CSE 403

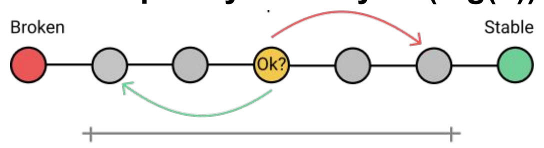
Software Engineering

Winter 2023

Software architecture

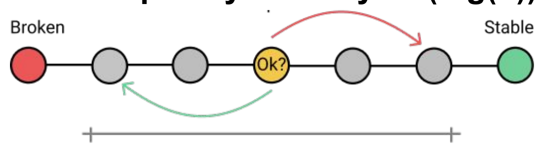
Recap: In-class exercise

- **Git bisect time complexity is always O(log(n))**

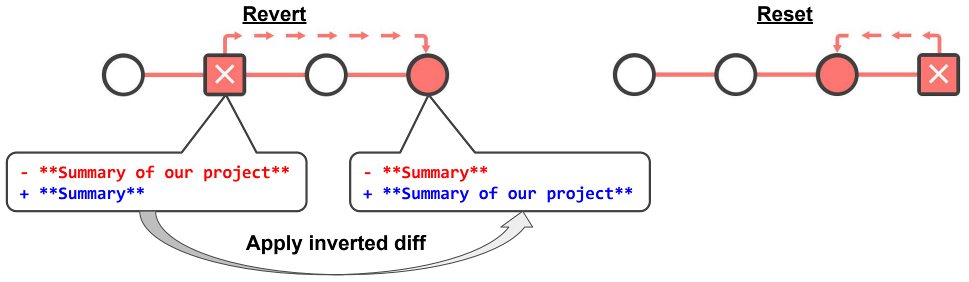


Recap: In-class exercise

- **Git bisect time complexity is always O(log(n))**

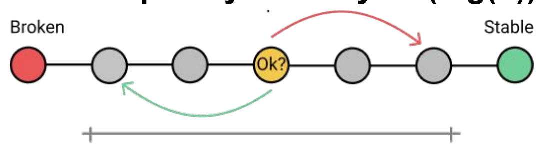


- **Git revert vs. git reset**



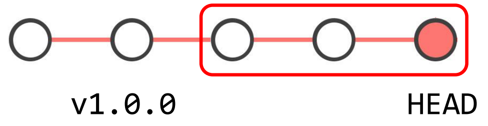
Recap: In-class exercise

- **Git bisect time complexity is always O(log(n))**



- **Git revert vs. git reset**

- `git rev-list v1.0.0..HEAD (or HEAD ^v1.0.0)`



Today

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

Software architecture vs. software design

Why software architecture and design?

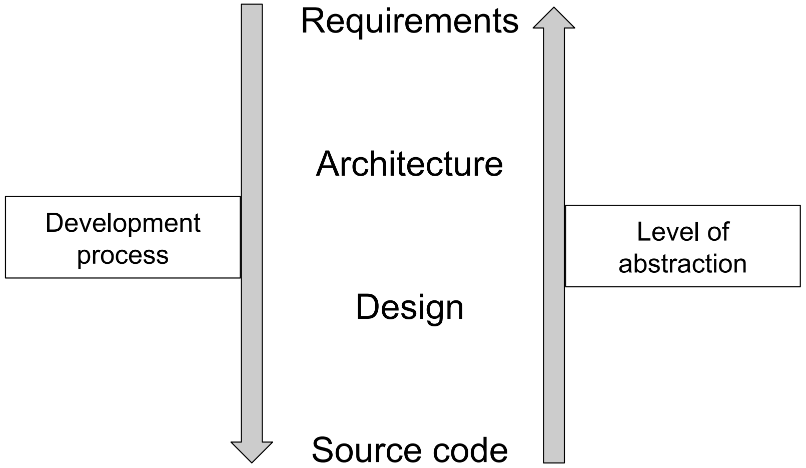
“There are two ways of constructing a software design:

