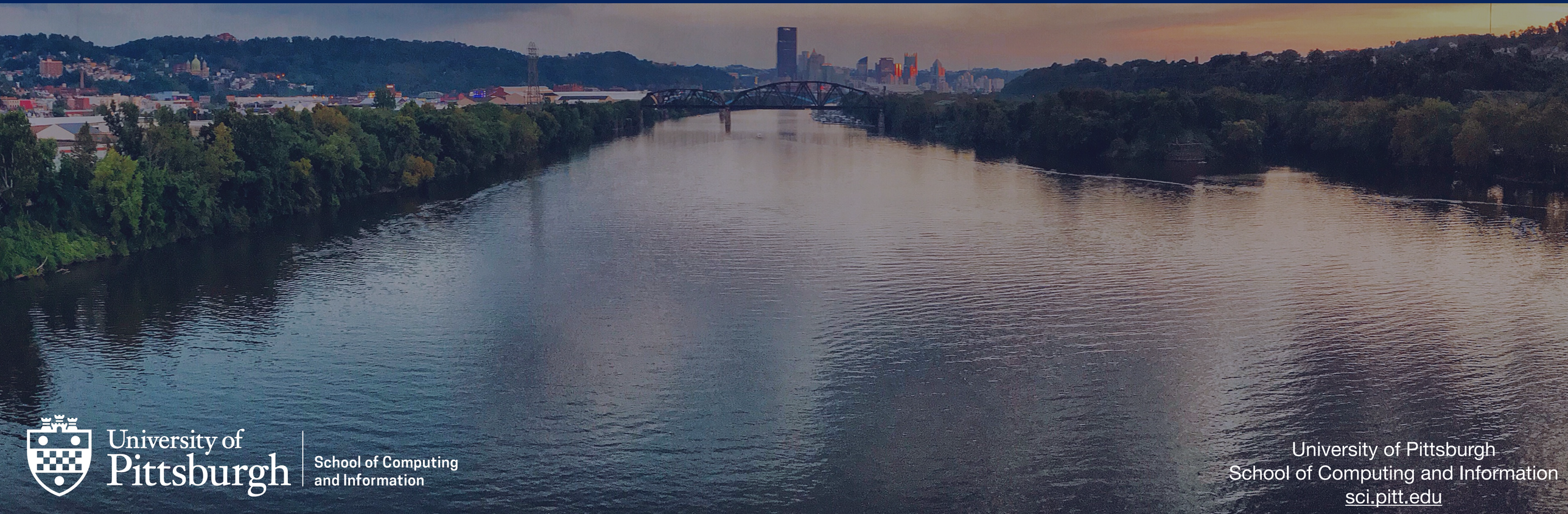
Fall 2025

L13 Intro to System Design

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

- Understand the purpose of system design
- You can formulate design goals
- You understand the purpose of modeling a workflow

Today's roadmap

- Recap: Analysis
  - Analysis object model
  - Modeling access
- System design overview
- Design goals

