

Introduction to Software Architecture

17-313: Foundations of Software Engineering

<https://cmu-313.github.io>

Josh Sunshine and Michael Hilton
Spring 2026

Smoking Section

- Last full row



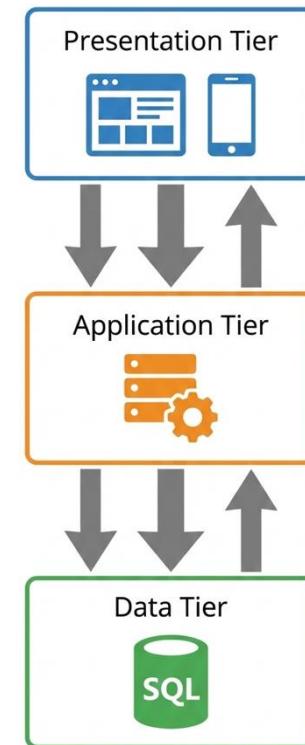
Learning Goals

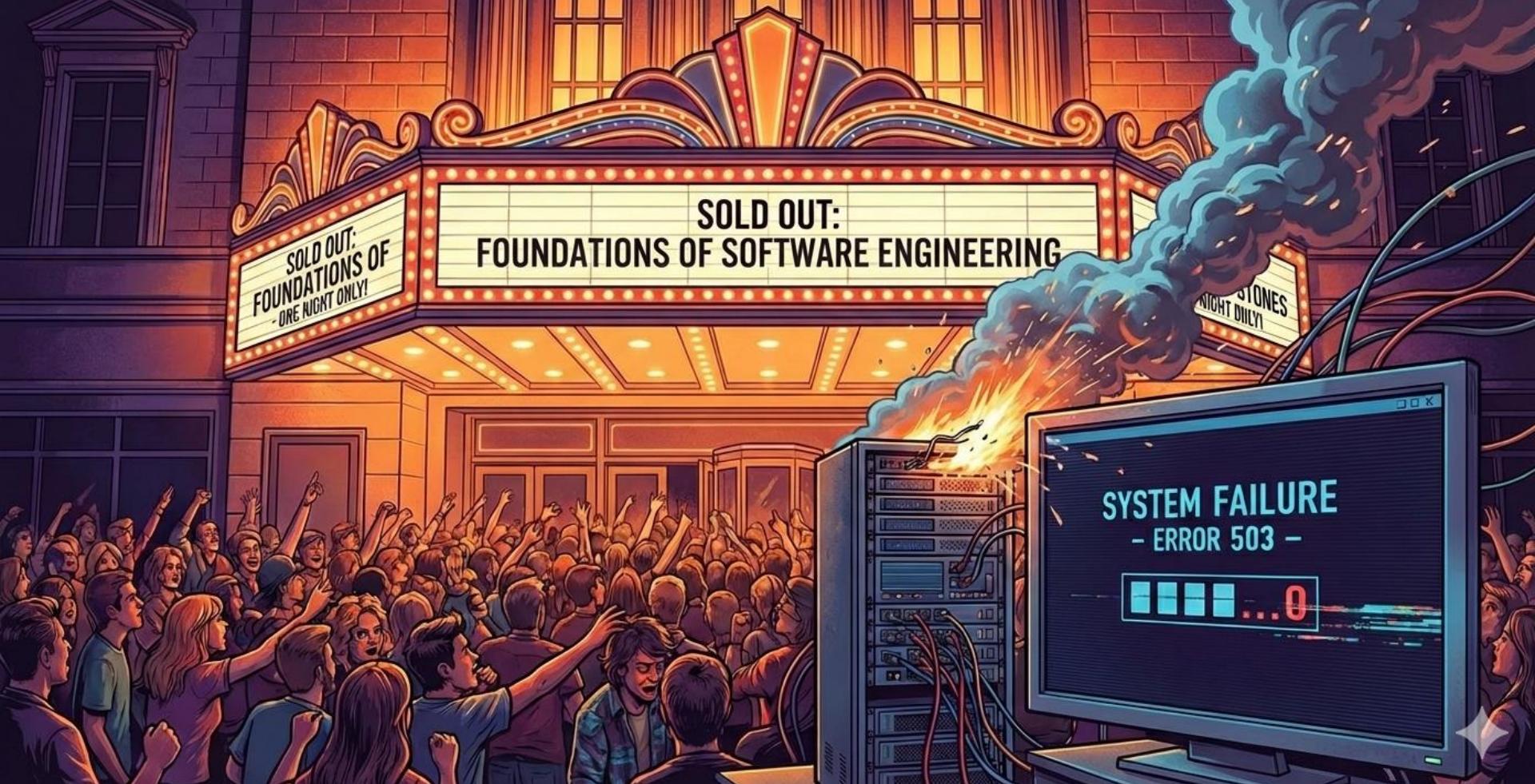
-  Understand the abstraction level of architectural reasoning
-  Appreciate how software systems can be viewed at different abstraction levels
-  Distinguish software architecture from (object-oriented) software design
-  Explain the importance of architectural decisions
-  Integrate architectural decisions into the software development process
-  Document architectures clearly, without ambiguity

Case Study: Victim of Success Theater

Case Study: Victim of Success Theater

- Simple web application for a local theater:
 - Sell tickets
 - 500 seats
- Architecture: Standard "3-Tier" setup:
UI → API → Database
- Happy path :
 - A user clicks "Buy"
 - API checks the DB, decrements the ticket count, and sends a confirmation
 - It works perfectly for months.





In-class activity

Where does the simple architecture of "Victim of Success Theater" break down?

1. Think on your own
2. Write down a hypothesis or two on your participation sheet
3. Discuss with your neighbor
4. Share with the class

Case Study: Quality Attributes

- **Scalability:** Can we handle the 10:00 AM rush?
- **Reliability:** Can we fail gracefully without double-charging people?
- **Maintainability:** Can we swap adjust the computing resources dedicated to the app without rewriting the code?

Outline

- Views and Abstraction
- Case Study: Victim of Success Theater
- **Software Architecture**
 - **Definitions, Importance**
 - **Software Design vs. Software Architecture**
- Architecting Software
 - Integrating Architectural Decisions into the SW Development Process
 - Common Software Architectures
 - Documentation

Software Architecture

The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them.



This definition is ambivalent to whether the architecture is known or whether it's any good!

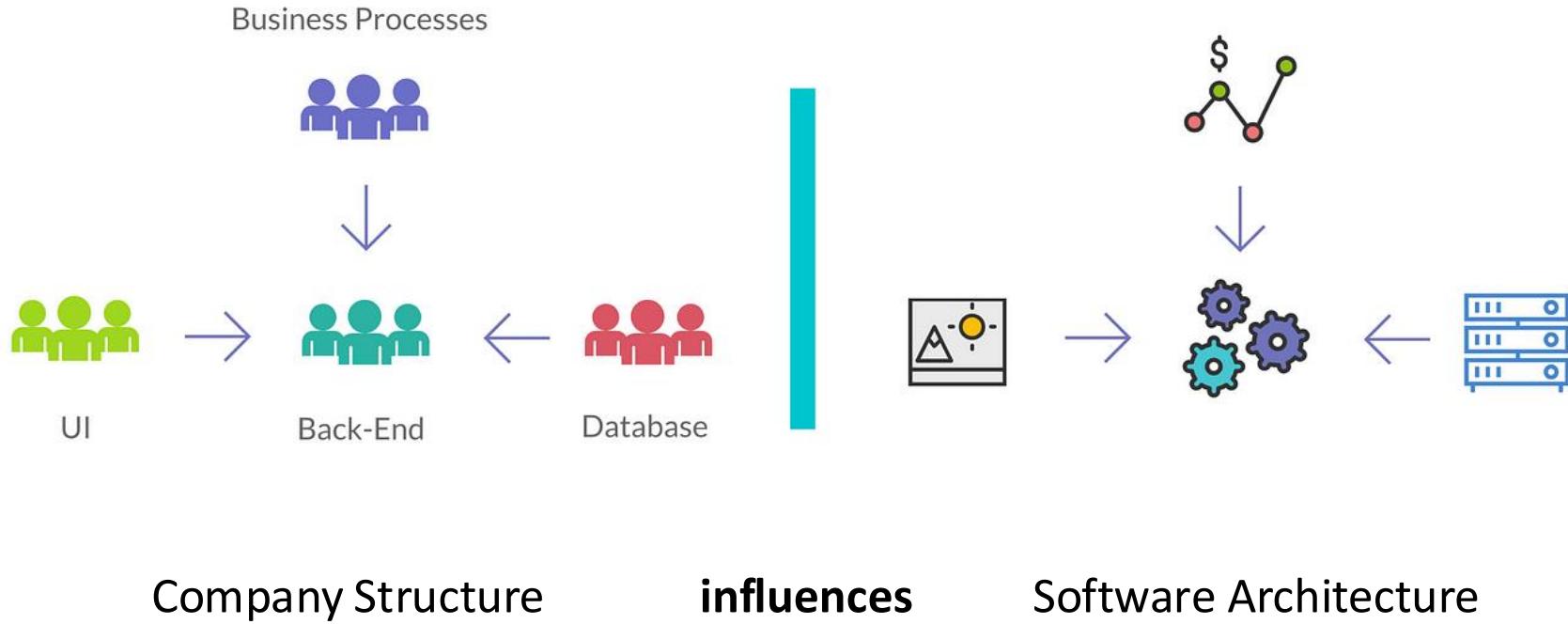
[Bass et al. 2003]

Software Architecture: Motivation

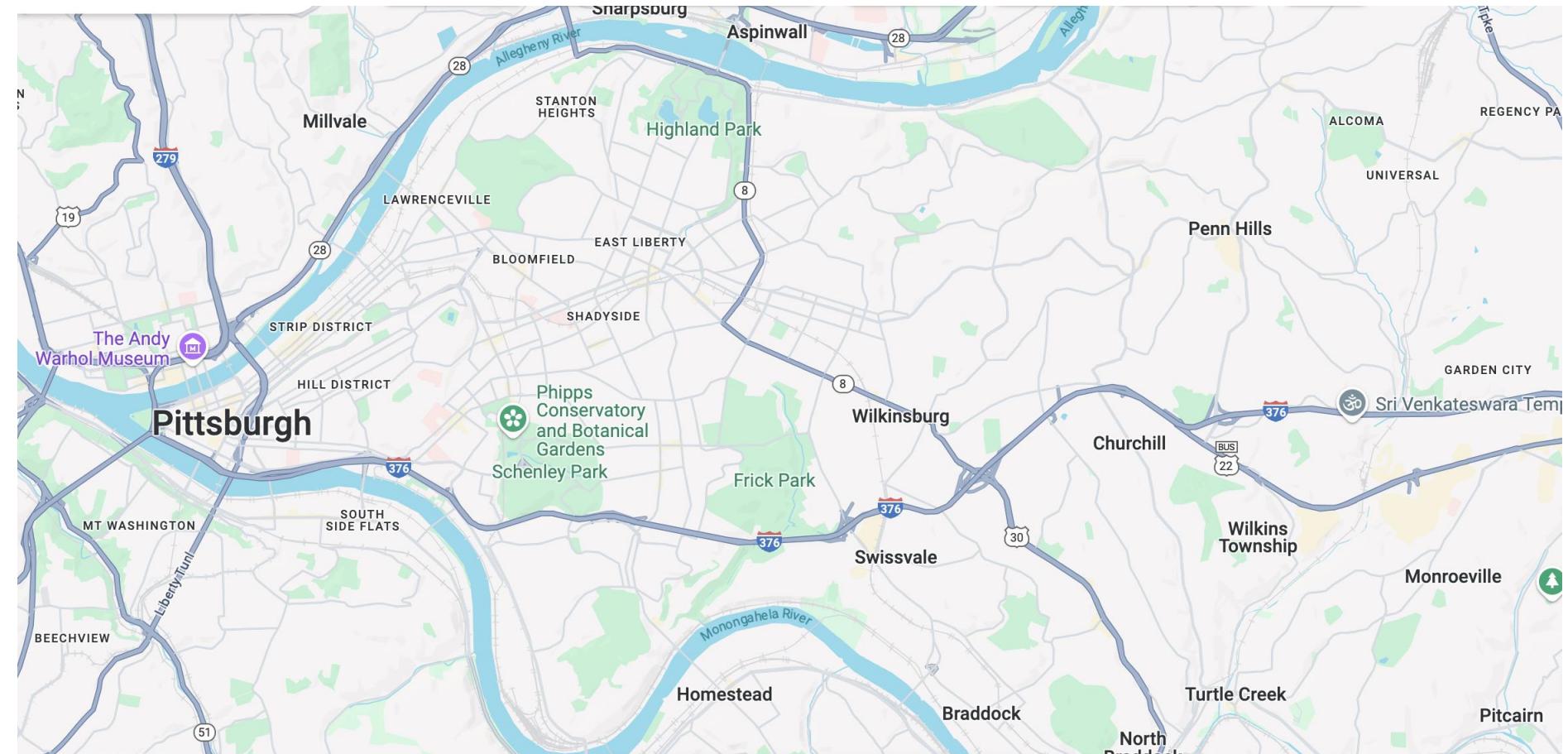
- Facilitates internal and external communication
- Describes design decisions and prescribes implementation constraints
- Relates to organizational structure
- Permits/precludes achieving non-functional requirements
- Control complexity
- Reason about and manage change
- Good basis for effort estimation
- ...