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



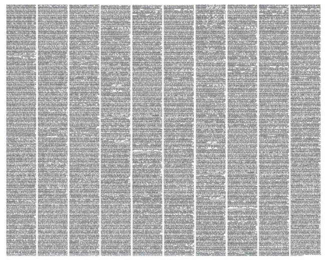
Abstraction

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



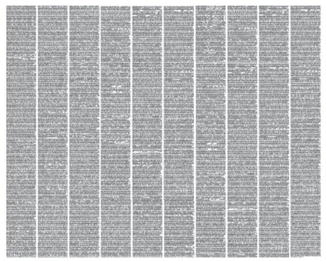
Example: Linux Kernel

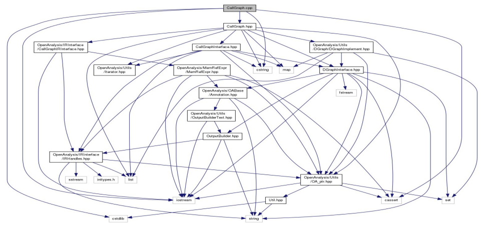
- 16 million Lines of Code!
- What does the code do?
- Are there dependencies?
- Are there different components?

Different levels of abstraction

Call graph

Source code





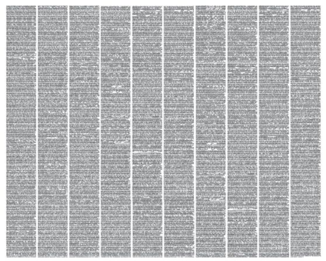
Example: Linux Kernel

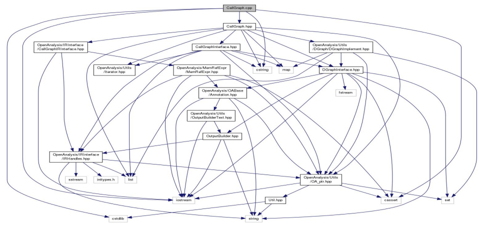
- 16 million Lines of Code!
- What does the code do?
- **Are there dependencies?**
- Are there different components?

Different levels of abstraction

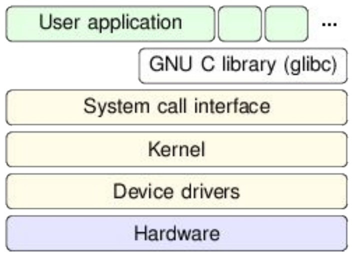
Call graph

Source code





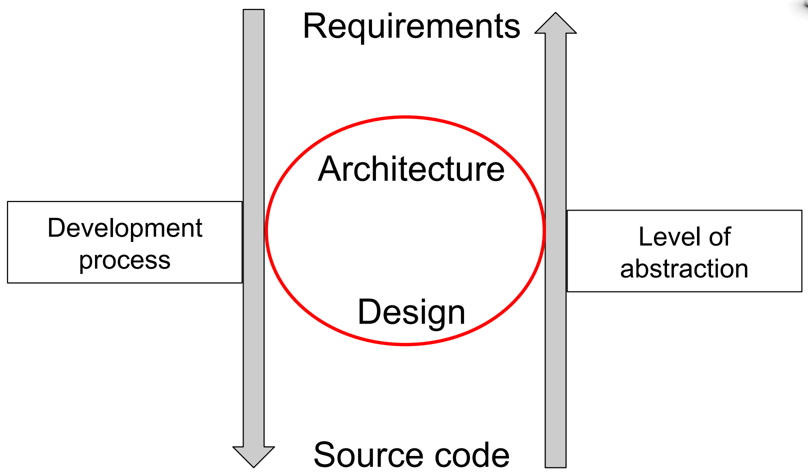
Layer diagram



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 is developed?)

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

Design (how are the components developed?)

