

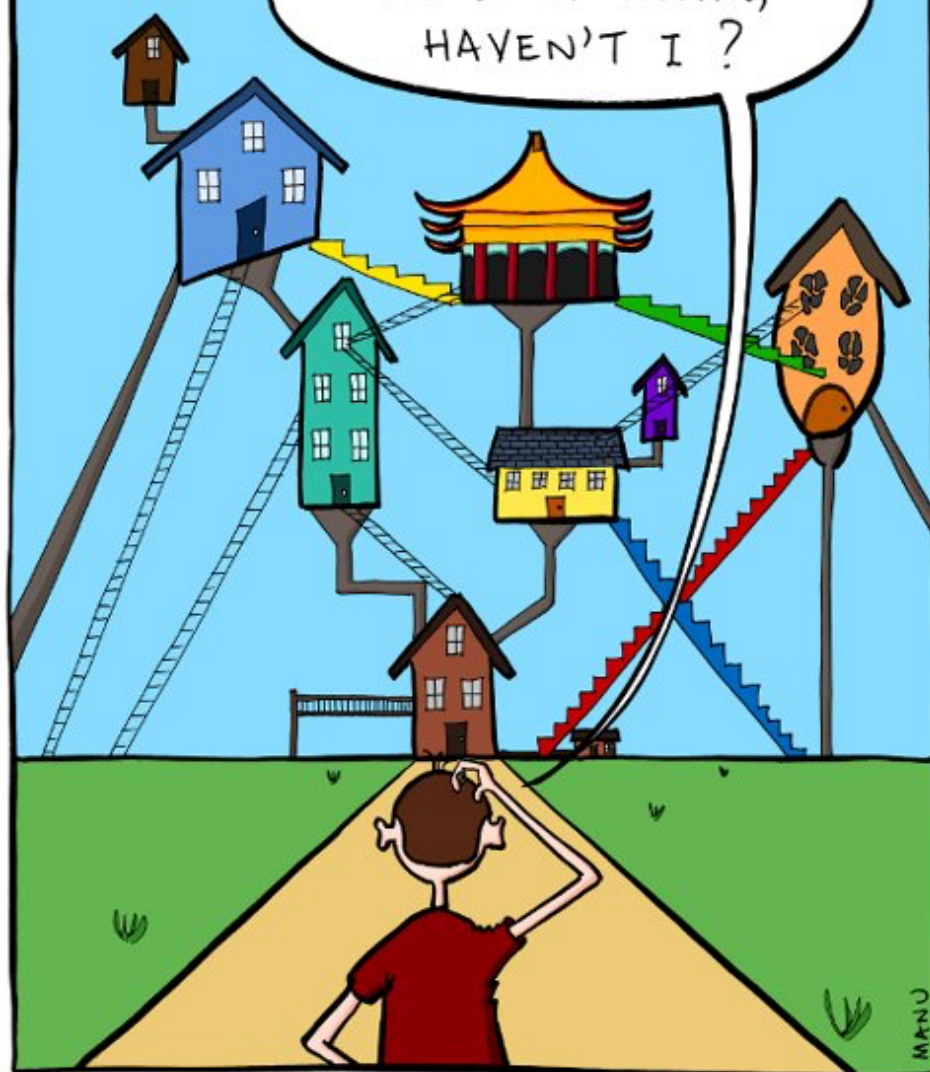
THE LIFE OF A SOFTWARE
ENGINEER.

CLEAN SLATE. SOLID
FOUNDATIONS. THIS TIME
I WILL BUILD THINGS THE
RIGHT WAY.

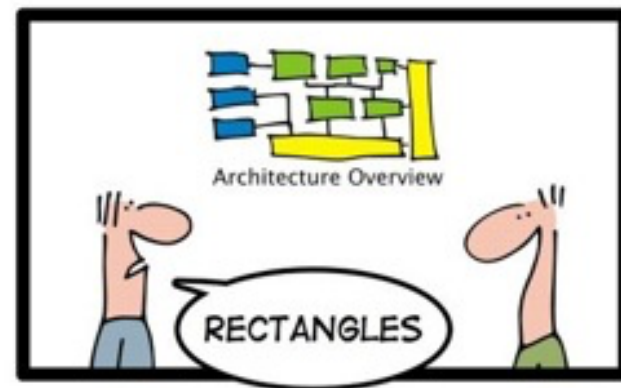
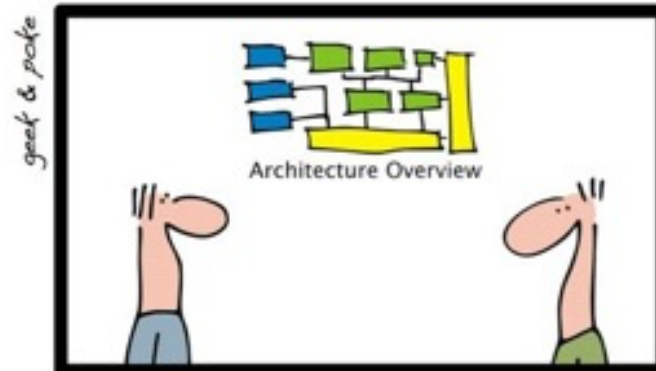
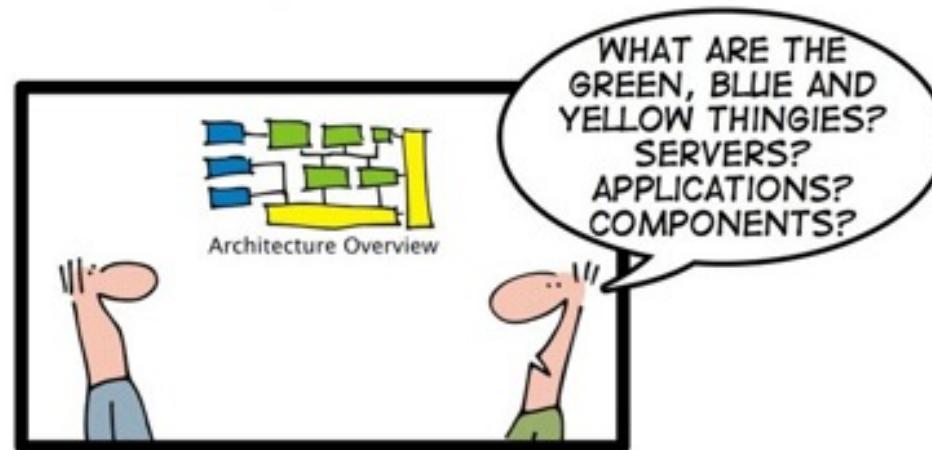


