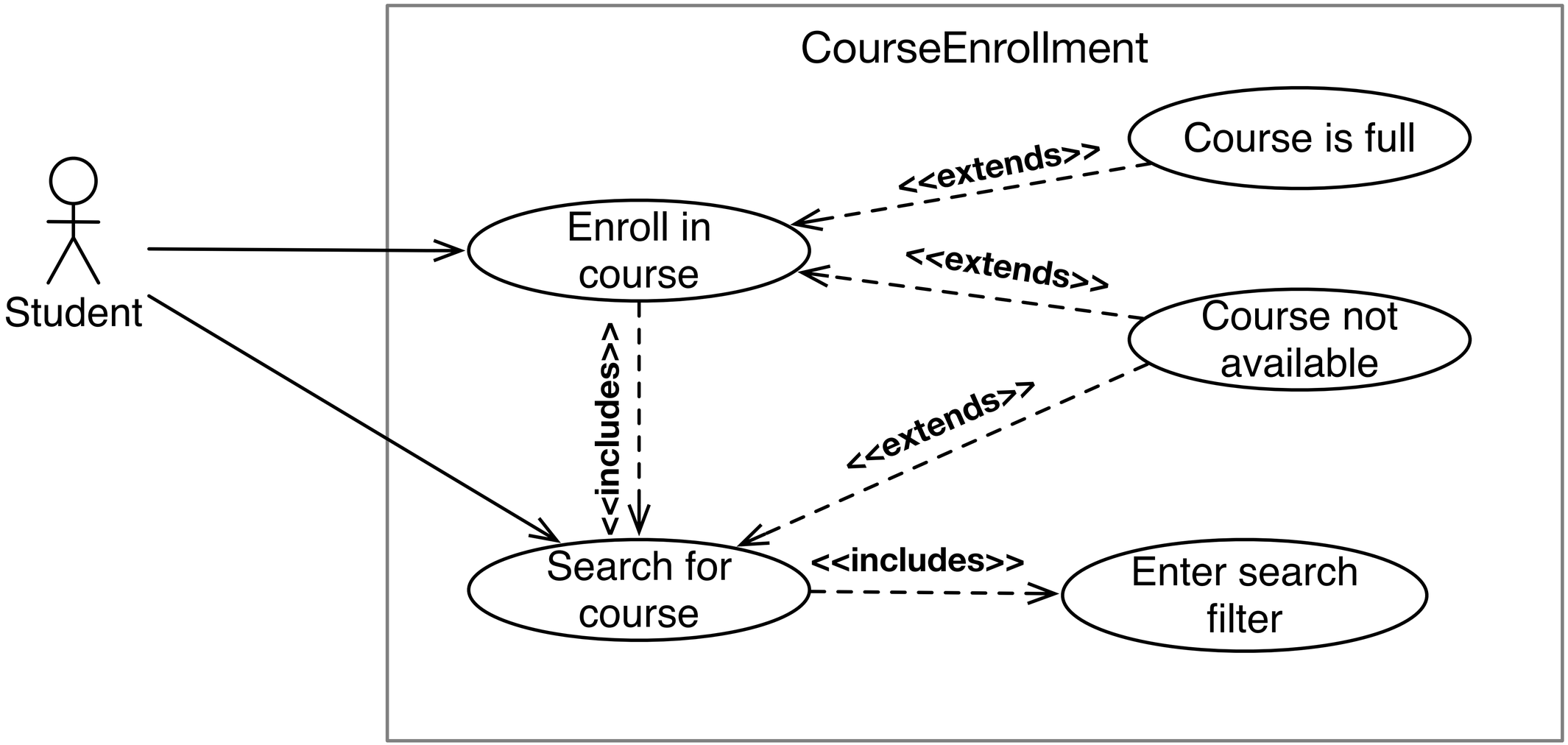
- Goal: Reduce complexity by identifying commonalities in different use cases
  - Incorporates one use case (the included use case) within another (the including use case
  - Ensures that the included use case is always part of the main functionality of the including use case
  - Enhances modularity and reusability by breaking down complex use cases into simpler, reusable components

