Design Abstractions

17-423/723 Designing Large-Scale Software Systems

Lecture 3
Jan 24, 2024

Learning Goals

- Describe the role of abstraction in communicating designs
- Describe common types of design abstractions
- Select and apply a suitable notation to model an aspect of a design
- Generate questions to explore further design decisions given a model

Logistics

- Homework 1 is out on Canvas; due next Wednesday, Jan 31
- Submission through Gradescope

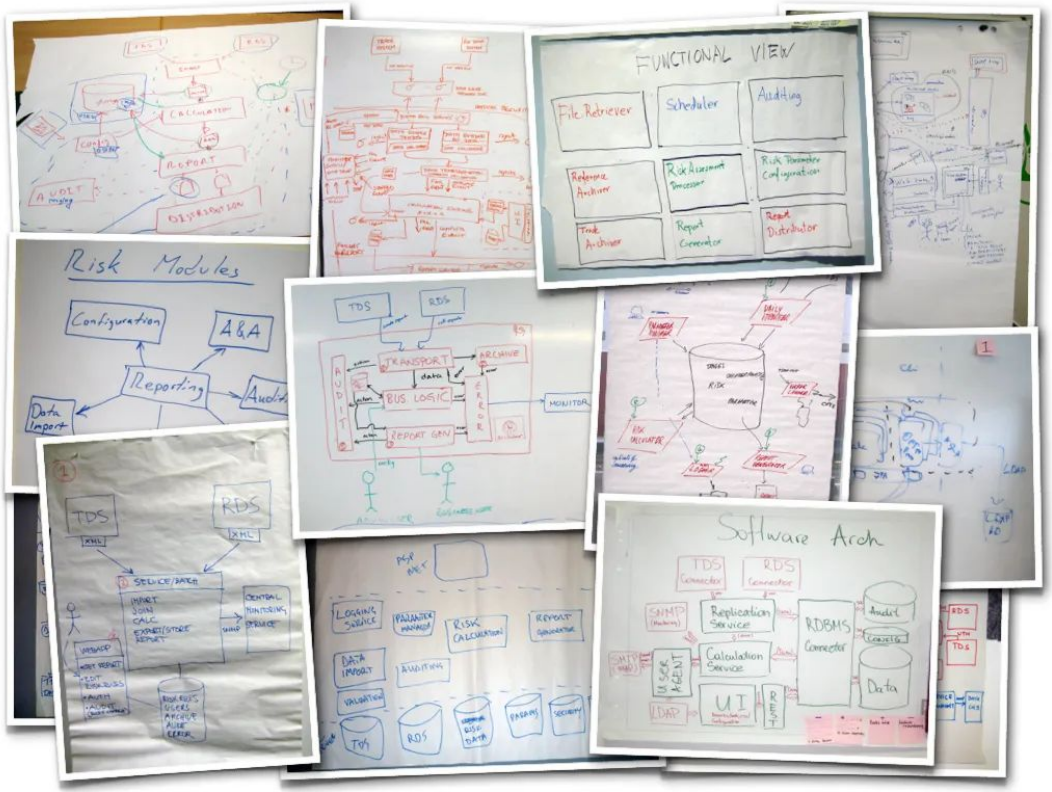
Communicating a Design

Last class: Use of a context model to understand the **problem space** (entities & assumptions)

This class: Models for communicating design ideas in the **solution space**

How do software designers communicate ideas?

Designers use sketches

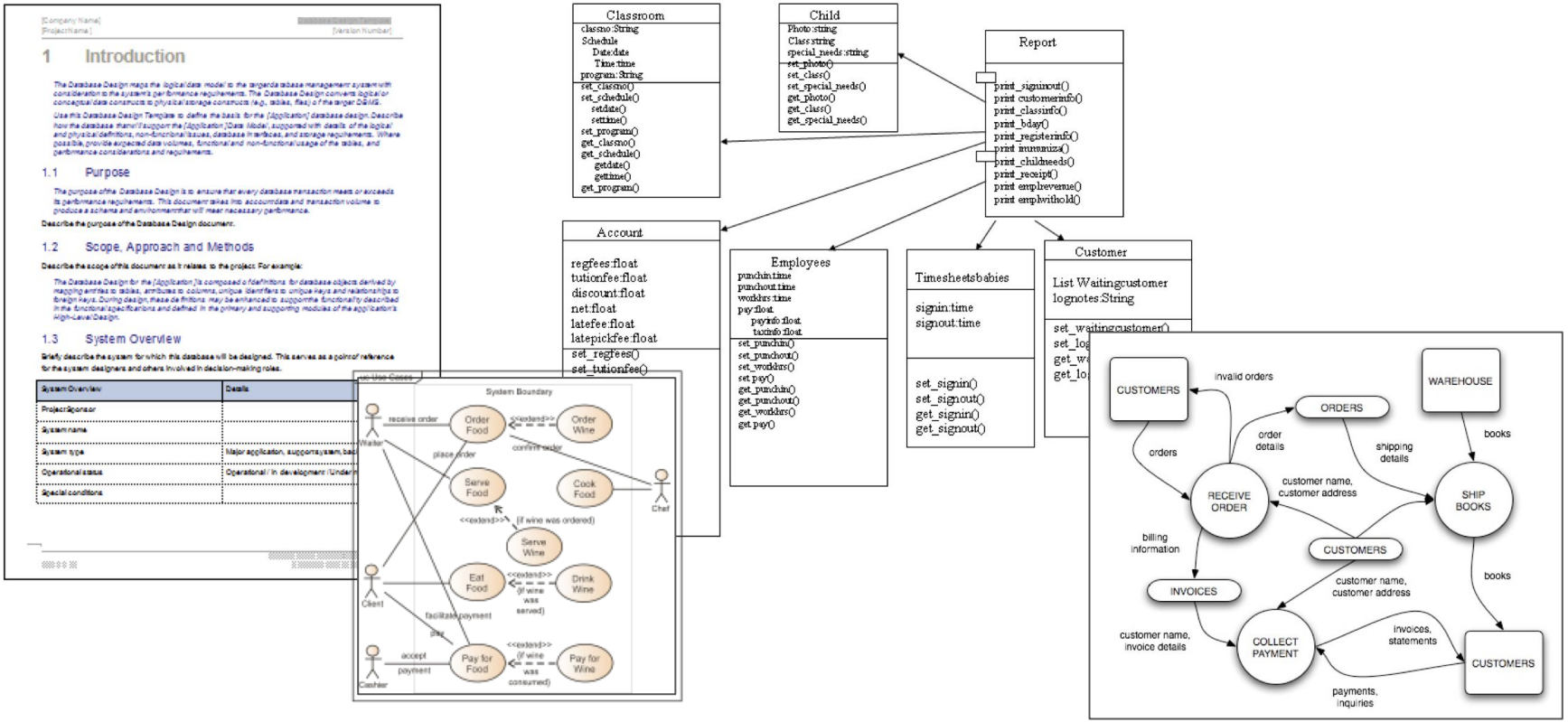


To brainstorm ideas

To explain how a design works

To ask questions about designs

Designers document designs



Viewpoint: Code as Design

Code is the final design and the only source of the truth

Source code listing is the only design documentation that we need

Q. What do you think? Arguments for & against?

Jack W. Reeves “Code as Design: Three Essays”
https://www.developerdotstar.com/mag/articles/reeves_design_main.html