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

Architecture vs. design

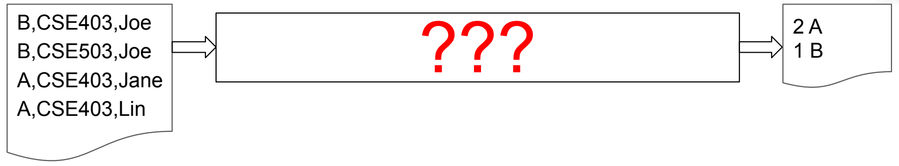
Architecture

[Gates Center Architecture, LMN]

Design

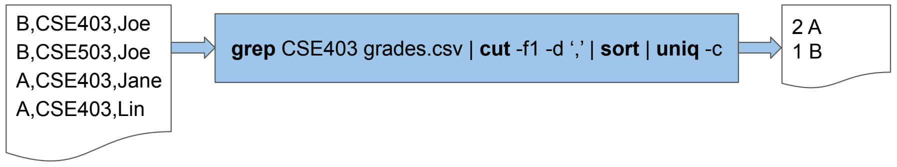
[Office design, New York Times]

A first example



Goal: group and count CSE403 letter grades.

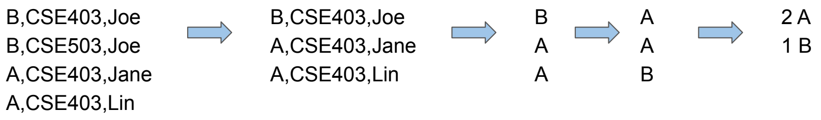
Pipe and filter



```
B,CSE403,Joe
B,CSE503,Joe
A,CSE403,Jane
A,CSE403,Lin
```

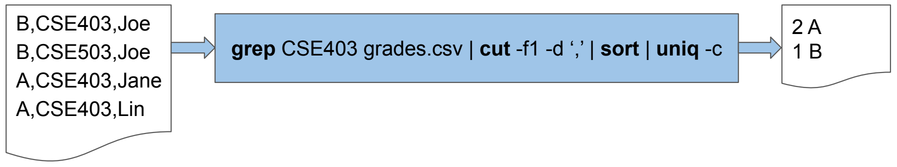
```
grep CSE403 grades.csv | cut -f1 -d ‘,’ | sort | uniq -c
```

```
2 A
1 B
```



```
B,CSE403,Joe    →   B,CSE403,Joe    →   B  →  A  →   2 A
B,CSE503,Joe        A,CSE403,Jane       A     A       1 B
A,CSE403,Jane       A,CSE403,Lin        A     B
A,CSE403,Lin
```

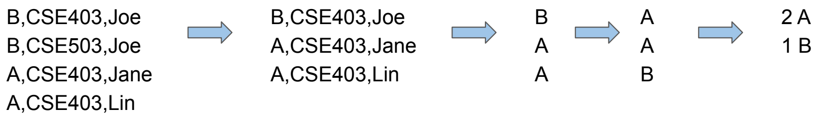
Pipe and filter



```
B,CSE403,Joe
B,CSE503,Joe
A,CSE403,Jane
A,CSE403,Lin
```

```
grep CSE403 grades.csv | cut -f1 -d ‘,’ | sort | uniq -c
```

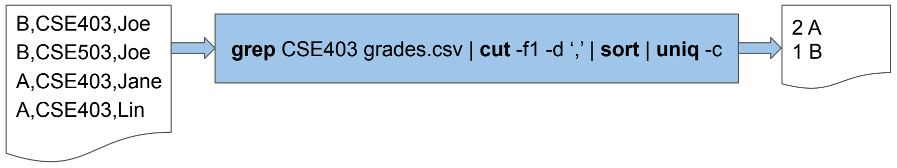
```
2 A
1 B
```



```
B,CSE403,Joe    →   B,CSE403,Joe    →   B  →  A  →   2 A
B,CSE503,Joe        A,CSE403,Jane       A     A       1 B
A,CSE403,Jane       A,CSE403,Lin        A     B
A,CSE403,Lin
```

Pipe and filter is an architecture (not a design) pattern, why?