UML use case elements — **<<extends>>**

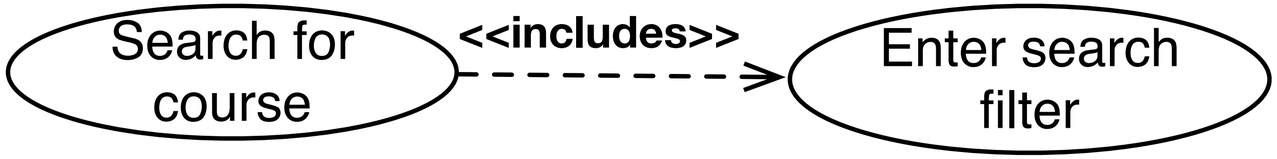


- Goal: Reduce complexity by extending use cases with optional and conditional behavior that occurs within a use case
- Allows for the inclusion of additional functionality, often triggered by **exceptional** or **rare scenarios**
- Enhances flexibility and adaptability in system behavior without cluttering the primary use case with exceptional paths

[Example] UML Use Case Model



[Example] UML Use Case Model



Use case name	Search for course
Participating actors	Student
Flow of events	1. The student selects to search for a course. 2. The system shows a list of courses. 3. The student enters a filter to search for a course. *The use case '*Enter search filter*' is invoked.* 4. The system displays a list of courses that match the student's filter.
Entry condition	The student is logged in.
Exit condition	A list of courses that match the search filter is produced.
Quality requirements	It should take the student at most three clicks to search for a course (NFR5)

Actor-step

System-step

I09 — Use case model

⏳ 10 min

👩🏽‍💻👨🏻‍💻 Pairs

University of Pittsburgh

Based on your functional requirements, model a use case diagram

continued L10!

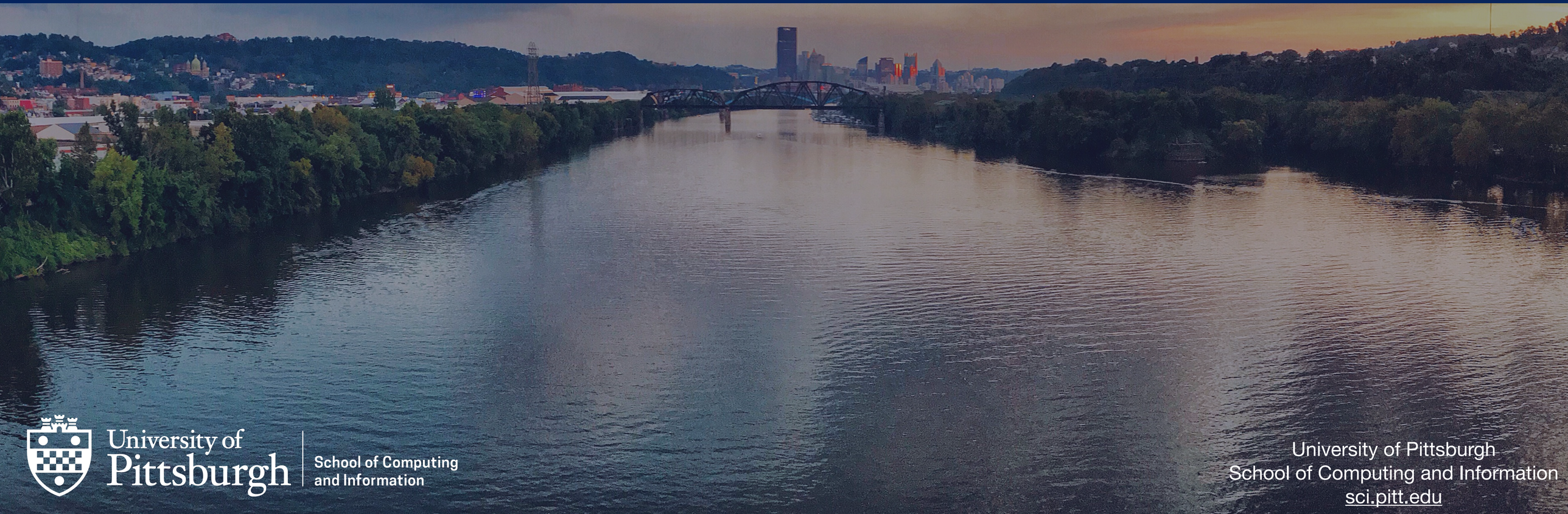
Use a modeling tool, e.g.,
[Commercial] OmniGraffle, Lucidchart, Visual Paradigm, gleek.io
[Free/open-source] draw.io, Gliffy, StarUML, PlantUML

Fall 2025

L09 Scenarios & Use Cases

CS 1530 Software Engineering

Nadine von Frankenberg



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