I12 Pizza ordering system

👀 Review

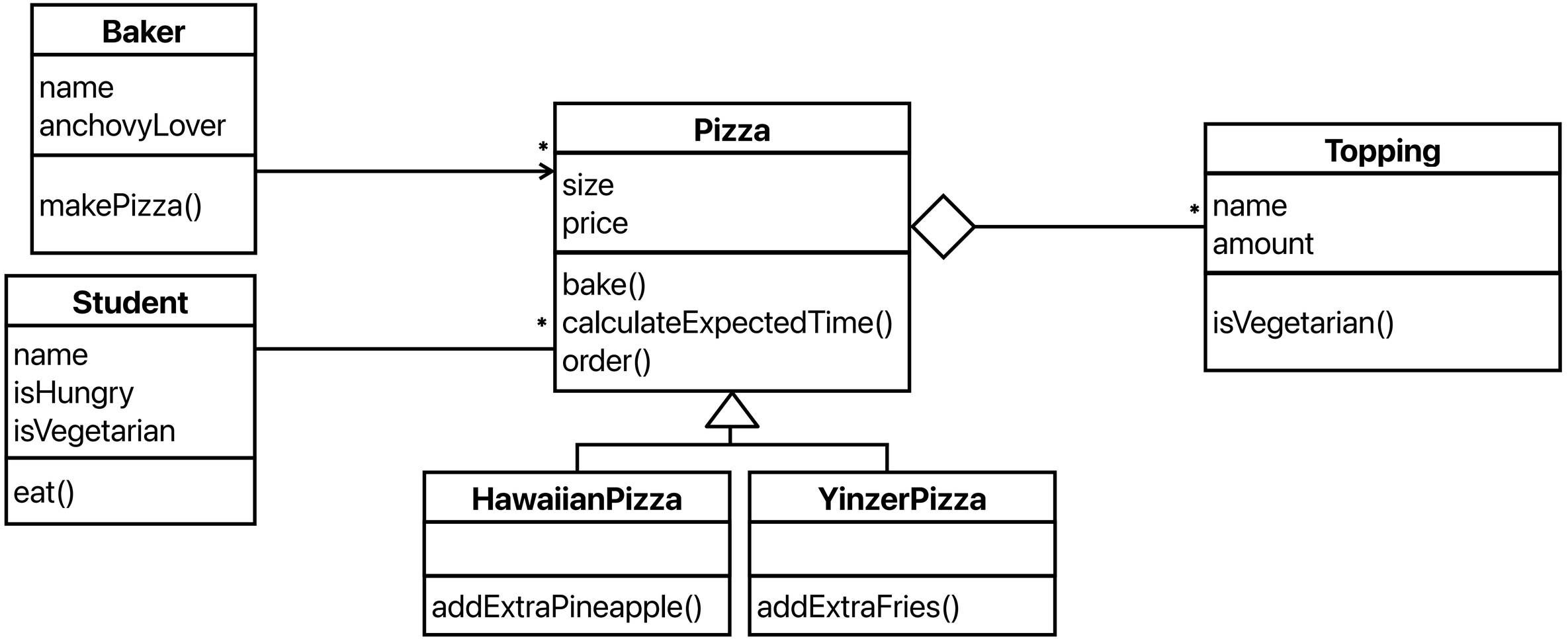
University of Pittsburgh

Problem statement — Design a pizza ordering system

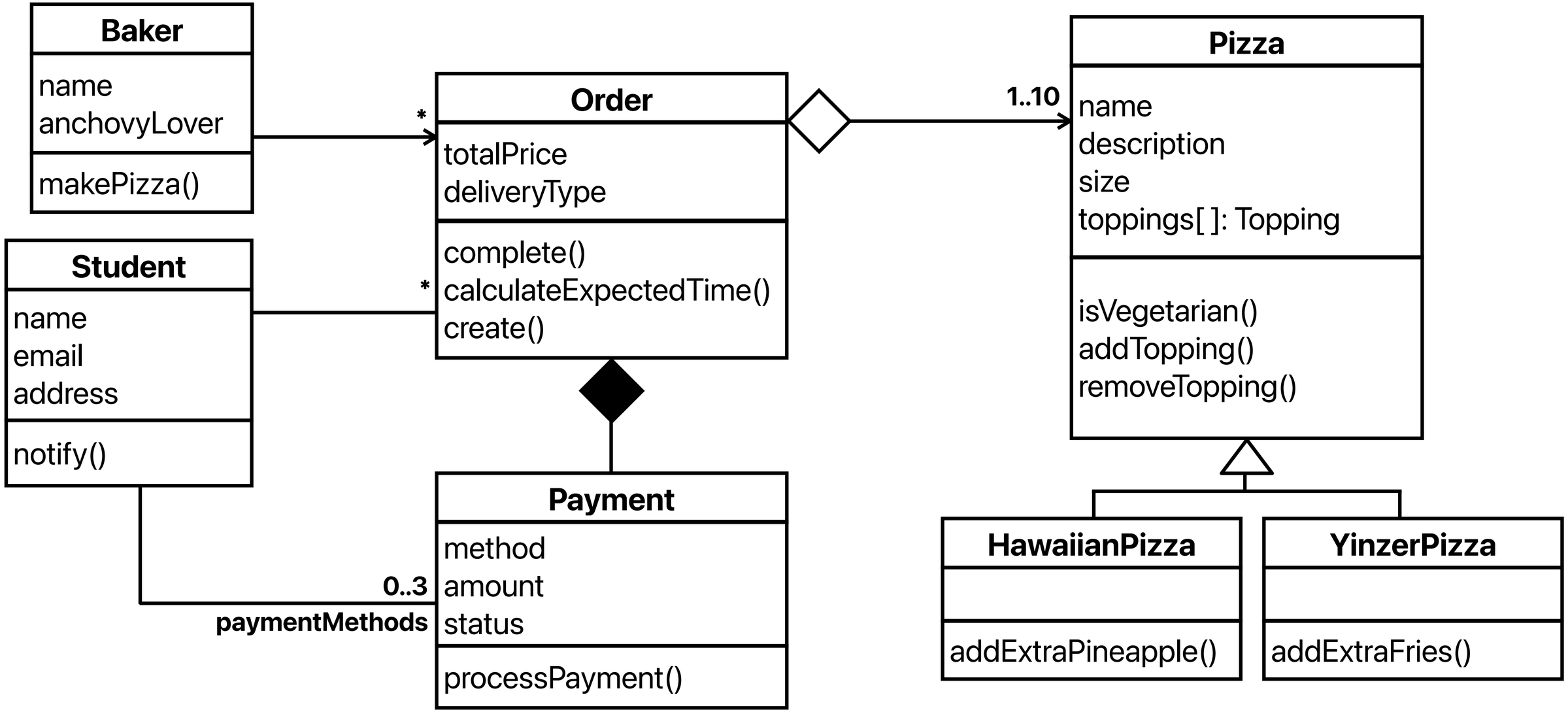
“Students can order pizzas with varying sizes and choose from various toppings. After selecting their pizzas, they confirm the delivery type (pickup or dorm delivery) and pay. Once an order is placed, the pizza baker receives it and confirms the order. Students can then check the expected time until the pizza is ready (for pickup) or until delivery. In addition to regular pizzas (Margherita, Pepperoni, Mushroom), the pizza baker can also prepare two special types of pizzas: Hawaiian or Yinzer."