MUCH LATER...

OH MY. I'VE
DONE IT AGAIN,
HAVEN'T I ?



ENTEPRISE ARCHITECTURE MADE EASY



PART 1: DON'T MESS WITH THE
GORY DETAILS

Introduction to Software Architecture

Michael Coblenz



Slide credit: Michael Hilton at CMU

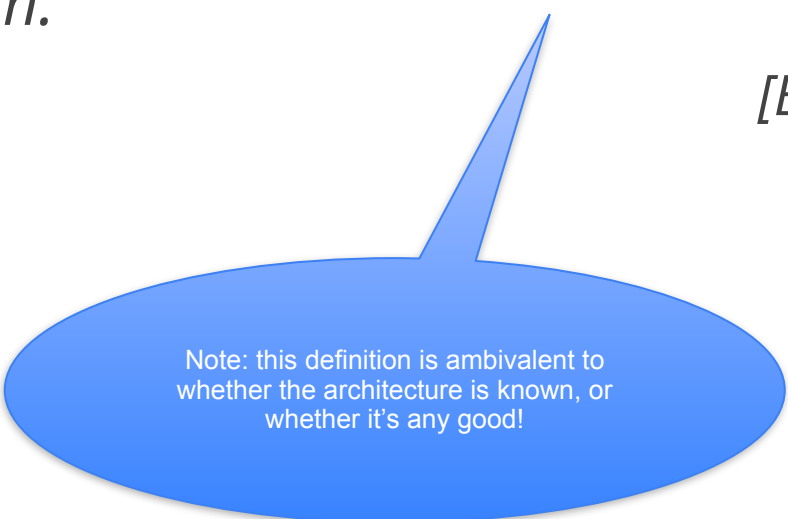
Learning Goals

- 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
- Use notation and views to describe the architecture suitable to the purpose
- Document architectures clearly, without ambiguity

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]

