

# CSE 403

Software Engineering

Spring 2023

## **#11: Software architecture**

# Logistics

## WEEK 4

04/17 L: Data modeling

04/18 T: DUE: [GPS!!!](#)

04/19 L: Architecture [Design & Architecture \(DnA\)](#)

04/20 P:

04/21 L: Design

# Today

- Software architecture vs. software design
- Common software architecture patterns

# **Software architecture vs. software design**

# Why software architecture and design?

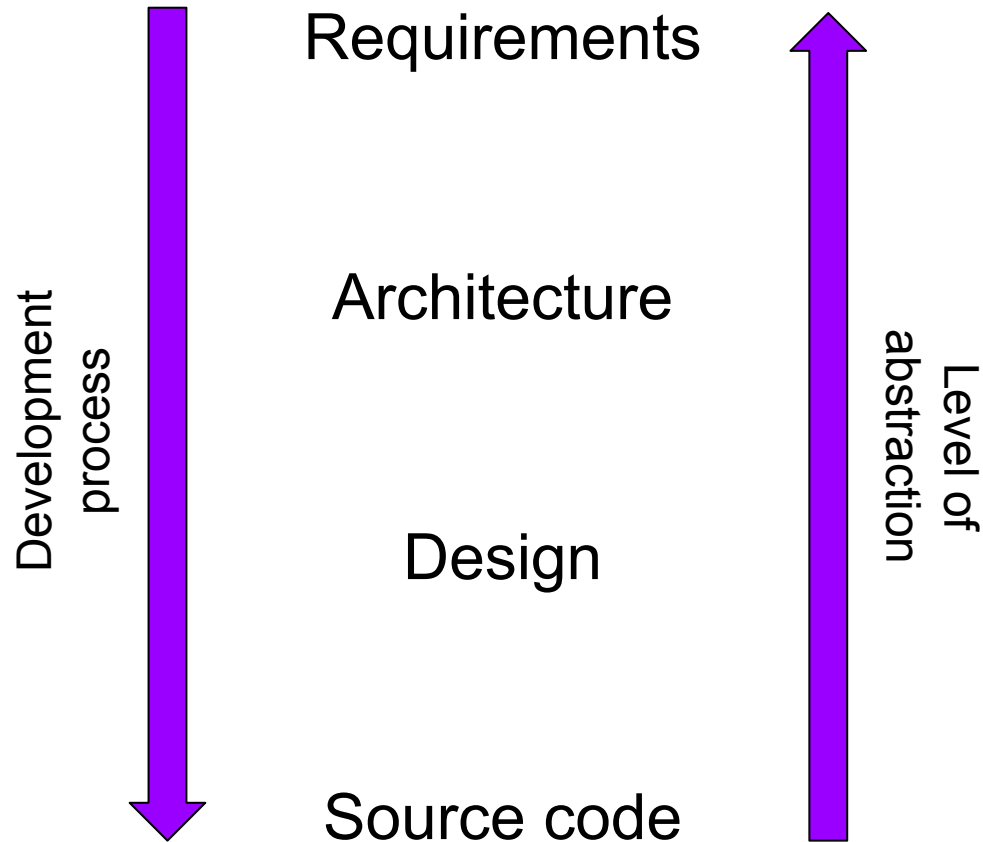
“There are two ways of constructing a software design:

one way is to make it so **simple** that there are *obviously no deficiencies*;

the other is to make it so **complicated** that there are *no obvious deficiencies*.” [Tony Hoare]

Goals: separation of concerns and modularity.

# Architecture vs. design



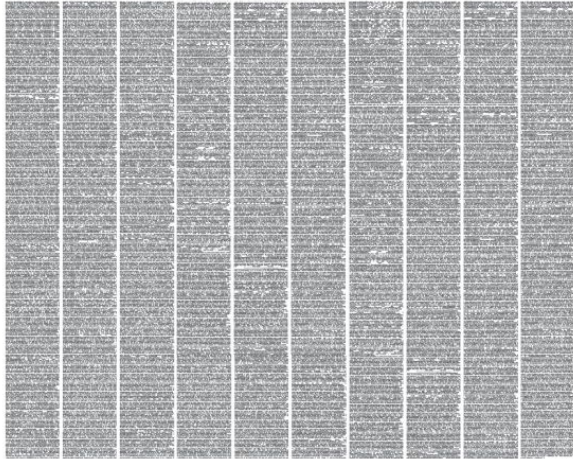
# Abstraction

## **Building an abstract representation of reality**

- Ignoring (insignificant) details.
- Focusing on the most important properties.
- Level of abstraction depends on viewpoint and purpose:
  - Communication
  - Component interfaces
  - Verification and validation

# Different levels of abstraction

Source code



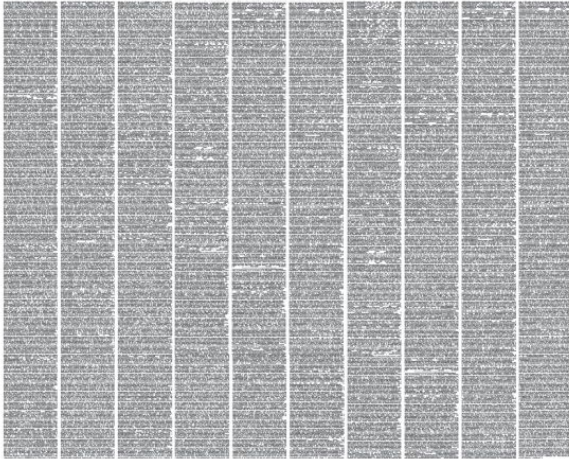
## Example: Linux Kernel

- 16 million Lines of Code!
- **What does the code do?**
- Are there dependencies?
- Are there different components?

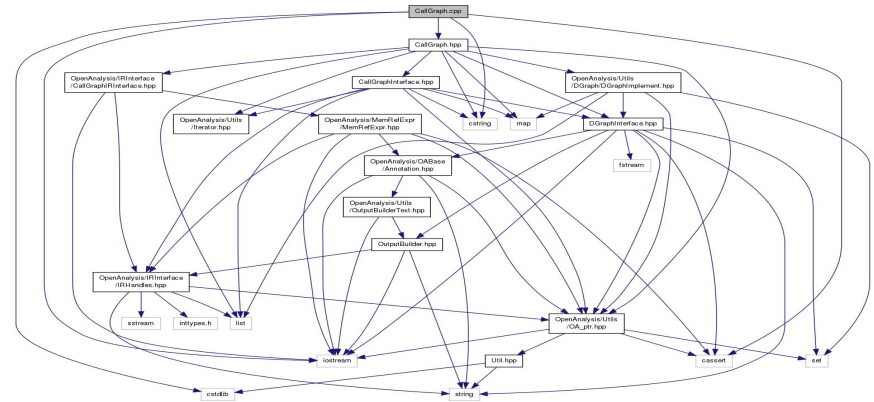


# Different levels of abstraction

## Source code



## Call graph

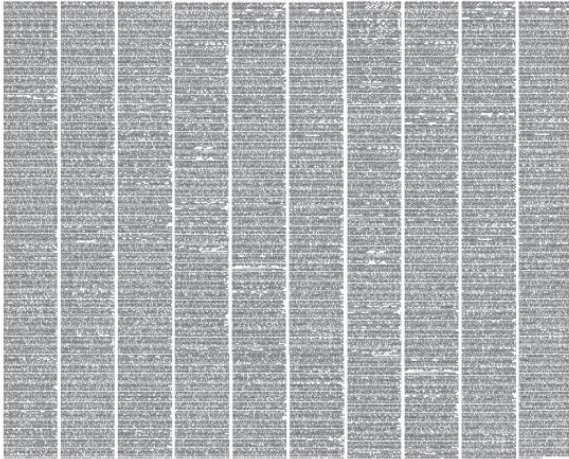


## Example: Linux Kernel

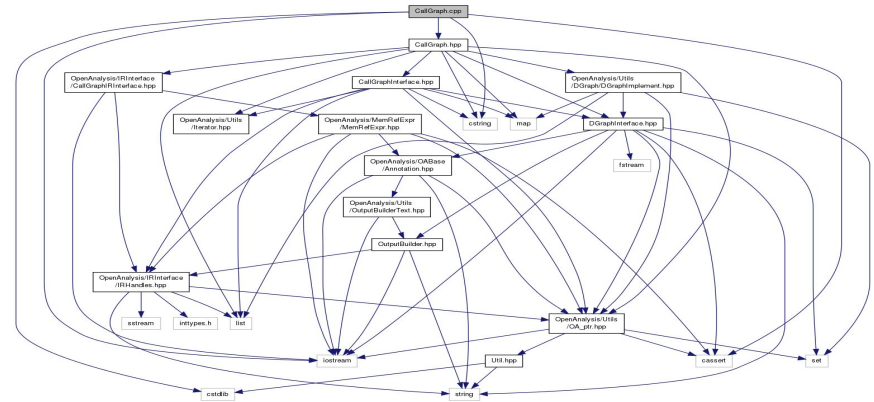
- 16 million Lines of Code!
- What does the code do?
- Are there dependencies?
- Are there different components?