Your task: Model this system using a UML class diagram.

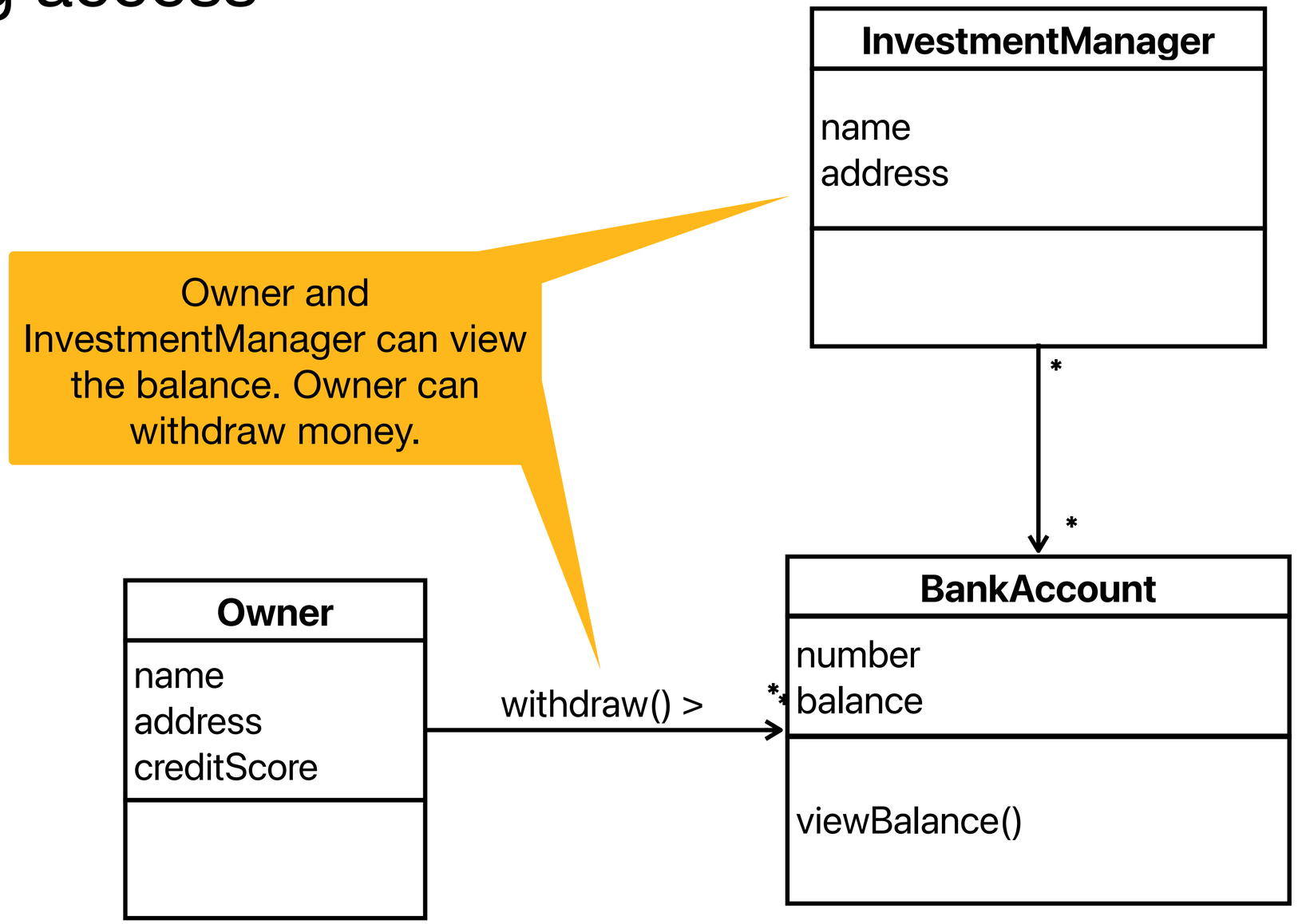
[Example] Pizza ordering system



[Example] Pizza ordering system



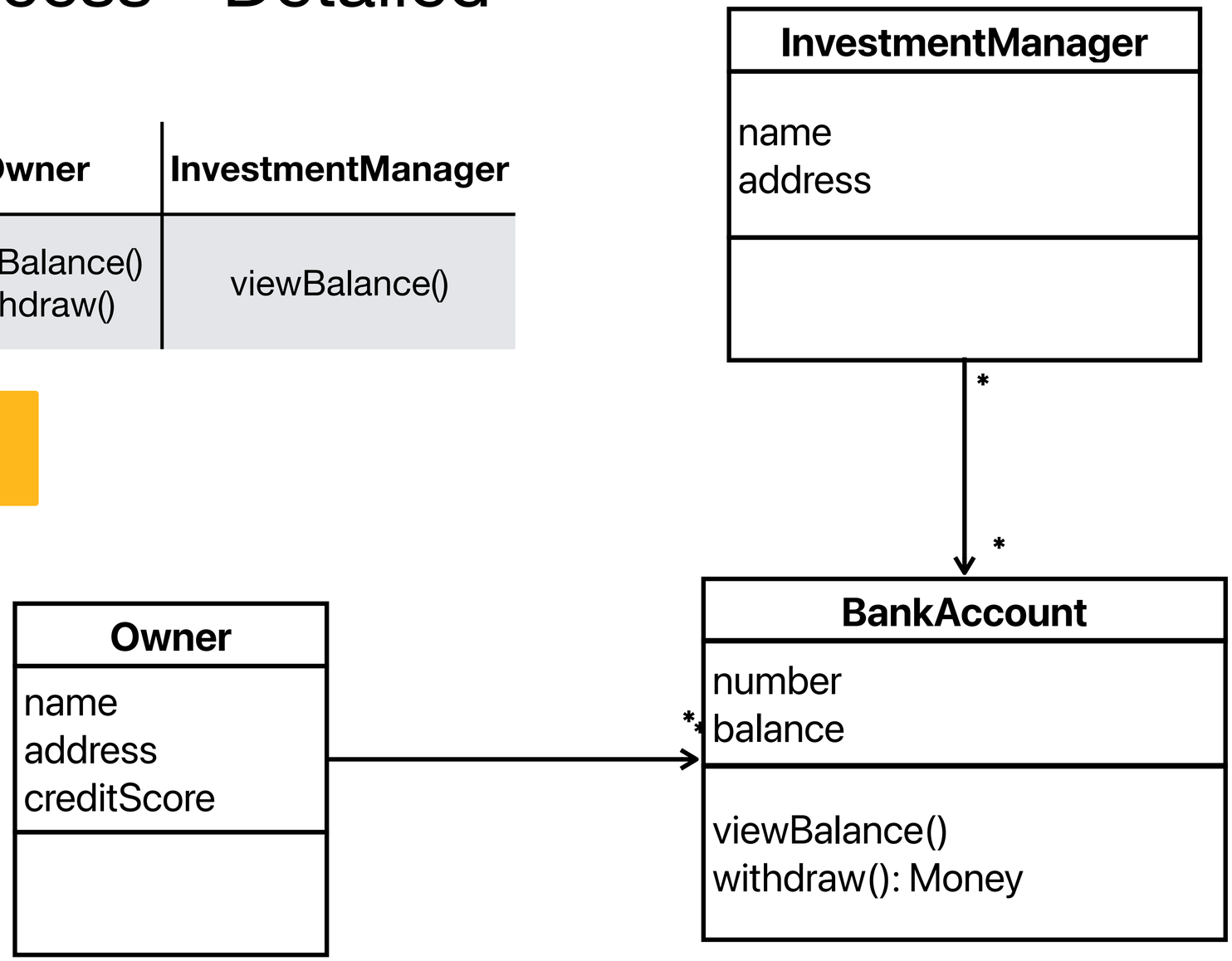
Identifying access



Identifying access - Detailed

	Owner	InvestmentManager
BankAccount	viewBalance() withdraw()	viewBalance()

Access matrix



Access Matrix

- Access control: Defines access permissions for entities and objects
- Permissions: Provides detailed, fine-grained access control
- Security: Forms the basis for system security analysis and policies
- The rows of the matrix represents the actors of the system
- The column represent classes whose access we want to control

	Owner	**InvestmentManager**
BankAccount	viewBalance() withdraw()	viewBalance()
SupportDesk	contact()	

An entry in the access matrix represents an **access right**

