#11: Software architecture
## Logistics

<table>
<thead>
<tr>
<th>Date</th>
<th>Activity</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>04/17</td>
<td>L: Data modeling</td>
<td></td>
</tr>
<tr>
<td>04/18</td>
<td>T:</td>
<td>DUE: GPS!!!</td>
</tr>
<tr>
<td>04/19</td>
<td>L: Architecture</td>
<td>Design &amp; Architecture (DnA)</td>
</tr>
<tr>
<td>04/20</td>
<td>P:</td>
<td></td>
</tr>
<tr>
<td>04/21</td>
<td>L: Design</td>
<td></td>
</tr>
</tbody>
</table>
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

- Requirements
- Architecture
- Design
- Source code

Development process

Level of abstraction
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

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

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

Design

[Gates Center Architecture, LMN]

[Office design, New York Times]
Provocation: How would you solve this problem?

Goal: count CSE403 letter grades.
Provocation: The "coding" way!

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

Lower level of abstraction: lang, code, run!
Provocation: The "design" way

Mid-level of abstraction: component specification!
Provocation: The "architecture" way

Process1() → Process2() → …() → ProcessN()

Higher level of abstraction: components concatenation!
The pipe-and-filter **architecture** doesn’t specify the **design** or **implementation** details of the individual components (the filters)!
SW Architecture #2: ???

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

Which architectural model we have already talked about here?
SW Architecture #2: Layered

Example: Linux Kernel
- 16 million Lines of Code!
- What does the code do?
- Are there dependencies?
- Are there different components?
SW Architecture #2: Layered

Layer 1

Layer 2

Layer 3

Layer N

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

Client X

Presentation layer

Business logic layer

Data access layer

Client Y

DB a

DB b

DB c

What if things in the server gets too complicated?
What if things in the server even more complicated and need help to process the load?
What if we want to focus on the Client and Server Interactions?
SW Architecture #6: MVC (or one version of it)

Separates data representation (Model), visualization (View), and client interaction (Controller).
SW Architecture #6: MVC *(or another version of it)*

There are different variations out there!
SW Architecture #6.2: MVC vs. MVP vs. MVVM

MVC

- View
- Controller
- Model

MVP

- View
- Presenter
- Model

MVVM

- View
- ViewModel
- Model
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!