Announcements, 3/16/2023

Today: Software Architecture

Break around 11:15am

1

Acknowledgements

Some of these slides are based on the lecture notes from Prof. Alex Kuhn, Prof. Emina Torlak at University of Washington, and Ian Sommerville's Software Engineering textbook

.

2

Thoughts on reading

- Was anything particularly interesting or unexpected?
- Any points that I could clarify further?
- Anything you disagreed with?

3

Outline

- What is software architecture?
- Why is software architecture design important?
- What makes good architecture?
- Examples of different architecture styles

4

What is software architecture?

How do we bridge requirements and code?

- **Software architecture:** The fundamental structure to build and evolve a software system
-
- Software architecture is similar to blueprints for an architect

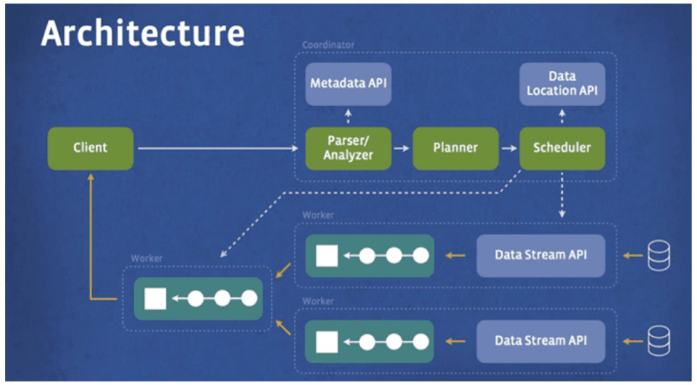
IEEE definition

Architecture is the fundamental organization of a software system embodied in its components, their relationships to each other and to the environment, and the principles guiding its design and evolution.

Architecture abstractions

- Multiple levels of abstractions
- On the small scale, architecture refers to how a program is decomposed into components
- On a large scale, it is concerned with the architecture of complex enterprise systems that include other systems and programs (distributed over many computers and potentially owned by different companies)

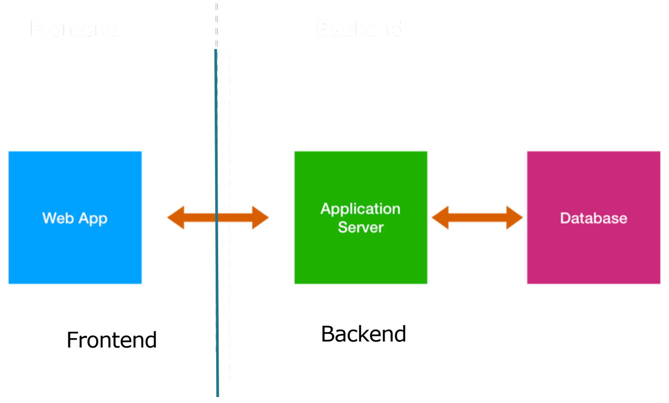
What does software architecture look like?



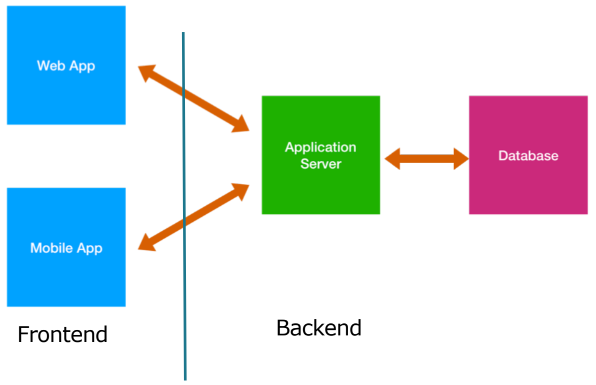
Presto DB Architecture – A High Performing Distributed Relational Database

From https://www.8bitmen.com/what-database-does-facebook-use-a-1000-feet-deep-dive/

Simple web architecture diagram



Supporting web + native app



Fundamental parts of architecture

- **Components**: a component is an element that implements a coherent set of functionality or features
  - Deliberately broad: can be a class, package, library, etc.
  - Usually interacts with other components through well defined interfaces or connectors

Fundamental parts of architecture

- **Connectors**: a connector defines how components are connected together
  - Deliberately broad definition for anything that transmits information between components
  - Can be function calls, API calls, requests, etc.
  - Connector mechanisms do not store state or functionality themselves

Why is software architecture design important?

(short answer)

- Requirements change
- Thus your code must also change
- Good architecture makes code easier to change

Additional reasons why architecture is important

- To create a reliable, secure and efficient product, you need to pay attention to architecture design which includes:
  - Overall organization
  - How the software is decomposed into components
  - Server structure
  - Technologies that you use to build the software
- The architecture of a software product affects its performance, usability, security, reliability and maintainability

Architecture design

- Need to design an overall system architecture early on
  - This design usually affects many components in the system, so refactoring the system architecture is quite expensive
- When designing software architecture, you do not need to decide how each individual component is implemented
  - You design the interface for the components first, and determine the implementation later in the process

17

Many factors influence architectural choices

- **Product lifetime** – How long will the product last? How many revisions?
- **Software comparability** – Does it need to be compatible with other software?
- **Number of users** – How many users do you need to support? Could this change rapidly?
- **Nonfunctional product characteristics** – Any security or performance requirements?
- **Software reuse** – Can you reuse large components from other products?

18

What makes a good architecture?

19

Goals for a good architecture

- Satisfies requirements
- Manages complexity of project
- Can handle changes and evolutions

20

Essence of good software design

Good design is easier to change than bad design

- Not always true, but a good general guide

21

Key architecture design principle

- Separation of concern between components
  - The less each component knows about the others, the easier to change
  - Use encapsulation to hide information
  - Makes for more modular programs

More modular systems are easier to understand, reuse, and evolve.