Lists the operations that can be executed by the actor on instances of the class

Today's roadmap

- Recap: Analysis
- System design overview
- Design goals

Designing software systems

There are two ways of constructing a software design.
One way is to make it so simple
that there are obviously no deficiencies.
And the other way is to make it so complicated
that there are no obvious deficiencies.

—C.A.R. Hoare

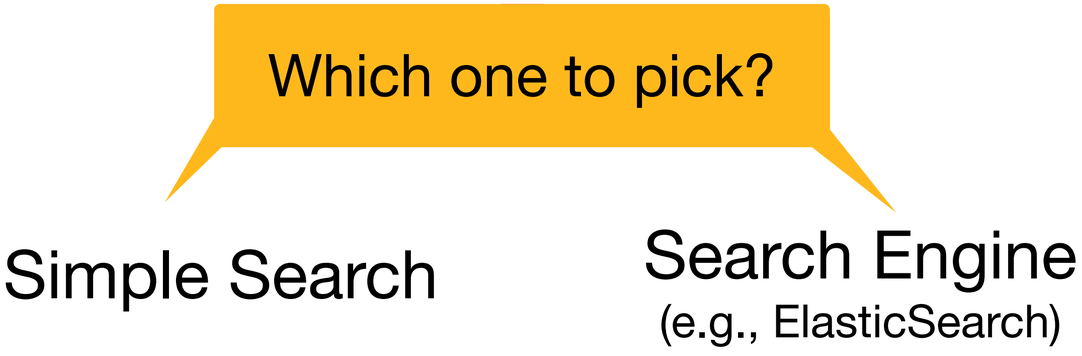
Designing software systems

- Design is a dynamic and creative problem-solving activity
- It is difficult - there is no concrete recipe or one-size-fits-all approach
- It requires adaptability and tailoring to the specific context
- The quality of a system's design and its designer directly impacts the system's performance, longevity, and overall success