# Different levels of abstraction

## Source code



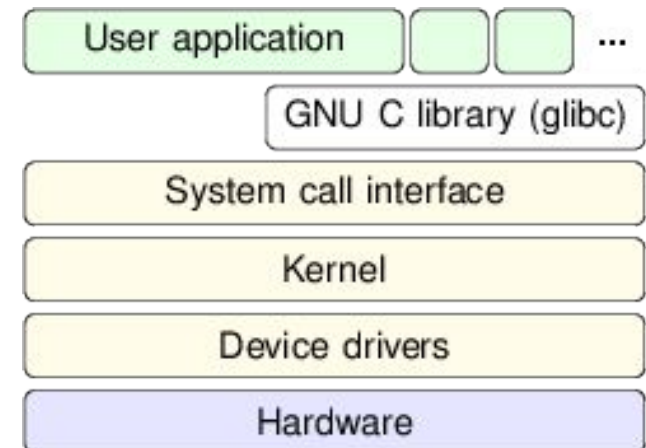
## Call graph



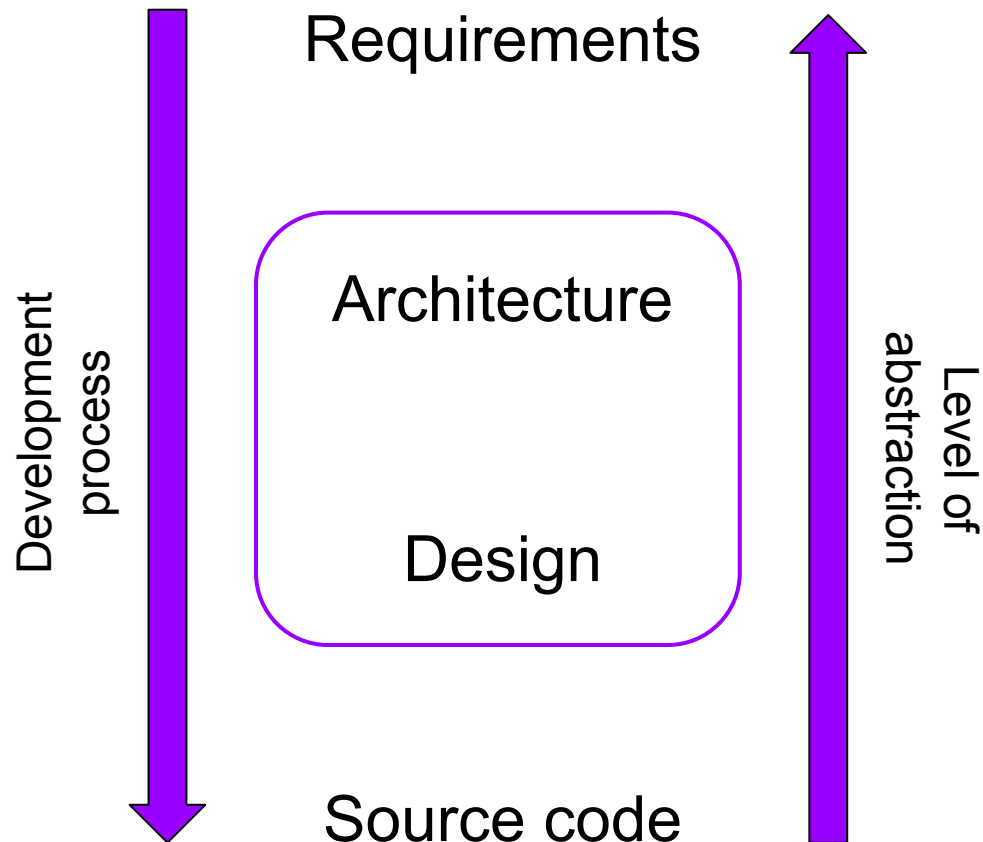
## Layer diagram

### Example: Linux Kernel

- 16 million Lines of Code!
- What does the code do?
- Are there dependencies?
- Are there different components?



# Architecture vs. design



What's the difference?

# Architecture vs. design

## **Architecture (what components are needed?)**

- High-level view of the overall system:
  - What components do exist?
  - What are the protocols between components?
  - ...

## **Design (how are the components developed?)**

- Considers individual components:
  - Data representation
  - Interfaces, Class hierarchy
  - ...

# Architecture vs. design

## Architecture



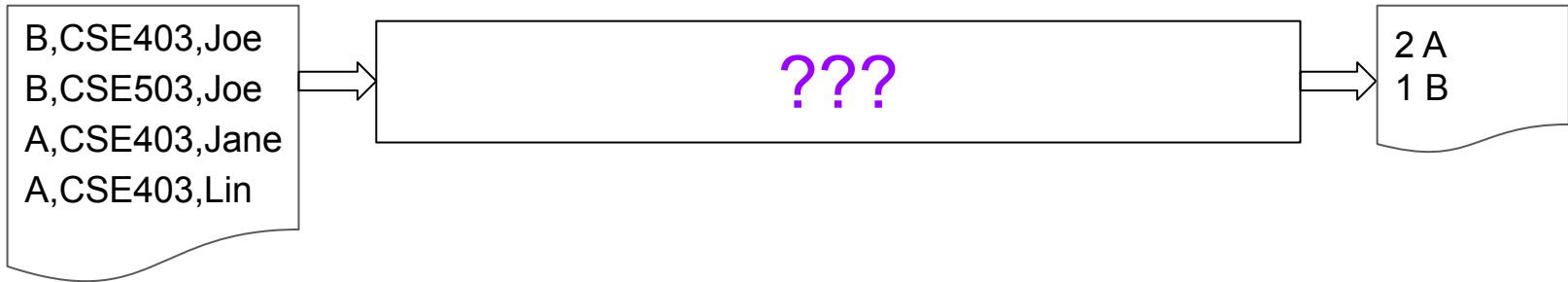
[Gates Center Architecture, LMN]

## Design



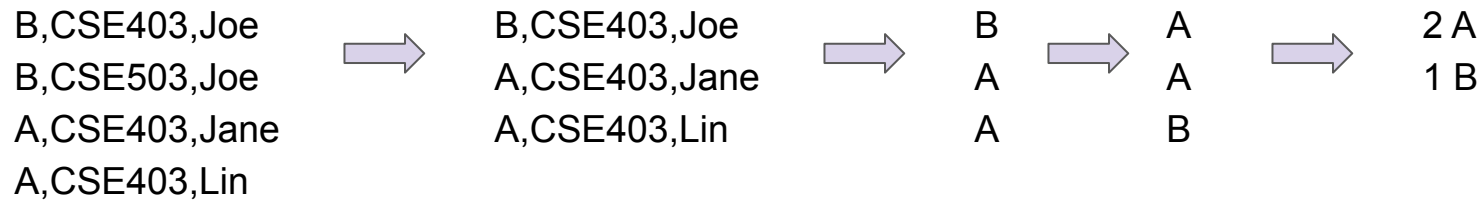
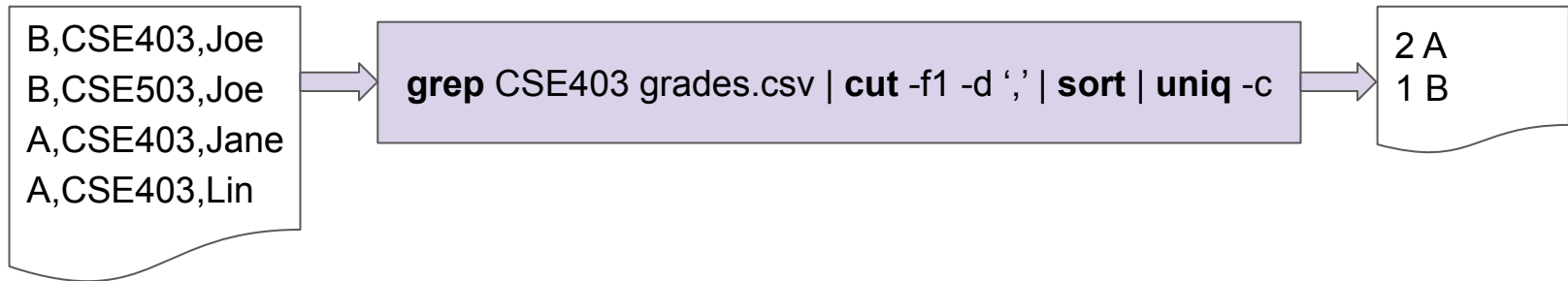
[Office design, New York Times]