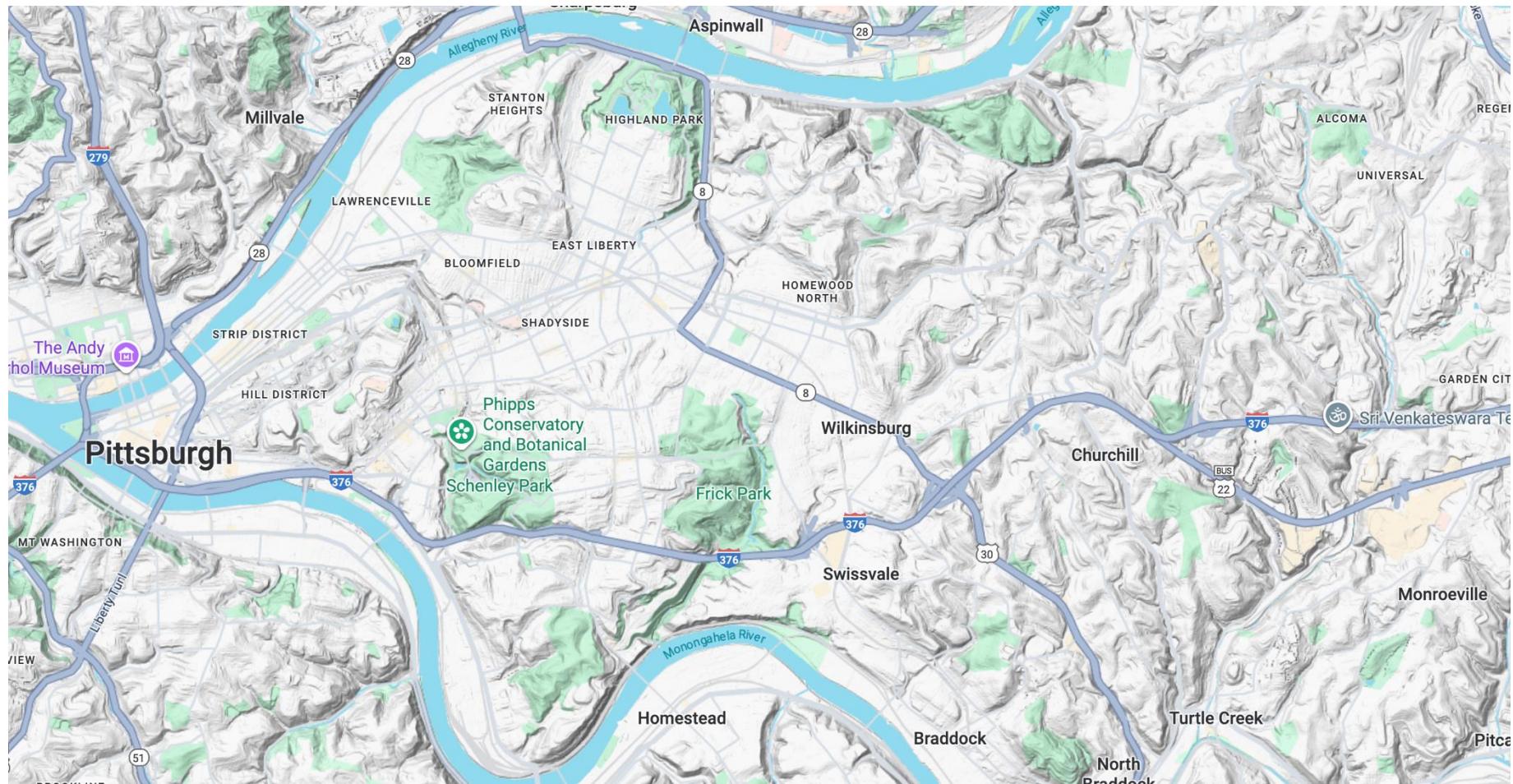
Architecting Software the SEI Way - Software Architecture Fundamentals: Technical, Business, and Social Influences. Robert Wojcik. 2012