System Design

- For each feature, there are many variants to accomplish the desired behavior
  - What are the differences between the variants?
  - Which variant should we choose?

Example FR11 Search for Product: The consumer can search for a product.



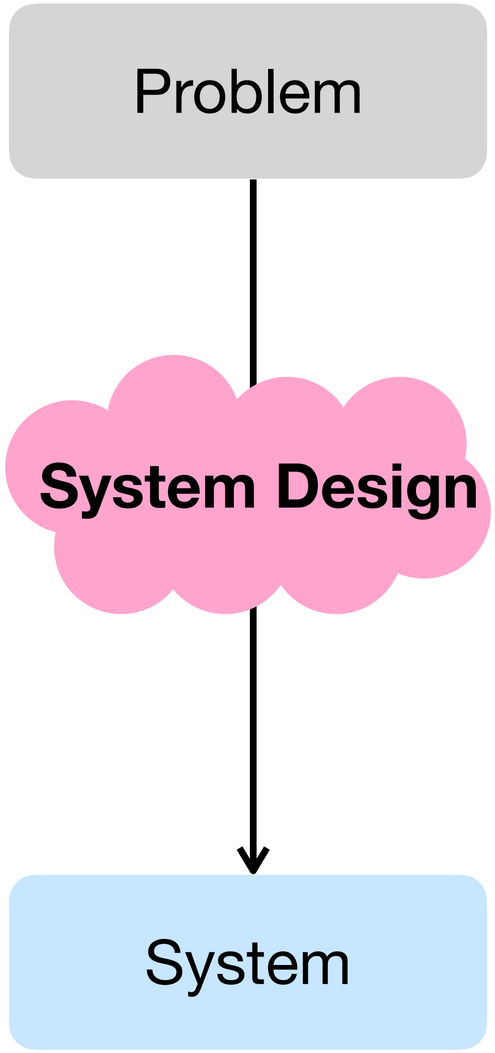
It depends!

on the goals

on the context

How to implement a system?

- System design bridges the gap between requirements analysis and implementation
- Goal: Allocate the requirements to hardware and software components
  - It helps to translate abstract ideas into concrete, organized systems
- Comprises three main steps:
  1. Choose a **high-level strategy for solving-problem**
  2. Decide how to **organize the system** into subsystems
  3. **Map subsystems to** HW and SW **components**
     - Decide on the "tech stack"
     - Decide on "build or buy"
     - ...



Designing Software Systems

University of Pittsburgh

- Object-Oriented Requirements **Analysis**
  - Understand the problem
  - Identify the key concepts and their relationships
  - Build a (visual) vocabulary
  - Create an application domain model (conceptual model)
- Object-Oriented **Design**
  - Assign responsibilities (attributes, methods)
  - Explore behavior with interaction diagrams
  - Explore design alternatives
- **Implementation**
  - Map designs to code, implementing classes and methods

Analysis Object Model

Important system design principles

1. **Separation of concerns**
   - Divide the system into separate aspects or concerns
   - Helps to keep the system organized
2. **Abstraction** hides complex implementation details, simplified interfaces
3. **Modularity** promotes code reusability and simplifies maintenance
4. **Encapsulation** restricts direct access to data (data integrity)
   - Makes changes to the implementation easier without affecting other parts of the system
5. **Consistency** promotes readability, maintainability, integrity
6. **Simplicity** reduces complexity

System design concepts — Overview

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

Today's roadmap

- Recap: Analysis
- System design
- Design goals
  - Types
  - Trade-offs

So far …

- Identified requirements
- Analysis object model (domain model)

- Now, define the concrete messages and behavior of/between objects
- How?
  - How should concepts be implemented by classes?
  - How should the objects interact exactly? Time-frame?

This is a critical, important, and non-trivial task!

