Introduction to Software Architecture

17-313 Fall 2024

Foundations of Software Engineering

https://cmu-313.github.io

Michael Hilton and Rohan Padhye

Administrivia

- Project 2B due tonight (Sep 24th)
- Project 2B: UI changes require theme repo
  - New instructions on Slack (see “#fall-24-announcements”)
  - Tl;dr – You need to clone and modify a separate repo for updating some front-end components (menus, sidebar, etc.). Submit both repos.
  - This is an ***excellent lesson*** in software architecture (this lecture)
  - Due to the delay in releasing new instructions, we will not penalize team members for missing front-end commits in Sprint 1 (but do it if you can)
    - Make sure to document your challenges in the issue/reflections. See Slack for more.

Smoking Section

- Last **two** full rows

Learning Goals

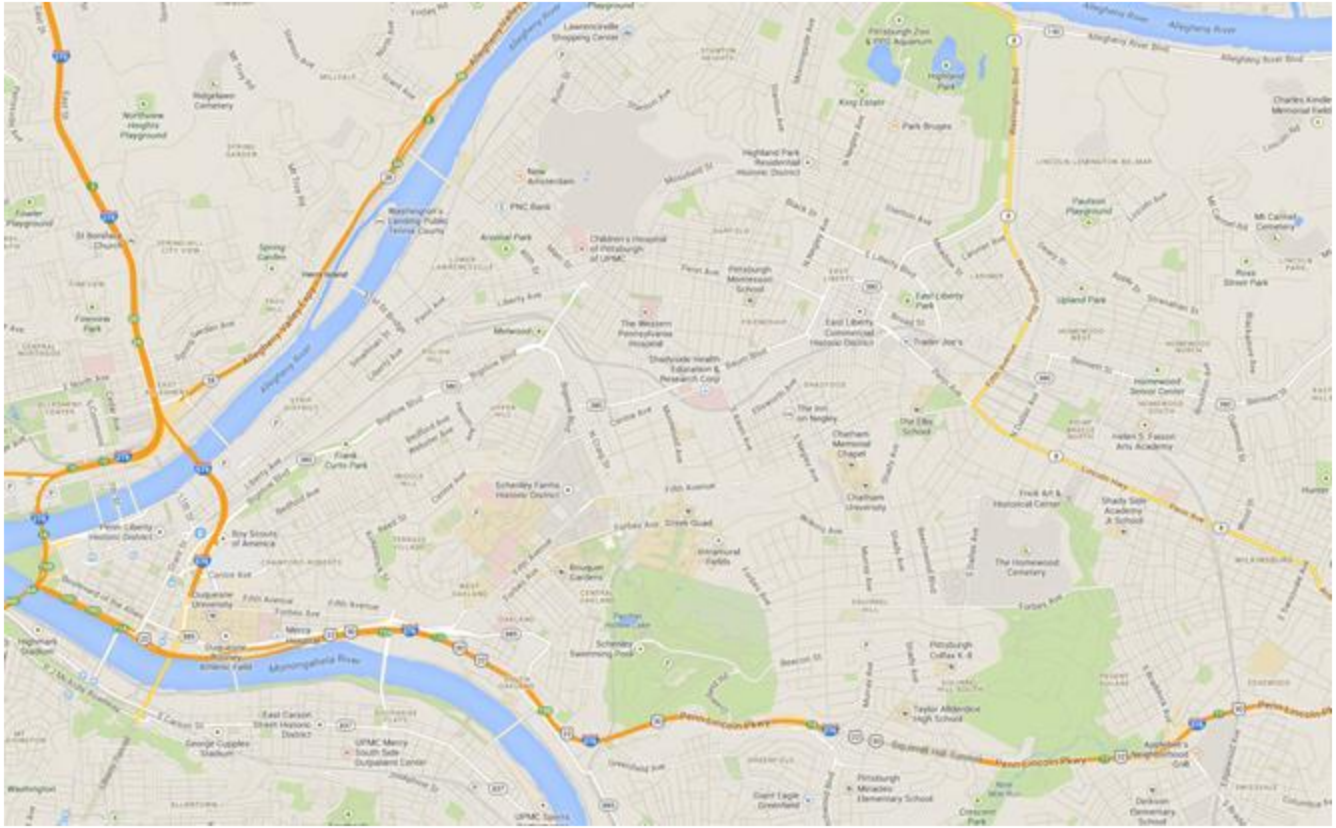
- Understand the abstraction level of architectural reasoning
- Appreciate how software systems can be viewed at different abstraction levels
- Distinguish software architecture from (object-oriented) software design
- Explain the importance of architectural decisions
- Integrate architectural decisions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

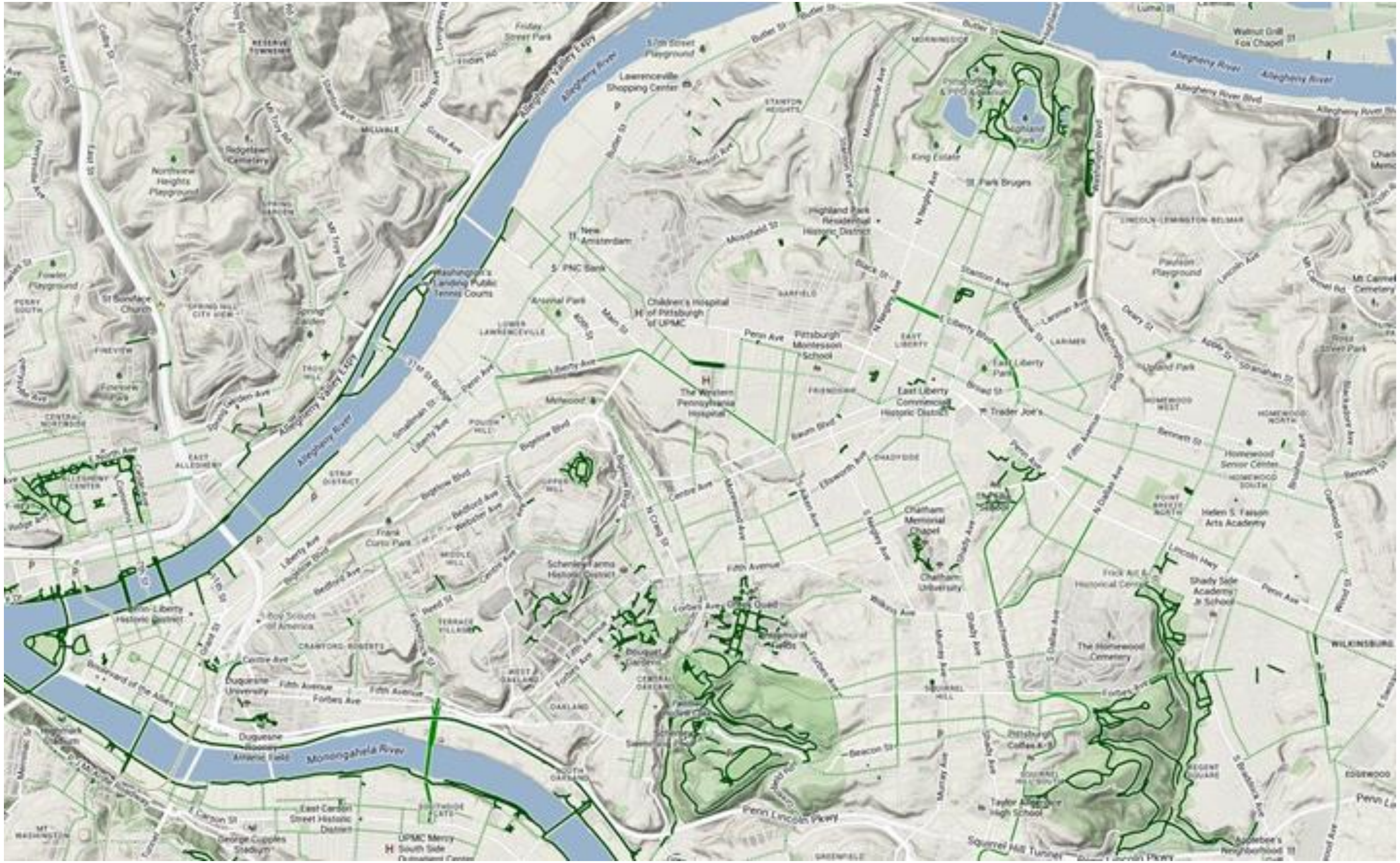
Outline

- Views and Abstraction
- Case Study: Autonomous Vehicles
- Software Architecture
  - Definitions, Importance
  - Software Design vs. Software Architecture
- Architecting software
  - Integrating Architectural Decisions into the SW Development Process
  - Common Software Architectures
  - Documentation