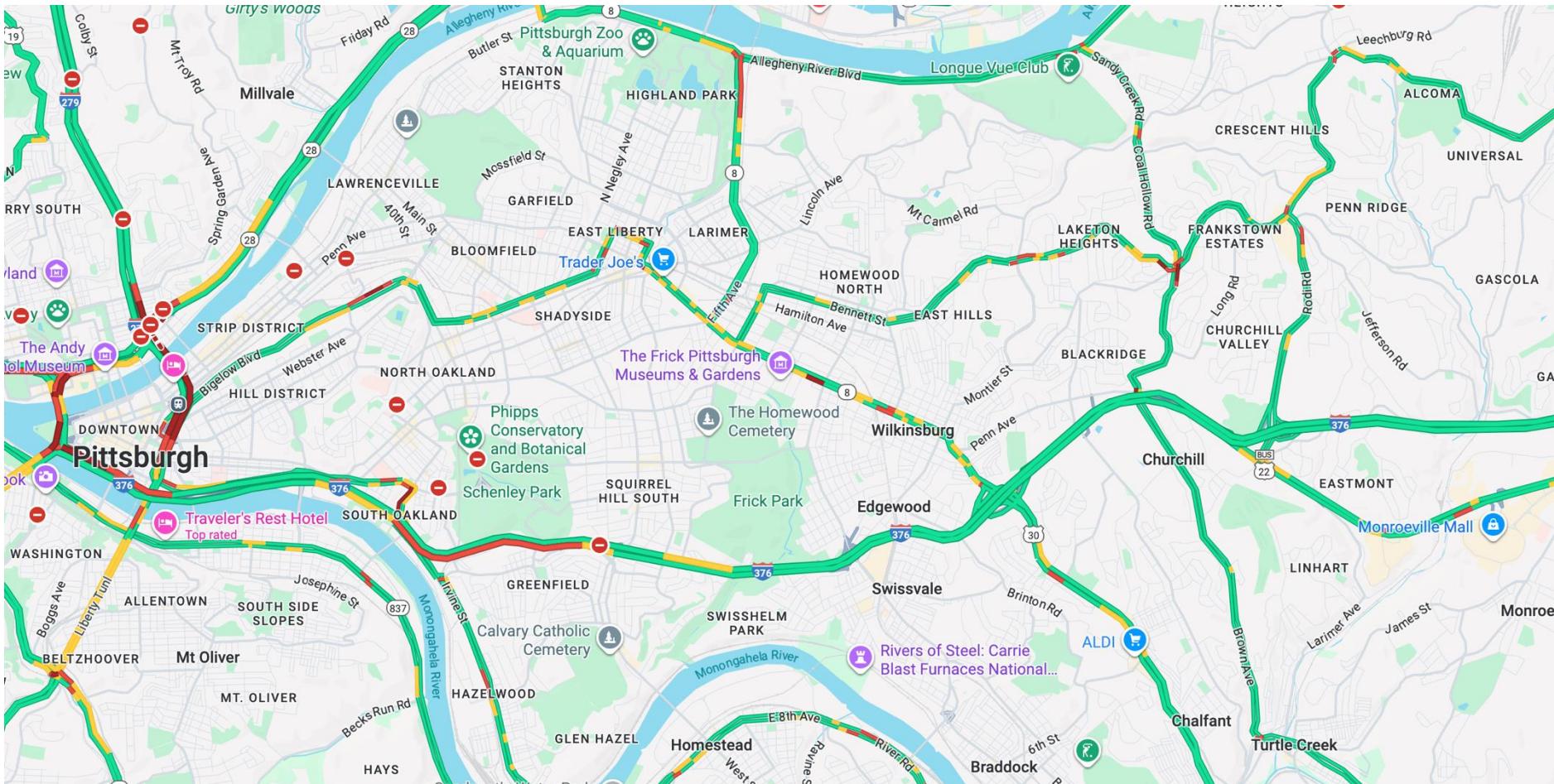
Conway's Law

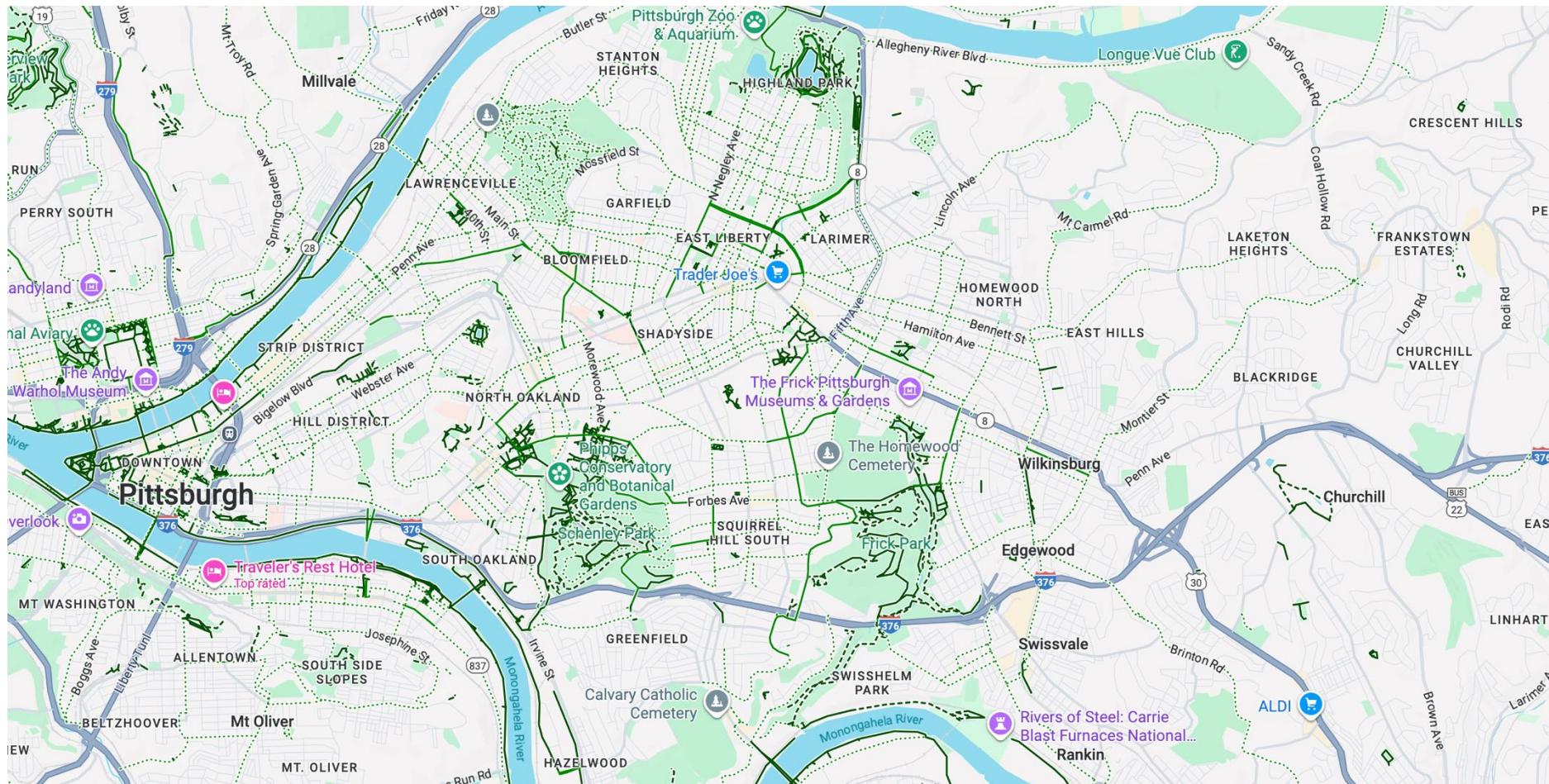


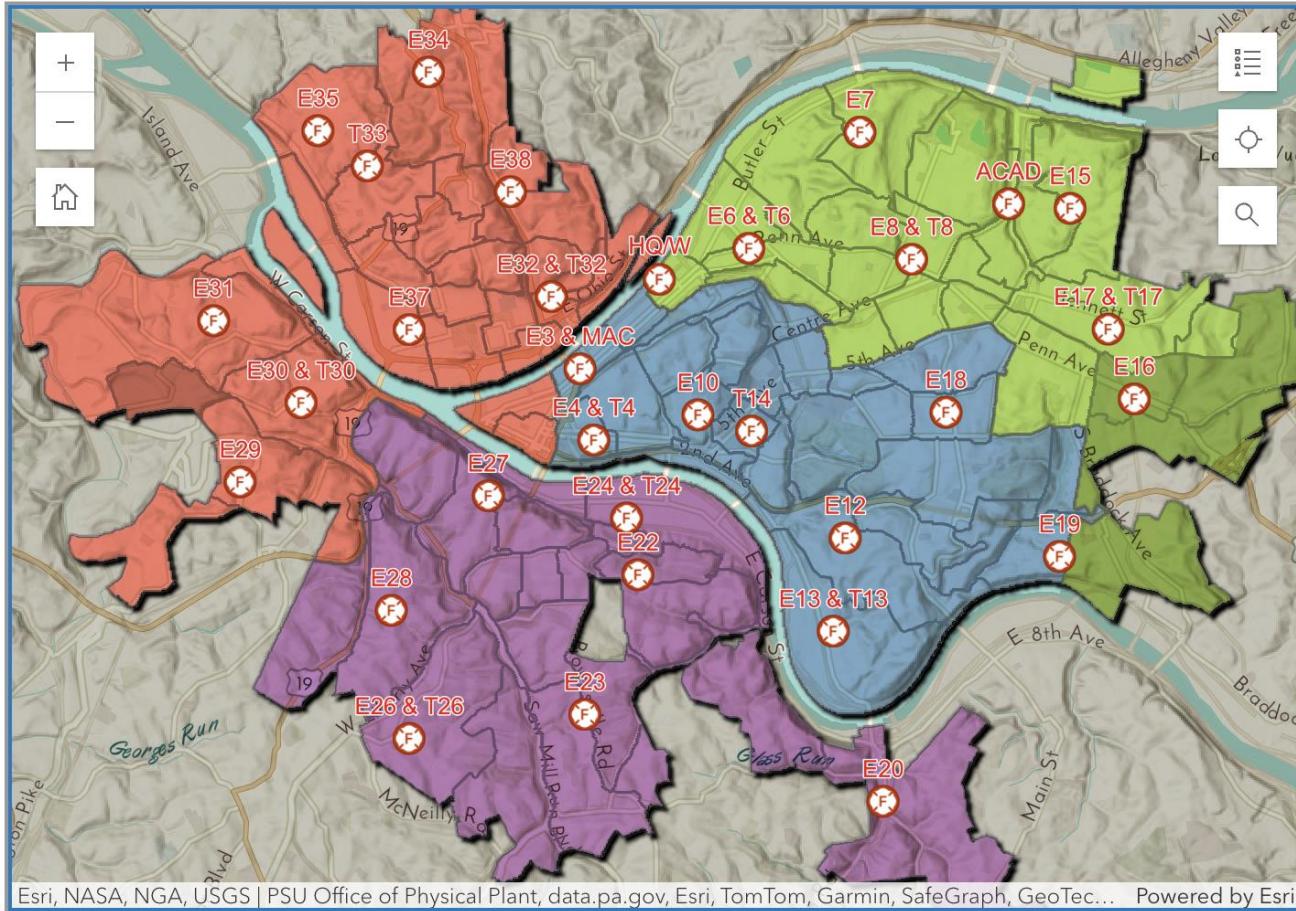
Architectural Views









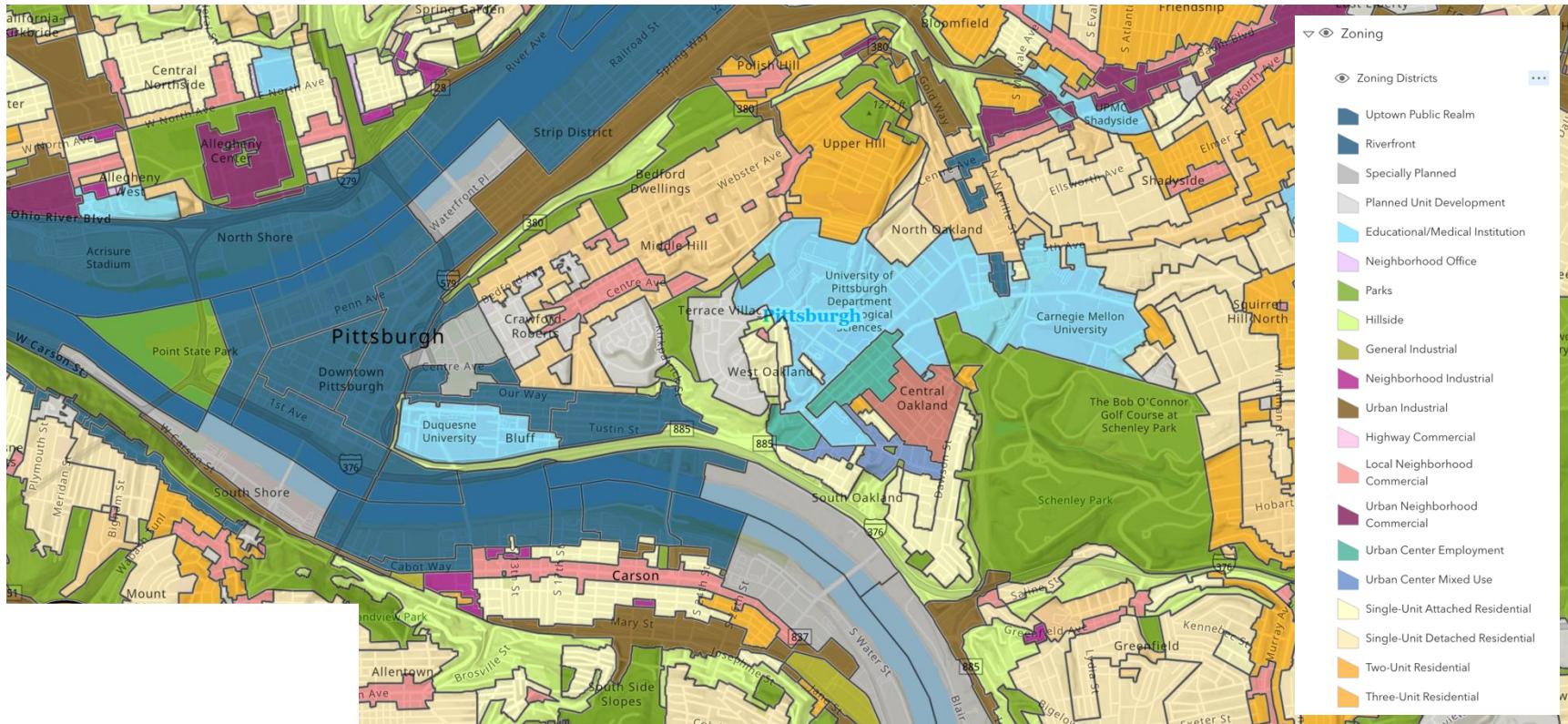


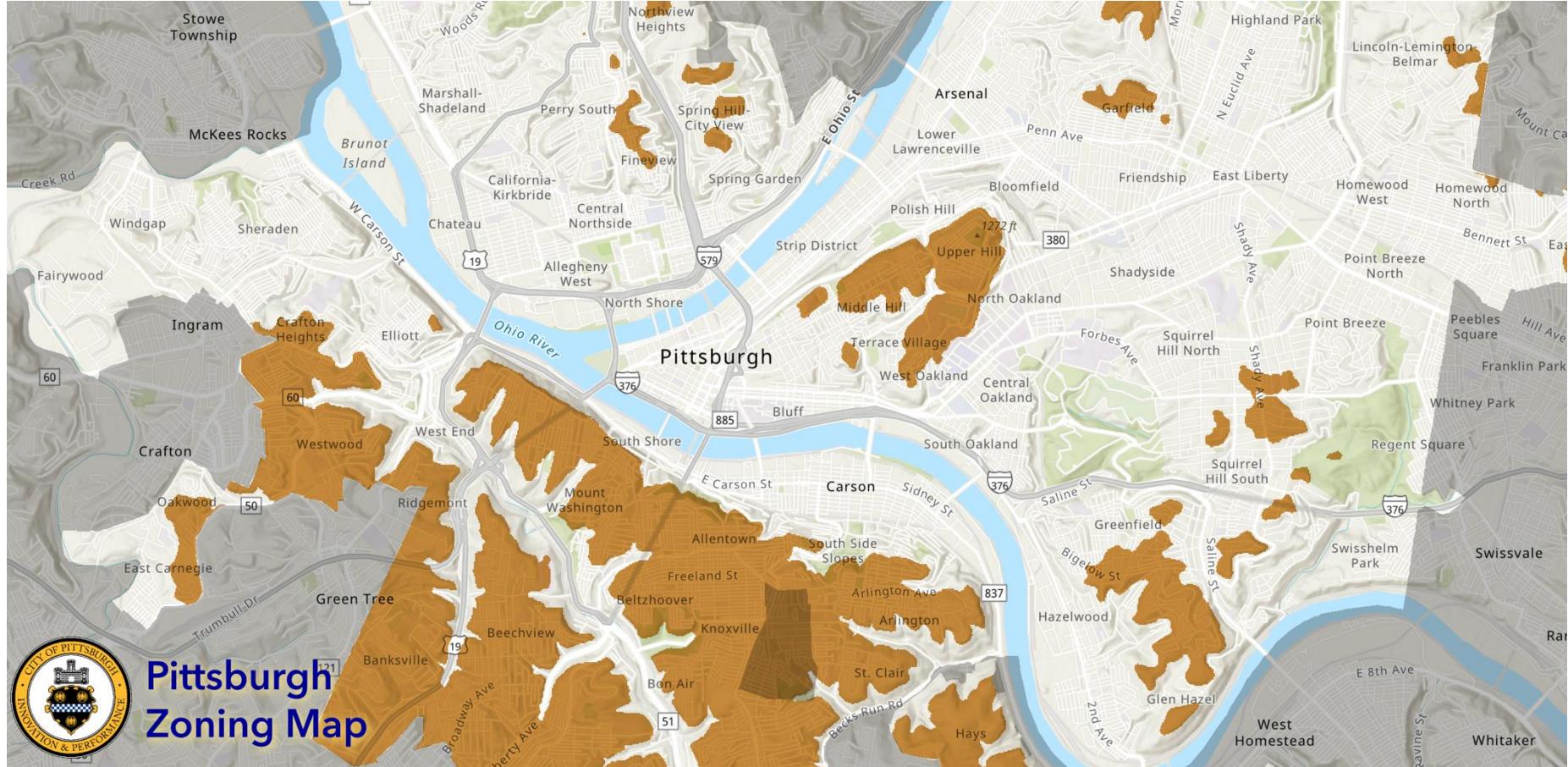
Fire Station



Fire Districts

- 1
- 2
- 3
- 4





Pittsburgh Zoning Map



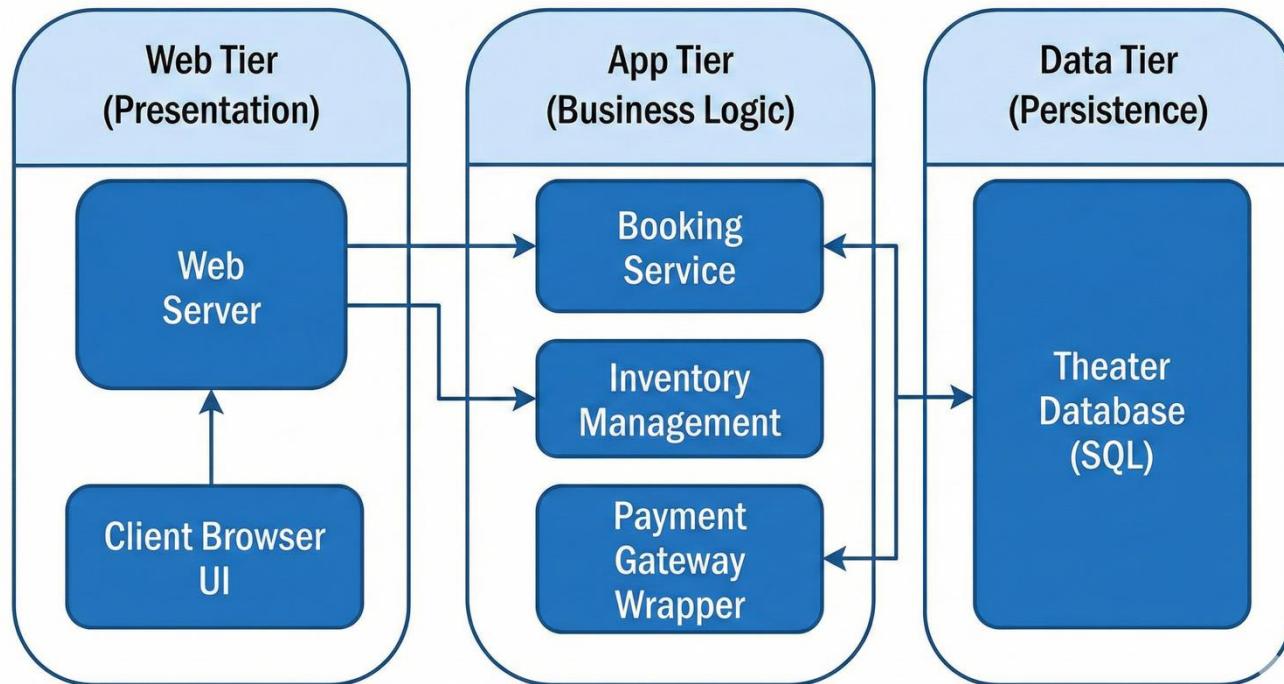
Software and Societal
Systems Department