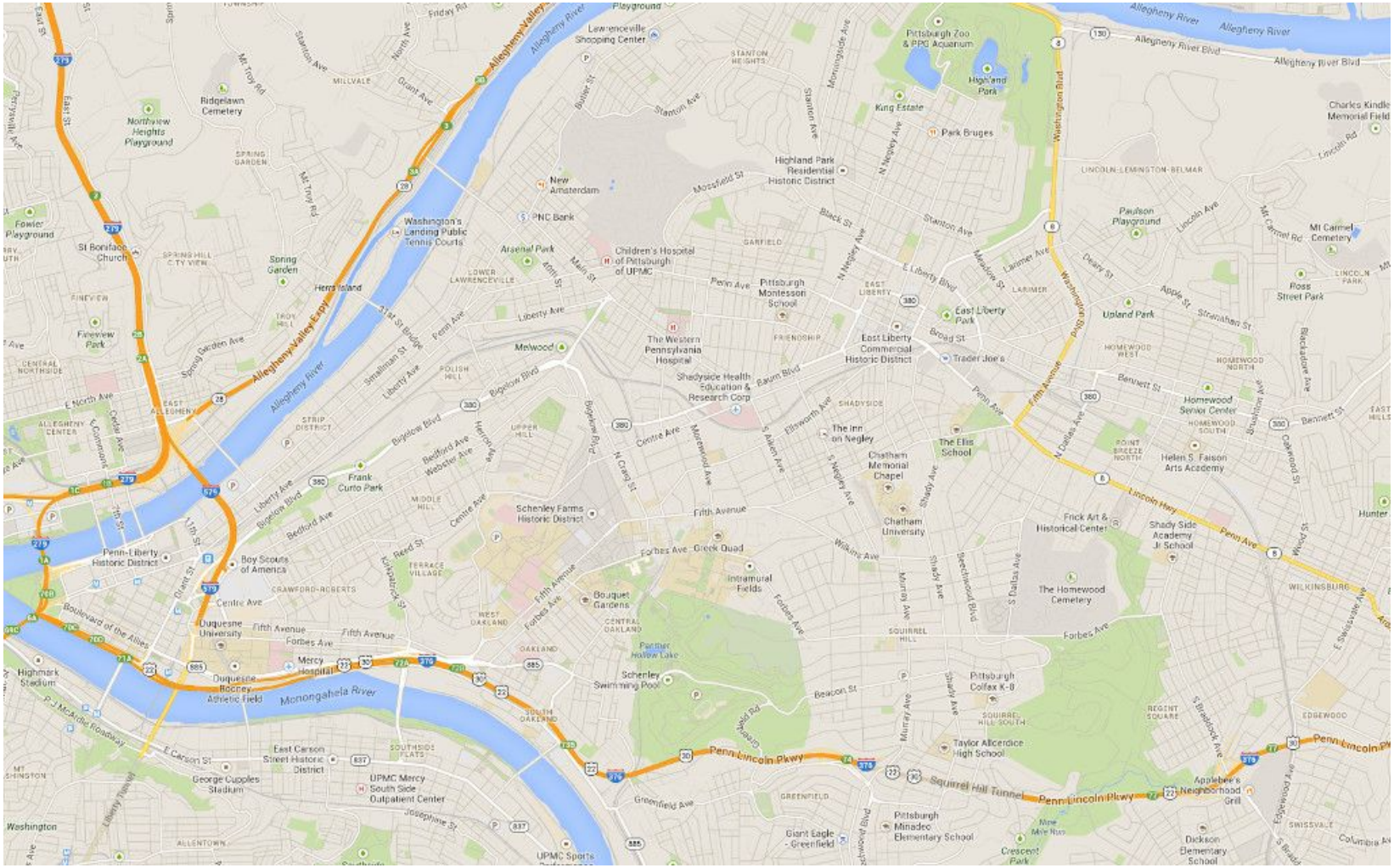
Designers use abstractions

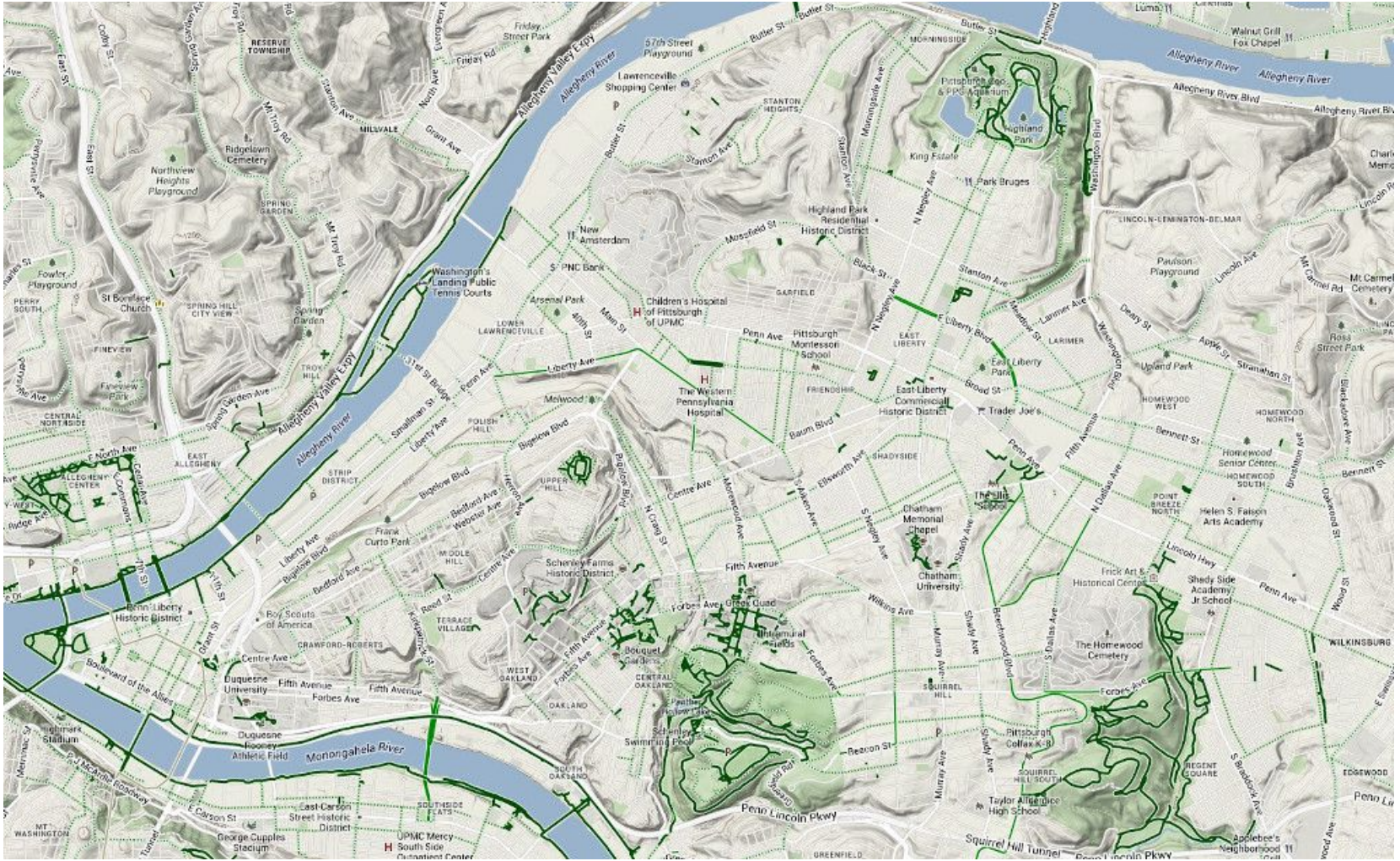
For most systems, code is too complex for a single person to understand