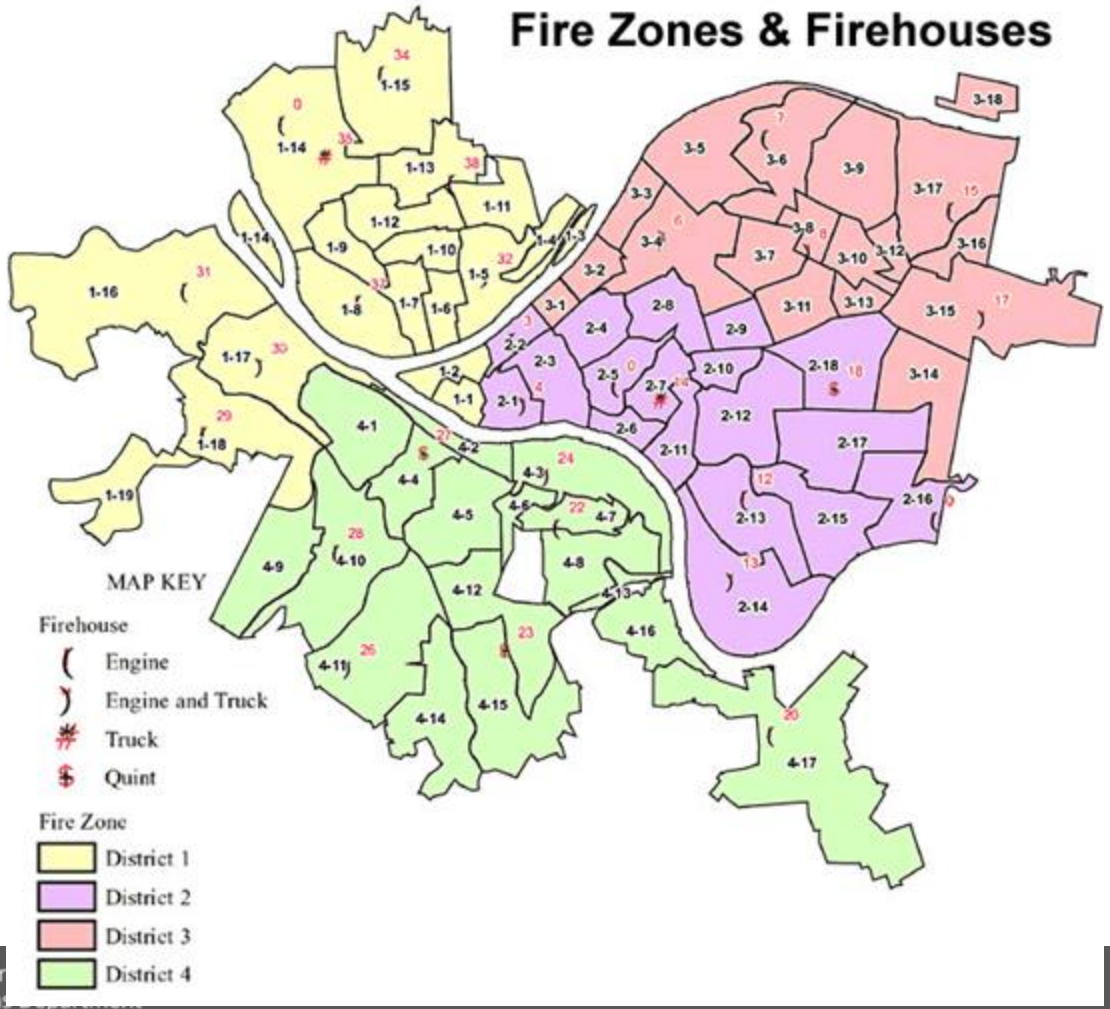
Outline

- Views and Abstraction
- Case Study: Autonomous Vehicles
- Software Architecture
  - Definitions, Importance
  - Software Design vs. Software Architecture
- Architecting software
  - Integrating Architectural Decisions into the SW Development Process
  - Common Software Architectures
  - Documentation

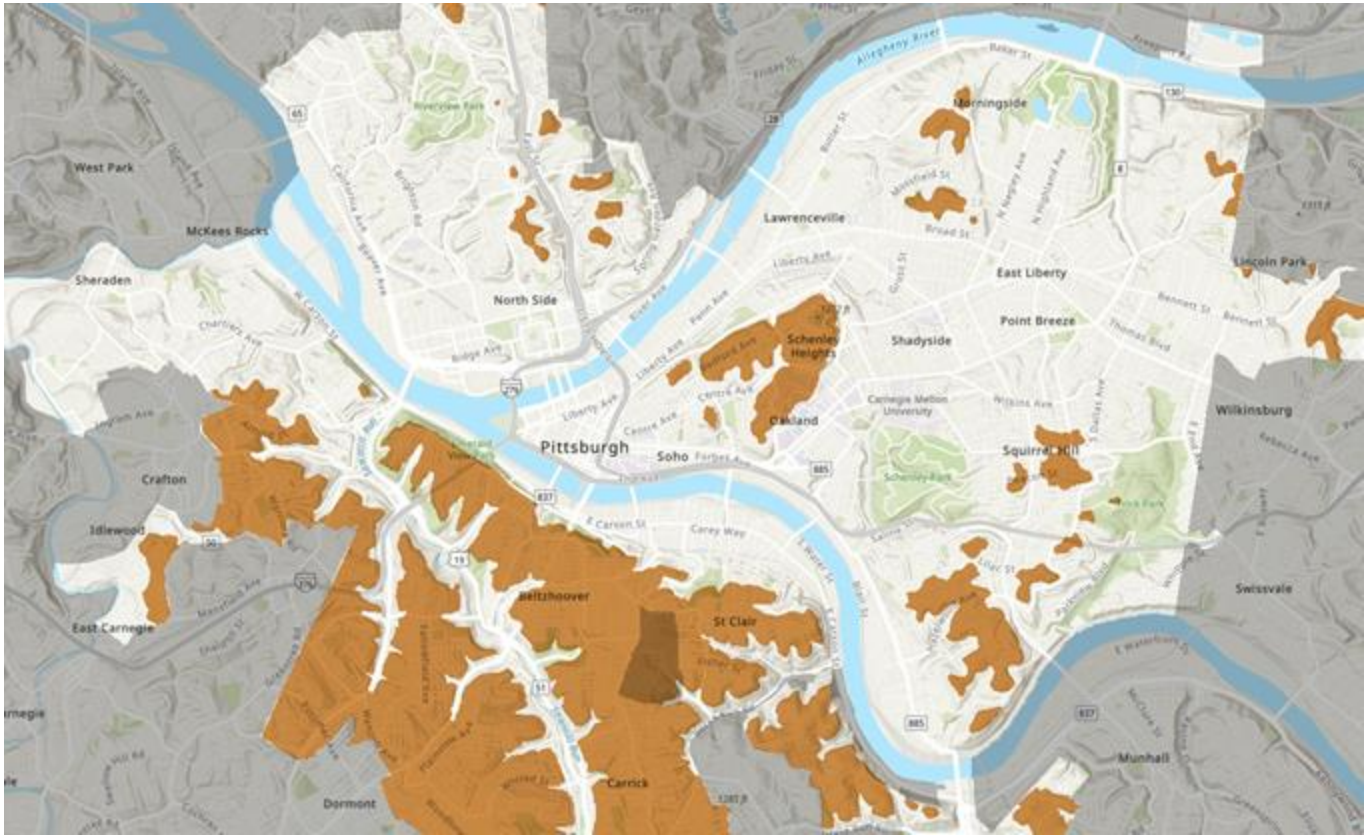

Allegheny River
Monongahela River


Allegheny River
Monongahela River
Fifth Avenue
Forbes Ave
Penn Ave
Penn Lincoln Pkwy
Squirrel Hill Tunnel
WILKINSBURG