Carnegie
Mellon
University

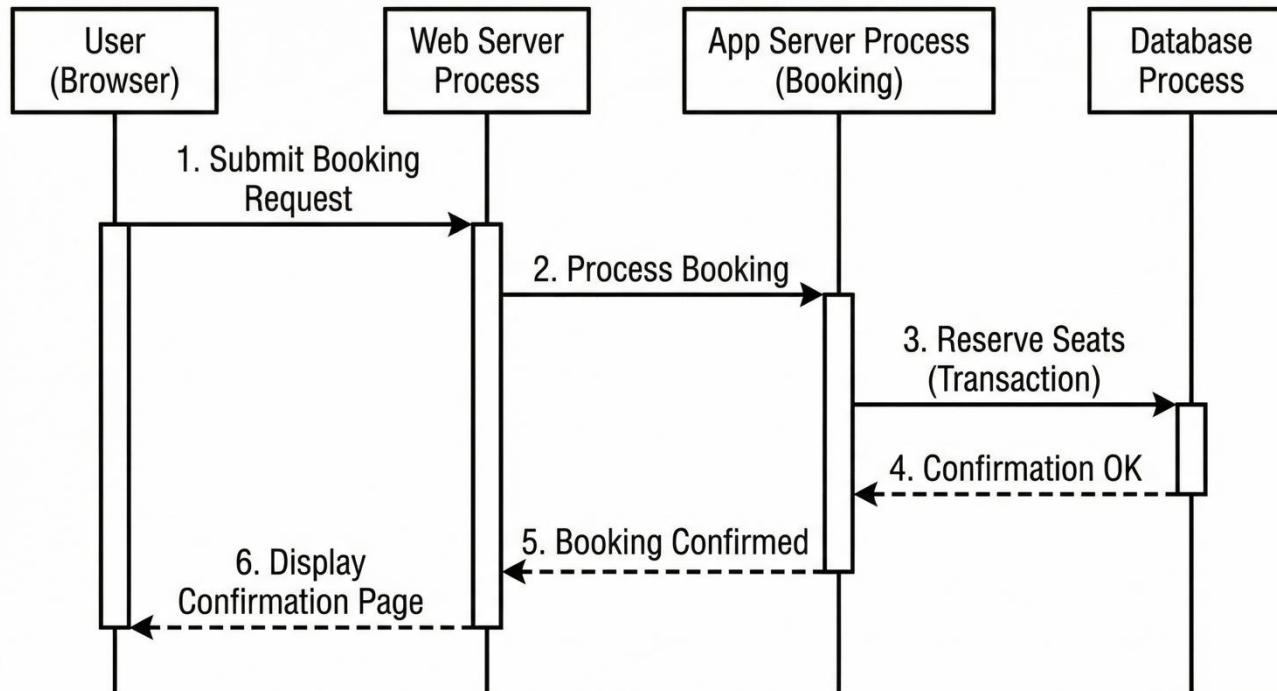
Each views abstracts reality to focus on conveying specific information

- They have a **well-defined purpose**
- Show only **necessary information**
- **Abstract away** unnecessary details
- Use legends and annotations to **remove ambiguity**
- **Multiple views** together produce a **richer understanding**

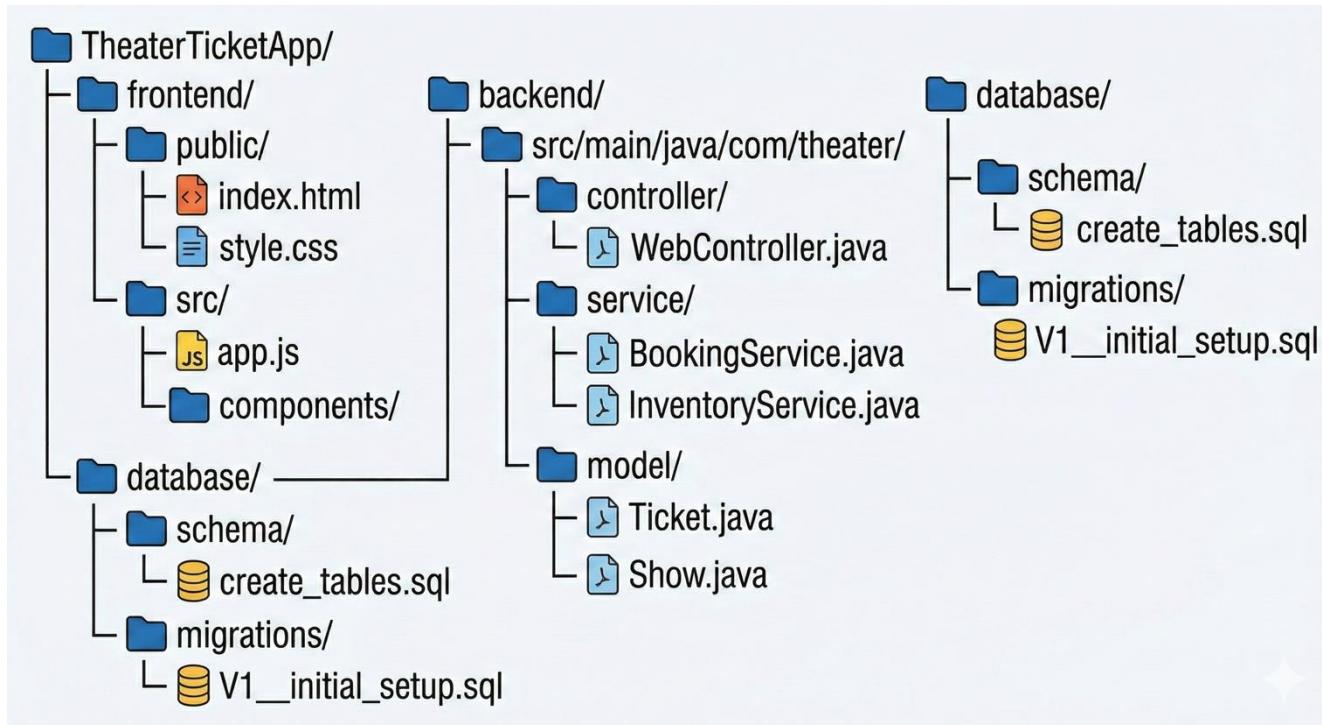
Logical View



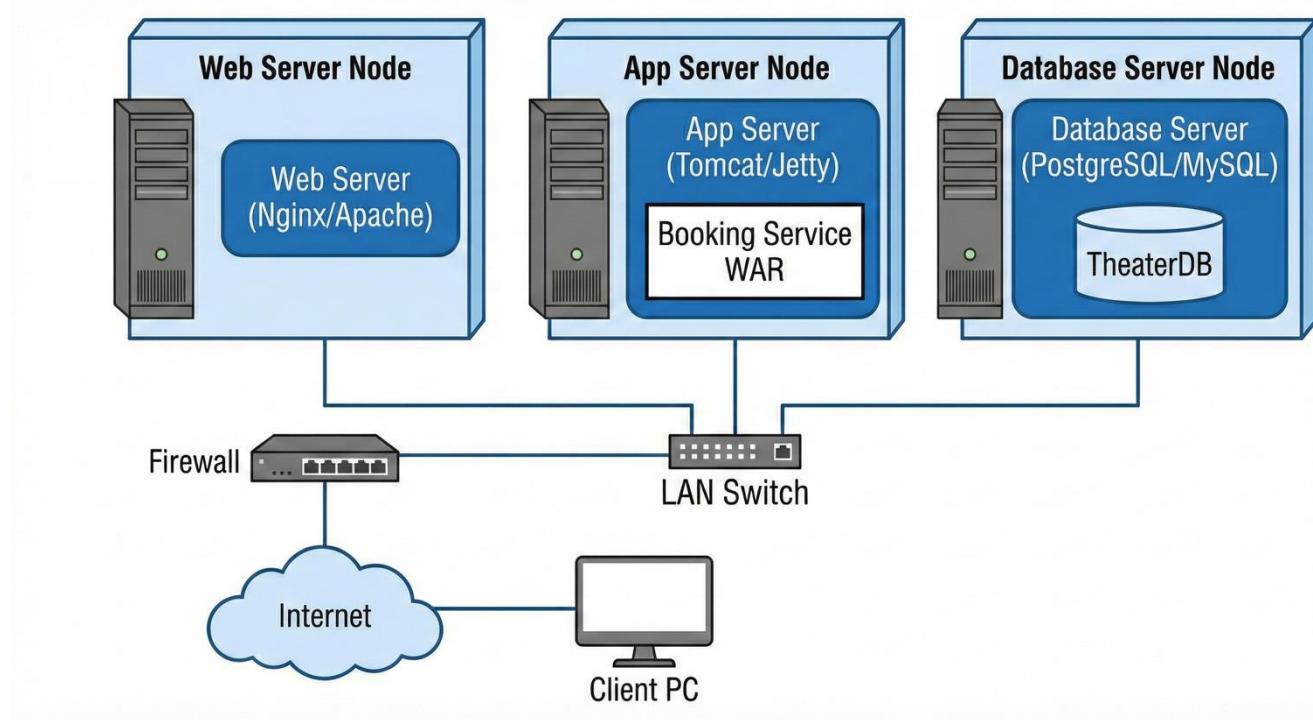
Process View



Development View



Physical View



Software Design vs. Architecture

Levels of Abstraction



Requirements

- high-level “what” needs to be done



Architecture (high-level design)

- high-level “how”, mid-level “what”



Software Design (low-level design, e.g., design patterns)

- mid-level “how”, low-level “what”



Code / Implementation

- low-level “how”

Design vs. Architecture

Design Questions

- How do I add a menu item in NodeBB?
- How can I make it easy to create posts in NodeBB?
- What lock protects this data?
- How does Google rank pages?
- What encoder should I use for secure communication?
- What is the interface between objects?

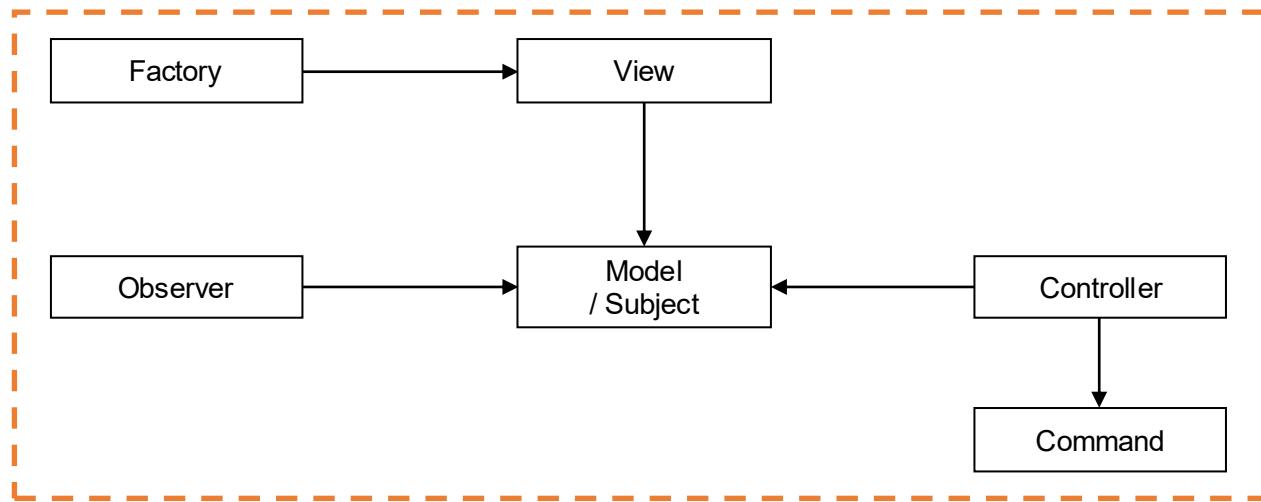
Architectural Questions

- How do I extend NodeBB with a plugin?
- What threads exist and how do they coordinate?
- How does Google scale to billions of hits per day?
- Where should I put my firewalls?
- What is the interface between subsystems?