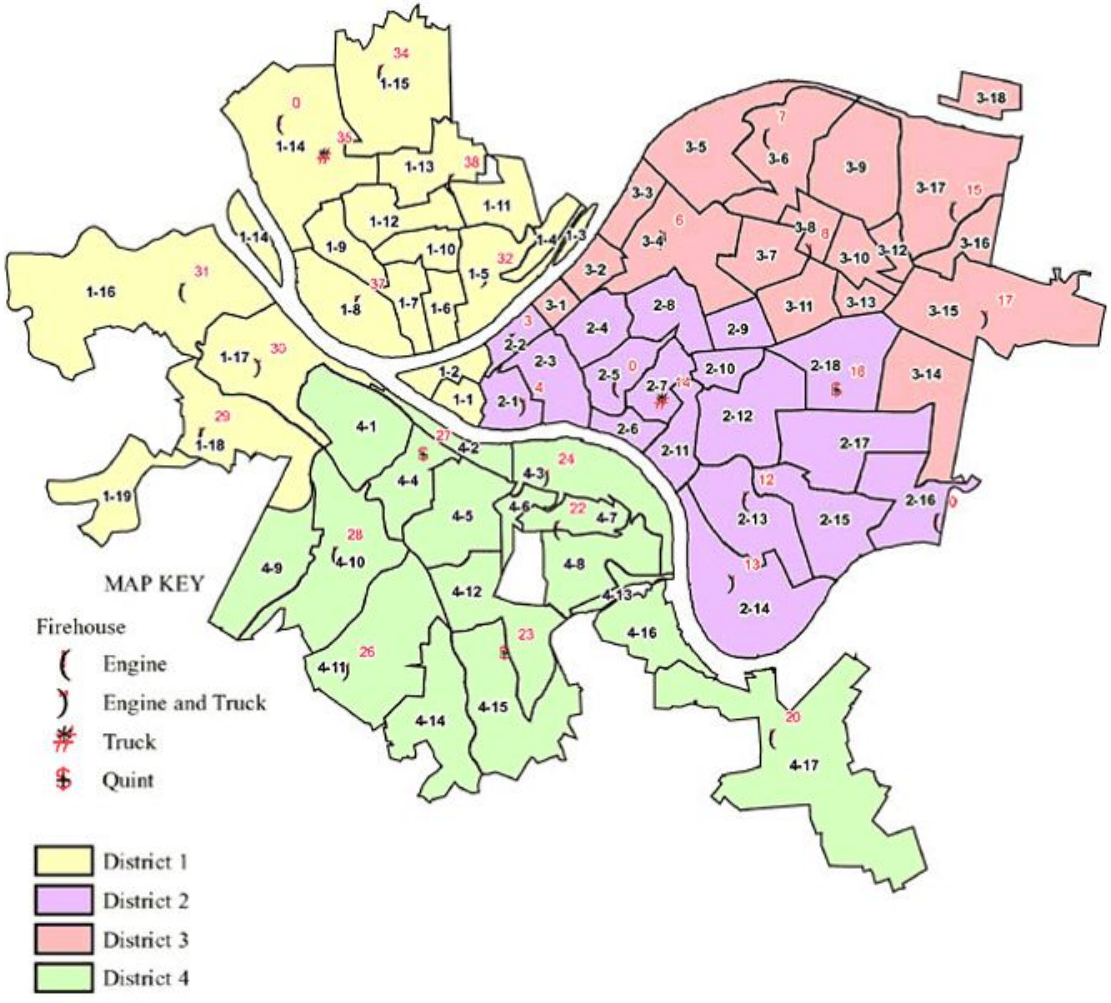
Code is not an ideal medium of communicating ideas

Code does not capture everything about design
- Q. What are some information that is not captured in the code?