Fire Zones & Firehouses




Pittsburgh
Zoning Map



Source: Pittsburgh Zoning Map (https://gis.pittsburghpa.gov/pghzoning/)

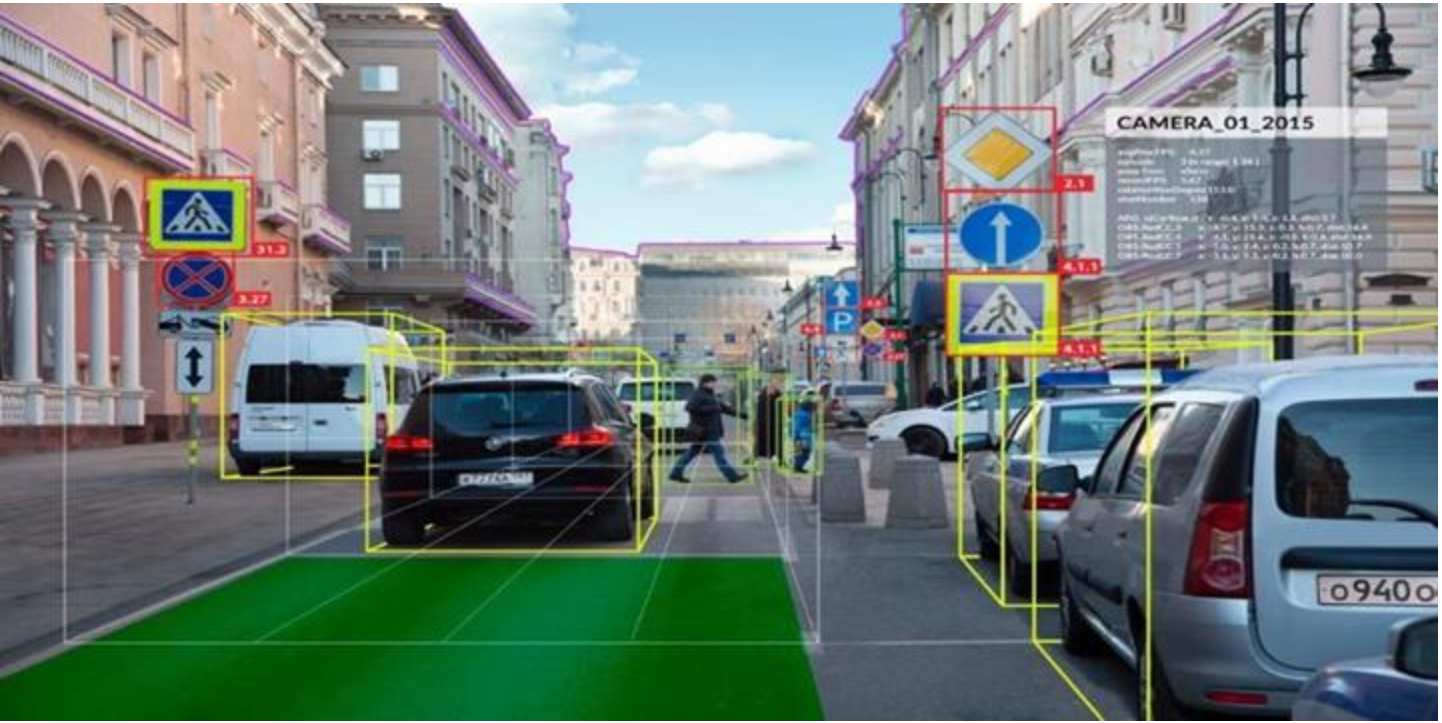
Abstracted views focus on conveying specific information

- They have a well-defined purpose
- Show only necessary information
- Abstract away unnecessary details
- Use legends/annotations to remove ambiguity
- Multiple views of the same object tell a larger story

Outline

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

Case Study: Autonomous Vehicle Software



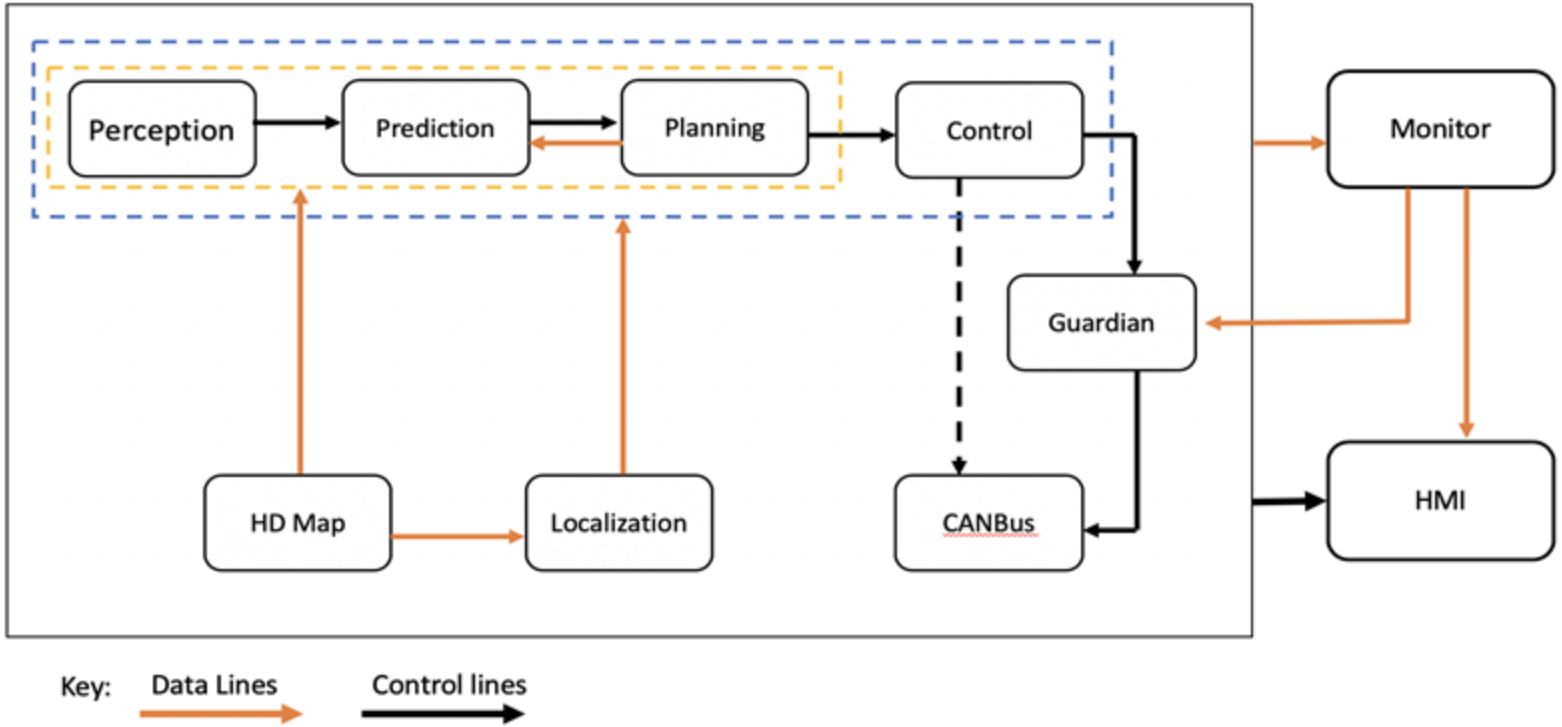
Case Study: Apollo

Source: https://github.com/ApolloAuto/apollo

Check out the “side pass” feature from the video:
https://www.youtube.com/watch?v=BXNDUtNZdM4