Software architecture: Pipe and Filter



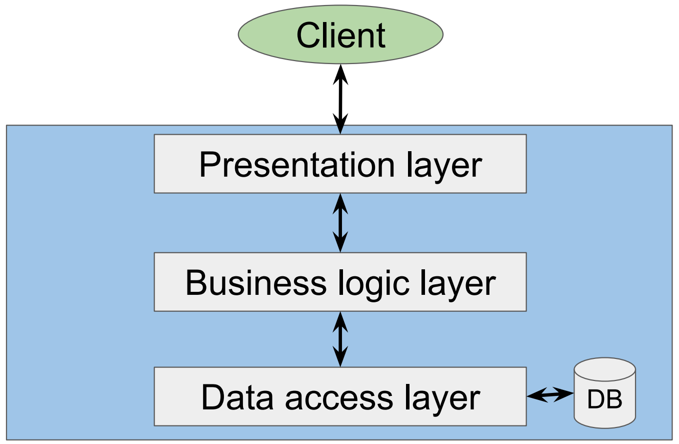
```
B,CSE403,Joe
B,CSE503,Joe
A,CSE403,Jane
A,CSE403,Lin
```

```
grep CSE403 grades.csv | cut -f1 -d ‘,’ | sort | uniq -c
```

```
2 A
1 B
```

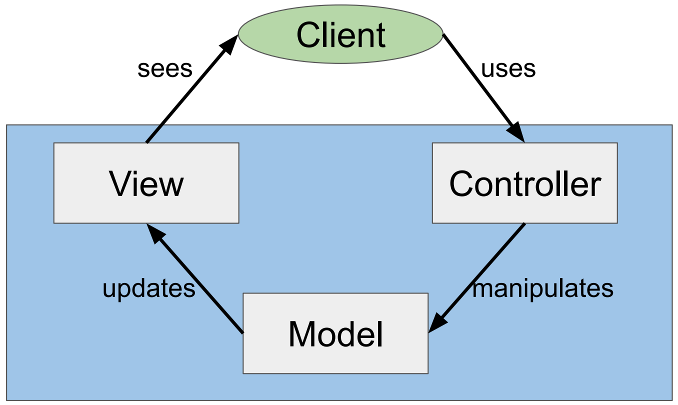
The pipe-and-filter architecture doesn’t specify the design or implementation details of the individual components (filters)!

Software architecture: Client-server / n-tier



Simplifies reusability, exchangeability, and distribution.

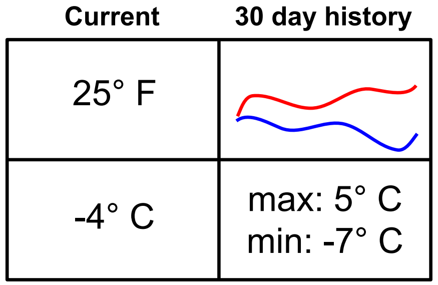
Software architecture: Model View Controller (MVC)