Instead, designers communicate through **an abstraction of a system** – a description that focuses on a particular aspect & ignores other details


Allegheny River
Monongahela River
Penn-Liberty Historic District
Schenley Farms Historic District
Penn Lincoln Pkwy
Squirrel Hill Tunnel
Penn Avenue
Fifth Avenue
Forbes Ave
Centre Ave
Liberty Ave
Bigelow Blvd
Boulevard of the Allies
Frank Curto Park
Schenley Park
Schenley Swimming Pool
Bouquet Gardens
Intramural Fields
Greek Quad
Chatham University
Chatham Memorial Chapel
The Inn on Negley
The Ellis School
Frick Art & Historical Center
Shady Side Academy Jr School
The Homewood Cemetery
Helen S. Faison Arts Academy
Homewood Senior Center
East Liberty Commercial Historic District
East Liberty Park
Trader Joe's
Pittsburgh Montessori School
The Western Pennsylvania Hospital
Shadyside Health Education & Research Corp
Children's Hospital of Pittsburgh of UPMC
Highland Park Residential Historic District
Highland Park
Pittsburgh Zoo & PPG Aquarium
King Estate
Park Bruges
Lawrenceville Shopping Center
PNC Bank
New Amsterdam
Washington's Landing Public Tennis Courts
Arsenal Park
Melwood
Ridgelawn Cemetery
Northview Heights Playground
Fowler Playground
St Boniface Church
Spring Garden
Herrs Island
Fairview Park
Paulson Playground
Mt Carmel Cemetery
Ross Street Park
Upland Park
Charles Kindle Memorial Field
Duquesne University
Mercy Hospital
Duquesne Boccey Athletic Field
Highmark Stadium
George Cupples Stadium
East Carson Street Historic District
UPMC Mercy South Side Outpatient Center
UPMC Sports
Giant Eagle - Greenfield
Pittsburgh Minadeo Elementary School
Pittsburgh Colfax K-8
Taylor Allderdice High School
Applebee's Neighborhood Grill
Dickson Elementary School
Crescent Park
Boy Scouts of America
Penn Lincoln Pkwy


Allegheny River
Monongahela River
Allegheny Valley Expy
Penn Lincoln Pkwy
Squirrel Hill Tunnel
Butler St
Liberty Ave
Penn Ave
Fifth Avenue
Forbes Ave
Centre Ave
Bigelow Blvd
Baum Blvd
Penn Liberty Historic District
Schenley Farms Historic District
Highland Park Residential Historic District
East Liberty Commercial Historic District
Children's Hospital of Pittsburgh of UPMC
The Western Pennsylvania Hospital
Pittsburgh Montessori School
Pittsburgh Zoo & PPG Aquarium
Highland Park
Lawrenceville Shopping Center
Washington's Landing Public Tennis Courts
Frank Curto Park
Schenley Swimming Pool
Duquesne University
Duquesne Rooney Athletic Field
Heinz Field
George Cupples Stadium
East Carson Street Historic District
The Homewood Cemetery
Chatham University
Chatham Memorial Chapel
Shady Side Academy Jr School
Frick Art & Historical Center
Taylor Allderdice High School
Pittsburgh Colfax K-8
Trader Joe's
Applebee's Neighborhood Grill
Ridgelawn Cemetery
Northview Heights Playground
Fowler Playground
Paulson Playground
Upland Park
East Liberty Park
Rose Street Park
Homewood Senior Center
Helen S. Faison Arts Academy
Walnut Grill Fox Chapel
Allegheny River Blvd
Washington Blvd
Lincoln Ave
Mt Carmel Cemetery
Mt Troy Rd
Grant Ave
Millvale
Reserve Township
Lincoln-Lemington-Belmar
Morningside
Stanton Heights
Garfield
East Liberty
Friendship
Shadyside
Point Breeze North
Homewood West
Homewood North
Homewood South
Squirrel Hill
Regent Square
Edgewood
Wilkinsburg
Greenfield
Mt Washington
Oakland
Central Oakland
West Oakland
South Oakland
Terrace Village
Crawford-Roberts
Middle Hill
Upper Hill
Polish Hill
Strip District
Central Northside
East Allegheny
Spring Hill City View
Troy Hill
Perry South
Lower Lawrenceville
Stanton Ave
Black St
Penn Ave
Liberty Ave
Centre Ave
Fifth Avenue
Forbes Ave
Wilkins Ave
Beacon St
Penn Lincoln Pkwy
Squirrel Hill Tunnel


MAP KEY
Firehouse
Engine
Engine and Truck
Truck
Quint
District 1
District 2
District 3
District 4

Purpose of an Abstraction

Abstractions aid in understanding

Each abstraction highlights particular aspects of a system and deliberately hides other details

Abstractions facilitate reasoning

Each abstraction encourages certain types of questions about the system

Abstractions are reusable

Each abstraction captures a commonality across multiple systems (within a single domain or sometimes across multiple domains)

What can we reason about?