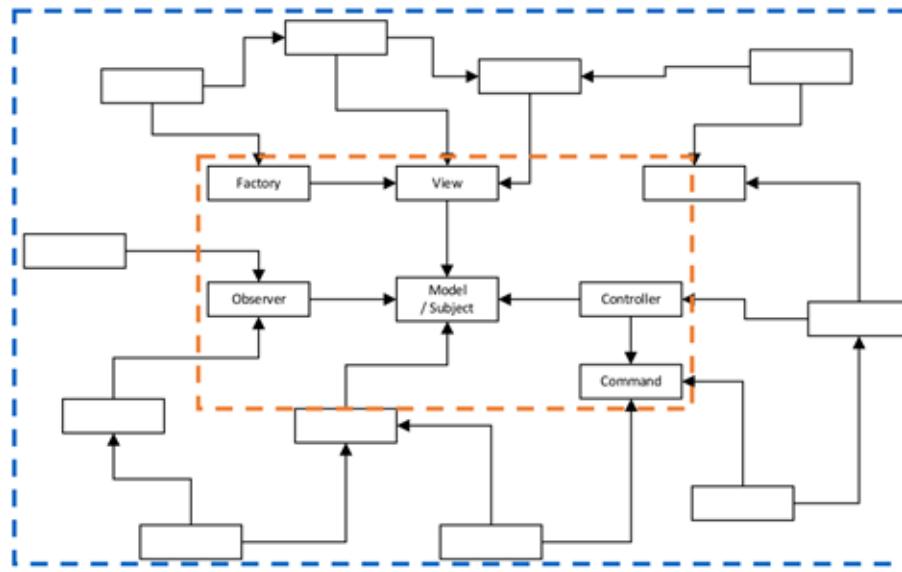
Objects

Model

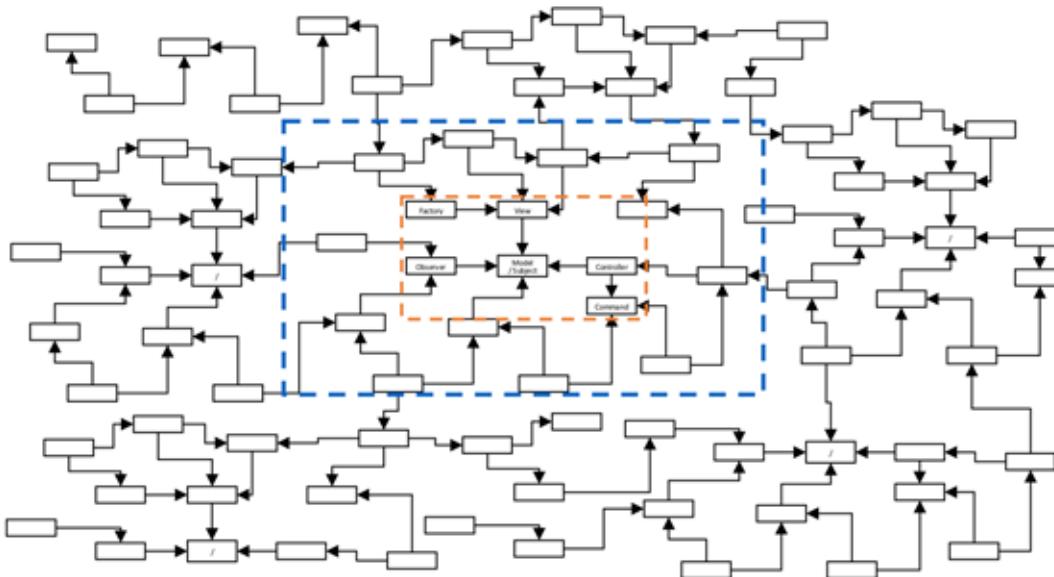
Objects → Design Patterns



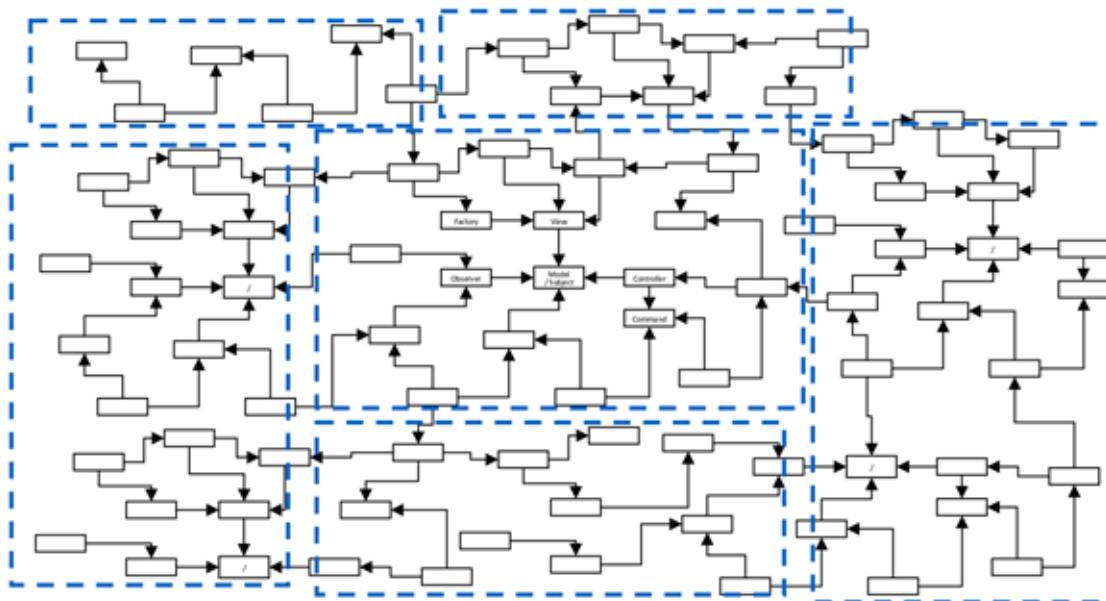
Design Patterns



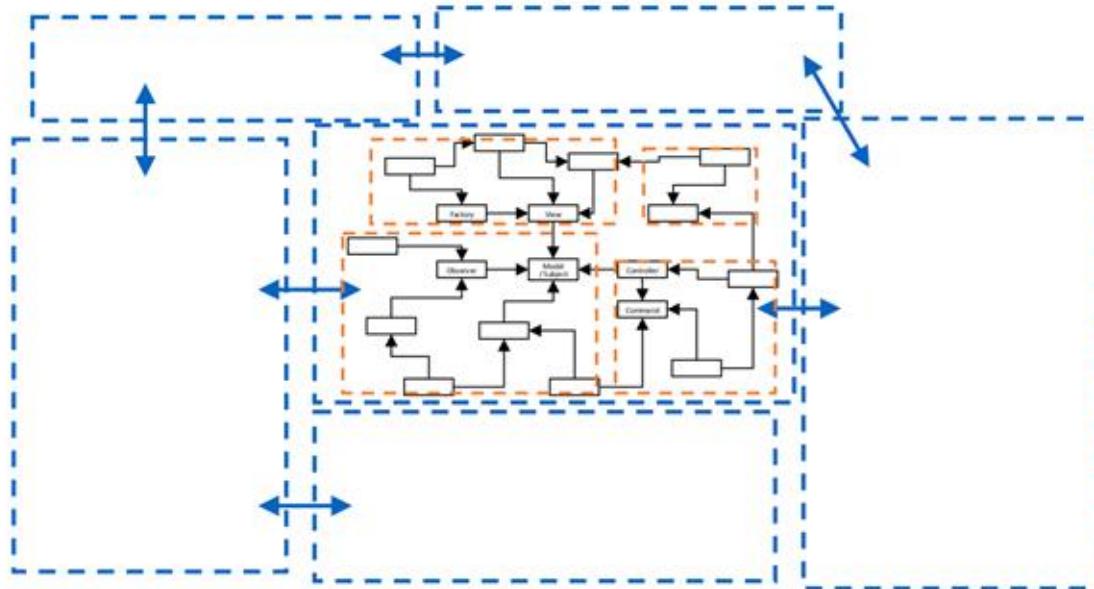
Design Patterns



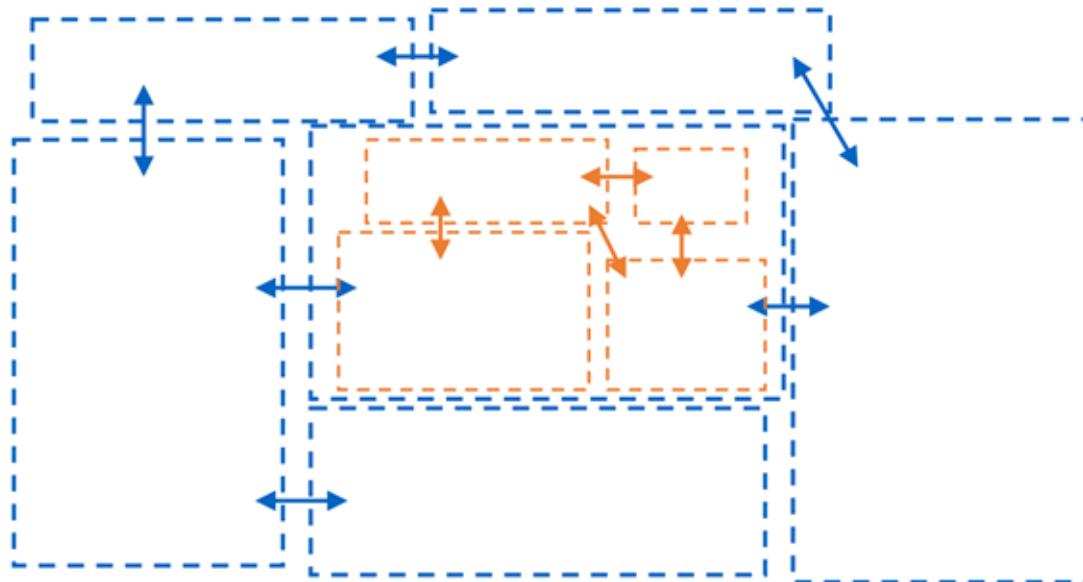
Architecture



Architecture



Architecture



Architecting Software

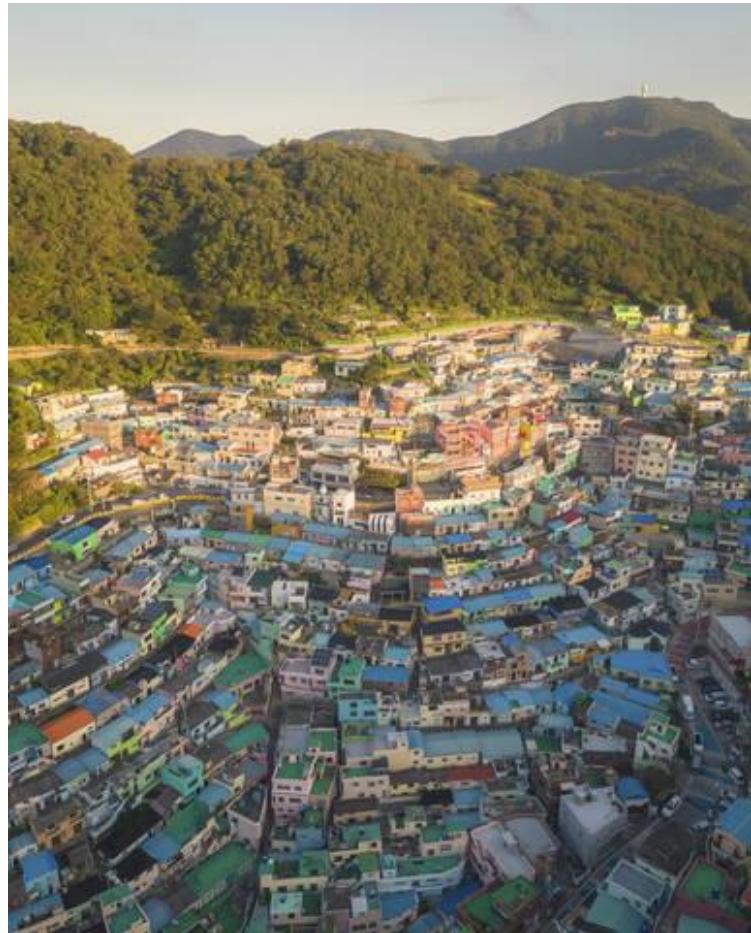
<https://www.archdaily.com/>



<https://www.instagram.com/architectanddesign>



<https://www.mykonosceramica.com/>



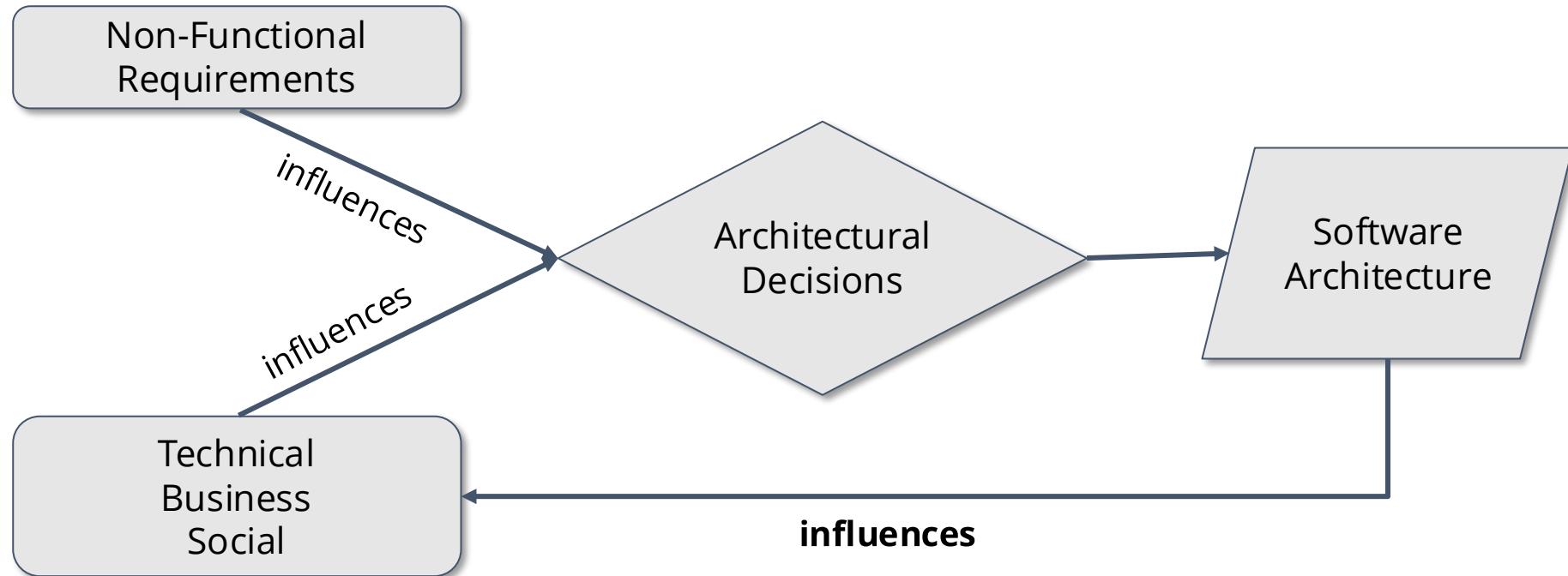
Every system has an architecture

- Whether you know it or not
- Whether you like it or not
- Whether it is documented or not

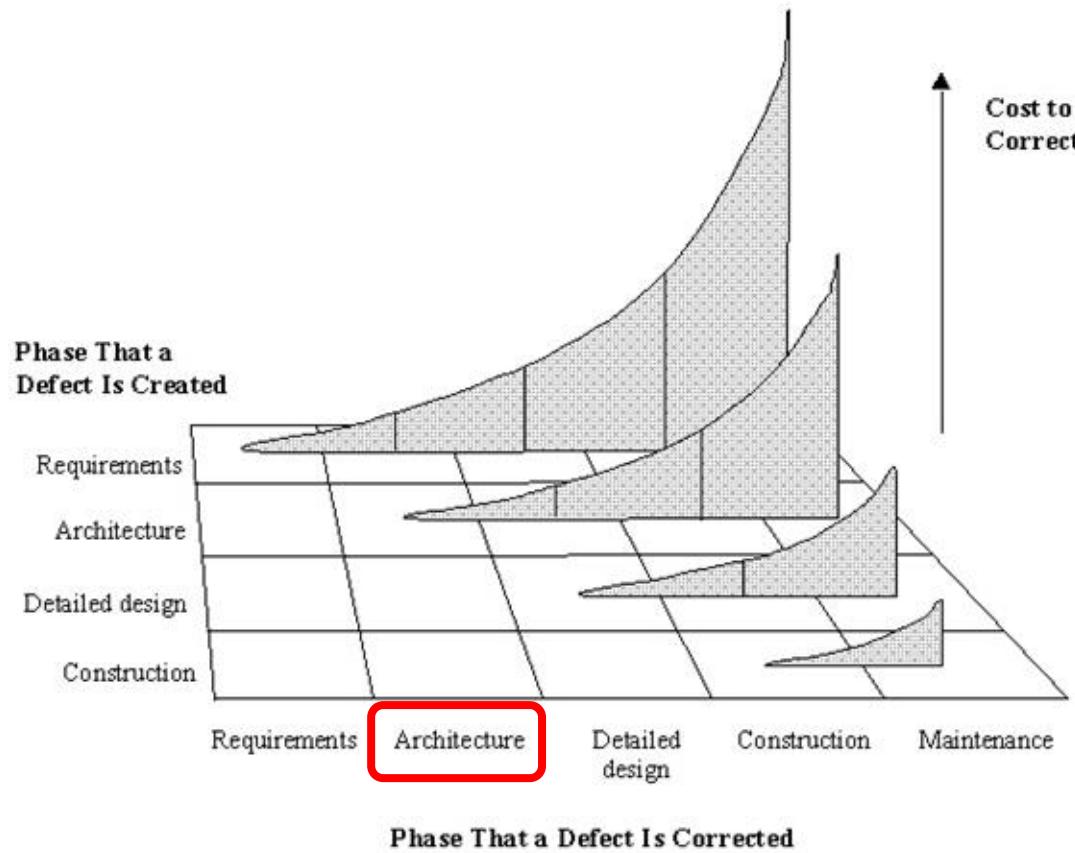
If you don't consciously elaborate the architecture, it will evolve by itself!



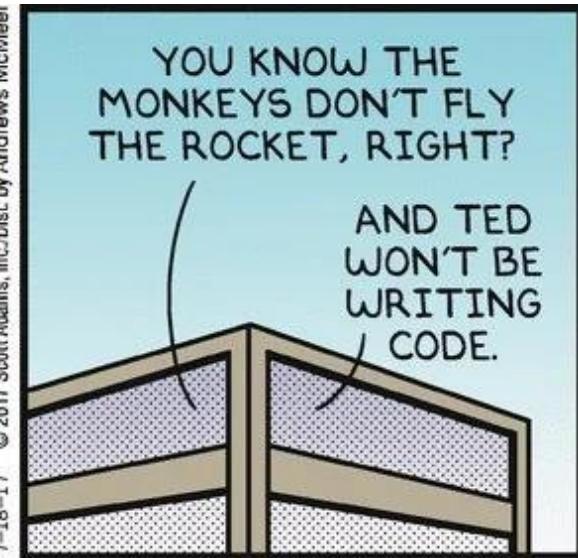