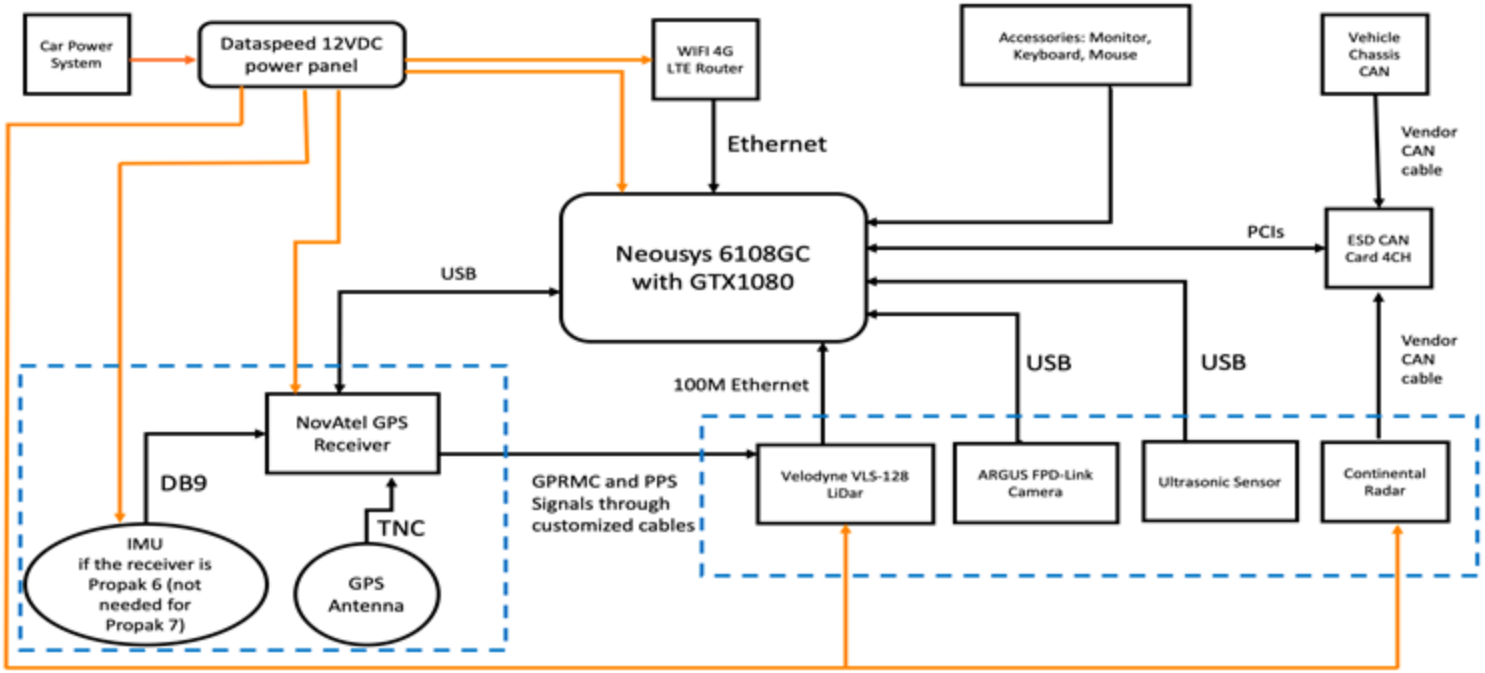
- Identify in teams of 3 what parts are associated with the **side pass feature**
- Remember to write down your names and Andrew IDs

Apollo Software Architecture



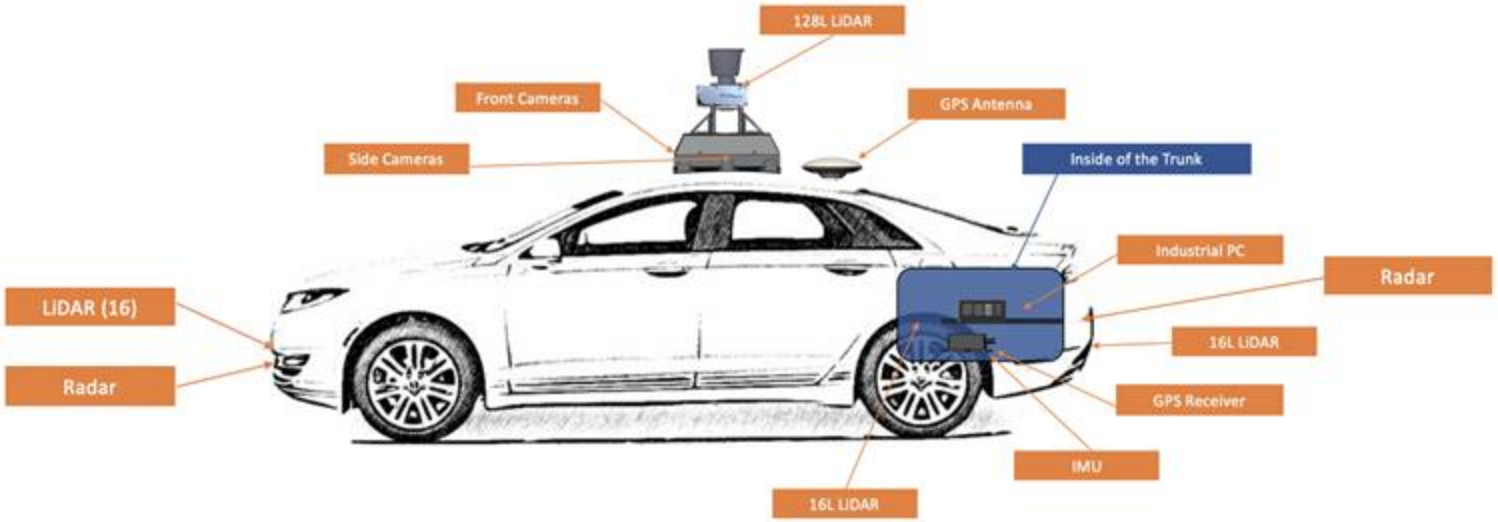
Source: https://github.com/ApolloAuto/apollo/blob/v6.0.0/docs/specs/Apollo_5.5_Software_Architecture.md

Apollo Hardware Architecture



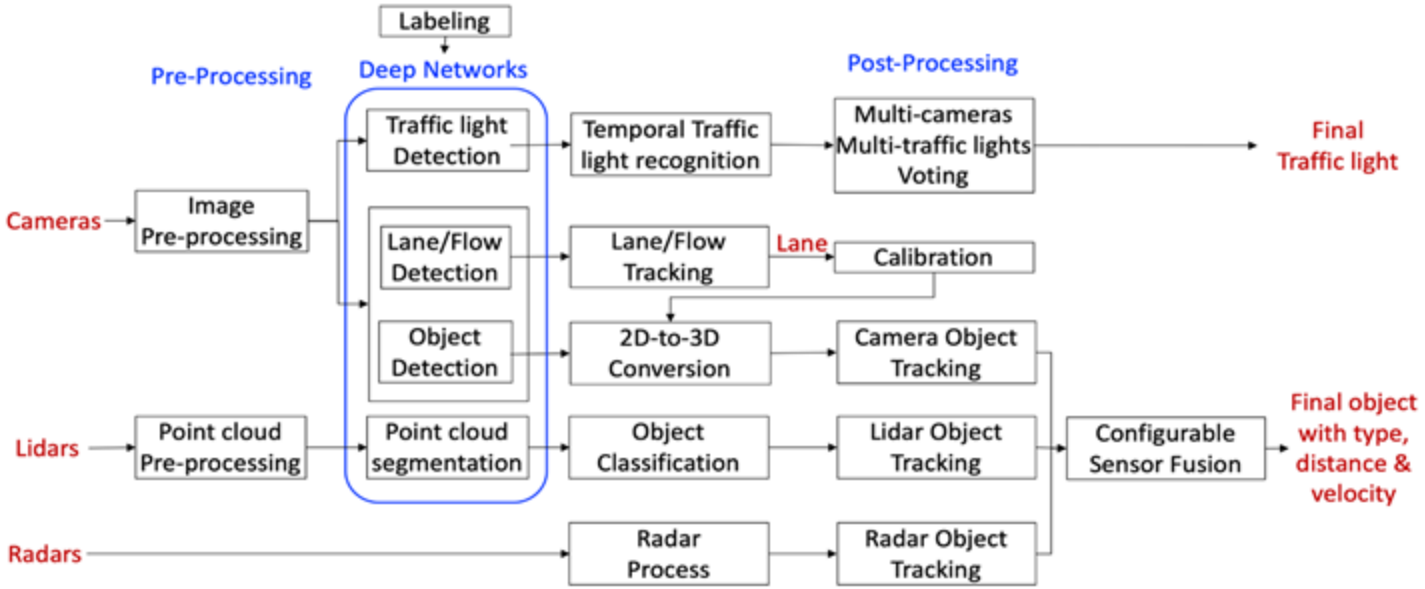
Source: https://github.com/ApolloAuto/apollo/blob/v6.0.0/README.md