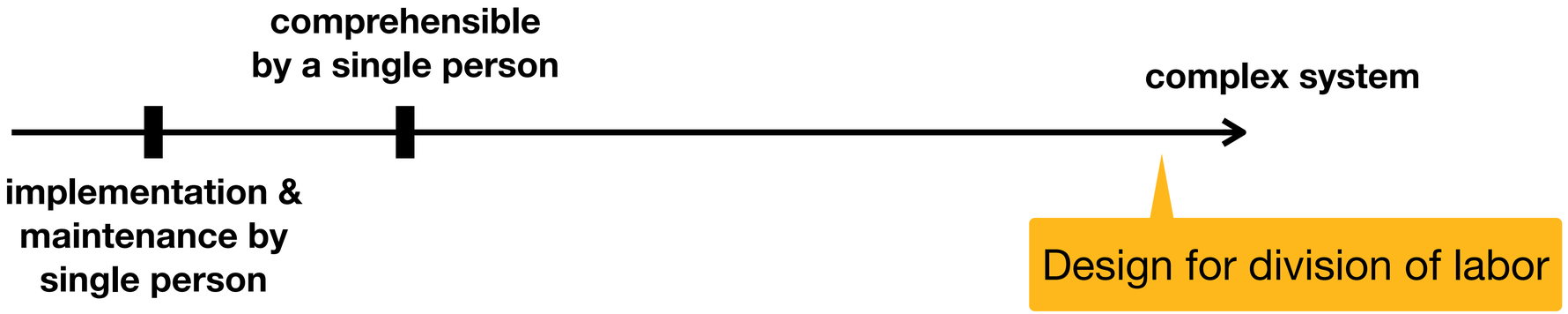
Goal of Software (Software Quality)

- Functional correctness
- Robustness
- Flexibility
- Reusability
- Efficiency
- Scalability
- Security

Designing for change

- *… accept the fact of change as a way of life, rather than an untoward and annoying exception*
  — Brooks, 1974
- *Software that does not change becomes useless over time*
  — Belady and Lehman

- The biggest cost is typically not building the system but evolving the system!
  - Reducing the cost of change is important when designing software systems



Design goals

Design goals identify and describe the **qualities the system should focus on**

- Often phrased as statements a team makes about the quality of experience they would like a system to attain
- Design goals can be **inferred from the non-functional requirements** or from the application domain
  - Design goals are often generalized non-functional requirements
  - Active Elicitation: Some design goals must be actively elicited through conversations with clients or stakeholders; metrics, design methodologies, and implementation goals are valuable sources

Design goals (cont.)

- Typically non-prescriptive and more abstract
- Focus on the concerns of designers, architects, and developers
- Often a prioritization of non-functional requirements: **trade-offs** guide the development
- Design goals describe the overarching objectives for the system's architecture and design

Types of design goals — Customer

- Cost
  - **Development cost** — Cost of developing the (initial) system
  - **Deployment cost** — Cost of installing the system and training the users
  - **Upgrade cost** — Expenses associated with data migration and backward compatibility requirements
  - **Maintenance cost** — Resources needed to handle future bug fixes and enhancements to the system
  - **Administration cost** — Cost required for administering and managing the system

Types of design goals — Customer

- Maintenance
  - **Extensibility** — How easy is it to add functionality or new classes to the system?
  - **Modifiability** — How easy is it to change the functionality of the system?
  - **Adaptability** — How easy is it to port the system to different application domains?
  - **Portability** — How easy is it to port the system to different platforms?
  - **Readability** — How easy is it to understand the system from reading the code?
  - **Traceability of requirements** — How easy is it to map the code to specific requirements?