# Provocation: How would you solve this problem?



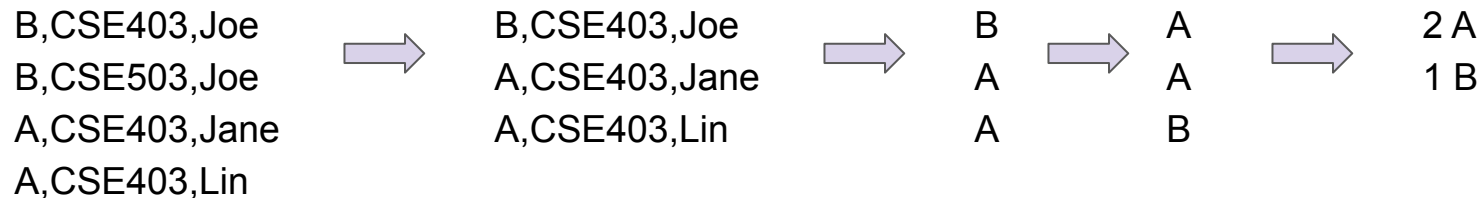
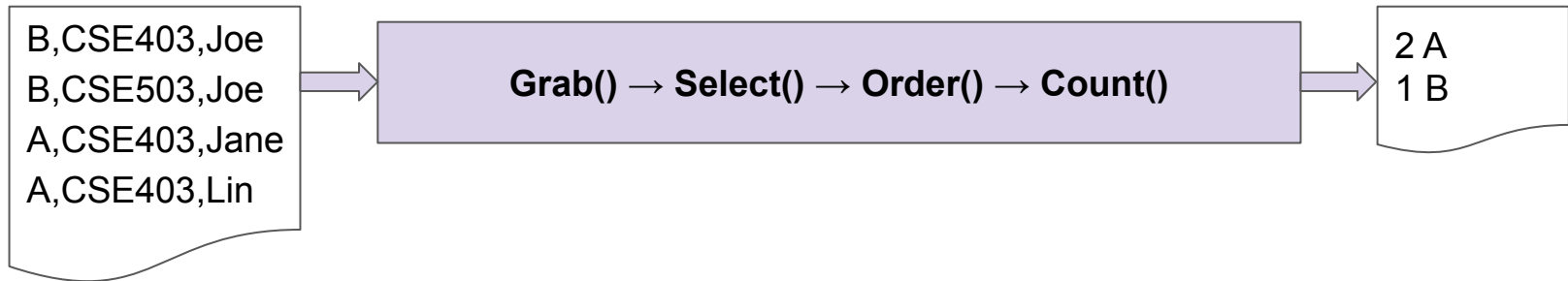
Goal: count CSE403 letter grades.

# Provocation: The "coding" way!



Lower level of abstraction: lang, code, run!

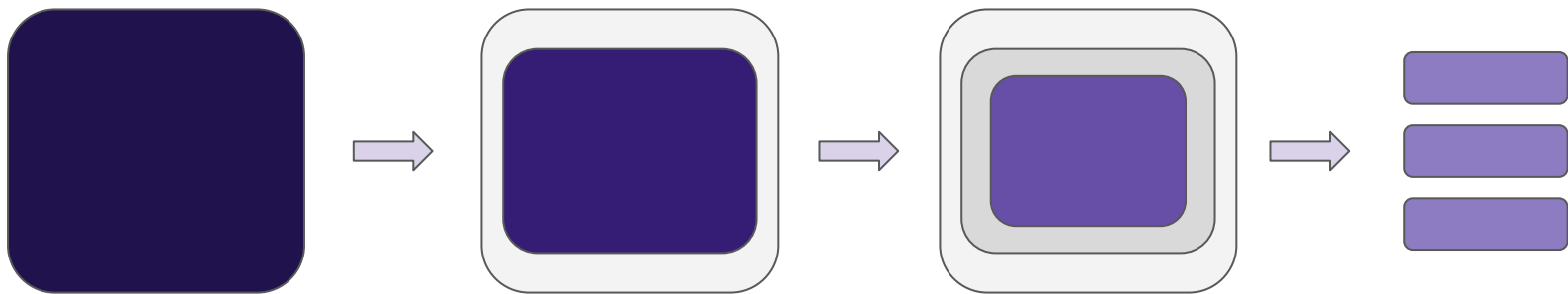
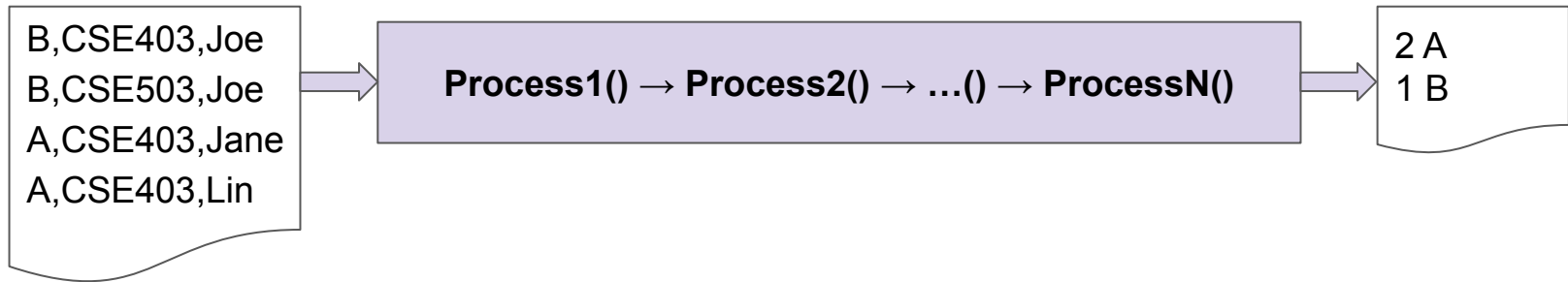
# Provocation: The "design" way



Mid-level of abstraction: component specification!

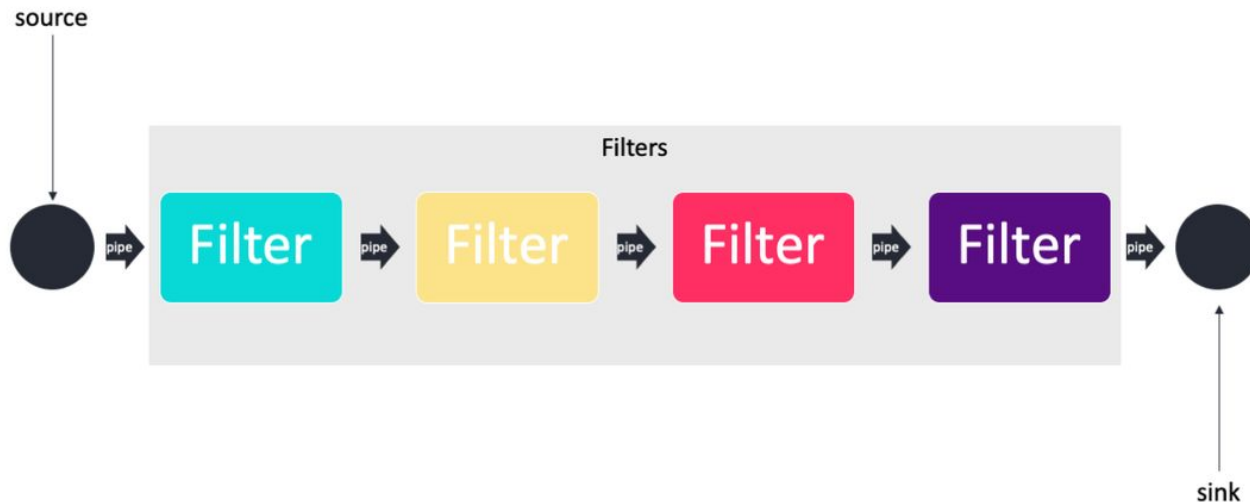


# Provocation: The "architecture" way



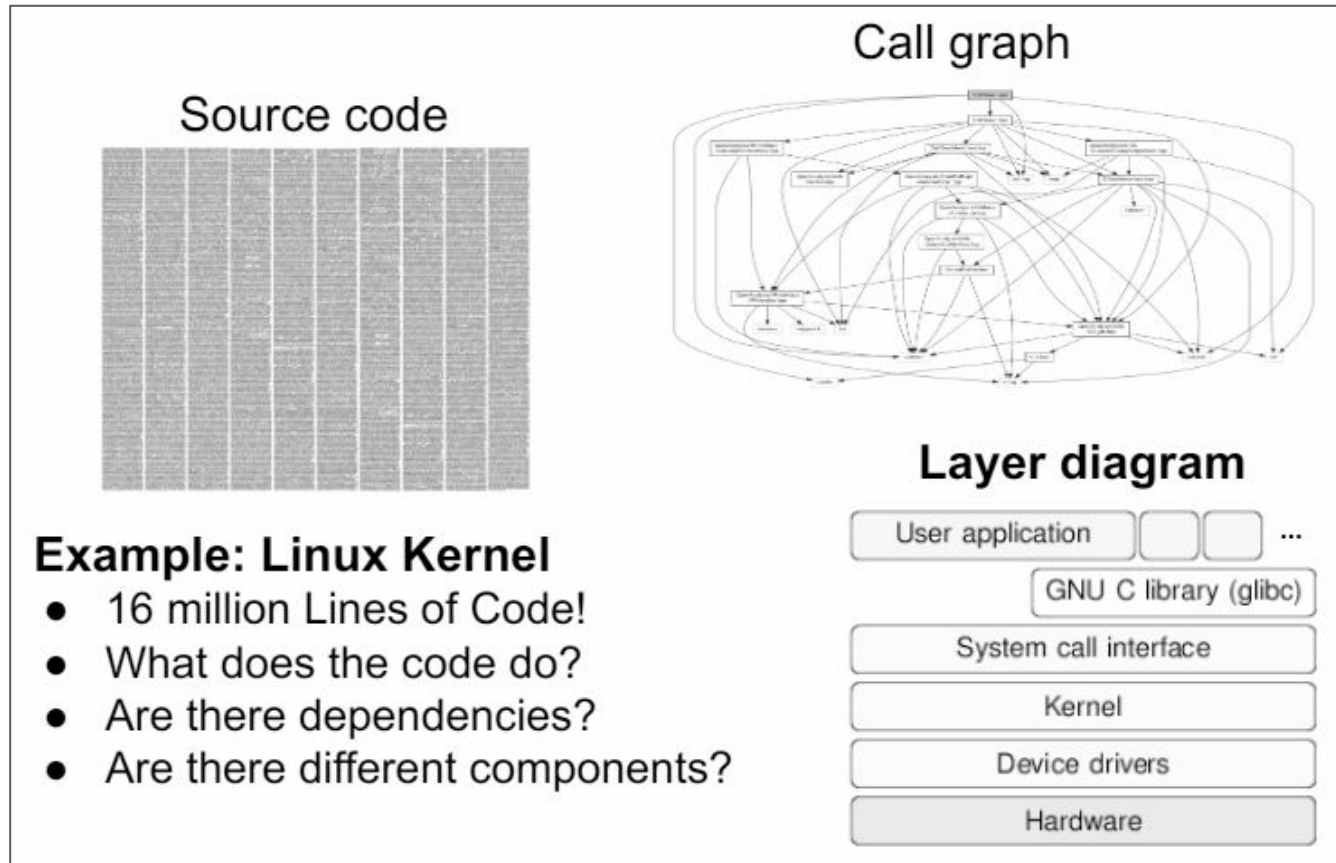
Higher level of abstraction: components concatenation!