Apollo Hardware/Vehicle Overview

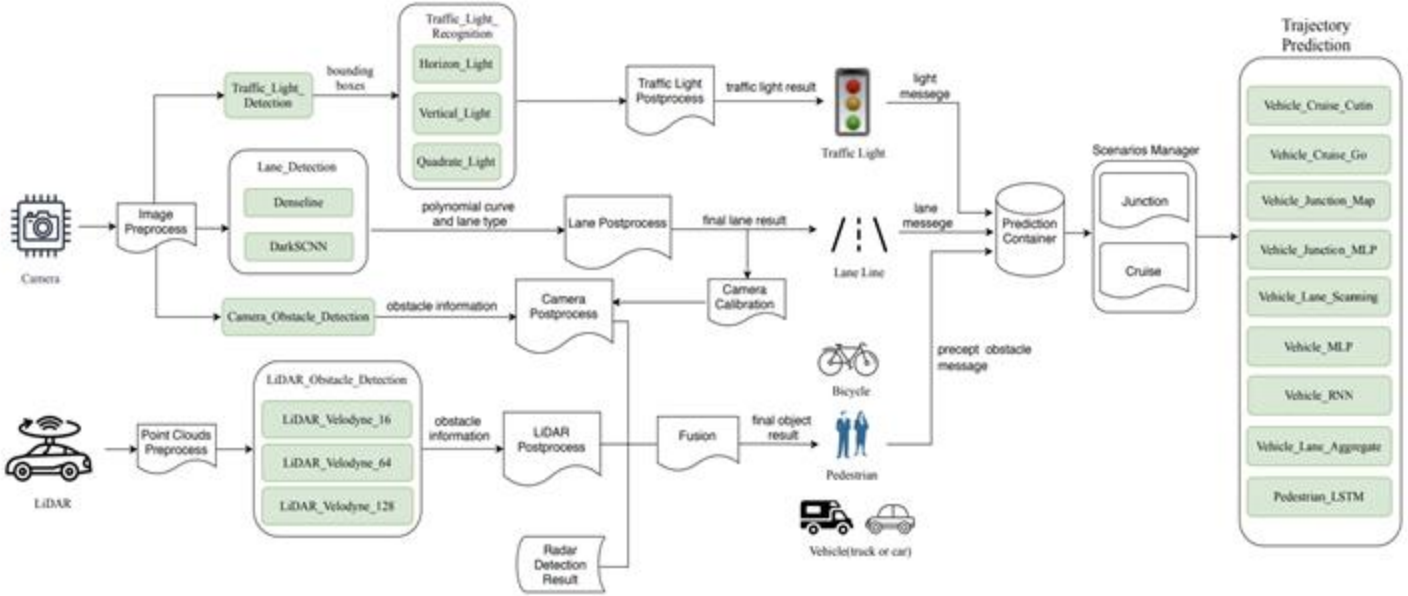


Source: https://github.com/ApolloAuto/apollo/blob/v6.0.0/README.md

Apollo Perception Module

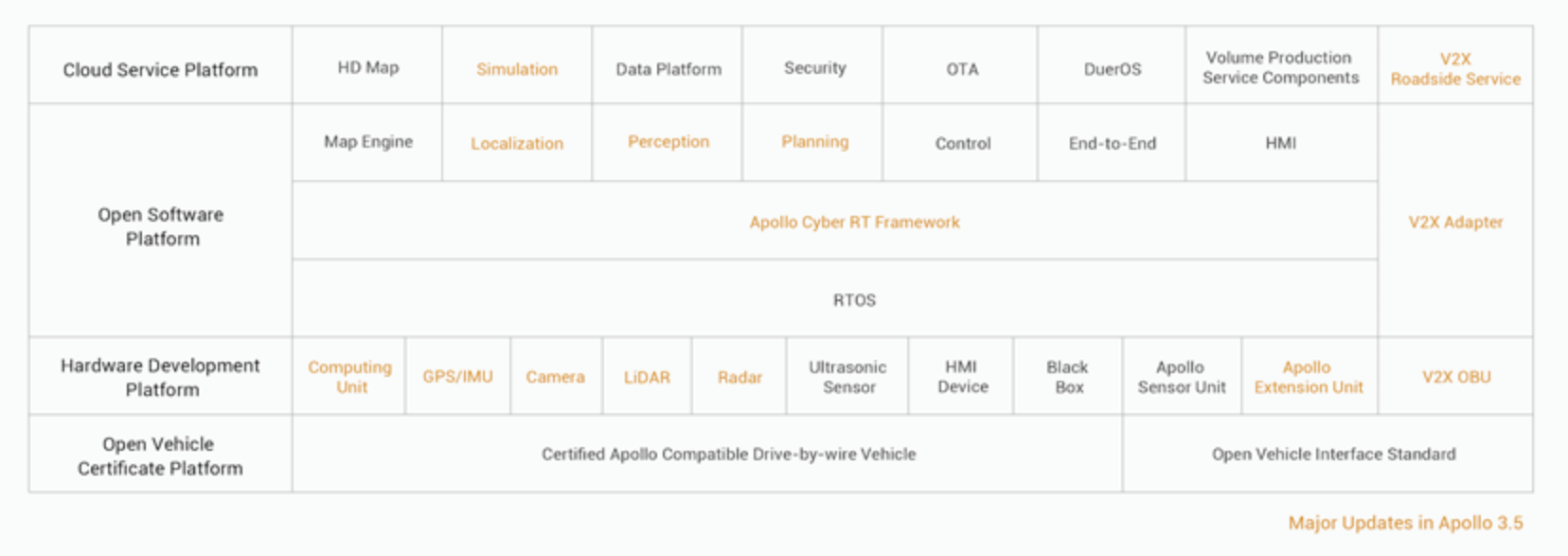


Apollo ML Models



Source: Zi Peng, Jinqiu Yang, Tse-Hsun (Peter) Chen, and Lei Ma. 2020. A First Look at the Integration of Machine Learning Models in Complex Autonomous Driving Systems: A Case Study on Apollo. In Proceedings of the 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '20), https://doi.org/10.1145/ 3368089.3417063

Apollo Software Stack



Source: https://github.com/ApolloAuto/

Outline

- Views and Abstraction
- Case Study: Autonomous Vehicles
- **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.

[Bass et al. 2003]

Note: this definition is ambivalent to whether the architecture is known or whether it's any good!

Software 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 does NodeBB support custom themes**?
- 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?

