Architecture and Design

CSE 403 Software Engineering
Autumn 2023
We are moving through the SDLC components
Questions about GitHub

Class flow -> Requirements -> Architecture -> Version control (git), CD/CI

But in practice!

- Need to be doing some of github setup in parallel
- Public/private is your decision, but staff need to have access
- Resources, as well as your TAs:
  - Projects tab on class website
  - Assignments, especially Git-Testing-CI assignment
  - Great material online and videos, including
    - Git, GitHub, and GitHubDesktop for beginners (youtube) (https://www.youtube.com/watch?v=8Dd7KRpKeaE)
    - Git tutorial (https://git-scm.com/docs/gittutorial), and
    - Becoming a Git Guru (https://www.atlassian.com/git/tutorials)
Today’s Outline

Architecture

1. What do we mean by architecture
2. How does it differ from design
3. What are some common architectures used in software
What does “Architecture” make you think of?

MIT Stata Center by Frank Gehry

Paul G. Allen Center by LMN Architects
In contrast, what comes to mind with “Design”?
Here’s another example close to home

Bill & Melinda Gates Center for UW CSE - LMN
Let’s transition the ideas to software engineering

- Requirements
- Architecture
- Design
- Source code

Development process

Level of abstraction
The level of abstraction is key

With both architecture and design, we’re building an abstract representation of reality

• Ignoring (insignificant details)
• Focusing on the most important properties
• Considering modularity (separation of concerns) and interconnections
Source code

Suppose you want to add a feature 16 million lines of code! Where would you start?

• What does the code do?
Case study – Linux kernel

Suppose you want to add a feature 16 million lines of code! Where would you start?

- What does the code do?
- **Are there dependencies?**
Case study – Linux kernel

Layer diagram

Suppose you want to add a feature
16 million lines of code!
Where would you start?

• What does the code do?
• Are there dependencies?
• What are the different components?
How about some definitions

Architecture (what components are needed)
• High-level view of the overall system:
  • What components do exist?
  • What are the connections and/or protocols between components?

Design (how the components are developed)
• Considers individual components:
  • Data representation
  • Interfaces, class hierarchy
Today’s Outline

Architecture
1. What do we mean by architecture
2. How does it differ from design
3. **What are some common architectures used in software**  
   We are here
The **pipe-and-filter** architecture talks about the main components and the way they connect.

Filter computes on the data.

Pipe passes the data.

It doesn’t specify the design or implementation details of the individual components (the filters).
SW Architecture #1 – let’s try it out

How would you attack this problem?
Count the CSE 403 letter grades

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

???

2 A
1 B
...

UW CSE 403 Au23
SW Architecture #1 – Pipe and filter

You might start by thinking of **components** and **successive filtering** (architecture)

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

**Process1() -> Process2() -> Processn()**

UW CSE 403 Au23
SW Architecture #1 – Pipe and filter

You might then consider the components’ inputs and outputs (design)

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

Grab() -> Select() -> Order() -> Count()

B,CSE403,Joe
B,CSE403,Joe
A,CSE403,Jane
A,CSE403,Lin
...

2 A
1 B
Finally, you get to **code**

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

```
B,CSE403,Joe  B,CSE403,Joe  B  2 A
B,CSE503,Joe  A,CSE403,Jane  A  1 B
A,CSE403,Jane  A,CSE403,Lin  A  A
A,CSE403,Lin  A  A  B
...  ...
```
SW Architecture #2 – Layered (n-tier)

- Layers use services provided (only) by the layers directly below them
- Layers of isolation – limits dependencies
- Good modularity and separation of concerns
SW Architecture #2 – Layered

Linux Architecture

Enterprise System Architecture

Pros / cons?
SW Architecture #2 – Layered

User Space
-------------------------------
Support Services & daemons
-------------------------------
Libraries
-------------------------------
Operating System
-------------------------------

Runs on all Service and IO Nodes

Linux
-------------------------------
Common
-------------------------------

Dedicated Network & External RAID IO

IO Node
-------------------------------
Specialized by Node Function

IO & FS daemons

RCA daemon

Portals

Boot

IP

Parallel File Systems

RCA

Portals Driver Support

Custom

3rd Party

3rd Party Integration

Custom

Open Source

Application code

Message passing code

Low Level Network code

UW CSE 403 Au23
SW Architecture #3 – Client Server

Clients can be software that depends on a shared database/service.

What might be a con of this and how might it be avoided?
SW Architecture combinations!

Client-Server may be too high a level of abstraction for your purpose. Consider combining with other patterns (e.g., layered).
SW Architecture combinations^2

How detailed should an architecture description be?
SW Architecture #4 – Model View Controller

Separates
• data representation (Model)
• visualization (View)
• client interaction (Controller)
## SW Architecture #4 – MVC Example

<table>
<thead>
<tr>
<th>Current</th>
<th>30 day history</th>
</tr>
</thead>
<tbody>
<tr>
<td>54°F</td>
<td></td>
</tr>
<tr>
<td>12.2°C</td>
<td>Max: 60°F</td>
</tr>
<tr>
<td></td>
<td>Min: 52°F</td>
</tr>
</tbody>
</table>

- 09/09, 8am, 50°F
- 09/09, 4pm, 51°F
- 09/10, 8am, 52°F
- 09/10, 4pm, 54°F

Today's temperature is forecast to be **NEARLY THE SAME** as yesterday.
SW Architecture #4 – MVC Example

Current

54 F

12.2 C

30 day history

Max: 60 F
Min: 52 F

Reset

09/09, 8am, 50
09/09, 4pm, 51
09/10, 8am, 52
09/10, 4pm, 54
...

UW CSE 403 Au23
SW Architecture – many variants of MVC

Consider the connections (* == many)
As an architect (and designer), consider ...

**Level of Abstraction**
- Components (modules) and their interconnections (apis)

**Separation of concerns**
- Strong cohesion – tight relationships within a component (module)
- Loose coupling – interconnections between components (module)

**Modularity**
- Decomposable designs
- Composable components
- Localized changes (due to requirement changes)
- Span of impact (how far can an error spread)
A good architecture is critical to success

Helps with:

• **System understanding**
  • components and their interactions

• **Reuse**
  • high-level view shows opportunity for reuse

• **Development**
  • breaks development down into manageable pieces; provides a path from requirements to code

• **Management**
  • helps understand scope of work and track progress

• **Communication, shared vision**
  • provides vocabulary; pictures say 1000 words
Questions?

Architecture

1. What do we mean by architecture
2. How does it differ from design
3. What are some common architectures used in software
   1. Pipe and filter
   2. Layer diagram
   3. Client – Server
   4. Model – View – Controller
   5. … Message passing …
   6. Many variations!

Use your project to get experience with defining an architecture

Discuss questions with your 403 TA (senior mgr/mentor)