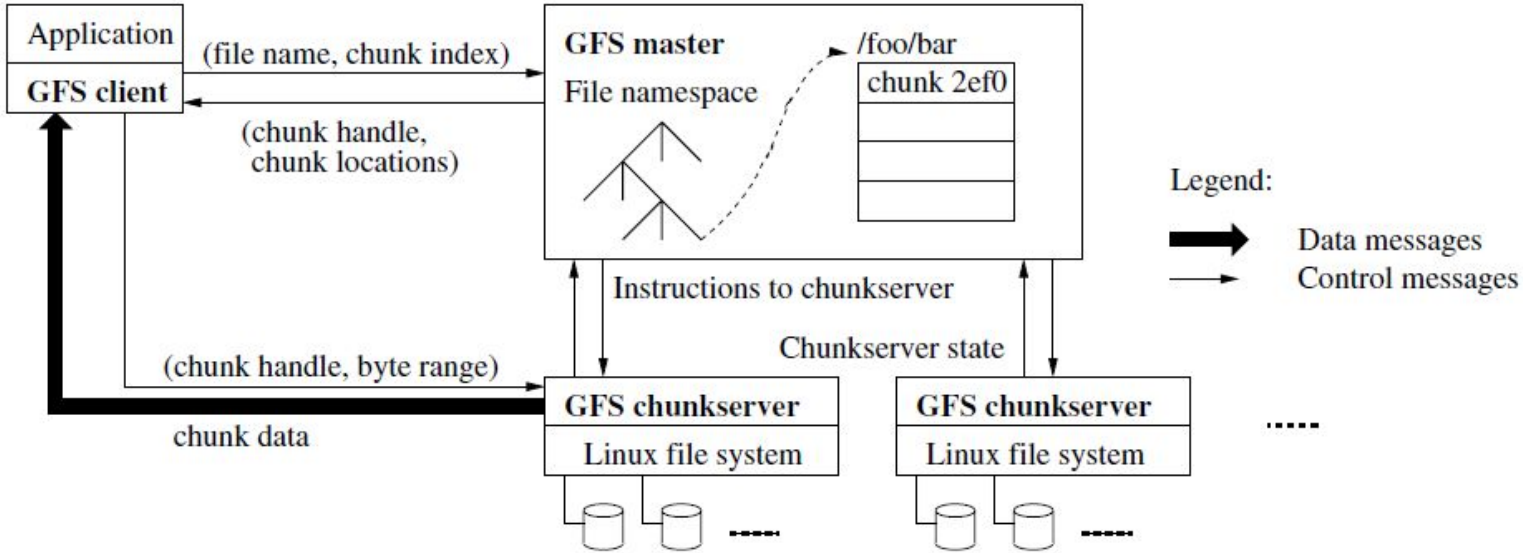
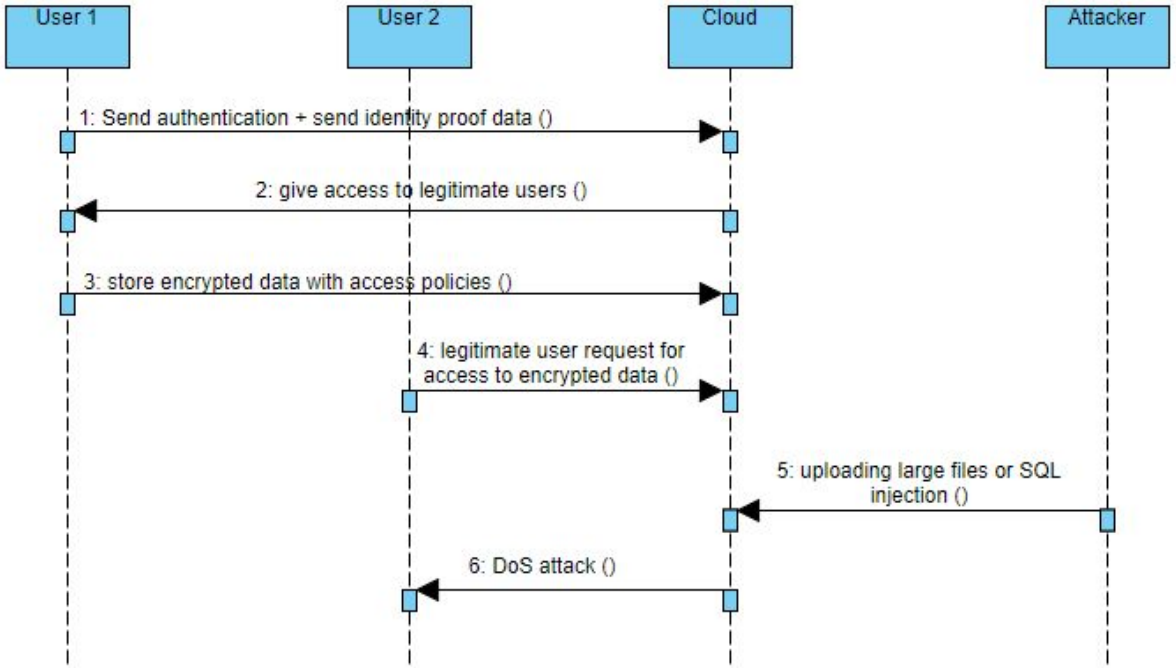


Figure 1: GFS Architecture

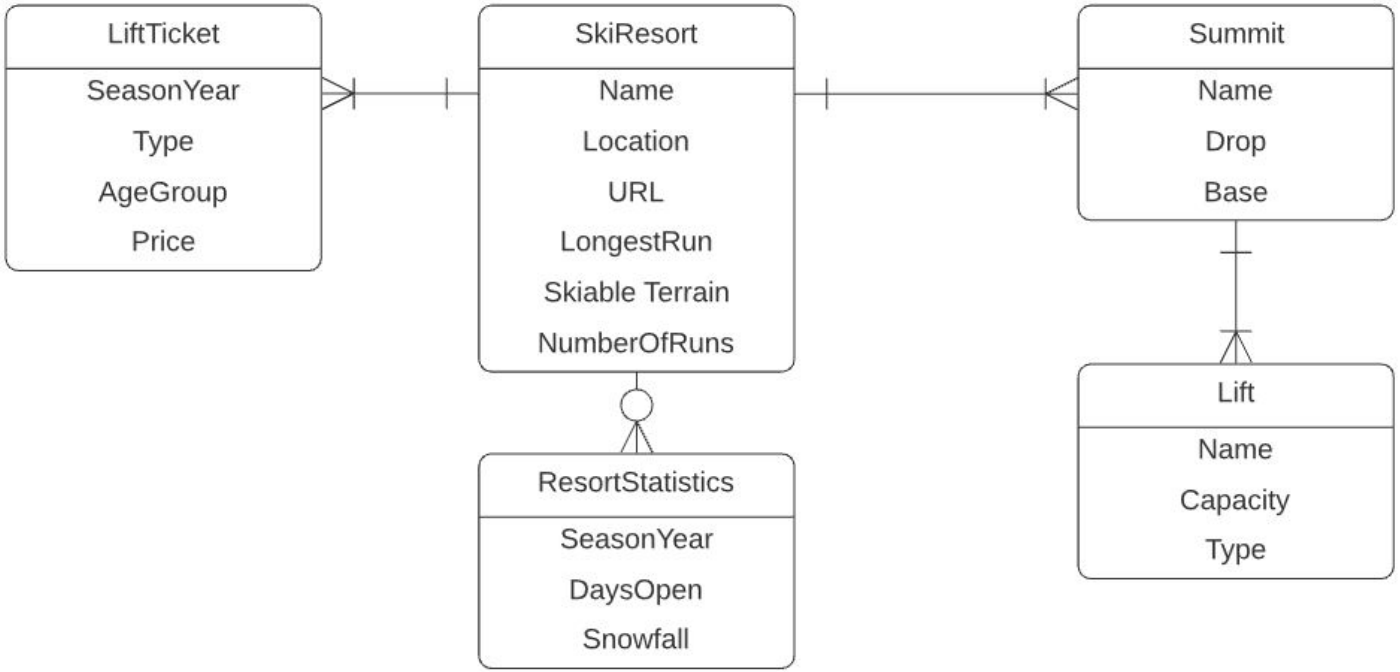
Ghemawat, Sanjay, Howard Gobioff, and Shun-Tak Leung. "The Google file system." ACM SIGOPS operating systems review. Vol. 37. No. 5. ACM, 2003.