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

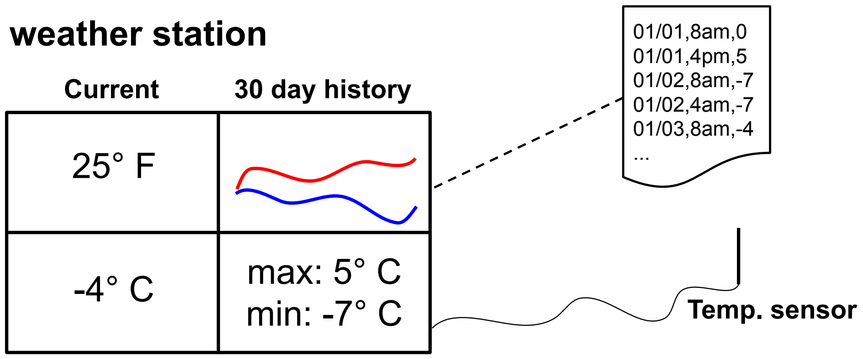
Model View Controller: example

Simple weather station



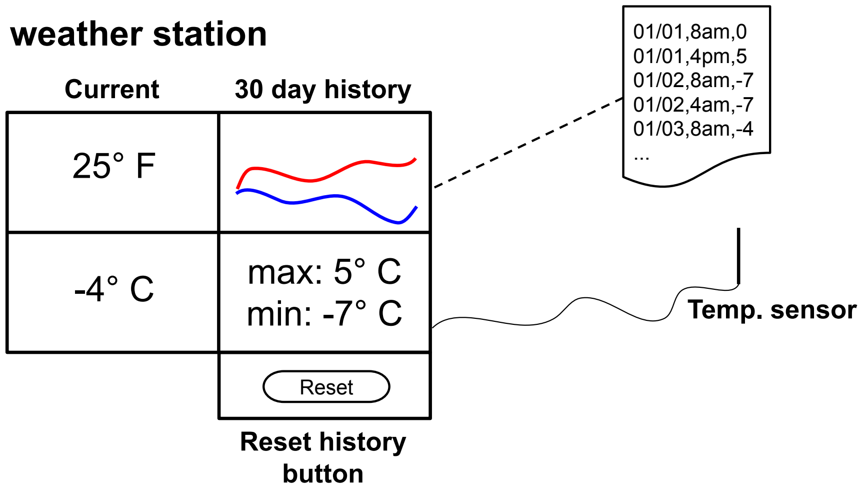
Model View Controller: example

Simple weather station