# SW Architecture #1: Pipe and Filter



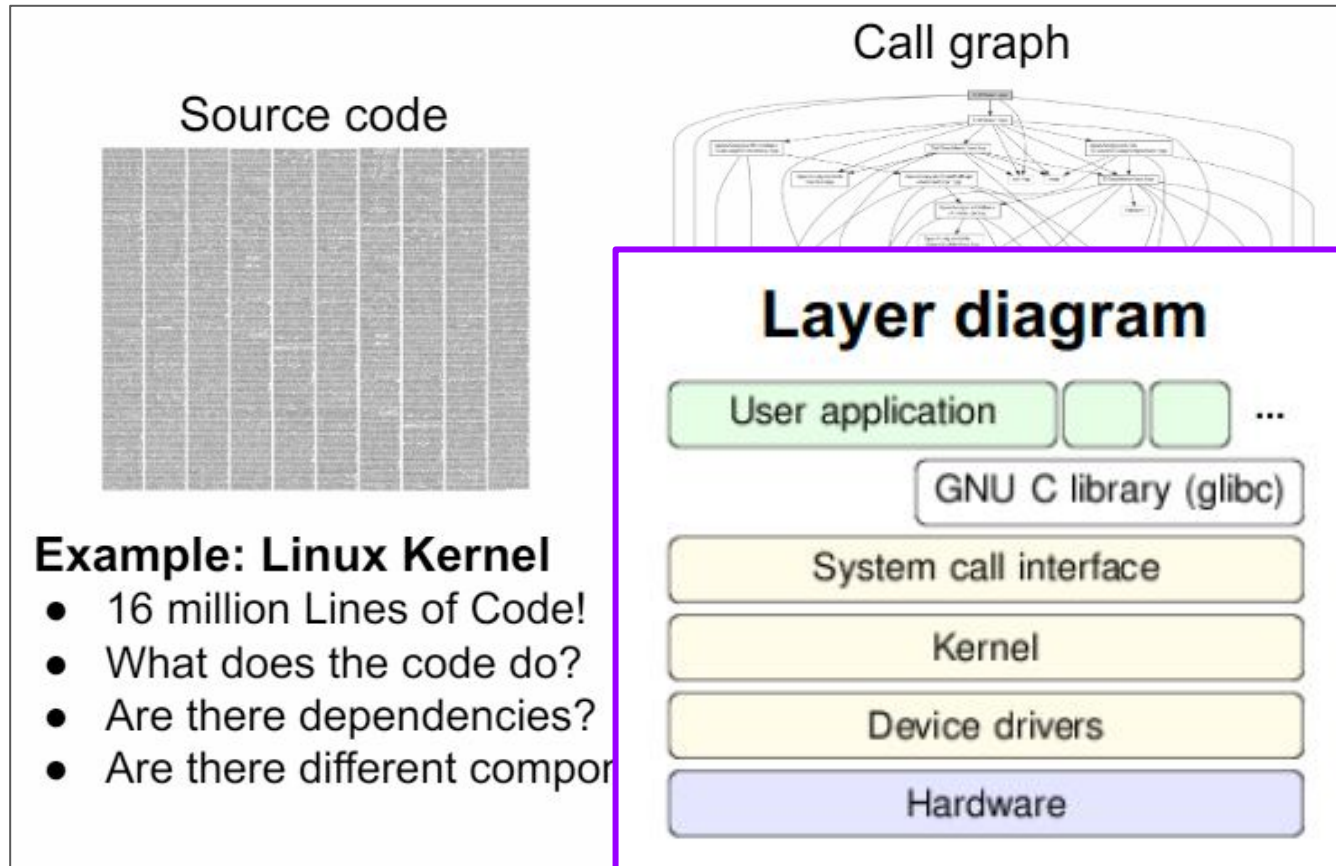
The pipe-and-filter **architecture** doesn't specify the **design** or **implementation** details of the individual components (the filters)!

# SW Architecture #2: ???

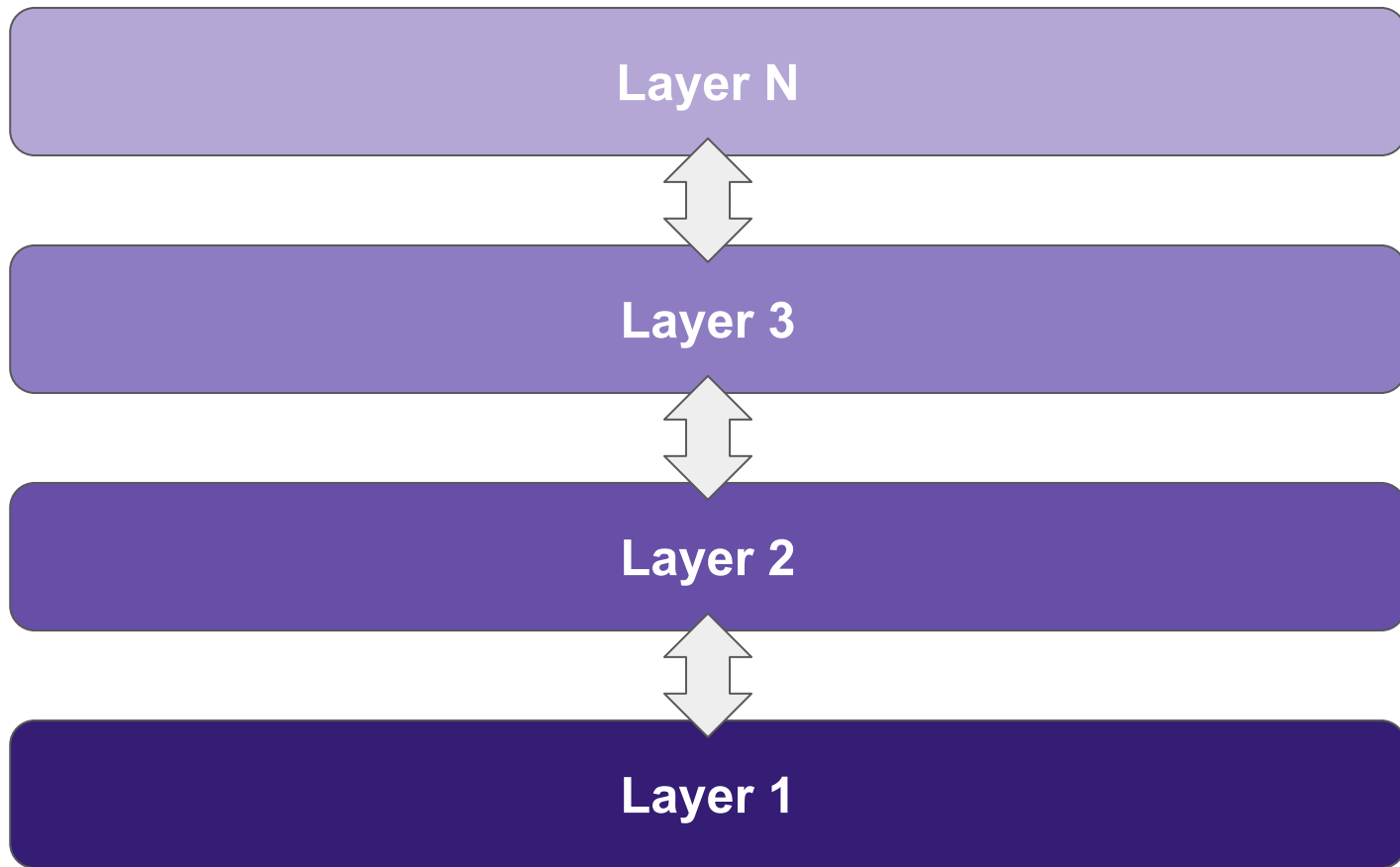


Which architectural model we have already talked about here?