What can we reason about?



Web of Things: Security Challenges and Mechanisms. Rhumba Sardar and Tayyaba Anees. IEEE Access (2021).

What can we reason about?



Notations for Common Design Abstractions

Component diagrams: What are major components in the system and how do they communicate with each other?

Data models: What types of data does the system store and what are their relationships?

Sequence diagrams: How different actors (domain entities & system components) collaborate to carry out a piece of functionality?

State machines: What states can the system be in and what events cause it to change its state?

Modeling in this class

Sometimes, models are themselves developed as a formal, first-class artifact

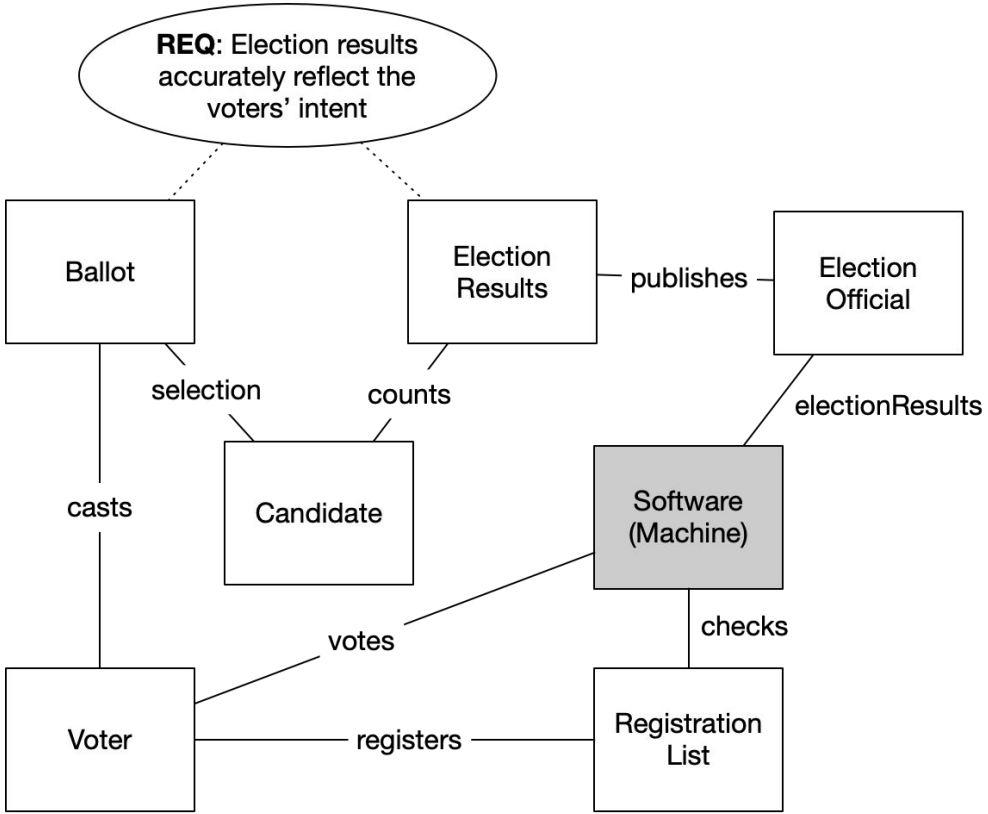
- To rigorously specify & verify properties about the system
- To generate a working implementation (e.g., model-driven engineering)

In this class, the goal is to **communicate**, not to achieve completeness

- To explain a design to someone who will work on the system later (sometimes, yourself)
- To provide a high-level overview of the system
- To explain an aspect of the system that is particularly complex

Example: Electronic Voting System

Context Model for the Voting System



Component Diagram

Component Diagram

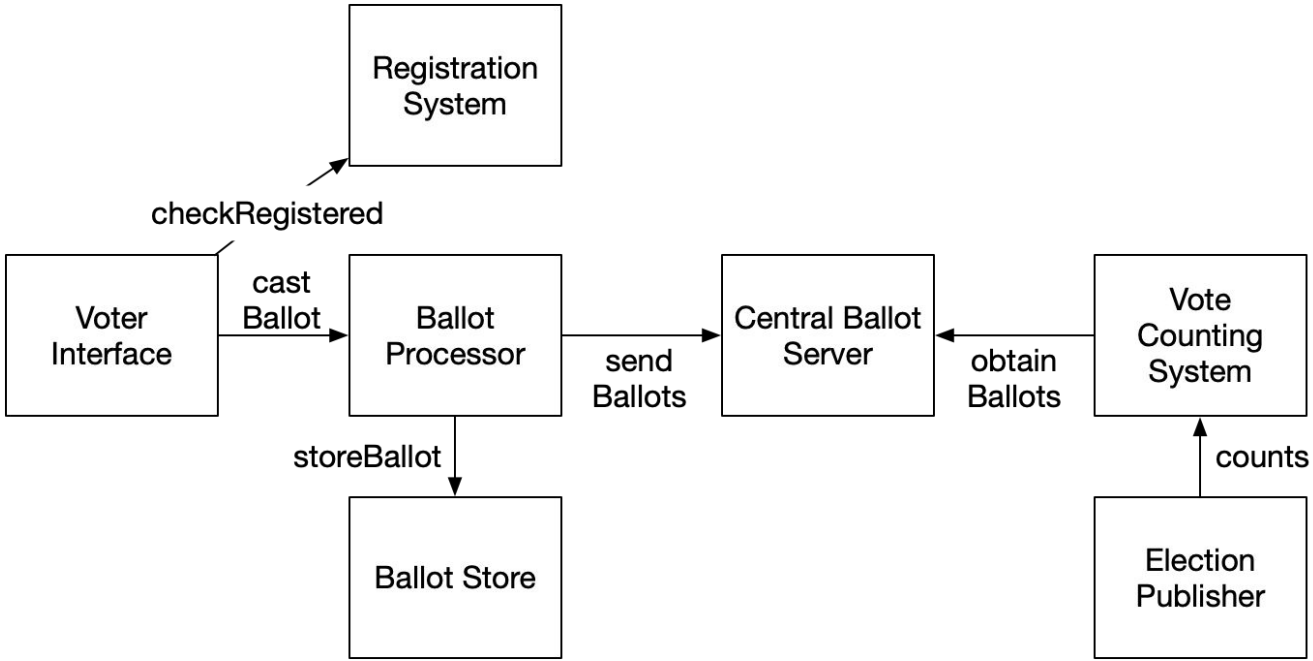
Purpose

Describe the major components in the software system, which components communicate with each other, and how they communicate.