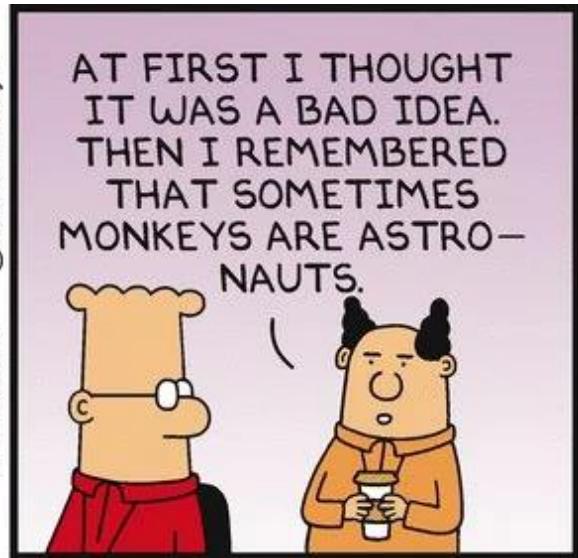
Architecting Software the SEI Way - Software Architecture Fundamentals: Technical, Business, and Social Influences. Robert Wojcik. 2012



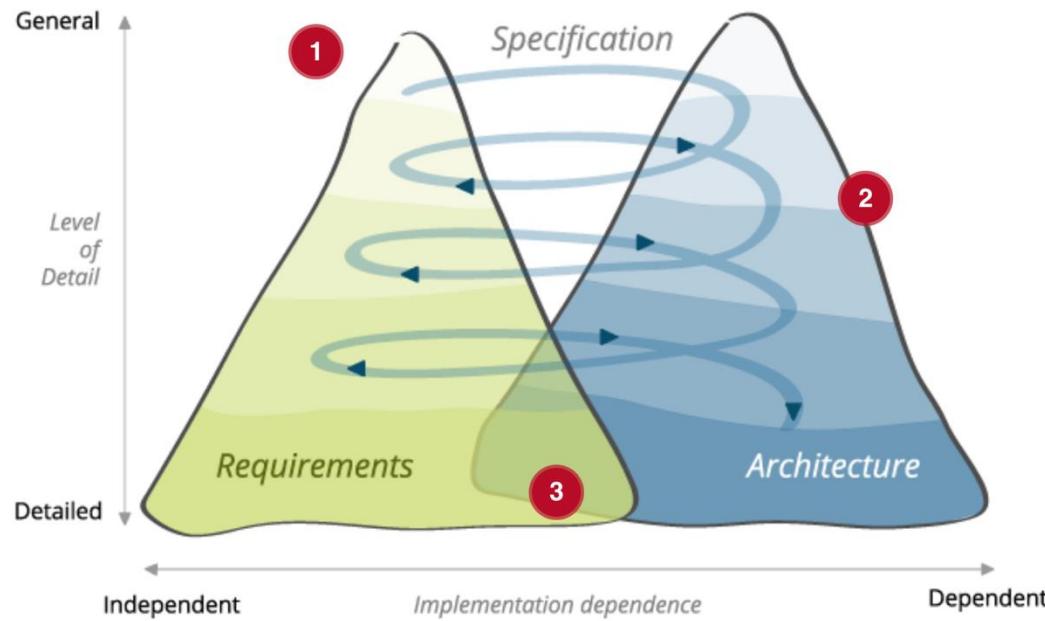
Architecting Software the SEI Way - Software Architecture Fundamentals: Technical, Business, and Social Influences. Robert Wojcik. 2012



Copyright 1998 Steven C. McConnell. Reprinted with permission from *Software Project Survival Guide* (Microsoft Press, 1998).



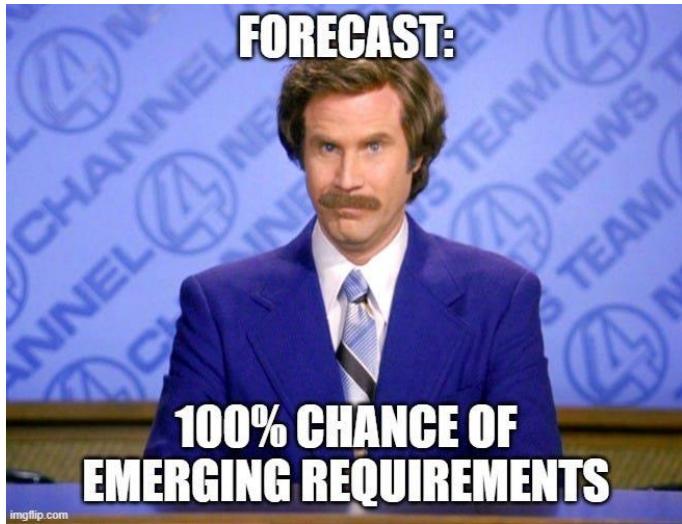
The Twin Peaks Model



B. Nuseibeh, "Weaving together requirements and architectures". 2001

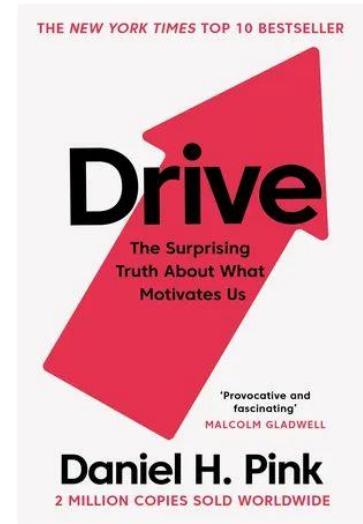
Agile and Architecture

*"The best architectures, requirements, and designs **emerge** from self-organizing teams". The Twelve Principles of the Agile Manifesto*



"Control leads to compliance; autonomy leads to engagement."