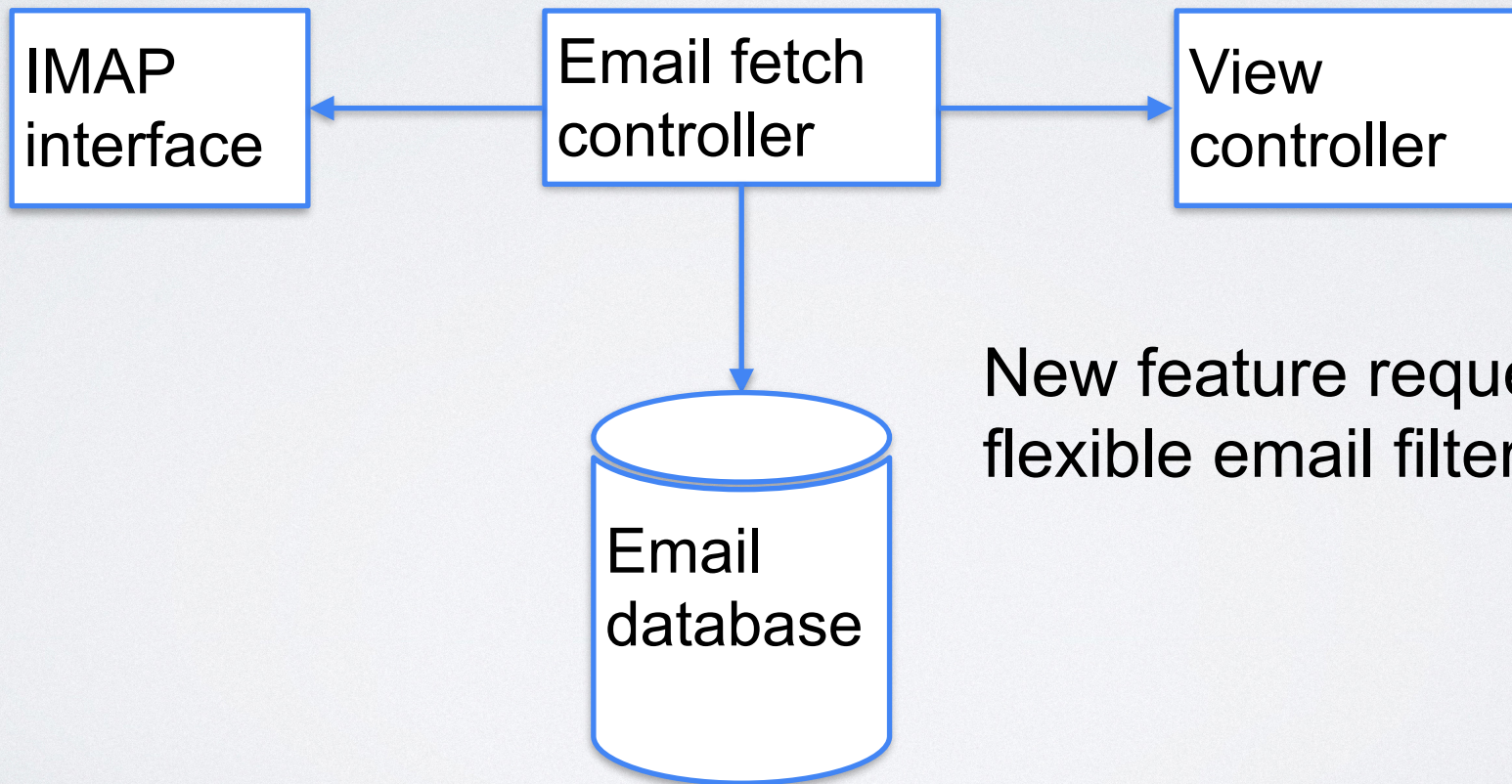


Note: this definition is ambivalent to whether the architecture is known, or whether it's any good!

Why Understand Architecture?

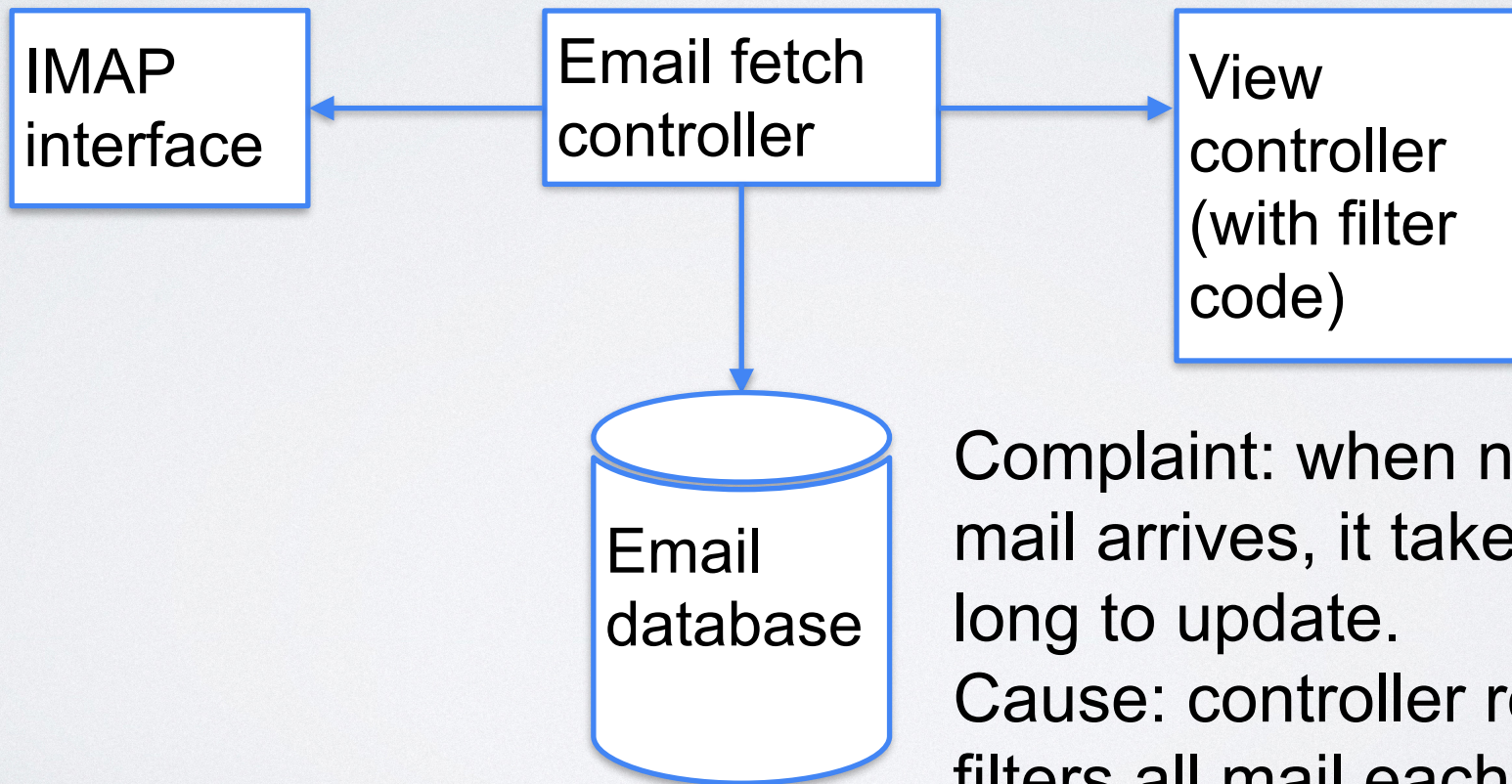
- Every system has an architecture
- But if you design the architecture intentionally, it's likely to be better!
- Let's look at an example