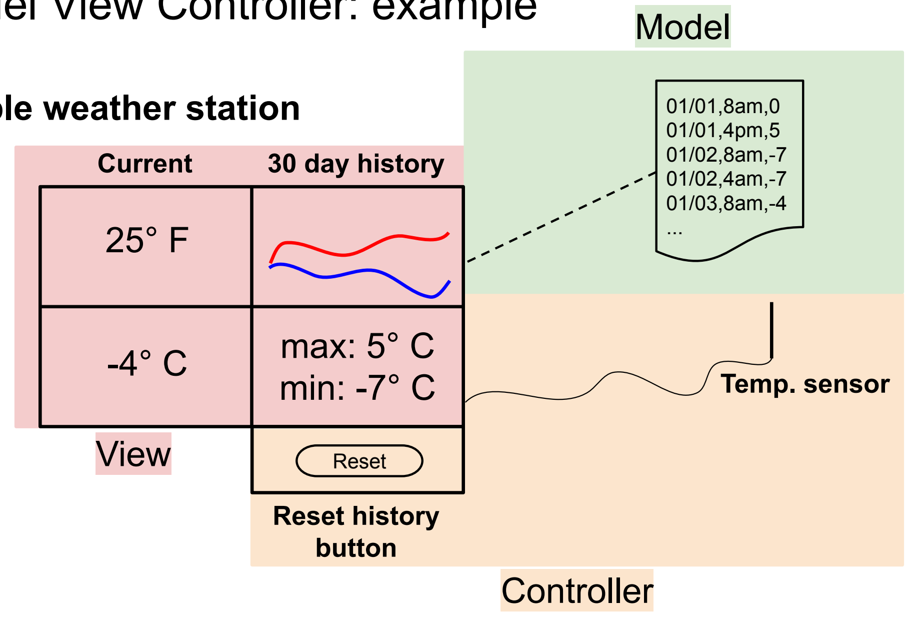
Model View Controller: example

Simple weather station

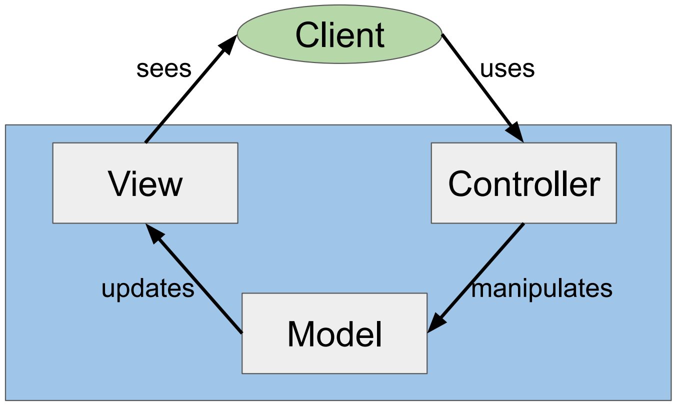


Model View Controller: example

Simple weather station

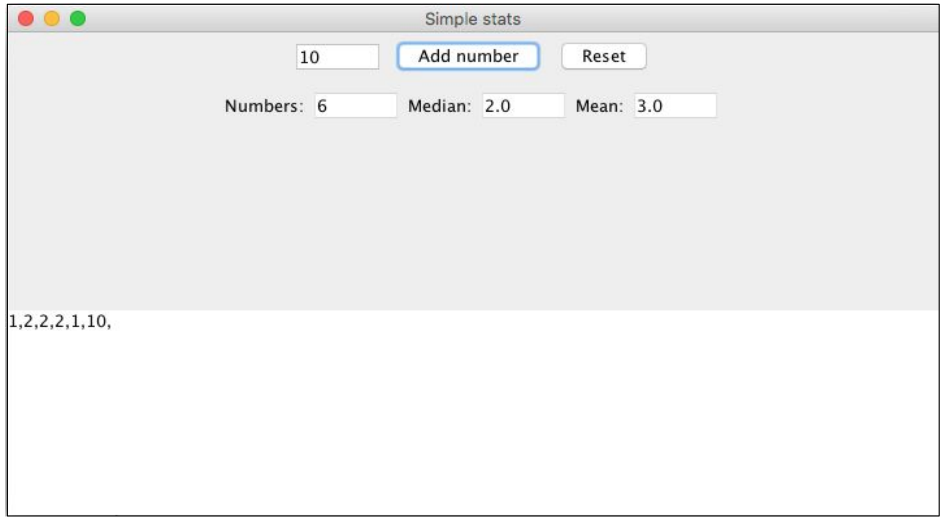


Software architecture: Model View Controller (MVC)