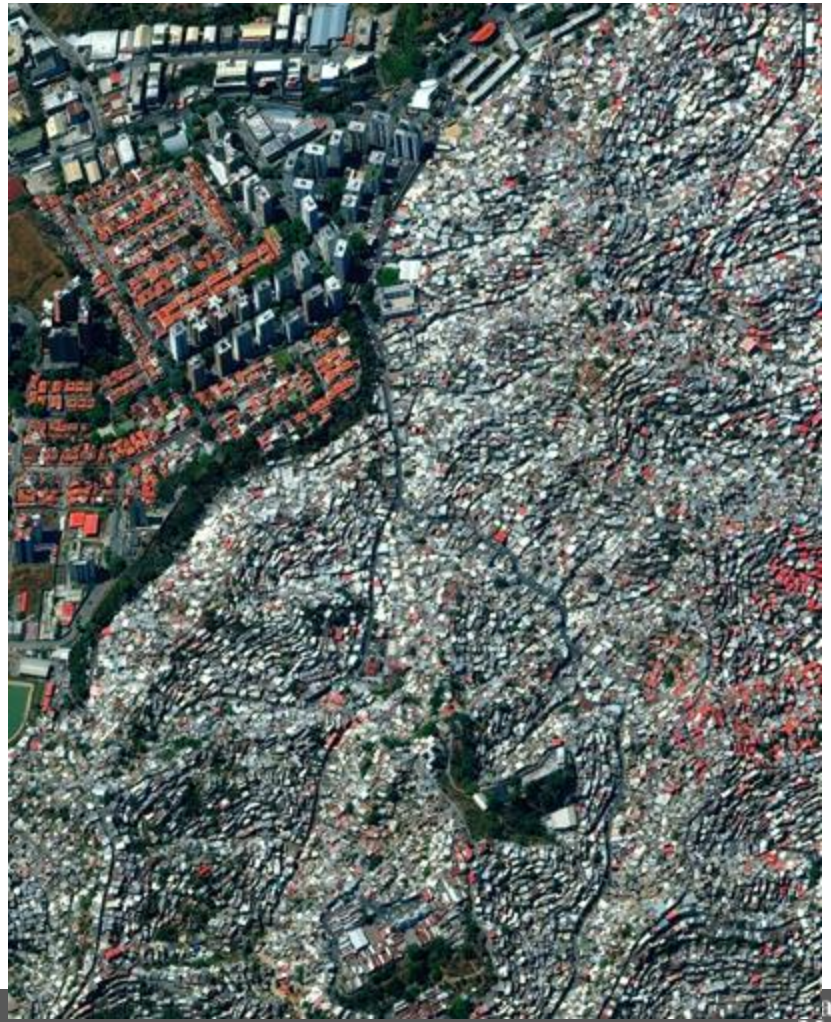
Outline

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

https://www.archdaily.com/

https://www.instagram.com/architectanddesign

https://www.mykonosceramica.com/



www.over-view.com

Every software system has an architecture

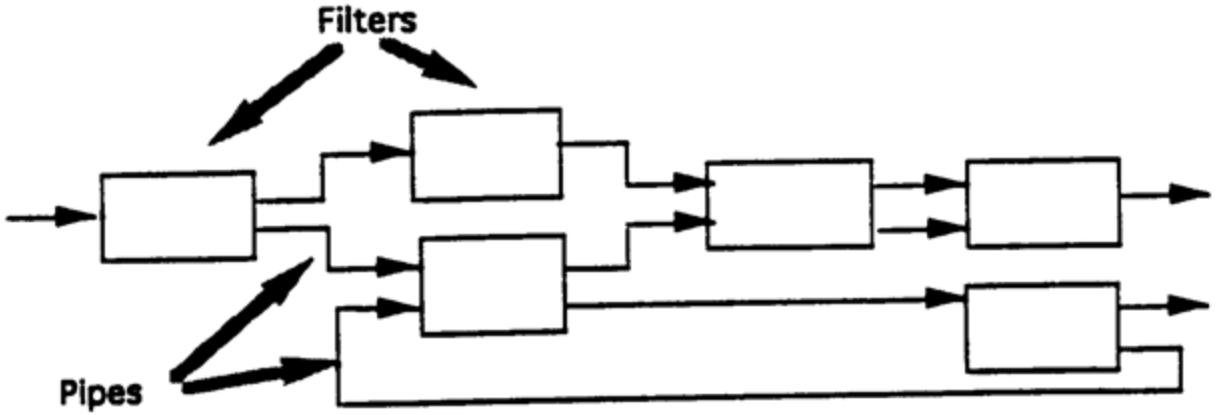
- Whether you know it or not
- Whether you like it or not
- Whether it's documented or not

