Understanding Systems with Design Models

17-423/723 Software System Design

Recitation 2
Jan 23, 2026

Recap: Design Abstractions

Notations for Common Design Abstractions

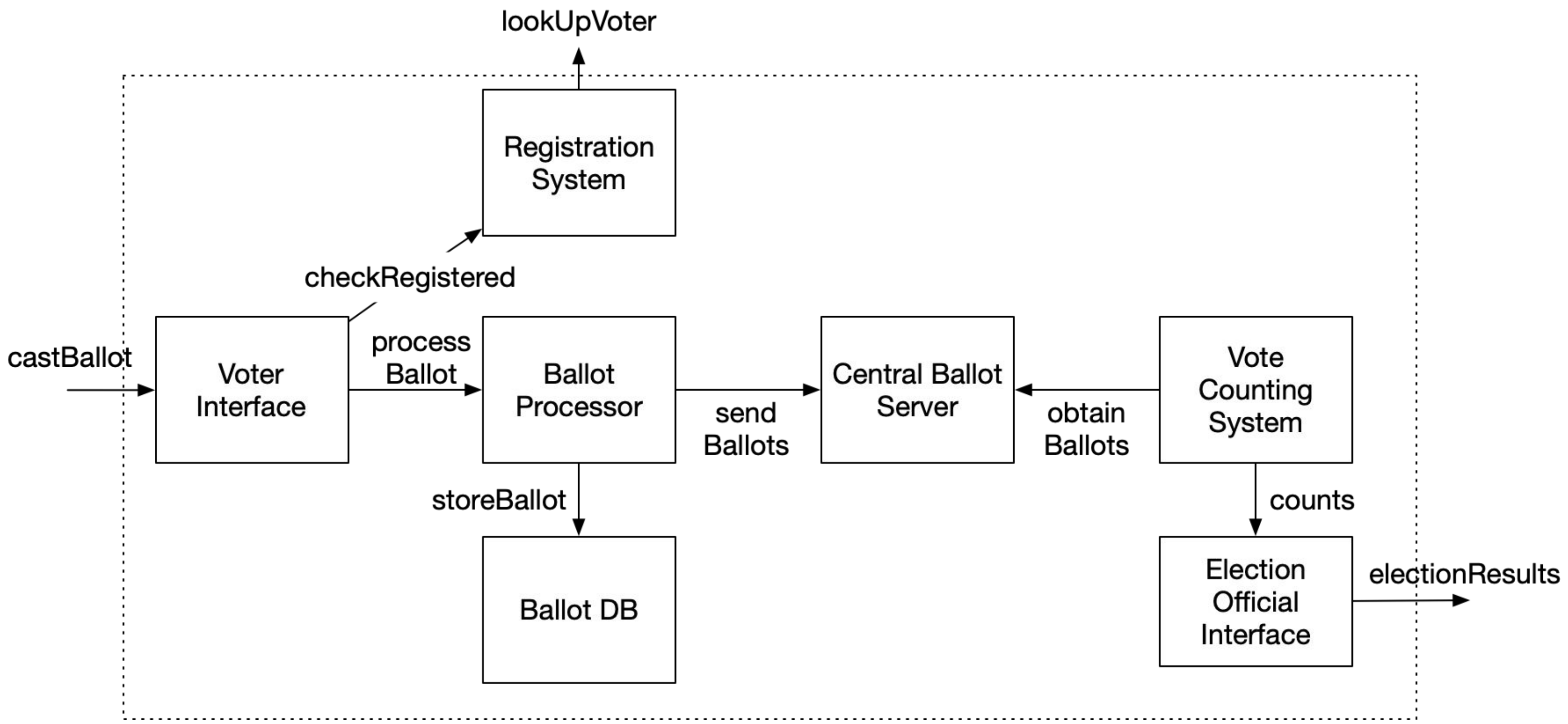
Component diagrams: What are major components in the system and how do they communicate with each other?