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

MVC: another example

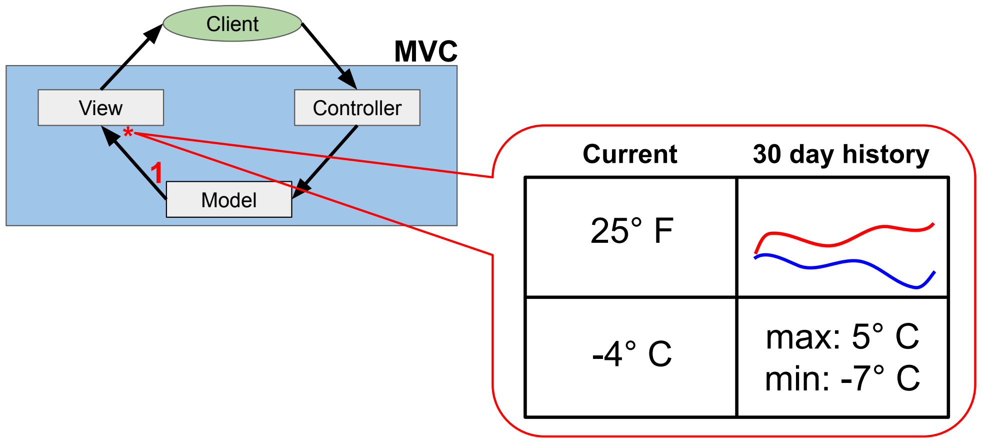


Numbers: 6 Median: 2.0 Mean: 3.0

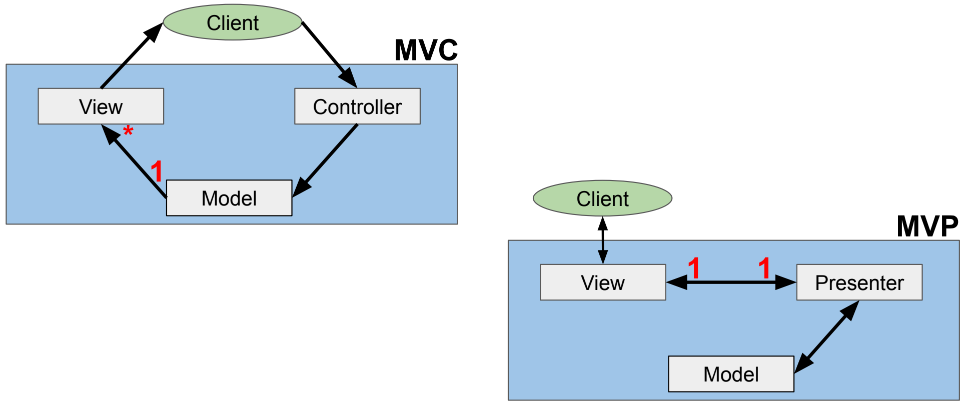
1,2,2,2,1,10,

https://bitbucket.org/rjust/basic-stats

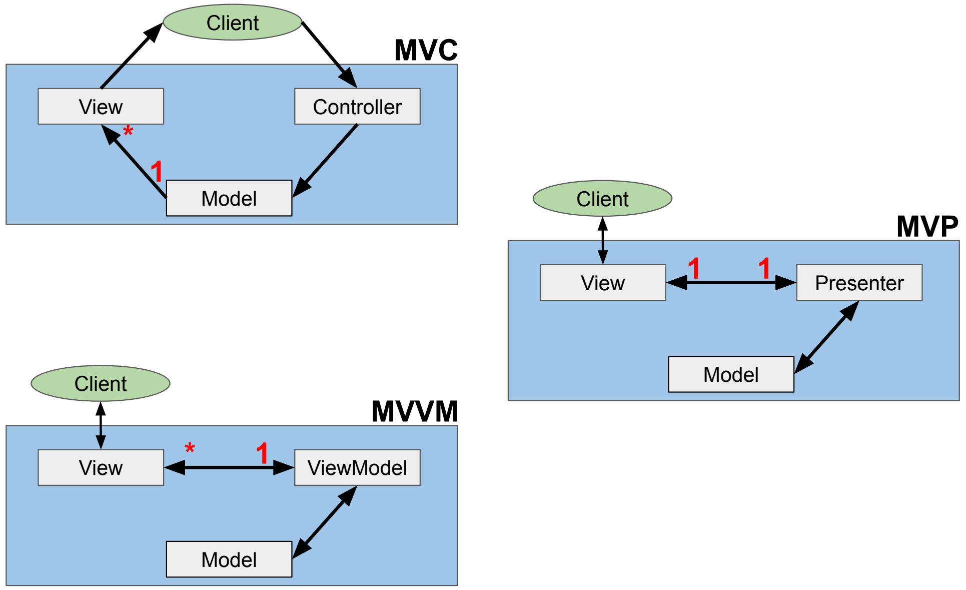
MVC vs. MVP vs. MVVM



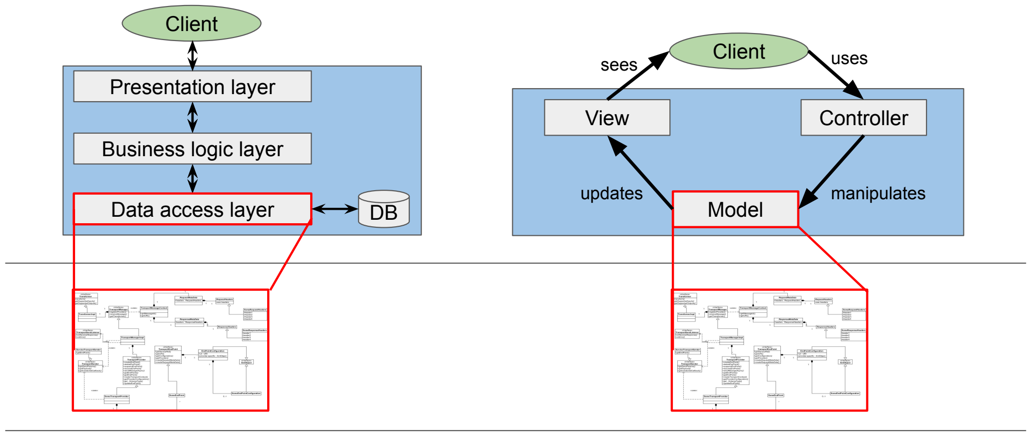
MVC vs. MVP vs. MVVM



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