Example: Email Client



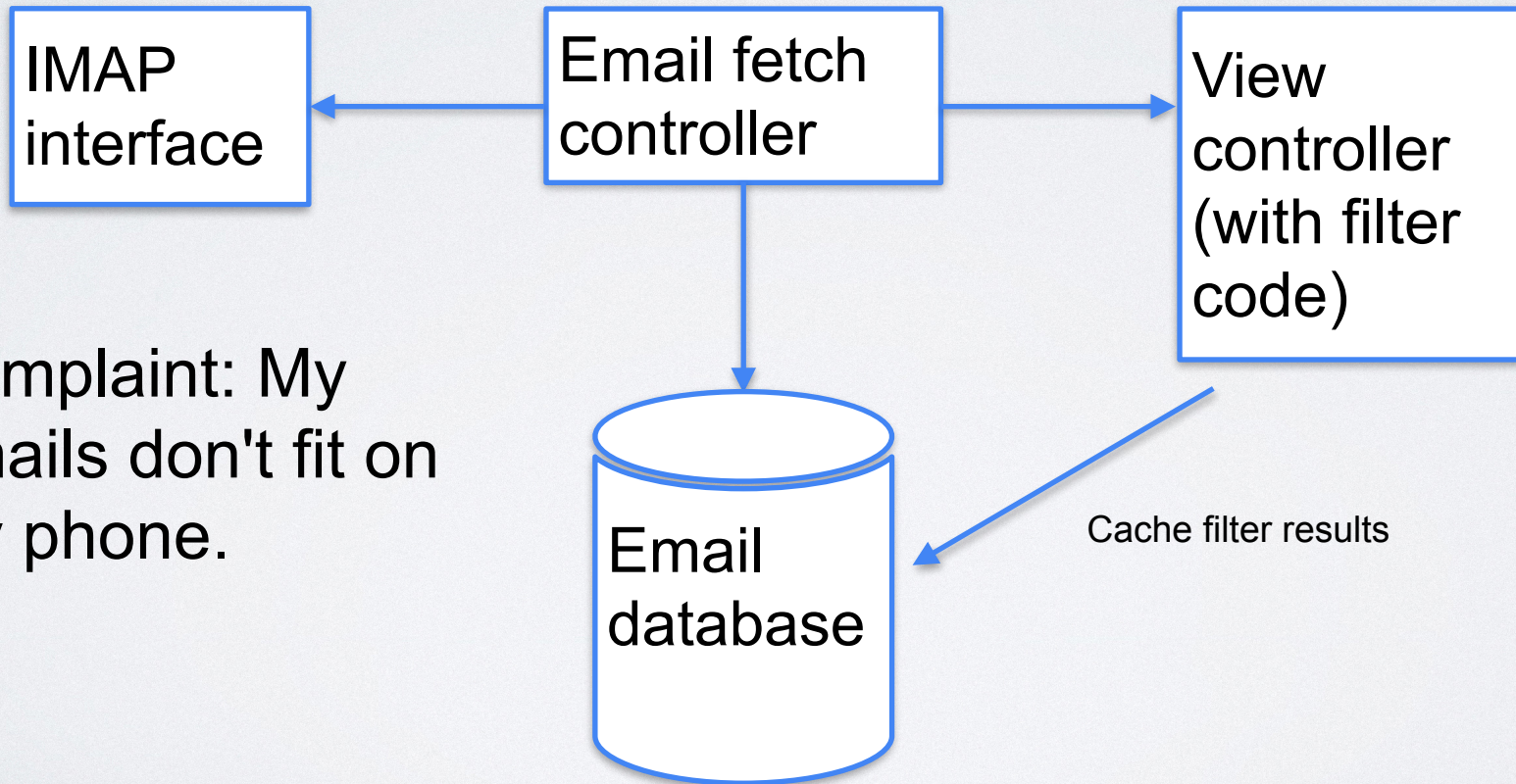
New feature request:
flexible email filters.

Example: Email Client



Complaint: when new mail arrives, it takes too long to update.
Cause: controller re-filters all mail each time.

Example: Email Client



Complaint: My emails don't fit on my phone.

Goal: Meet Quality Requirements

- Maintainability / Modifiability
- Performance
- Scalability
- Availability
- Usability

Key lesson: software architecture is about selecting a design that meets the desired quality attributes.

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”
- OO-Design (Low-level design, e.g. design patterns)
 - mid-level “how”, low-level “what”
- Code
 - low-level “how”

Design vs. Architecture

Design Questions

- How do I add a menu item in VSCode?
- How can I make it easy to add menu items in VSCode?
- 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