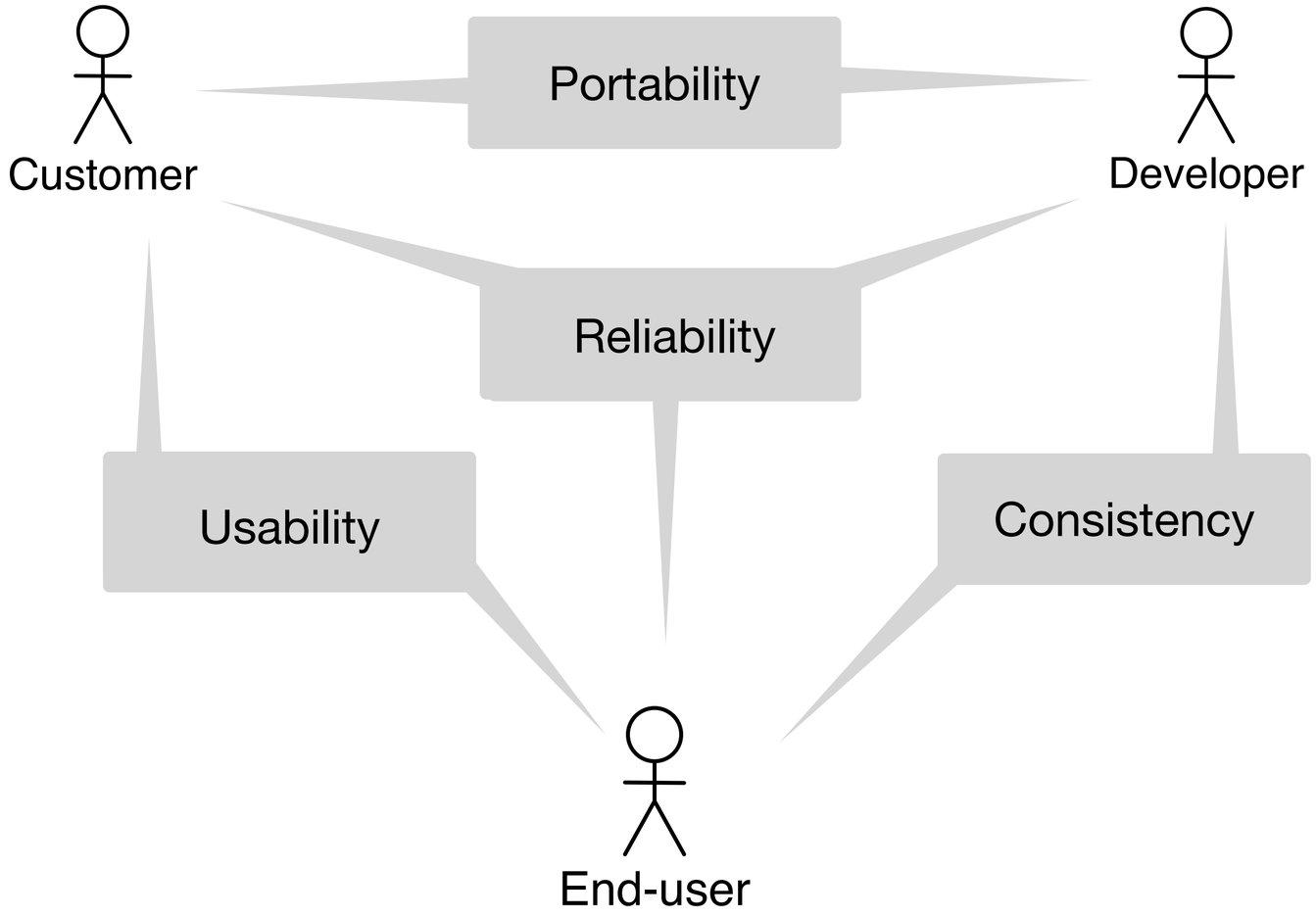
Types of design goals — End-user

- **Utility** — How well does the system support user tasks and goals?
- **Usability** — How easy is it for the user to interact with the system?
- **Cost** — How much does it cost to use the system? (free vs one-time payment vs future costs)

Types of design goals — Developer

- Performance
  - **Response time** — How quickly does the system respond after receiving a user requests?
  - **Throughput** — How many tasks can the system accomplish within a fixed time frame?
  - **Memory** — How much space is necessary for the system to operate efficiently?
- Dependability
  - **Robustness** — Ability to handle invalid user input
  - **Reliability** — Difference between specified and actual behavior
  - **Availability** — Percentage of time the system is available for normal tasks without downtime
  - **Fault tolerance** — Ability to continue operating in the presence of errors or failures
  - **Security** — Ability to protect against malicious attacks, unauthorized access, data breaches
  - **Safety** — System's ability to prevent endangering human lives, even in the presence of errors and failures

Design goals interests overlap (Example)



Requirements vs design goals

Requirements define **what the software system must do**

- "What does the system need to accomplish in terms of functionality, features, and constraints?"
- Typically detailed and specific, providing clear instructions on how the system should behave
- Focus on addressing the needs and expectations of stakeholders, including end-users, clients, and regulators
- Early stage definition (requirements elicitation, analysis)

Design goals identify and describe the **qualities the system should focus on**

- What should be prioritized?
- What is the overarching goal of the entire system?

[Example]

- Functional Requirements *simplified
  - **FR1 Access Lab Result:** Patients should be able to securely access their lab test results.
  - **FR2 Book Appointment:** Patients should be able to book an appointment with their healthcare provider.
  - **FR3 Authenticate:** Patients and doctors must authenticate themselves to protect patient data.
- Non-functional Requirements *simplified
  - **NFR 1 Security:** Ensure patient data privacy and compliance with healthcare data regulations (HIPAA).
  - **NFR 2 Usability:** Design an intuitive and user-friendly interface for patients of all technical skill levels, so that they can navigate it without needing a manual.

[Example]

Design Goals

- **DG1 Usability and User Experience:** Ensure the app is easy to navigate, with a clean and intuitive user interface to provide a positive user experience for patients, reducing the learning curve and frustration.
- **DG2 Security and Privacy:** Implement robust security measures to protect patient data to maintain patient trust and comply with healthcare data privacy regulations.
- **DG3 Scalability:** Design a scalable architecture to handle increased user loads to accommodate the number of patients using the system without performance degradation.

Designing Software Systems

University of Pittsburgh

- Object-Oriented Requirements **Analysis**
  - Understand the problem
  - Identify the key concepts and their relationships
  - Build a (visual) vocabulary
  - Create an application domain model (conceptual model)
- Object-Oriented **Design**
  - Assign responsibilities (attributes, methods)
  - Explore behavior with interaction diagrams
  - Explore design alternatives
- **Implementation**
  - Map designs to code, implementing classes and methods

Design vs Architecture

University of Pittsburgh

Architecture questions

- How does the user choose to view a map?
- How should the user's location data be protected?
- What other functionalities are connected to viewing a map?
- What are the interfaces between objects?