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

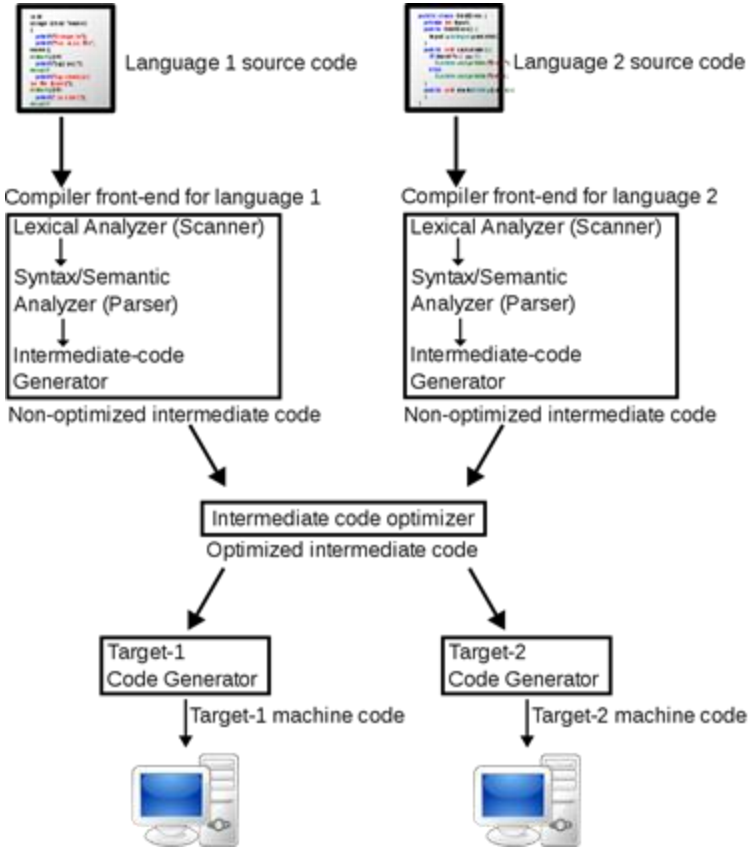
Common Software Architectures

1. Pipes and Filters

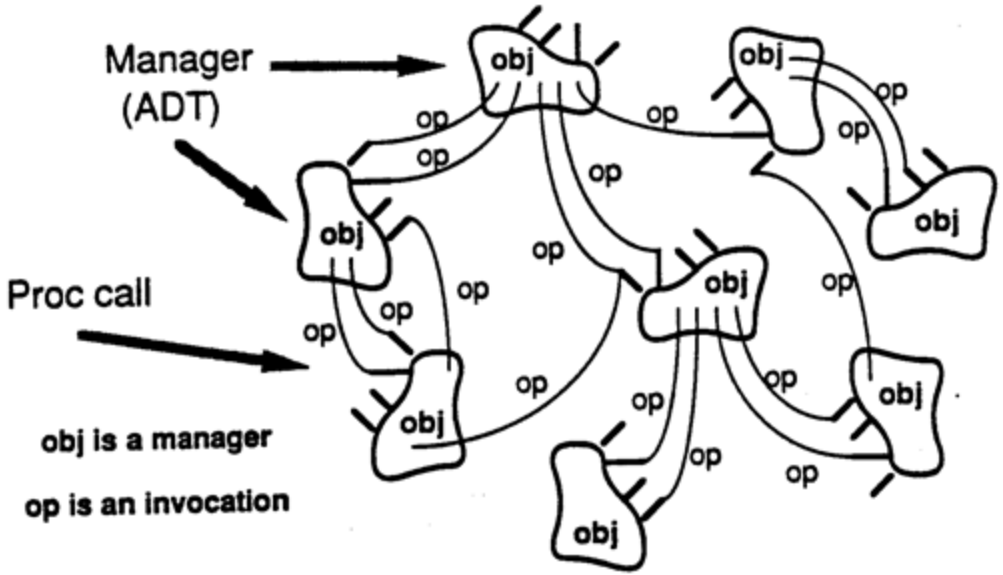


© David Garlan and Mary Shaw, CMU/SEI-94-TR-021

Example: Compilers

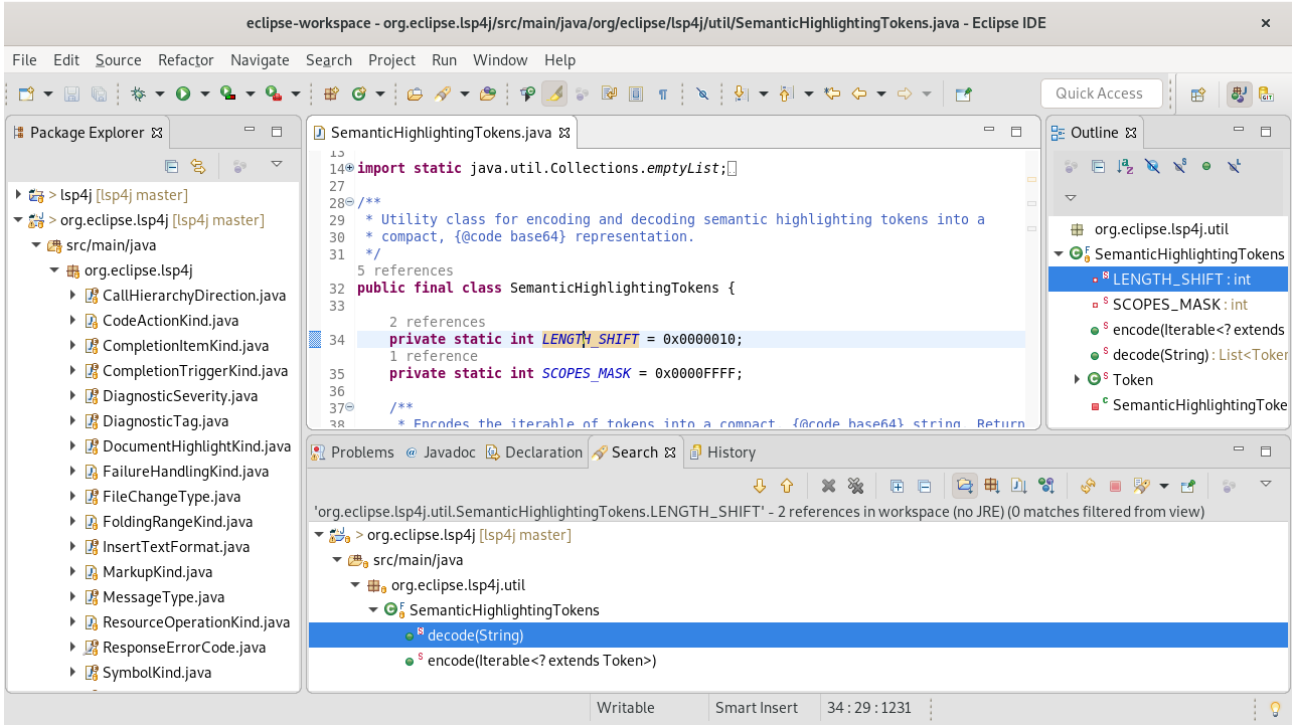


2. Object-Oriented Organization

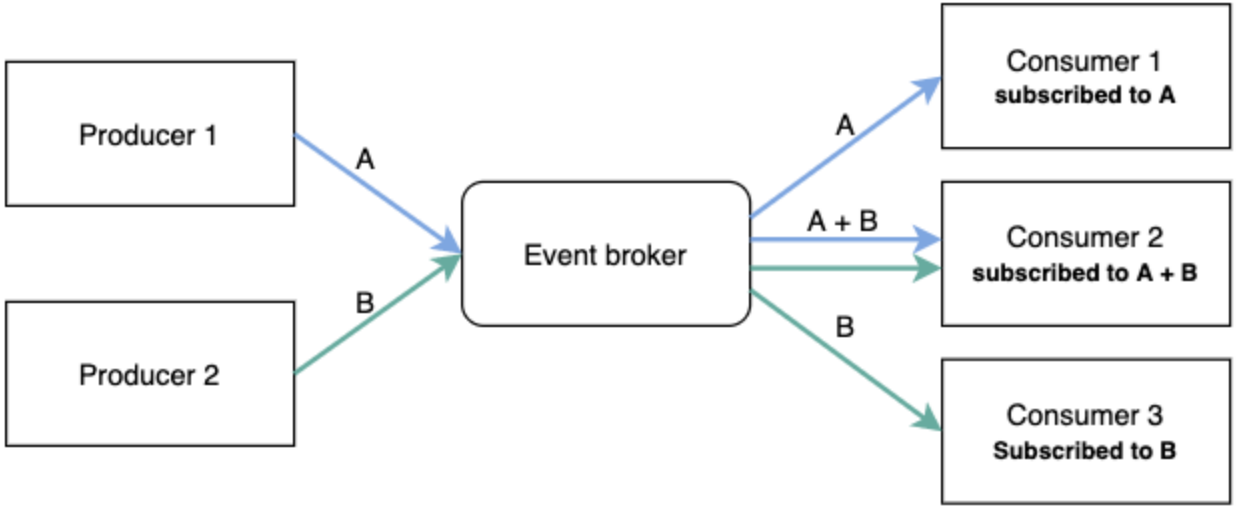


© David Garlan and Mary Shaw, CMU/SEI-94-TR-021

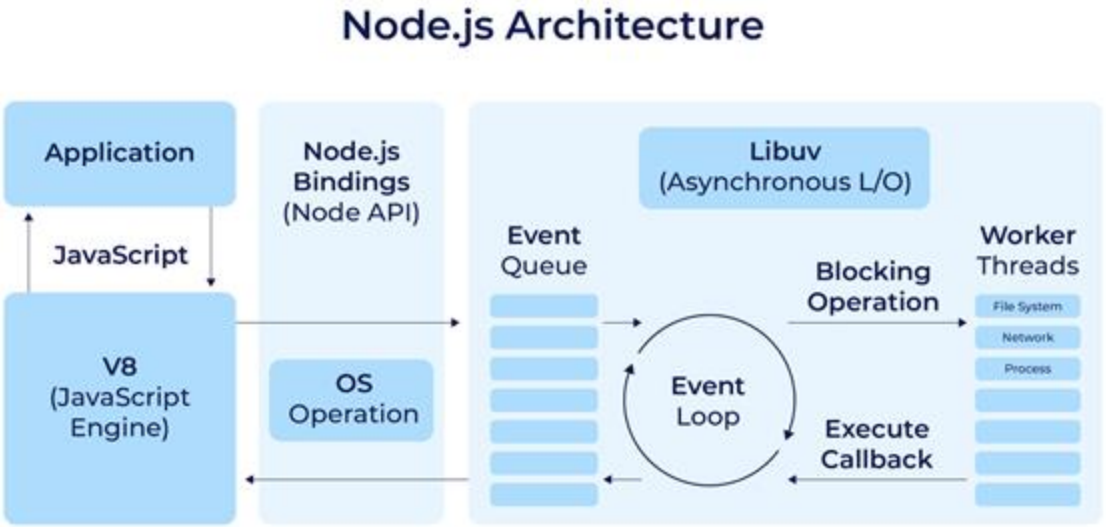
Example: Eclipse IDE



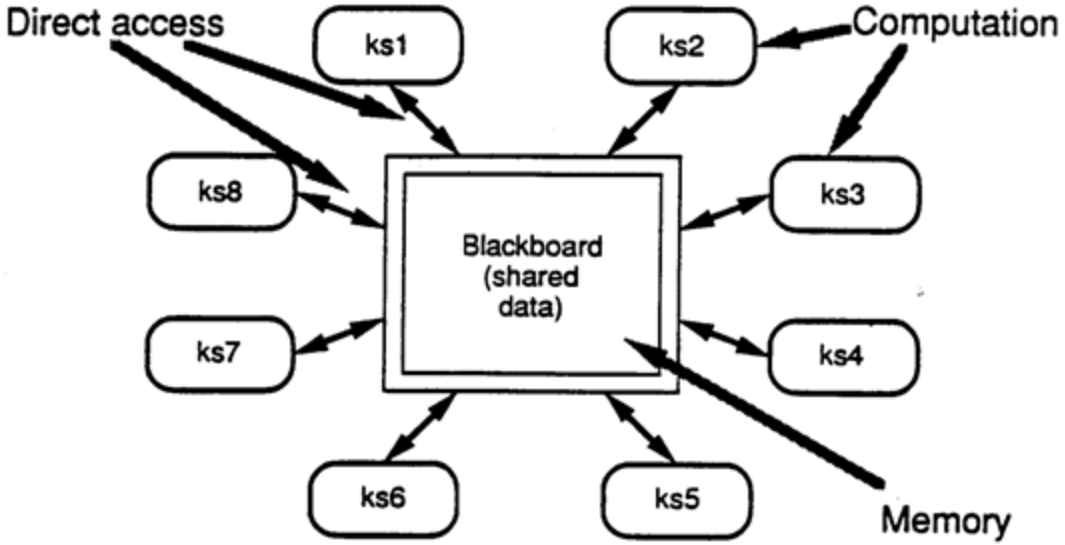
3. Event-Driven Architecture



Example: Node.js

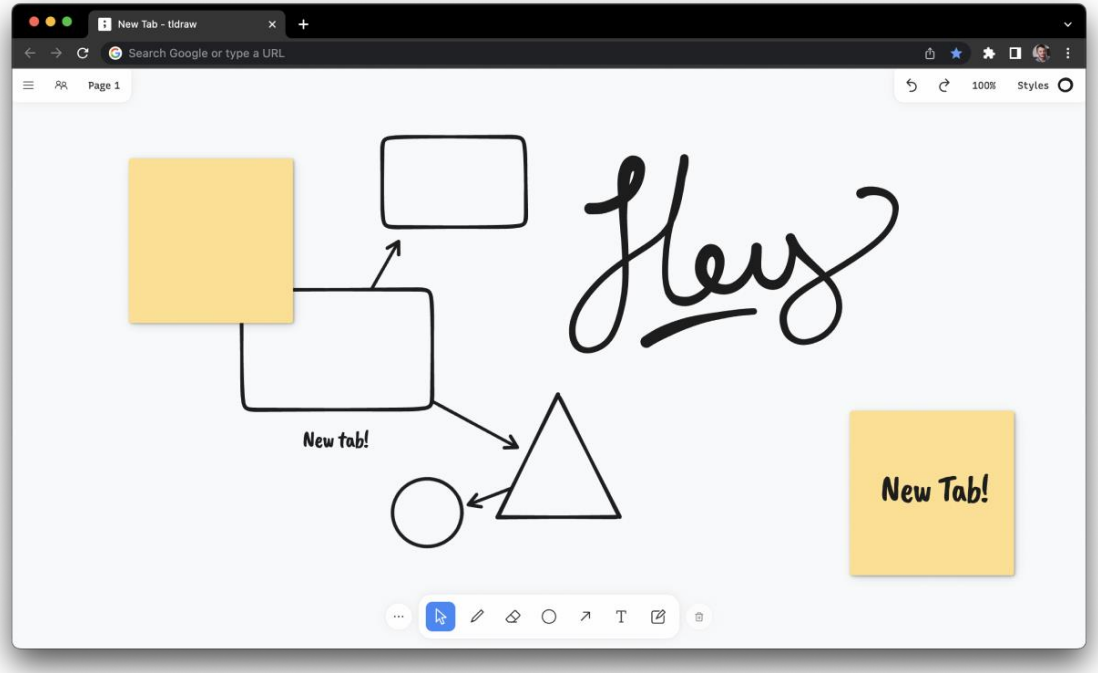


4. Blackboard Architecture