# SW Architecture #2: Layered

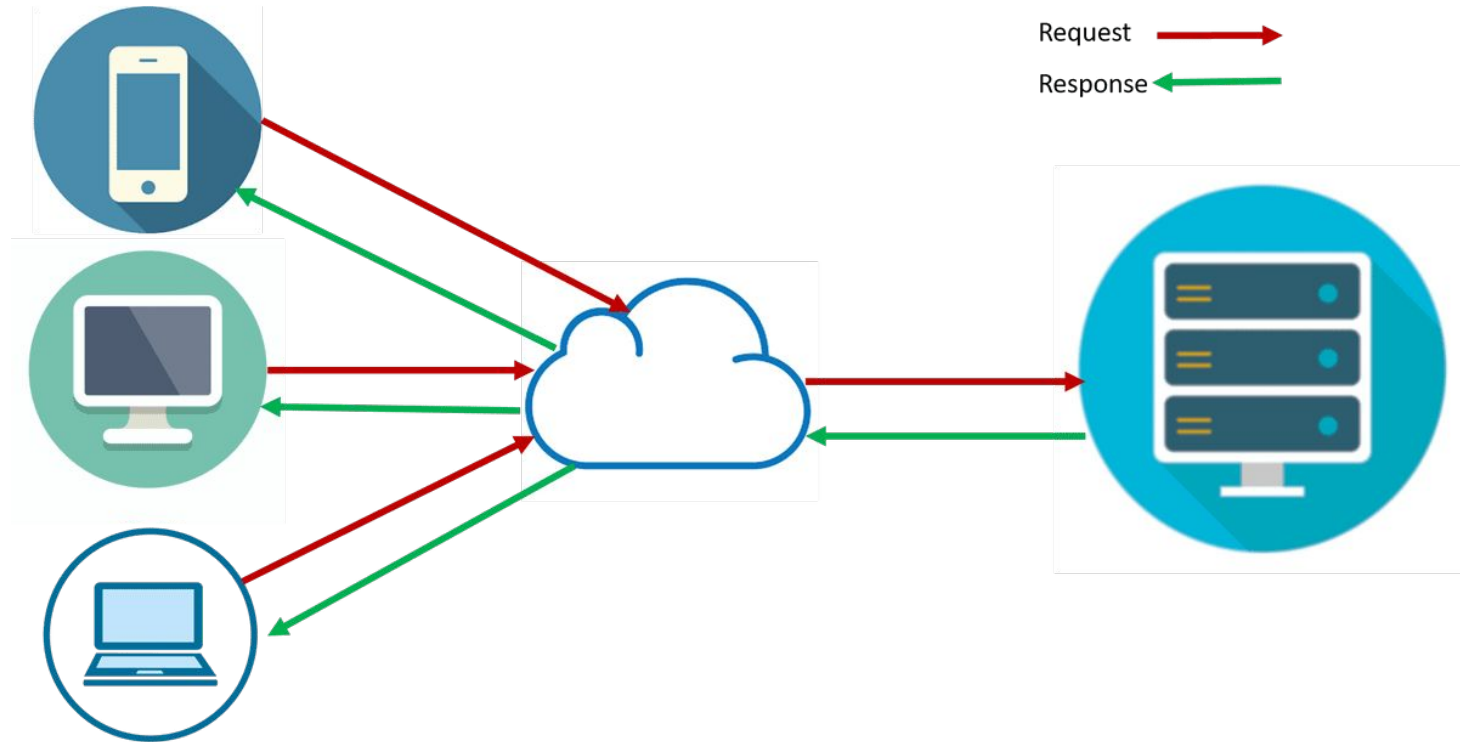


## SW Architecture #2: Layered



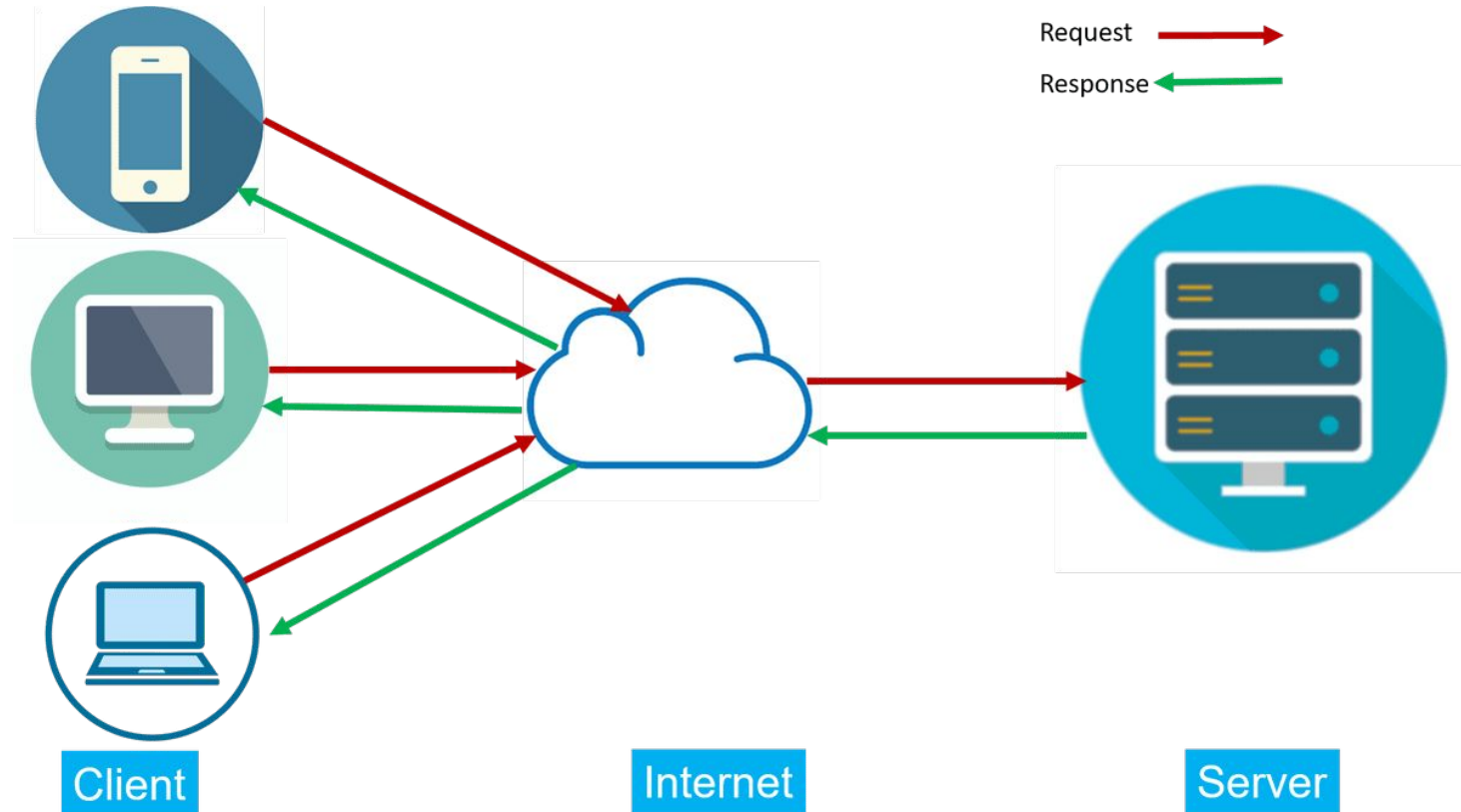
Each layer offers a service to the one on the top!