Building blocks

- Component: A software component, responsible for carrying out a distinct unit of functionality
- Connection: A **directed** connection between a pair of components, labeled with an event (e.g., an API call) or data flow

Component Diagram: Example



Component Diagram: Design Questions to Ask

What information is passed between one component to another? Is there a return value?

What type of communication mechanism is used for each connection? (e.g., HTTP/S, RPC, Bluetooth) Is the communication synchronous or asynchronous?

Which components interact with the entities in the problem space (e.g., users)?

Which hardware device is each component deployed in? Which components are deployed on the same device?

What if a component changes or fails? What other components does it affect?

Data Model

Data Model

Purpose

- Describe different types of data that the system needs to remember to fulfill its specification
- Serve as the conceptual schema for designing a database

Building blocks

- Data type: A collection of data elements or objects
- Relation: A **directed** relation between a pair of data types
- Multiplicity constraints: A constraint on a relation, specifying how many instances of the two data types can be related to each other

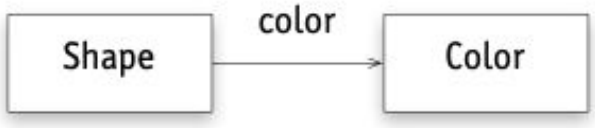
Data Model: Relations

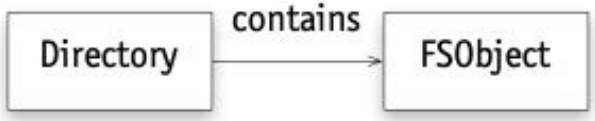
A relationship between different data types

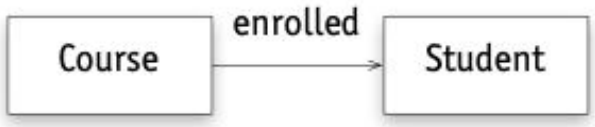
Kinds of relations:

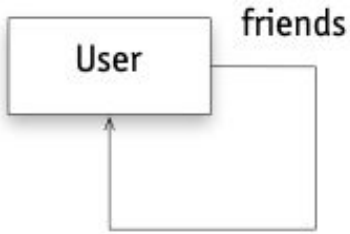
- Property
- Containment
- Association
- Naming

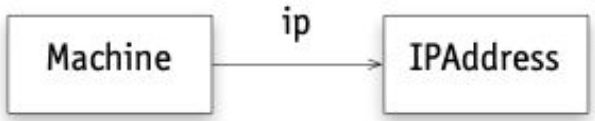
Q. What kinds of relations are these?







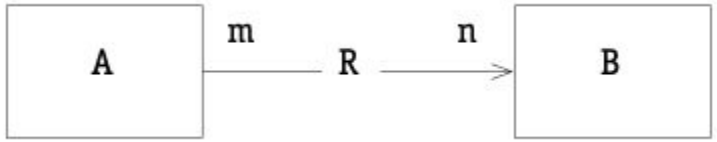




Data Model: Multiplicity Constraints

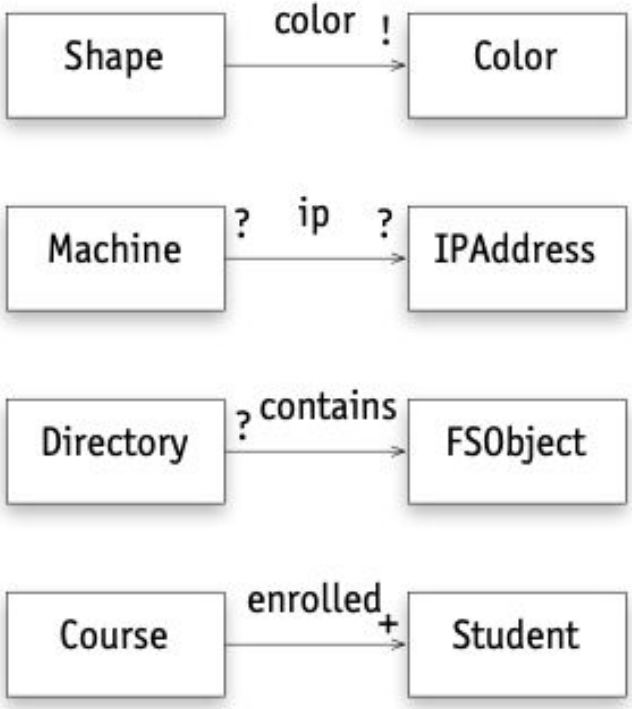
Constraints on relations

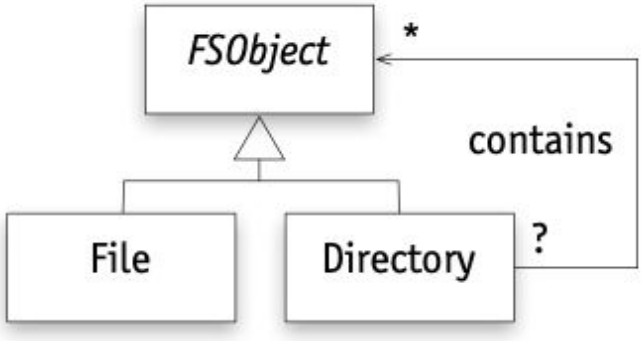
- R is a relation of type A to B
- R maps **m** A's to each B
- R maps each A to **n** B's