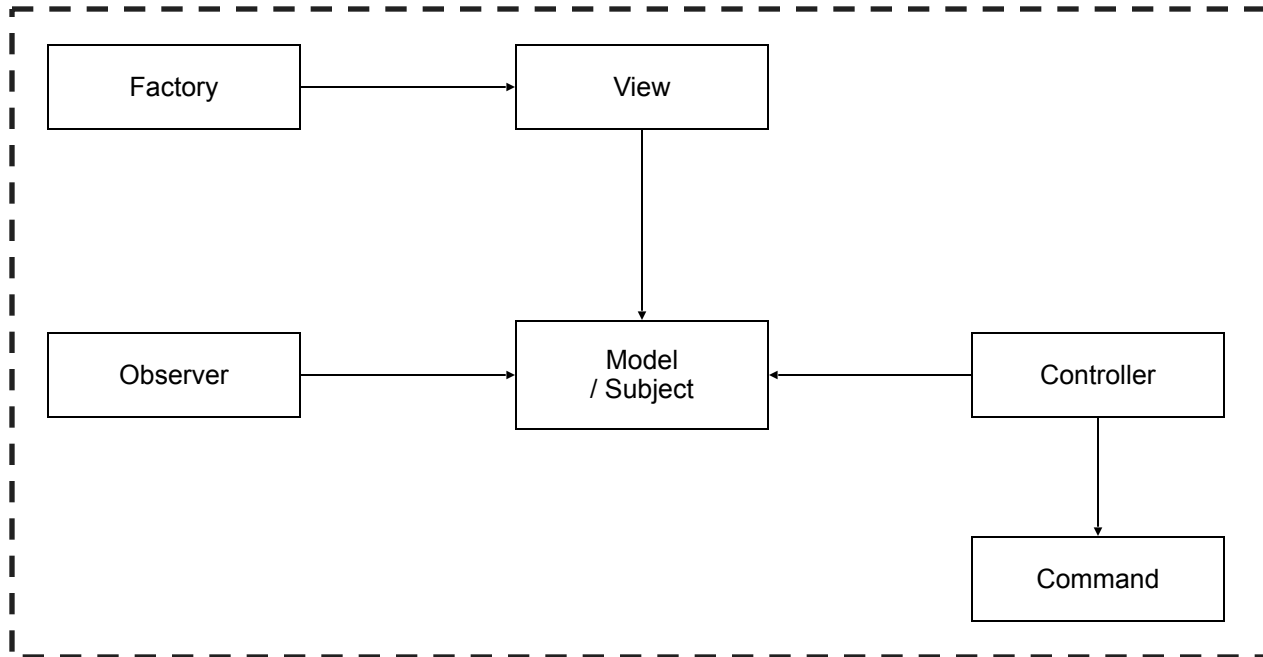
- How do I extend VSCode with a plugin?
- What threads exist and how do they coordinate?
- How does Google scale to billions of hits per day?
- Where should I put my firewalls?
- What is the interface between subsystems?