# SW Architecture #3: ???



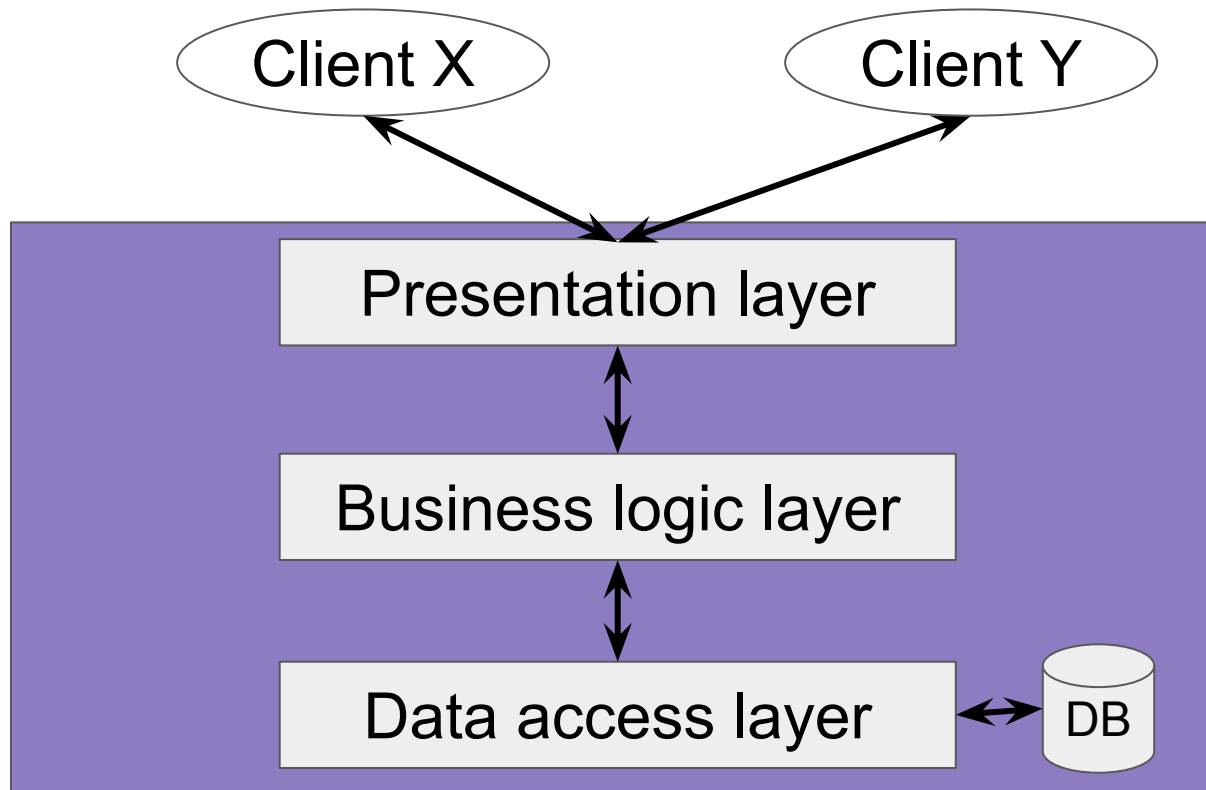
Pretty sure you know this one!

# SW Architecture #3: Client-Server



Breaks-up the whole problem into a server, clients, and communication medium!

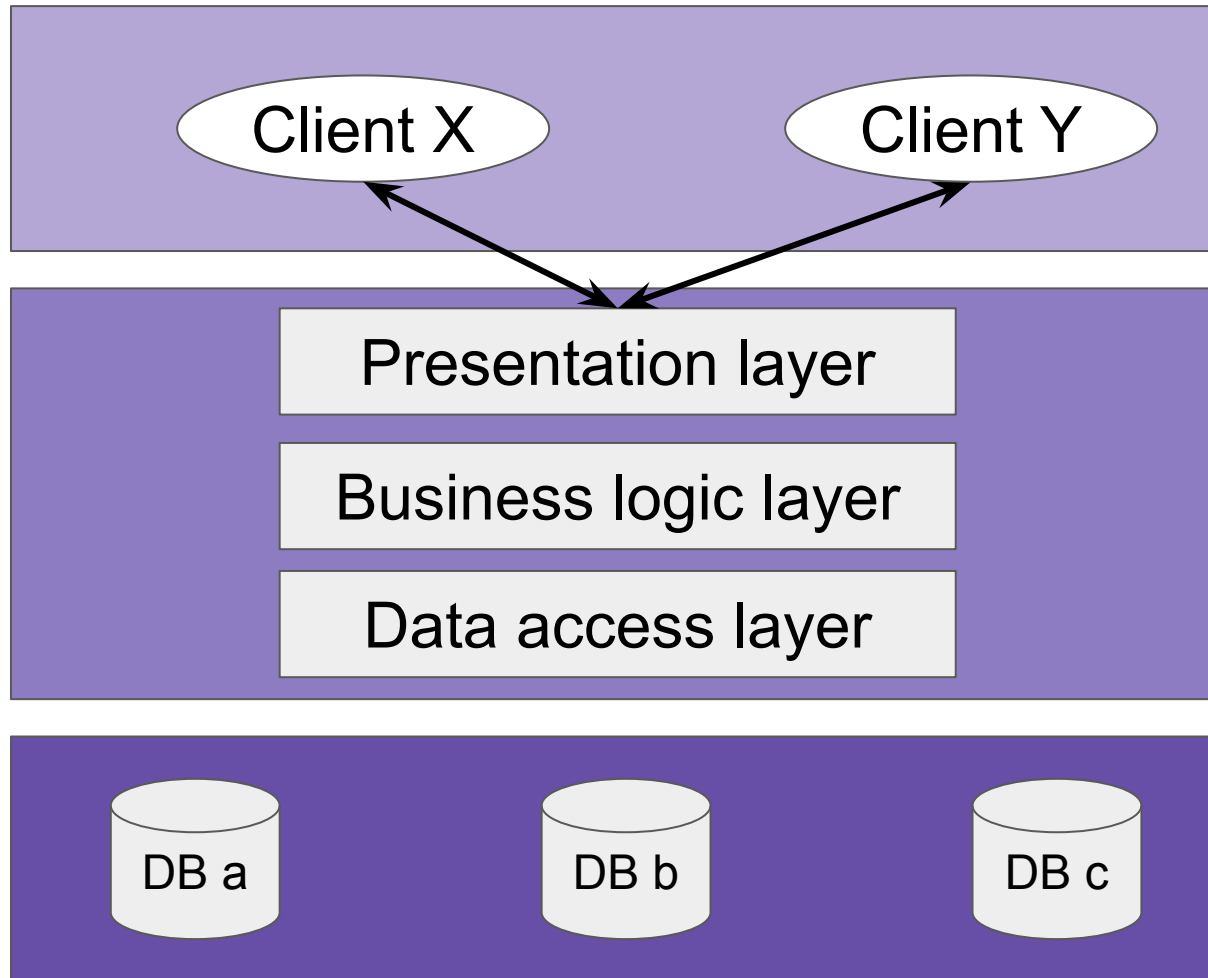
# SW Architecture #3 ½: Client-Server + Layers



What if your focus is the server part?