Daniel H. Pink

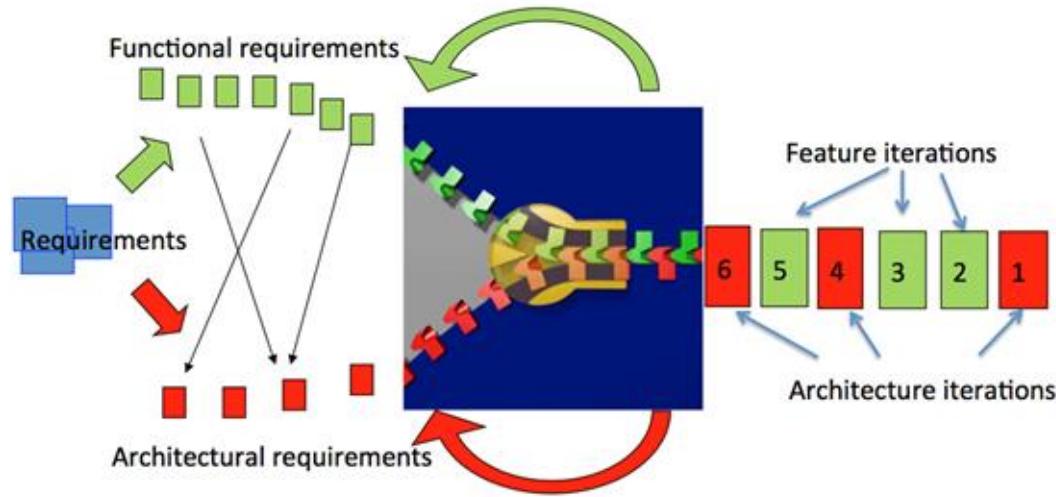


https://medium.com/@beatrix_66005/the-best-architectures-requirements-and-designs-emerge-from-self-organizing-teams-8b54ebc4c6b0

The Zipper Model

How to Agilely Architect an Agile Architecture

by Stephany Bellomo, Philippe Kruchten, Robert L. Nord, and Ipek Ozkaya

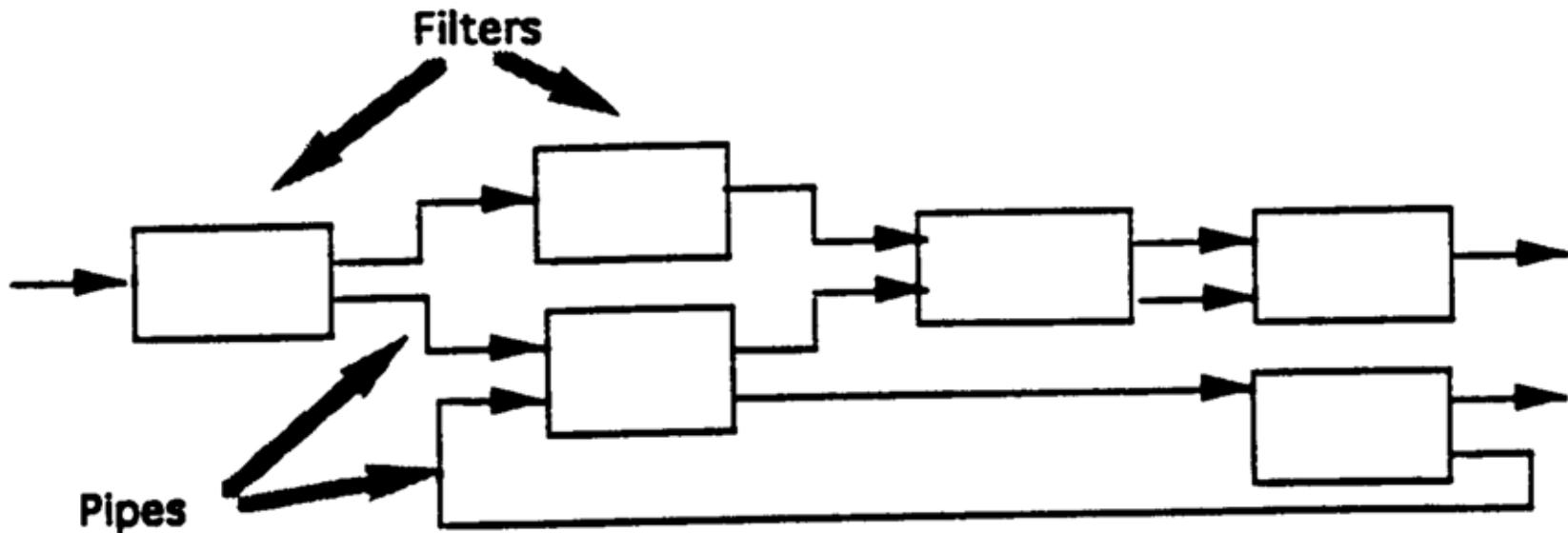


Common Architectural Styles



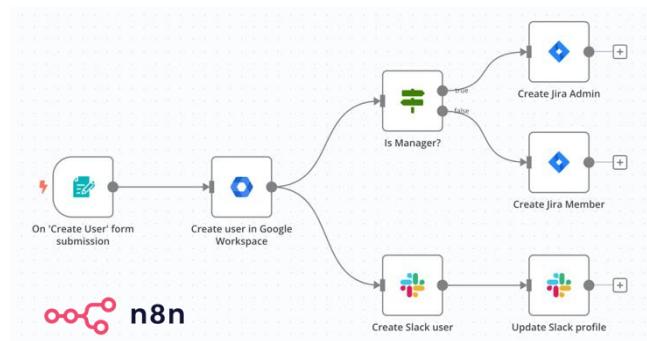
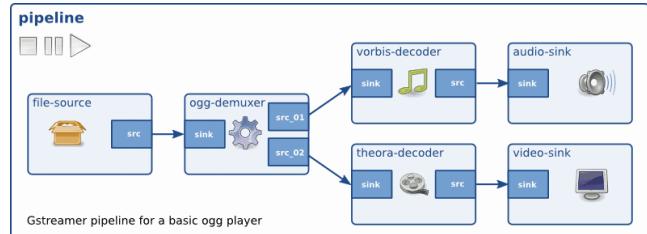
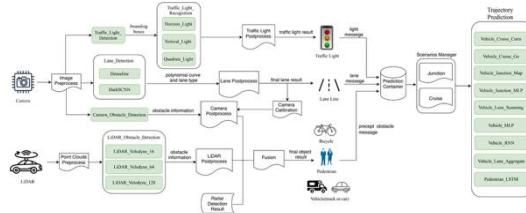
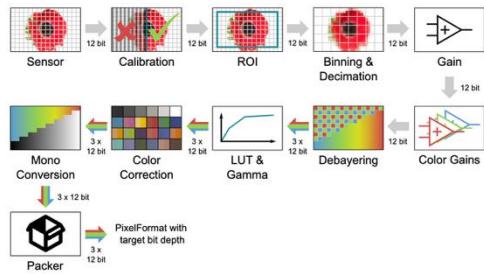
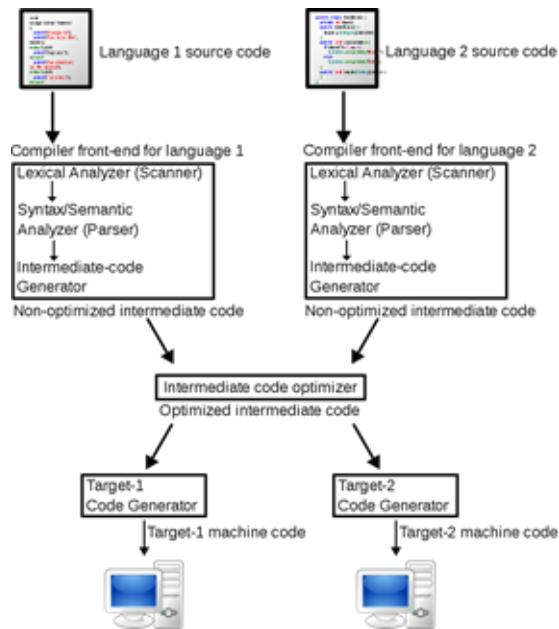
<https://www.thespruce.com/top-architectural-styles-4802083>