Objects

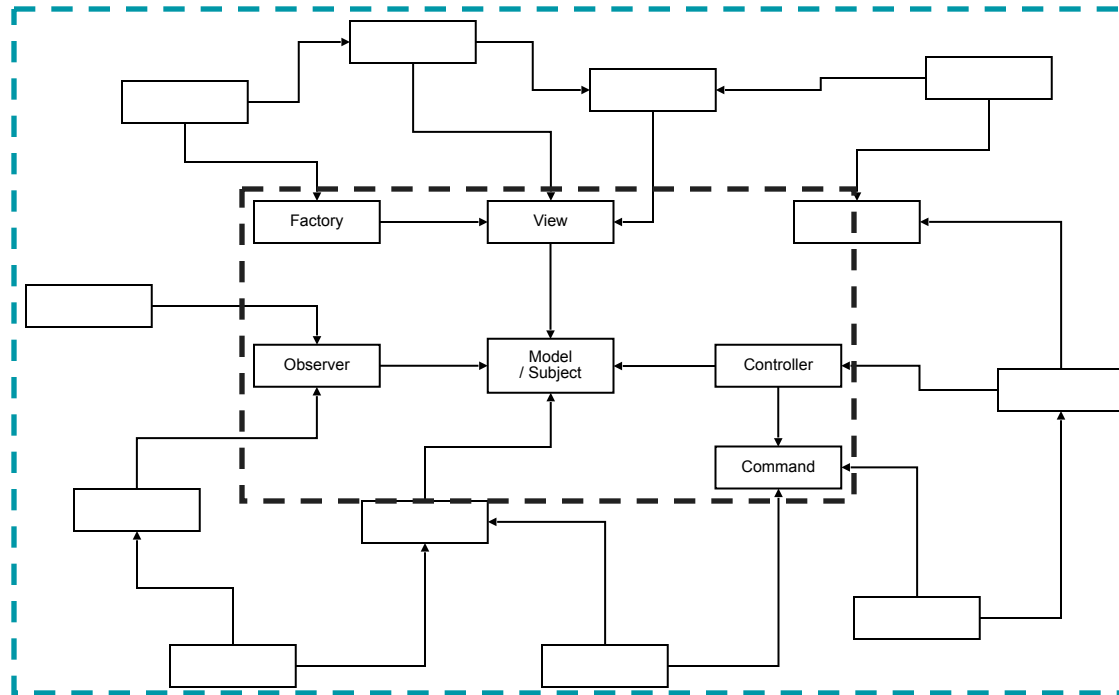


Model

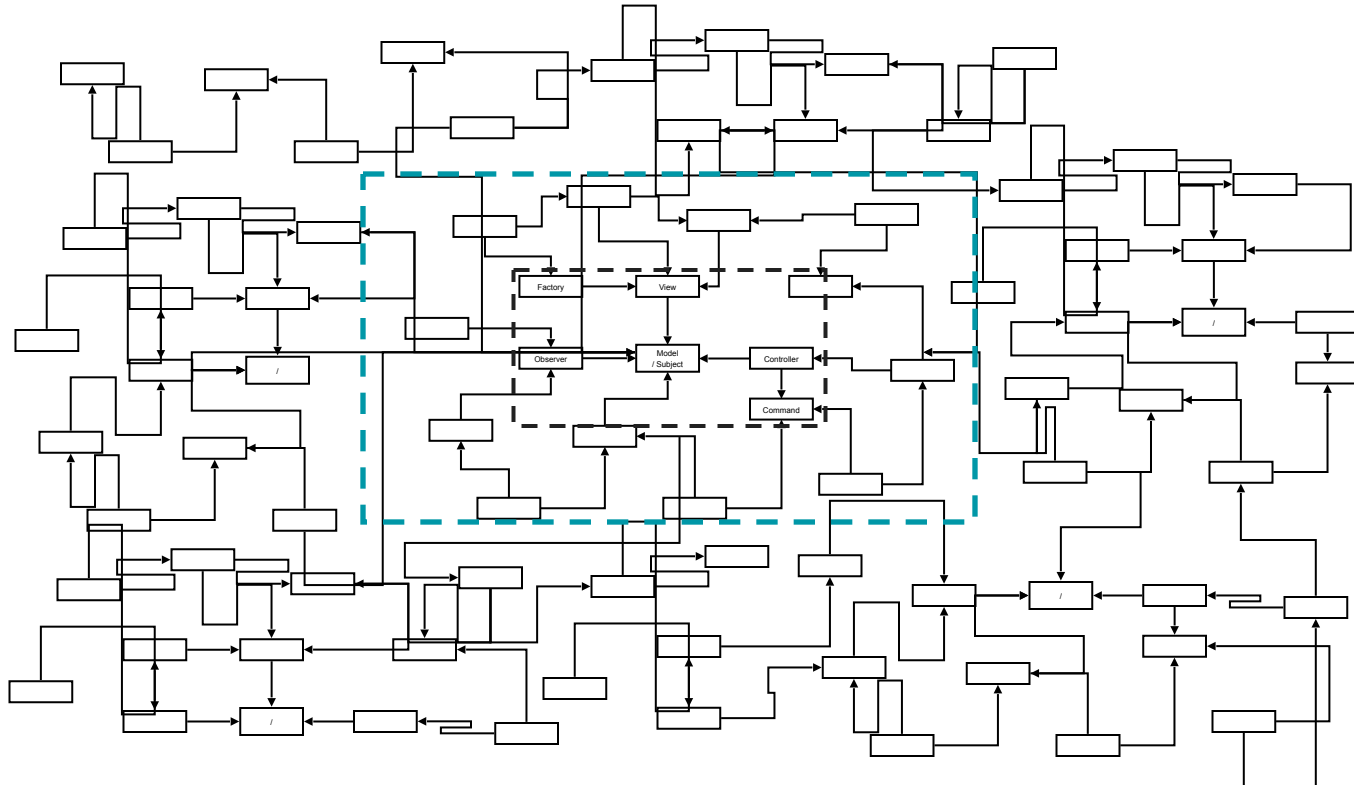
Design Patterns



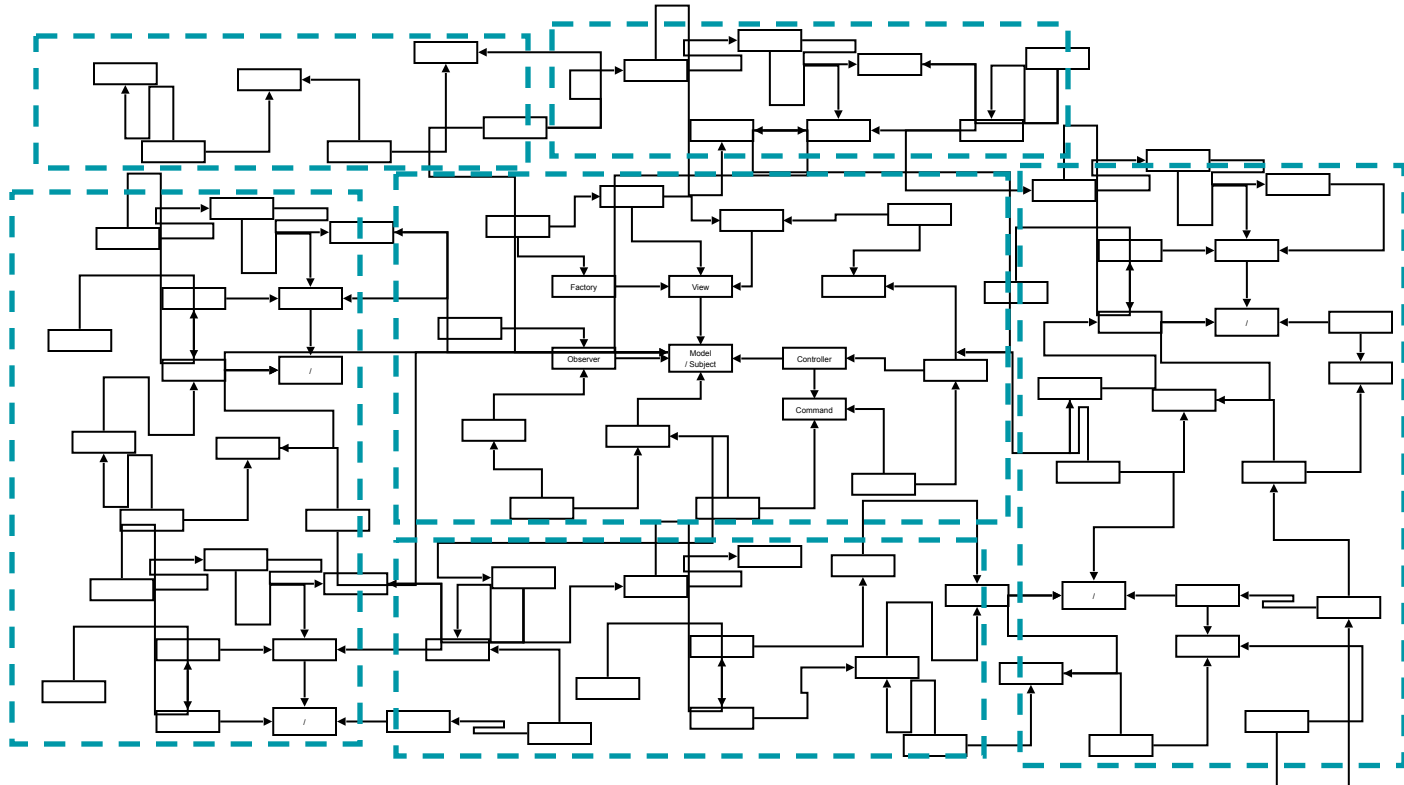
Design Patterns



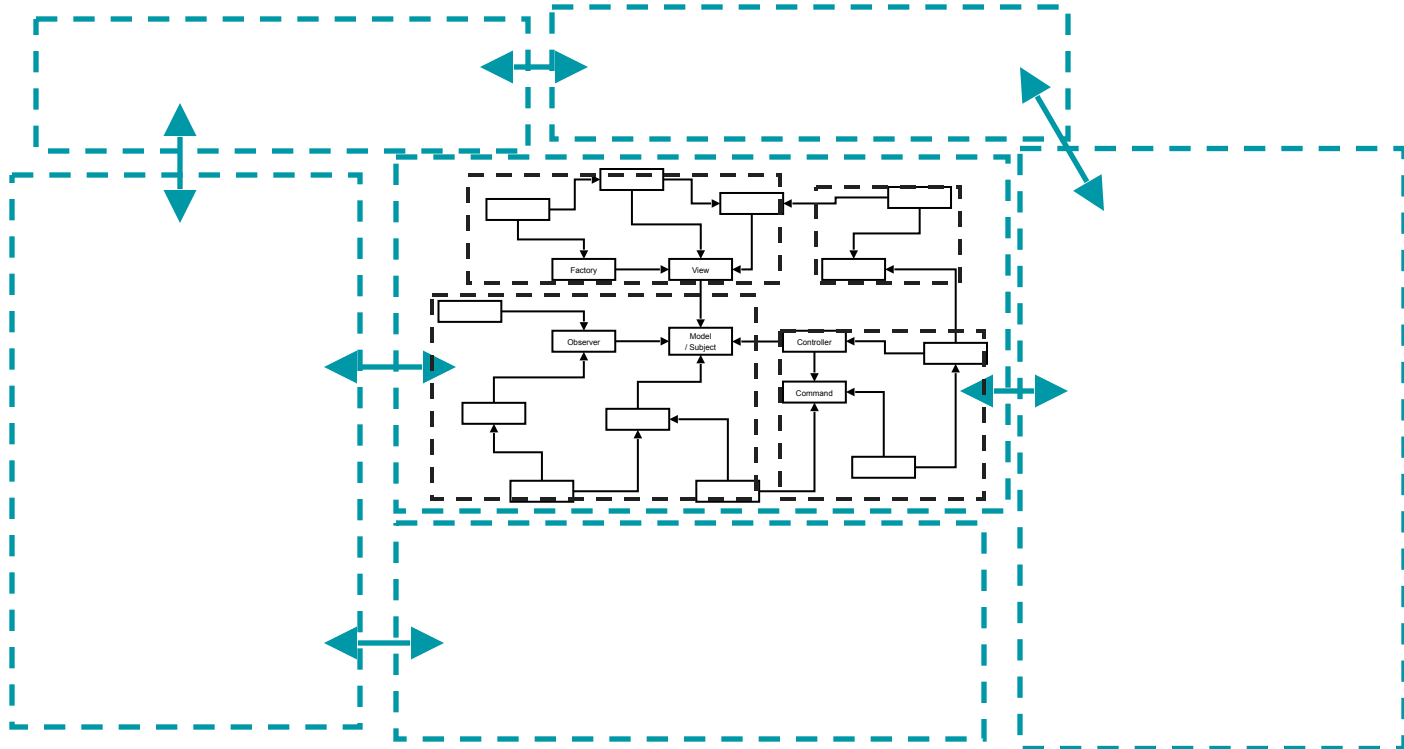
Design Patterns



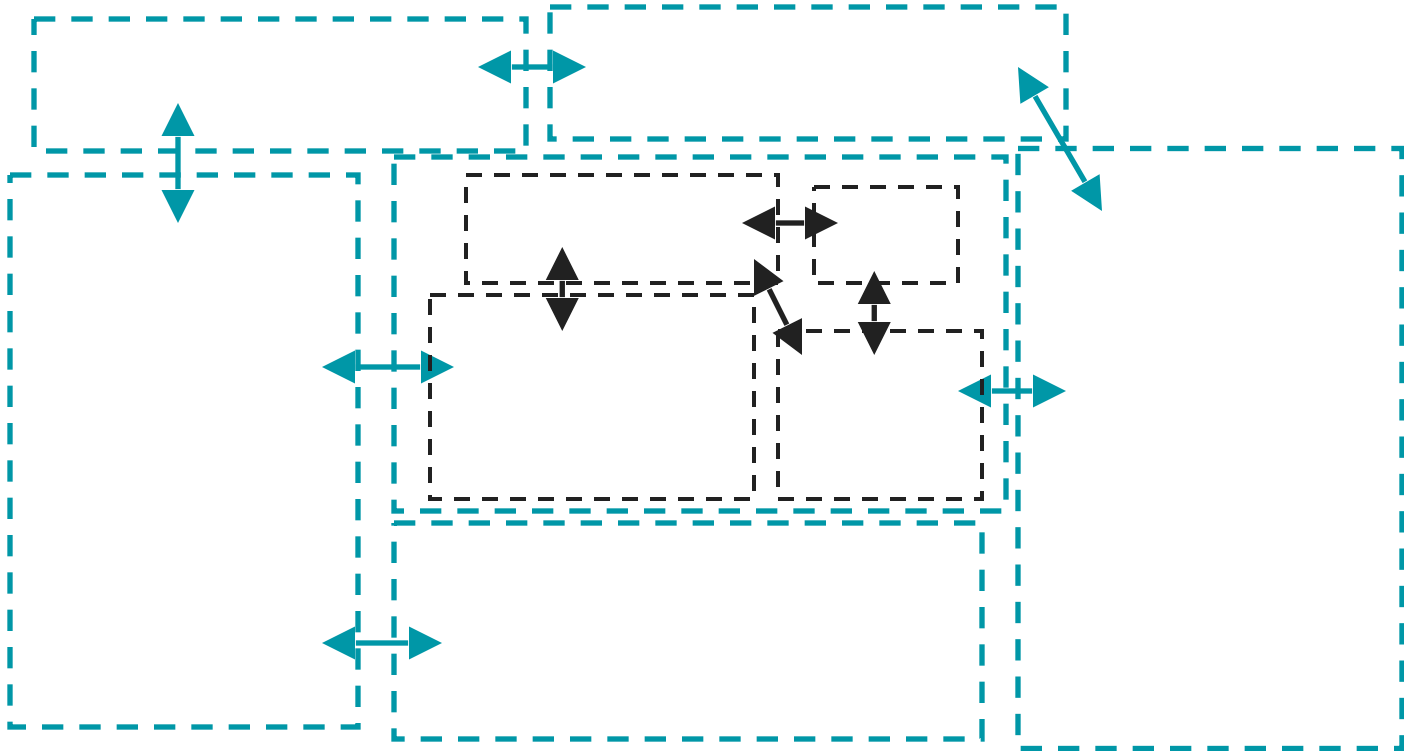
Architecture



Architecture



Architecture



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.

Views and Purposes