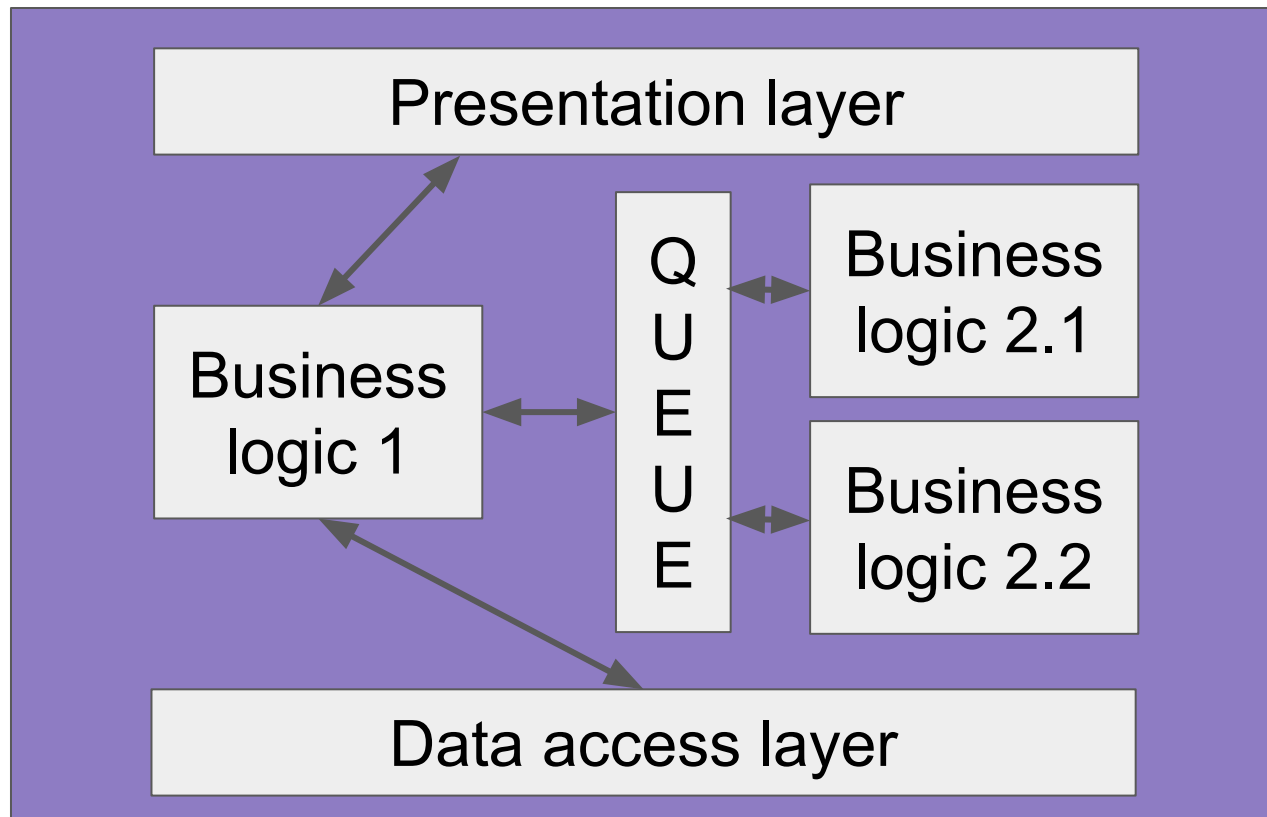


## SW Architecture #4: (n)-tiered



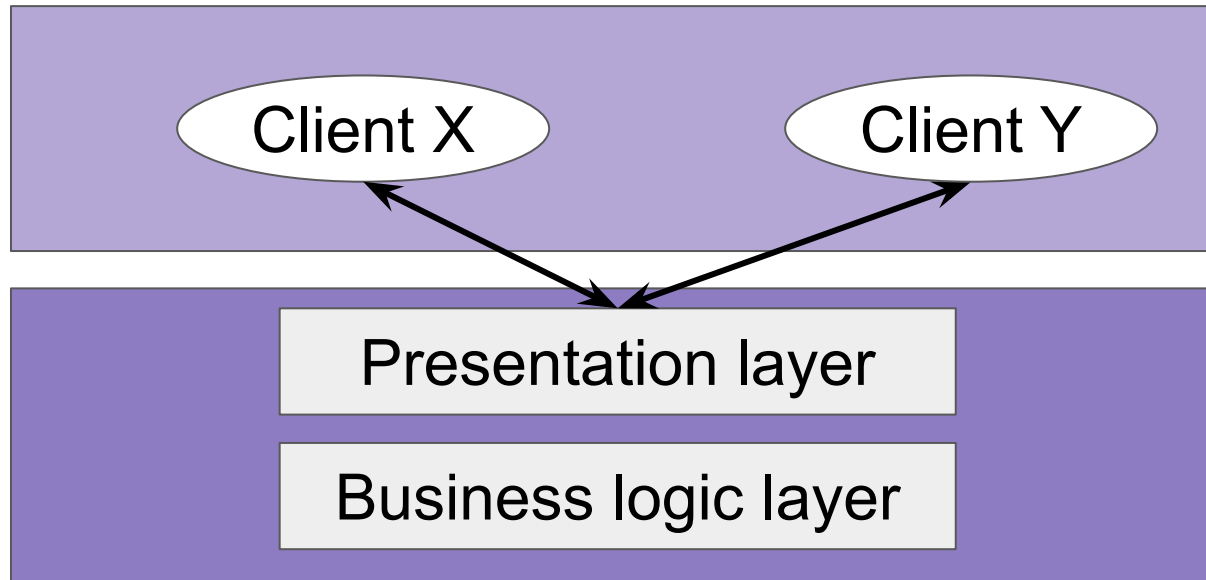
What if things in the server gets too complicated?

# SW Architecture #5: Message-Oriented



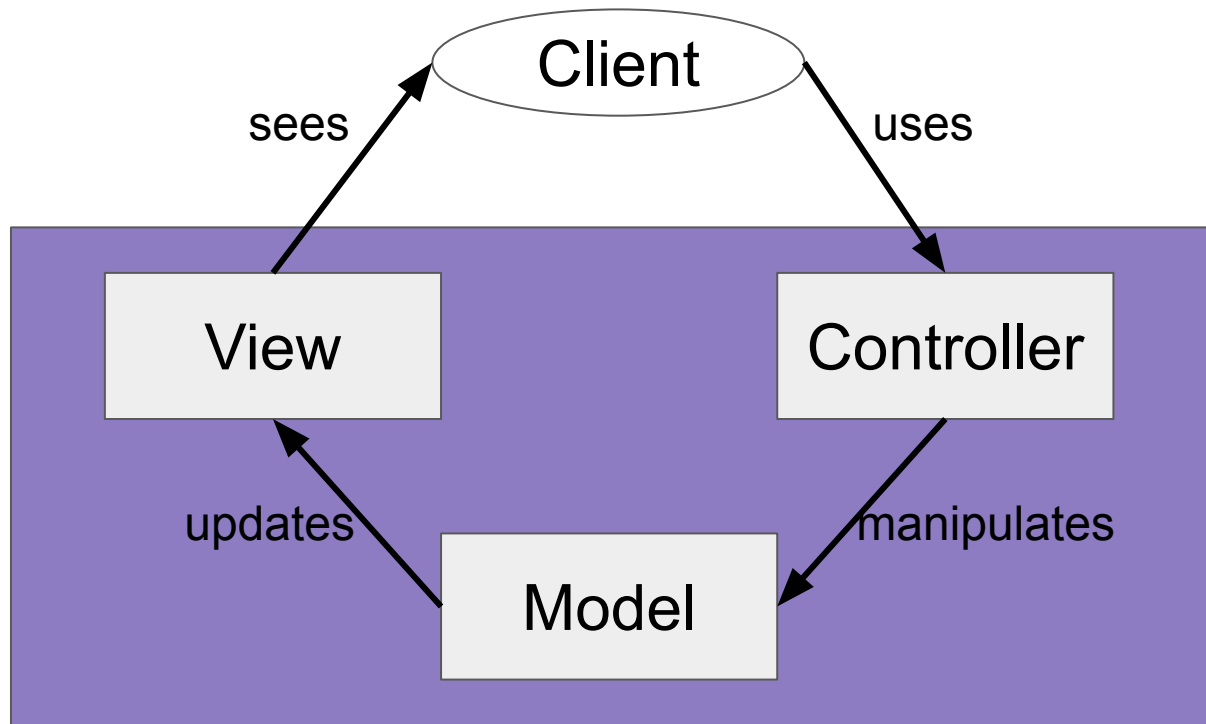
What if things in the server even more complicated and need help to process the load?

## SW Architecture #6: ???



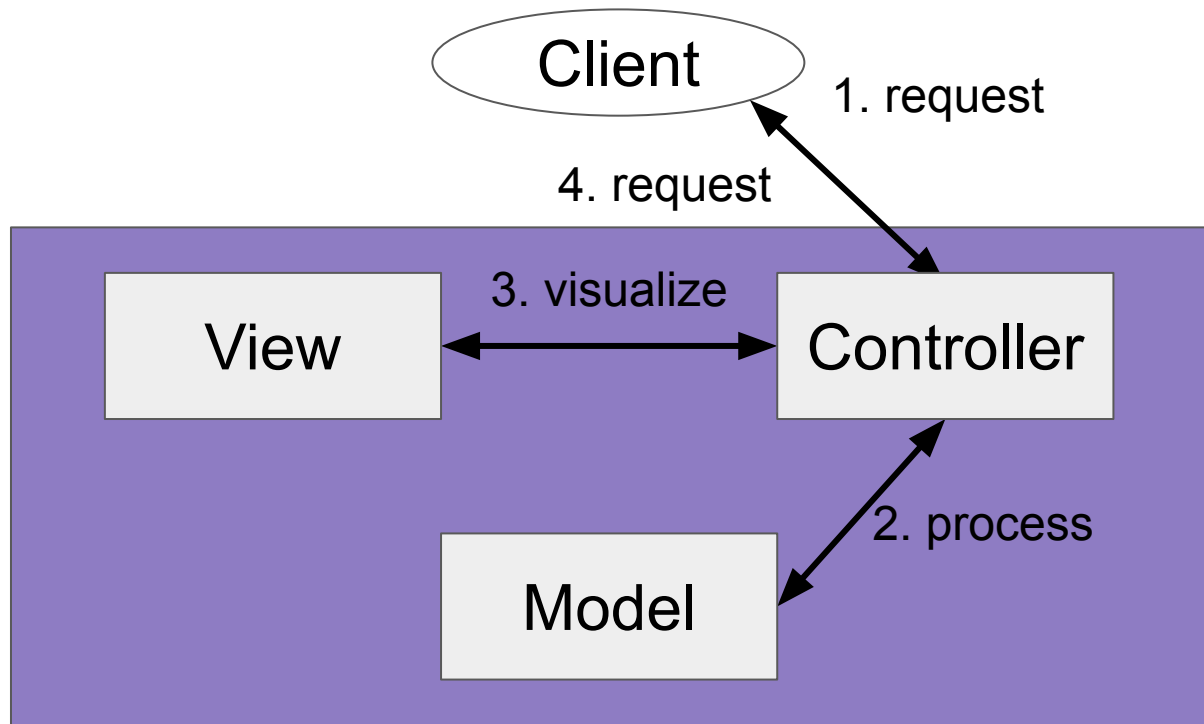
What if we want to focus on the Client and Server Interactions?