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

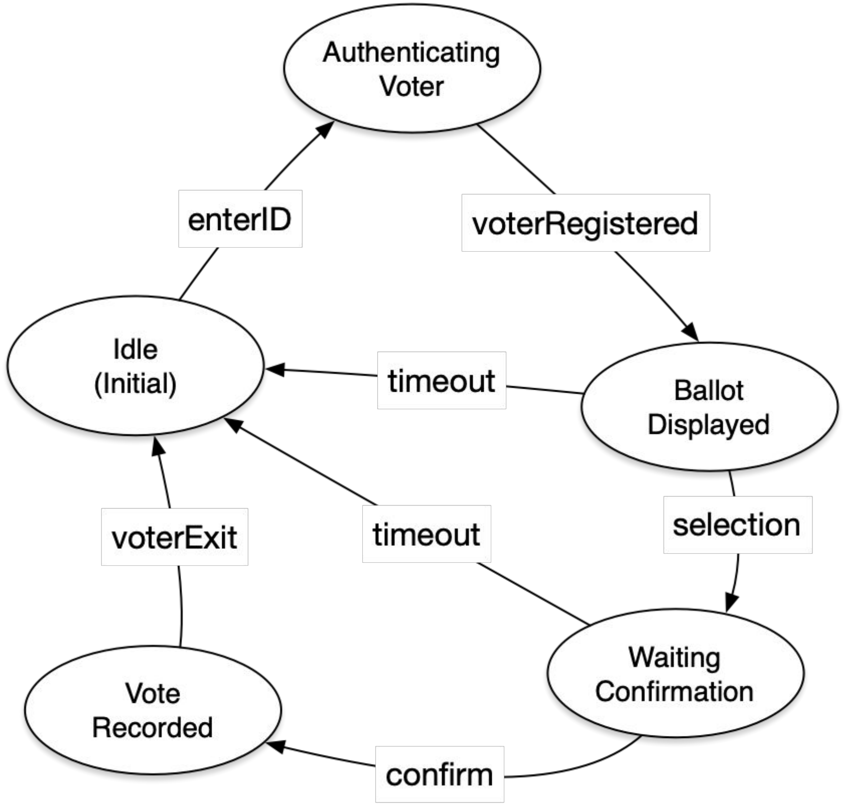
Data models: What types of data does the system store and what are their relationships?

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

Component Diagram: Voting System



State Machine: Voter Interface



Today: Using Design Models for System Understanding

- In addition to designing a new system, design models can also be used to understand the behavior and structure of an existing system
  - This is sometimes called **reverse engineering**
- There are many program analysis tools that can be used to extract different types of models from the code
- Today, we will explore how generative AI can be used to aid in this process

Case Study: OpenCode

- An open source AI coding assistant
- https://opencode.ai/
- https://github.com/anomalyco/opencode

Activity #1: Component Diagram

- Pair up with another person
- Use an LLM of your choice (e.g., Gemini, ChatGPT, Claude)
- Prompt the LLM to generate a component diagram
  - **Hint**: Ask it to generate the diagram in a format that can be displayed using an existing visualizer (e.g., Mermaid or PlantUML)
- Explore the diagram to answer the questions from the lecture; e.g.,
  - How do components communicate with each other?
  - What are the responsibilities of each component?
  - What if a component fails? How does it affect the overall system?
- If needed, use additional/modified prompts to simplify the diagram or extract more information about the system
- Post a screenshot of your component diagram on Slack #recitations

Component Diagram: Sample Prompt

I need your help understanding the system design of an open-source software project. The GitHub repository is: https://github.com/anomalyco/opencode

Please analyze this codebase and create a component diagram. A component diagram describes a high-level architecture of the system in terms of its major components and how they communicate with each other. The diagram should contain:
** Components, each of which is responsible for carrying out a distinct unit of functionality, represented as rectangular boxes*
** Connections, each a directed edge between a pair of components, labeled with an event (e.g., an API call) or data flow*
** External systems, actors, or libraries that the software interacts with*
** Component responsibilities, included as text annotation.*

You should produce the following deliverables:
** The diagram should be in PlantUML format so it can be rendered*
** A brief explanation of each diagram*
** Key insights about the system design and behavior based on the diagrams*

Activity #2: State Machine

- Prompt the LLM to generate a state machine for one of the core components in the system (identified through the component diagram)
- Explore the diagram to answer the questions from the lecture
    - What are possible error states? How can the machine reach those states?
    - Can the machine get stuck in a state due to an event that never takes place?
    - Can it get into a cycle, and is that acceptable?
    - Are there ways to improve the component logic based on what you found above?
- Post a screenshot of your state machine on Slack #recitations

State Machine: Sample Prompt

Please analyze the same codebase and generate a state machine from it. A state machine models the dynamic behavior of a component/system through its lifecycle. It should show:
** States that the system or key components can be in (represented as circles)*
** Initial state (shown as a bolded circle)*
** Transitions between states (arrows between states)*
** Events/triggers that cause state transitions (labels on arrows)*

Focus on the most important behavioral aspects, such as the application lifecycle, core workflow states, or the state management of critical entities.

Discussion

- What are benefits and potential risks of using design models to understand an existing system?
- How would you validate a generated model to ensure that it accurately reflects the system?

Summary

- Design models can help us understand existing systems, not just design new ones
- Different models reveal different aspects (structure vs. behavior vs. data)
- LLMs can be useful for reverse engineering but require critical evaluation
- Always validate AI-generated diagrams against actual code/behavior