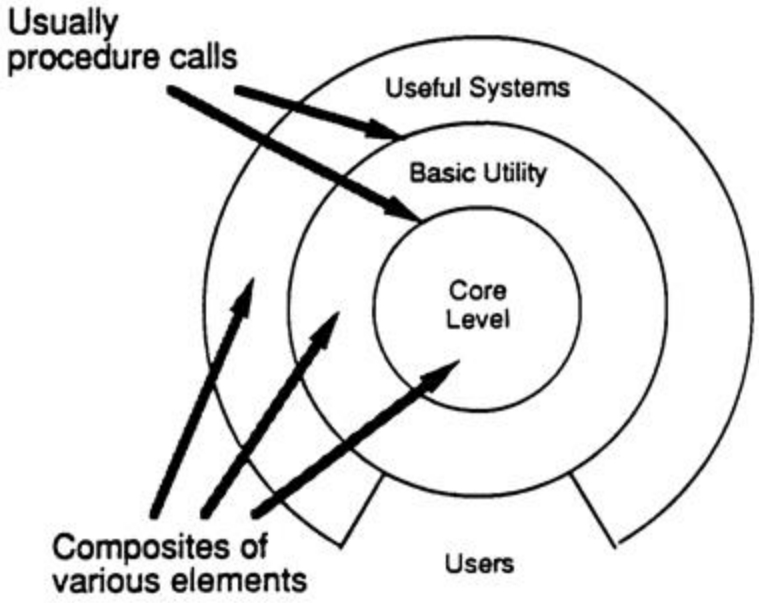


© David Garlan and Mary Shaw, CMU/SEI-94-TR-021

Example: tldraw

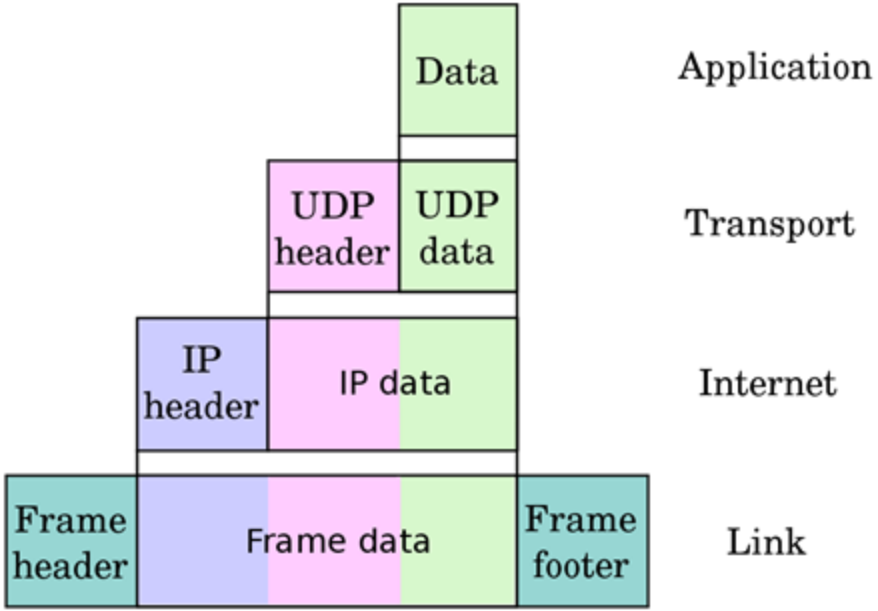


5. Layered Systems



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

Example: Internet Protocol Suite

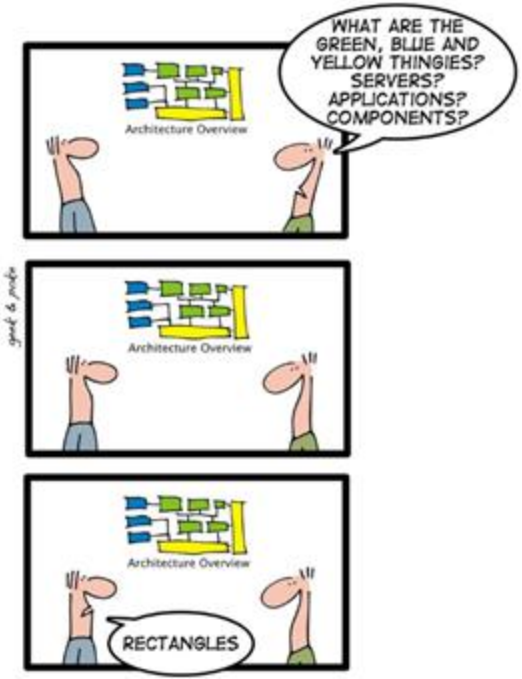


Why Document Architecture?

- 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, today and 20 years from today
  - As long as the system is built, maintained, and evolved according to its documented architecture
- Support traceability.

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



Aside: NodeBB Themes Architecture and Project 2