22

To achieve modularity

- Think about interfaces between components
- **Public interface:** code that can be seen and run by other components
- **Private implementation:** data and methods that are only accessible within the component

- You want stable interfaces between components that change slowly

23

Key properties of architectures

- **Cohesion:** How closely operations in a component are related (low versus high)
- **Coupling:** How interdependent components are (low versus high)

- **Desire high cohesion and low coupling**

Learn more at:
http://en.wikipedia.org/wiki/Coupling_(computer_programming)
http://en.wikipedia.org/wiki/Cohesion_(computer_science)

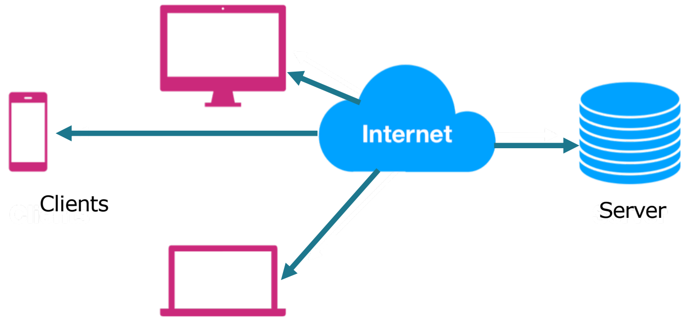
24

How is this class' *cohesion*?

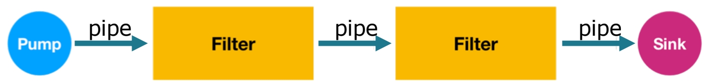
```
class Person {
    var name: String
    var address: Address
    var nearbyRestaurants: [Restaurant]
    var phoneNumber: String

    func validatePhoneNumber(number: String) {
        [...]
    }

    func updateAddressOnServer(address: Address) {
        [...]
    }

    func displayPersonInfoOnContactScreen() {
        [...]
    }
}
```

Example architecture styles

Client-server architecture

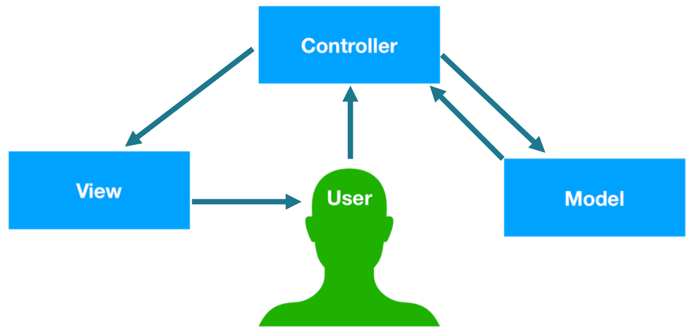


Pipe and filter



- Examples:
  - Command lines
  - Compilers

Model-view-controller (MVC)



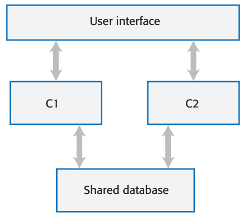
29

Architecture involves tradeoffs

30

Shared database architecture

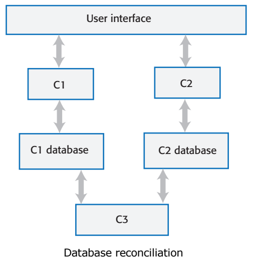
- Example of a system with two components (C1 and C2) that share a common database
  - Assume C1 runs slowly because it has to reorganize the information in the database before using it
  - The only way to make C1 faster might be to change the database
  - This means that C2 also has to be changed, which may affect its response time



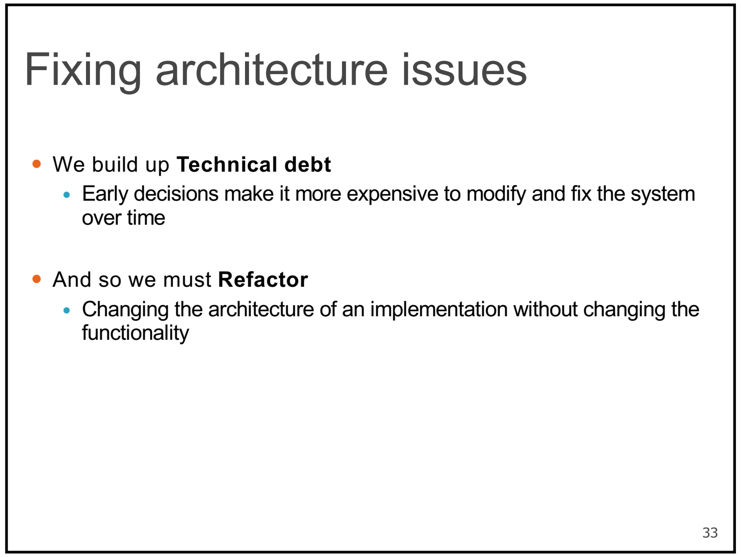
31

Multiple database architecture

- Here each component has its own copy of the parts of the database that it needs
  - If one component needs to change the database organization, this does not affect the other component
- However, a multi-database architecture may run more slowly and may cost more to implement and change
  - A multi-database architecture needs a mechanism (component C3) to ensure that the data shared by C1 and C2 is kept consistent when it is changed



32



Fixing architecture issues

- We build up **Technical debt**
  - Early decisions make it more expensive to modify and fix the system over time
- And so we must **Refactor**
  - Changing the architecture of an implementation without changing the functionality

33