Design questions

- How do I include the Google Maps API?
- How does the device's built-in encryption work?
- What threads exist and how do they coordinate?
- What are the interfaces between subsystems and third-party systems?

Architecture defines what the system's major components are and how they interact.
Design defines how each component is implemented and behaves internally.

Identifying design goals

- **Analyze requirements** — identify implicit design goals within these requirements
- **Stakeholder input** — insights and expectations regarding system quality and performance
- **Benchmark existing systems** — industry standards and best practices
- **Risk assessment** — potential risks and challenges associated with the project

Identifying design goals (cont.)

University of Pittsburgh

- **Technical feasibility** — design goals need to align with the chosen technology stack and infrastructure
- Prioritize and refine design goals based on their importance to stakeholders: specific, measurable, achievable, relevant, and time-bound (**SMART**)
- **Iterate and collaborate**
  - Involve **cross-functional teams** in goal derivation and validation
  - **Iterate on design goals** as the project progresses and new insights emerge

SMART

- Specific
- Measurable
- Achievable
- Relevant
- Time-bound

Design goal trade-offs

- Design goals are often conflicting
- Typical trade-offs
  - Functionality vs. usability
  - Cost vs. robustness
  - Efficiency vs. portability
  - Rapid development vs. functionality
  - Cost vs. reusability
  - Backward compatibility vs. readability

[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: Any user (regardless of skill level) can navigate through the system and recall the main features (...) of the system after following a tutorial.
- DG2 Reliability: The system should ensure data integrity by accurately reflecting sinkhole status updates in a timely manner.
- 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.

Take Away: Designing good systems is hard!

University of Pittsburgh

- Typically, systems are divided into smaller, manageable pieces to deal with complexity
- Design goals guide decisions made by developers
- Design goals encompass the entire system to be able to realize the different parts independently
- Trade-offs between design goals need to be addressed
  - E.g., "a user can navigate to every functionality using at most 2 clicks from the home page" vs. "a user must be able to use the system without needing a manual"

References

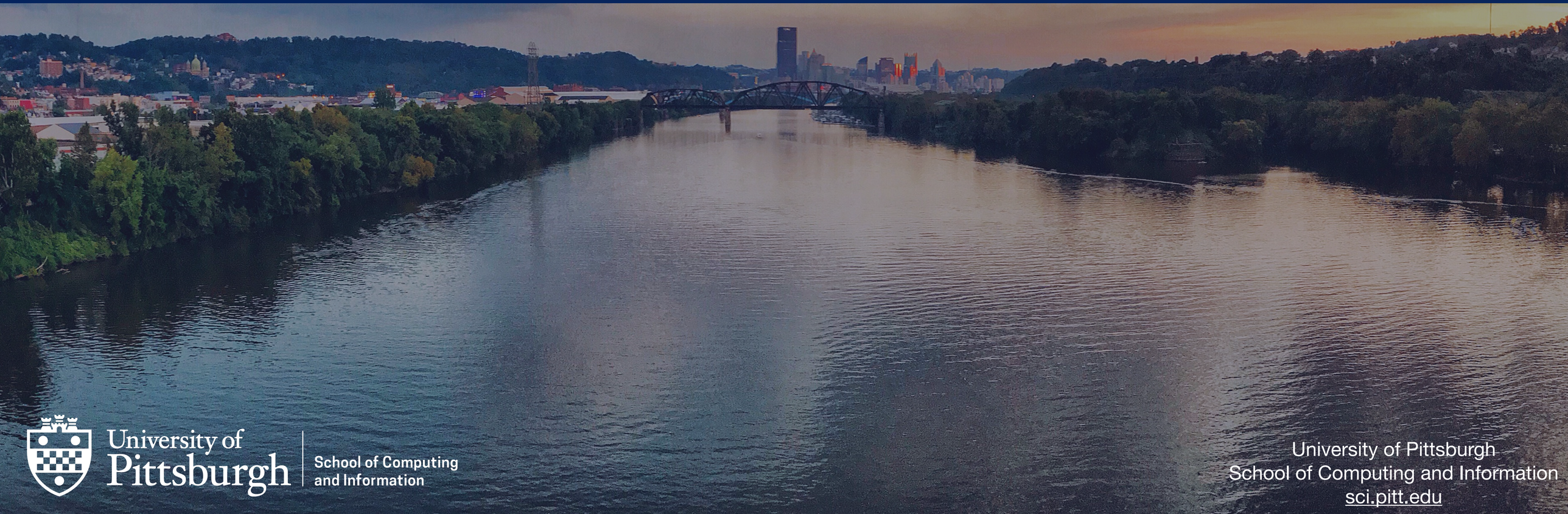
- Bruegge, & Dutoit. Object–oriented software engineering. using UML, patterns, and Java. Pearson, 2009.
- Object Management Group. Unified Modeling Language. Version 2.5.1, 2017

Fall 2025

L13 Intro to System Design

CS 1530 Software Engineering

Nadine von Frankenberg



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