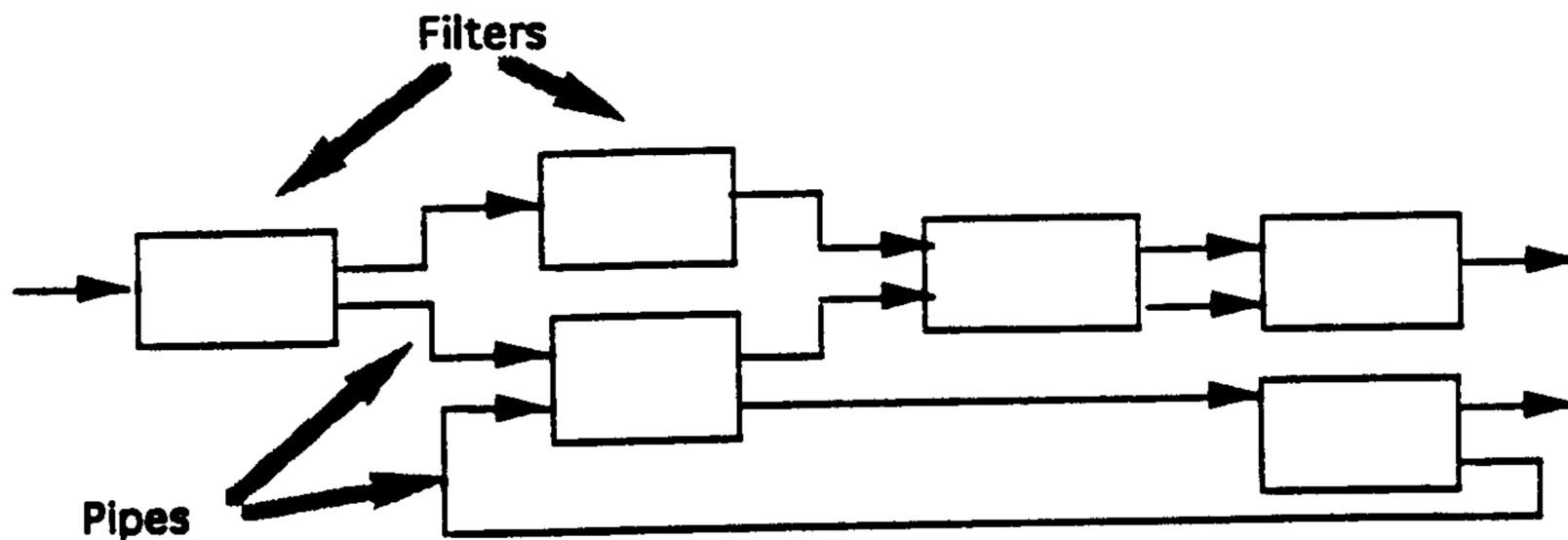
- Every view should align with a purpose
- Views should only represent information relevant to that purpose
 - Abstract away other details
 - Annotate view to guide understanding where needed
- Different views are suitable for different reasoning aspects (different quality goals), e.g.,
 - Performance
 - Extensibility
 - Security
 - Scalability
 - ...

Common Views in Documenting Software Architecture

- Static View
 - Modules (subsystems, structures) and their relations (dependencies, ...)
- Dynamic View
 - Components (processes, runnable entities) and connectors (messages, data flow, ...)
- Physical View (Deployment)
 - Hardware structures and their connections

Common Software Architectures

1. Pipes and Filters



Example: Compilers