Pipes and Filters

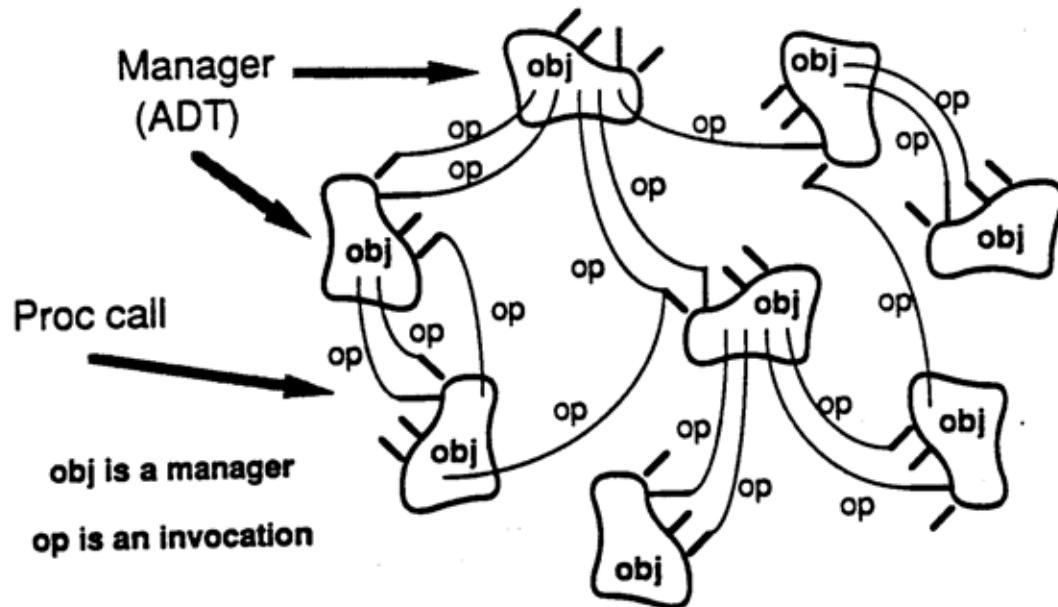


© David Garlan and Mary Shaw, CMU/SEI-94-TR-021

Pipes and Filters in the Wild

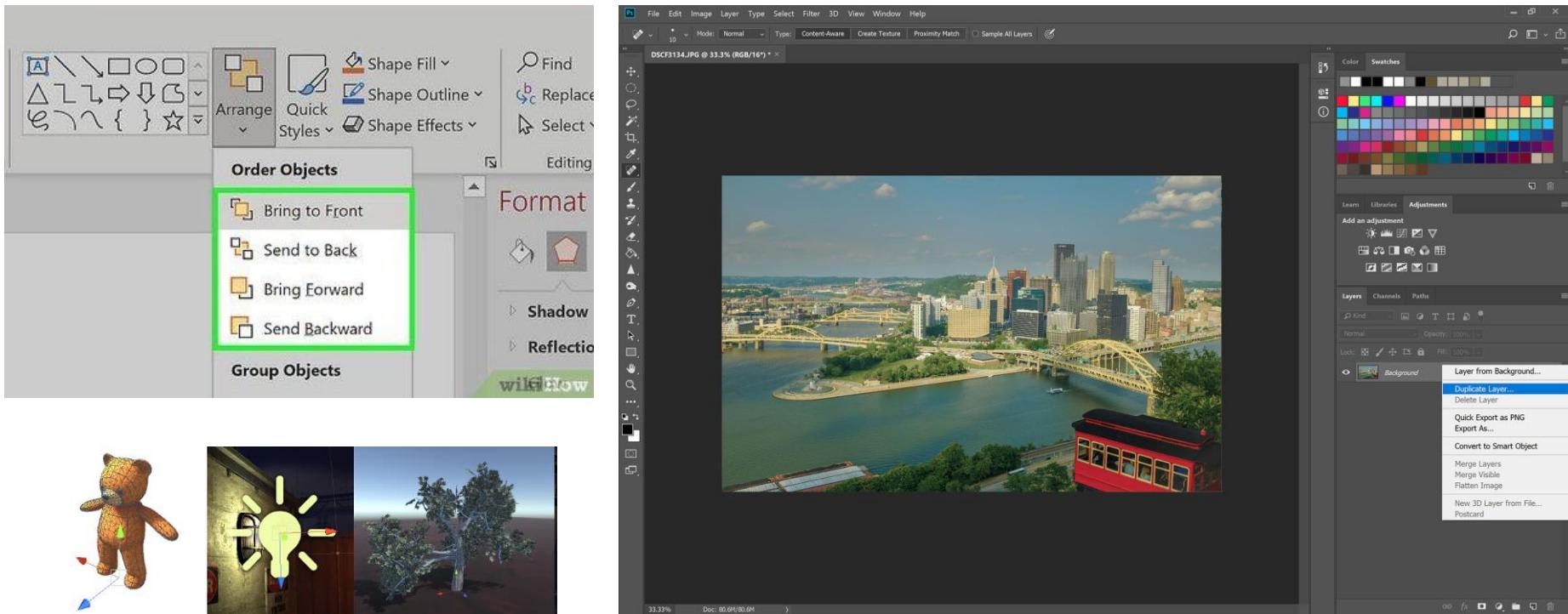


Object-Oriented Organization

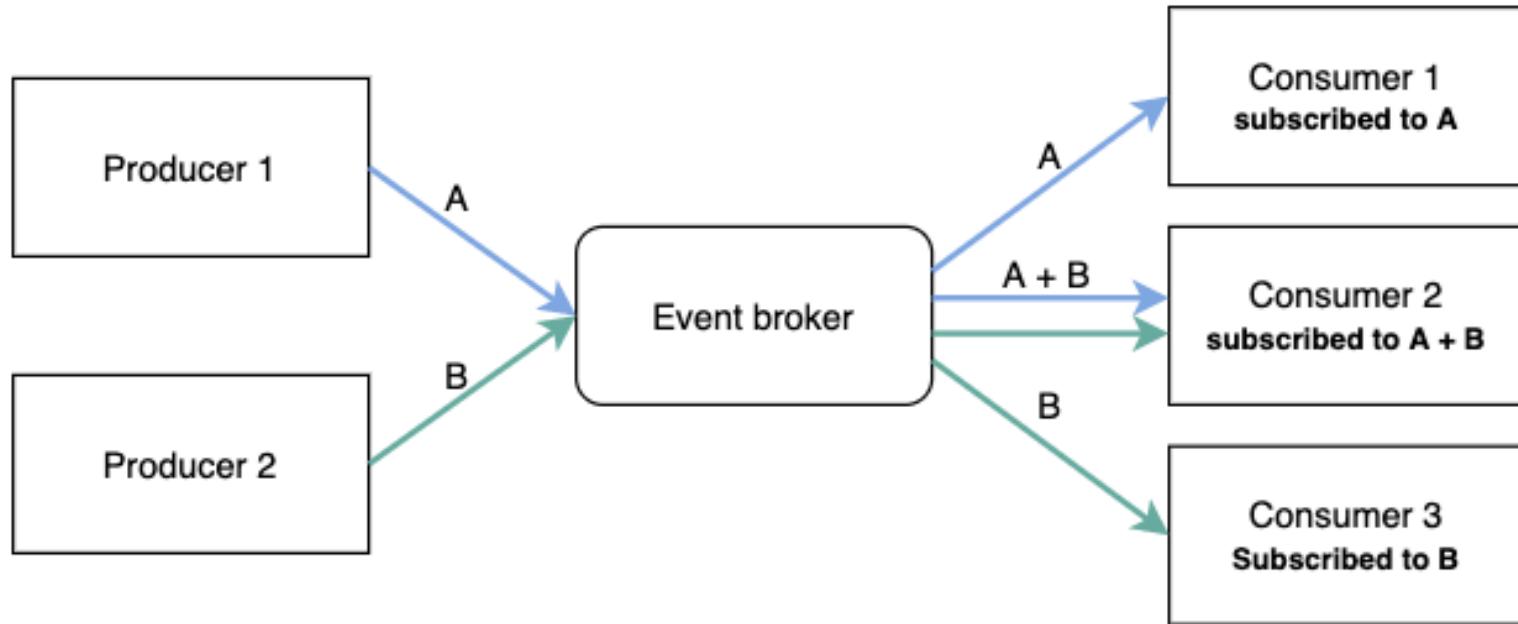


© David Garlan and Mary Shaw, CMU/SEI-94-TR-021

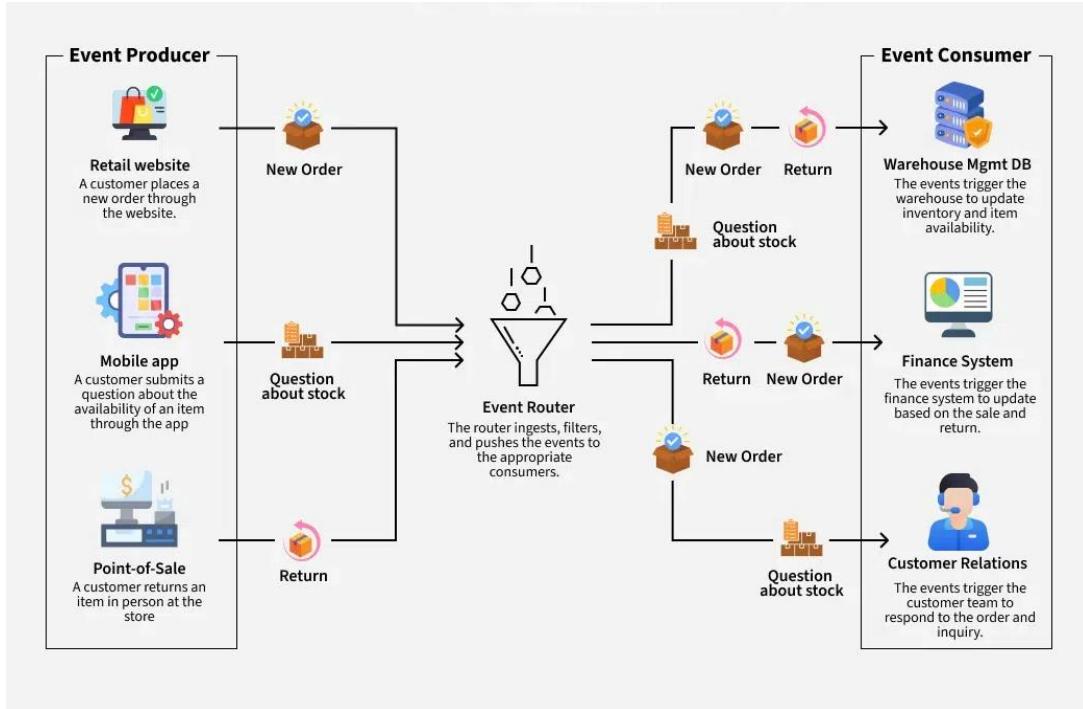
Object-Oriented Organization in the Wild



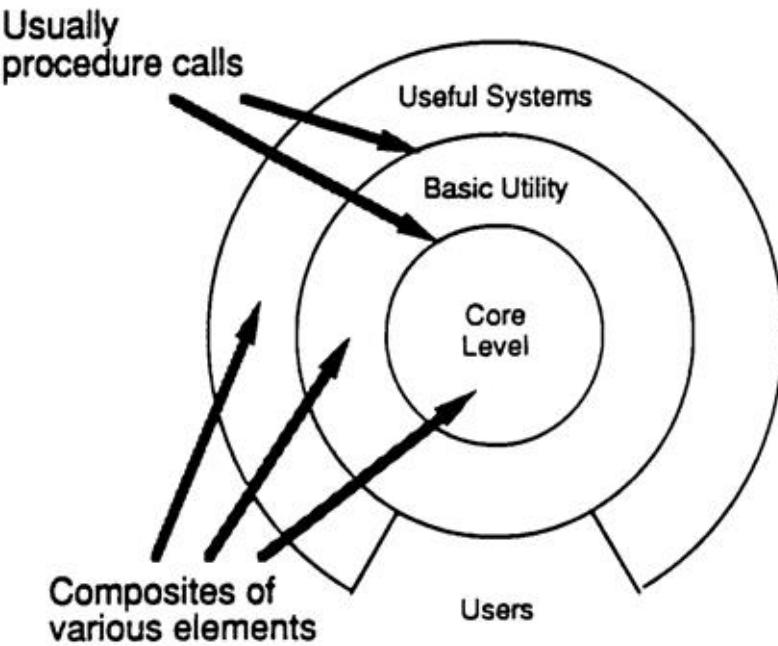
Event-Driven Architecture



Event-Driven Architectures in the Wild

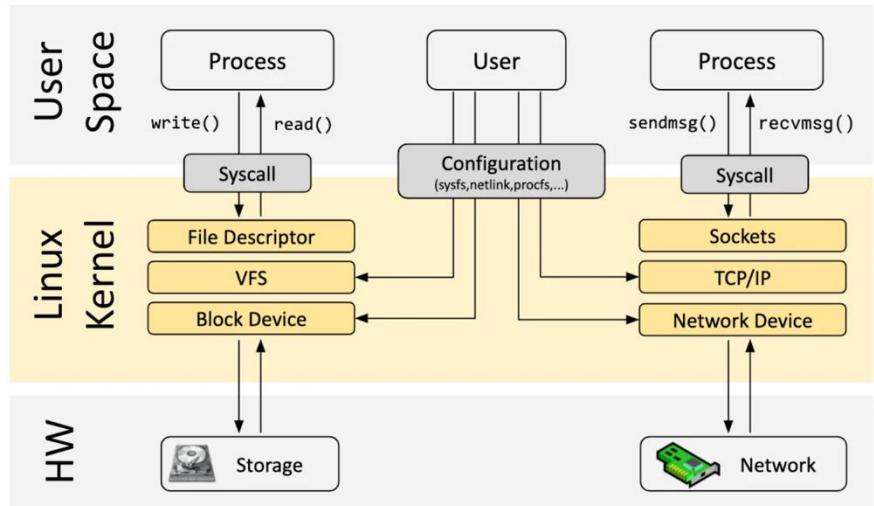
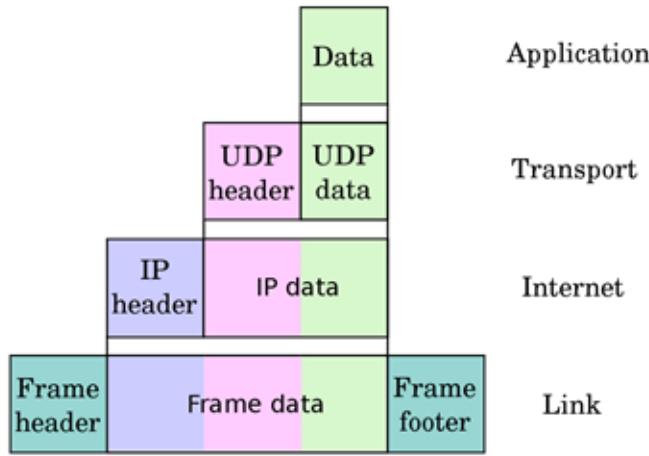


Layered Systems



© David Garlan and Mary Shaw, CMU/SEI-94-TR-021

Layered Systems in the Wild



Why Document Architecture?

- Blueprint for the system
 - Artifact for early analysis
 - Primary carrier of quality attributes
 - Key to post-deployment maintenance and enhancement
- Documentation speaks for the architect, both today and 20 years from today
 - As long as the system is built, maintained, and evolved according to its documented architecture
- Support traceability



Btw, I'd like to apologize for Twitter being super slow in many countries.
App is doing >1000 poorly batched RPCs just to render a home timeline!

1:00 PM - Nov 13, 2022

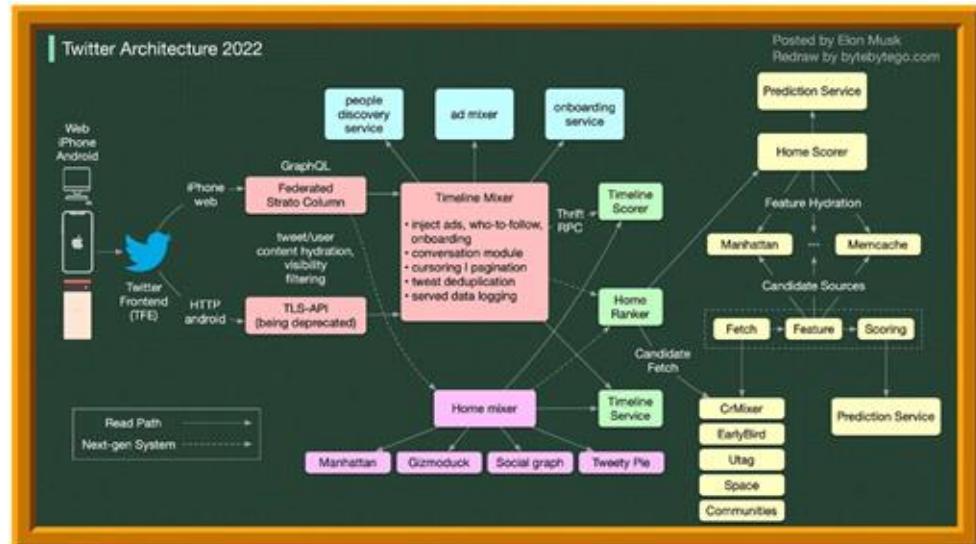


Just leaving Twitter HQ code review



4:28 AM - Nov 19, 2022

36.9K Retweets 16.1K Quote Tweets 464K Likes



Guidelines for selecting a notation

- Suitable for purpose
- Often visual for compact representation
- Usually, boxes and arrows
- UML possible (semi-formal), but possibly constraining
 - Note the different abstraction level – Subsystems or processes, not classes or objects
- Formal notations available
- Decompose diagrams hierarchically and in views
- Always include a legend
- Define precisely what the boxes mean
- Define precisely what the lines mean
- Do not try to do too much in one diagram
 - Each view of architecture should fit on a page
 - Use hierarchy