+ one or more
* zero or more
! exactly one
? at most one
omitted = *

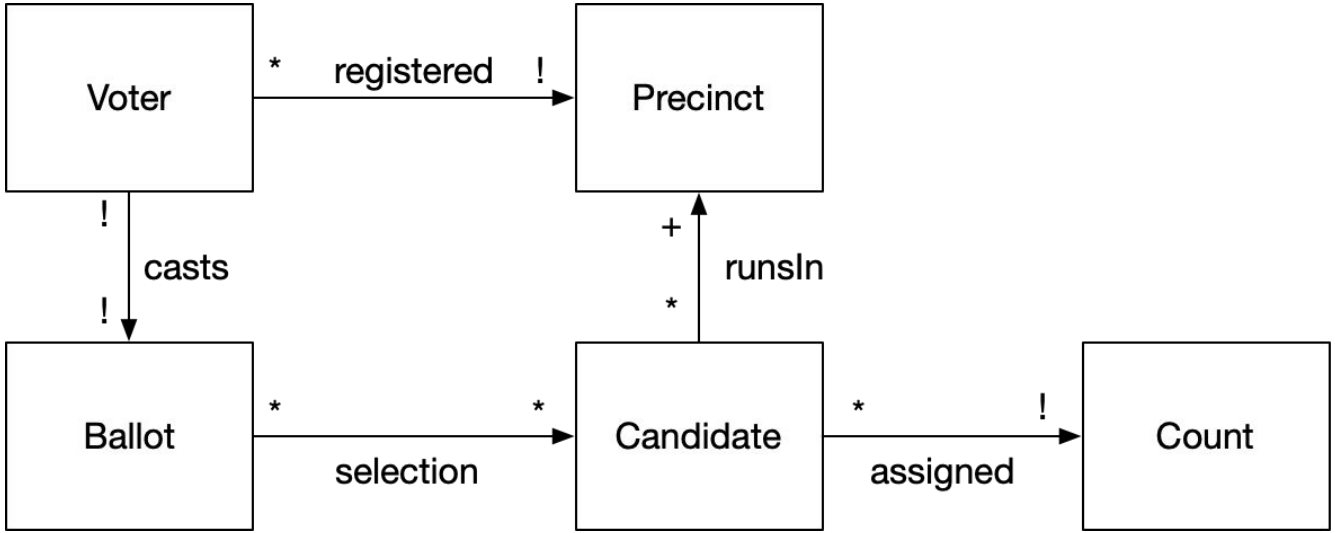
Data Model: Multiplicity Examples





+ one or more
* zero or more
! exactly one
? at most one
omitted = *

Data Model: Example



Data Model: Design Questions to Ask

Are we capturing all information that the system needs to function (and also achieve its quality attributes - e.g., security, availability)?

Do the multiplicity constraints reflect the real world scenarios? Are there any missing constraints? Are some constraints too strong?

Will all relations be stored on the same database, or be distributed across multiple databases? If distributed, do we need to consider consistency issues?

Is some of the data potentially sensitive? Do we need additional security or privacy mechanisms?

Sequence Diagram

Sequence Diagram

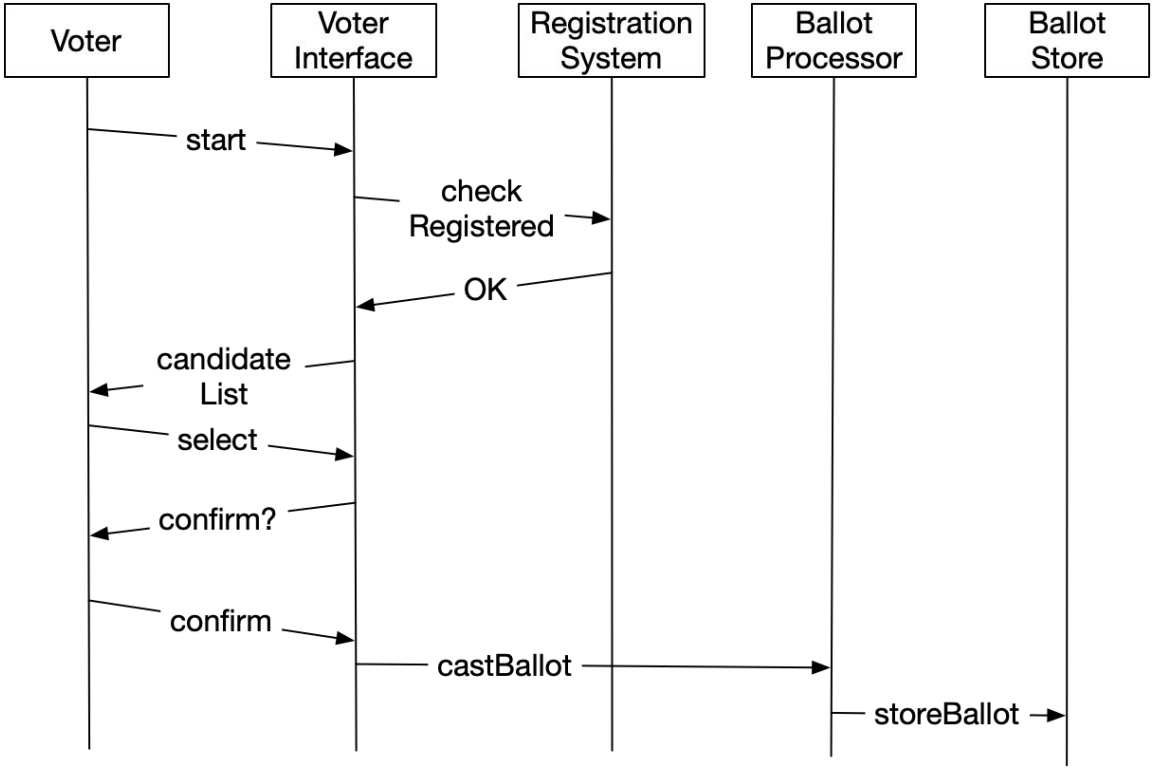
Purpose

Describe how a set of domain entities and system components collaborate in sequence to achieve a piece of functionality

Building blocks

- Process: A domain entity or system component
- Message: A message passed from one process to another

Sequence Diagram: Example



Sequence Diagram: Design Questions to Ask

What happens if a process terminates its activity early?

Is it possible for a message to be lost, and how does the system handle this? What if it arrives late?

What if a process receives two messages out of order? Does the order of execution matter for functionality or a quality attribute?

Are there any other processes that we are missing in this scenario?

Tips for Using Diagrams

If you need to introduce additional concepts in a diagram, that's OK! (e.g., a new type of edge in a component diagram)

- But be sure include a legend or text to explain new concepts

Keep diagrams to a reasonable size. If a diagram gets too big, break it into multiple ones.

Annotate a diagram with text to explain a concept (e.g., an event or a component) that is not obvious.

Use intuitive names! Avoid meaningless names (e.g., component named "Service").

Tips on Modeling

We will ask you document your design using some of these models throughout project milestones

Use these models in your team discussions! We will ask for your reflections on how they helped (or not) with designing your system

Treat these models as a tool for brainstorming & communicating, not for documenting everything perfectly

- Focus on aspects that are most important for your system (recall “risk-driven” approach to design)
- Use models to ask further questions about the system!

Summary

Exit ticket!