## SW Architecture #5: MVC *(or one version of it)*



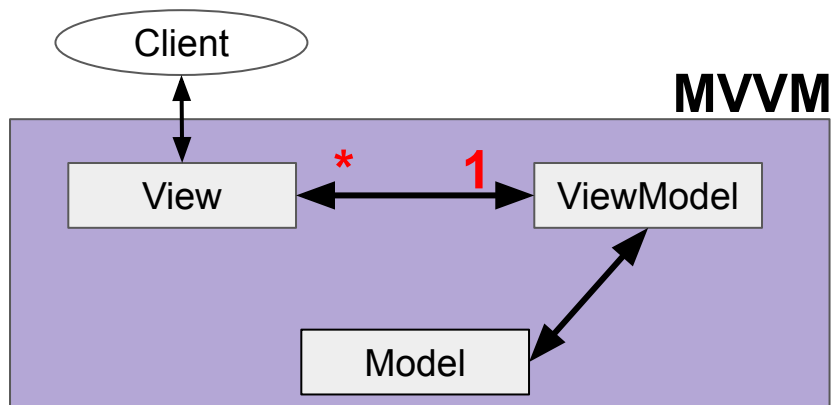
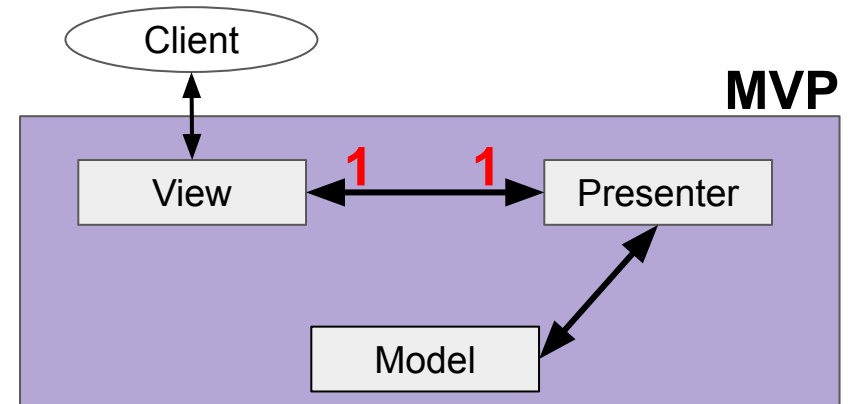
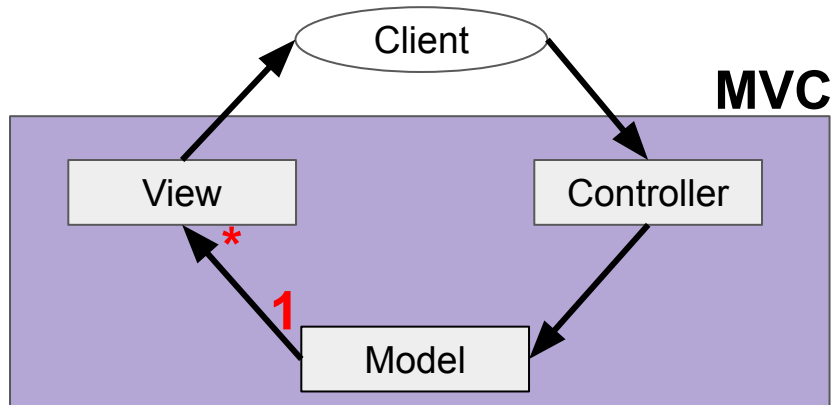
Separates data representation (Model),  
visualization (View), and client interaction (Controller)

# SW Architecture #5: MVC *(or another version of it)*

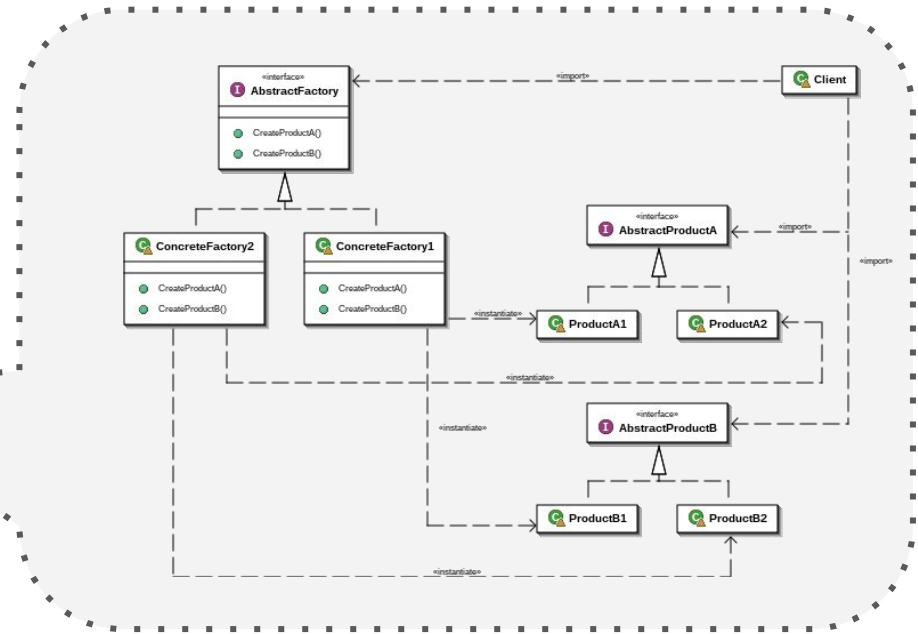
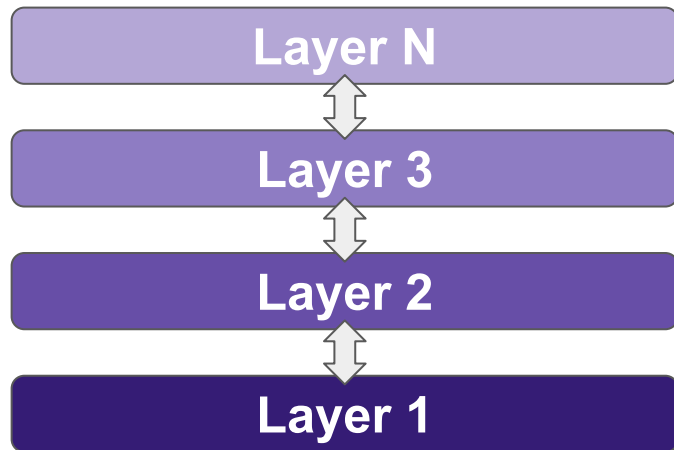


There are different variations out there!

# SW Architecture #5.2: MVC vs. MVP vs. MVVM



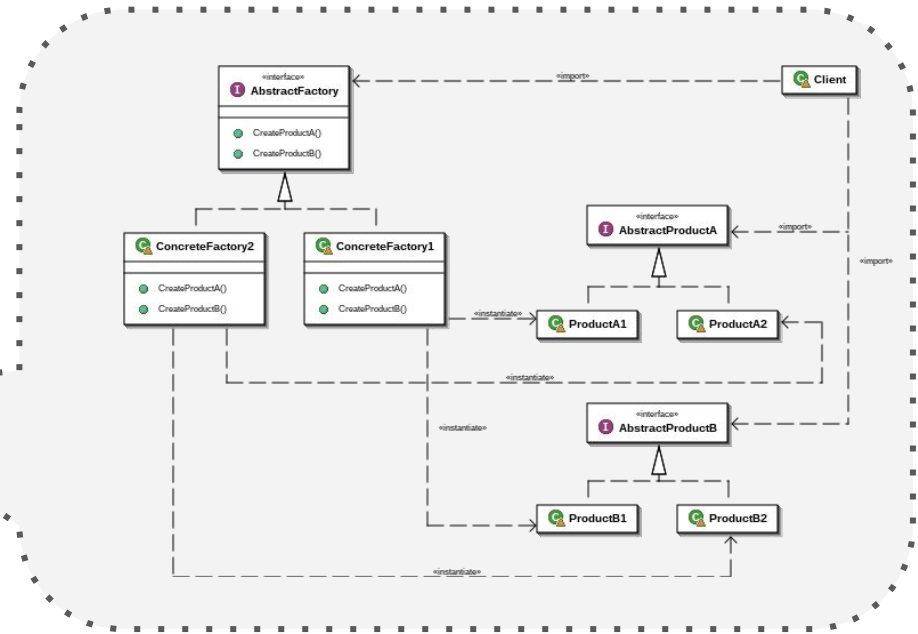
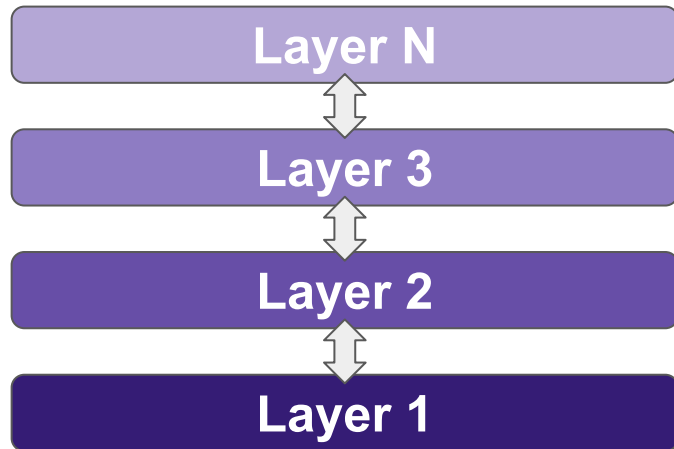
# Software architecture vs. design: summary



## Architecture and design

- Components and interfaces: understand, communicate, reuse
- Manage complexity: modularity and separation of concerns
- Process: allow effort estimation and progress monitoring

# Software architecture vs. design: summary



Questions, please!