Recap: Logistics

- Will meet in person on Tuesday and Thursday, 10 AM – 11:15 AM and will also be recorded
  - Lectures, in-class exercises, course project presentations
- Course material, policies, and schedule on web site: https://people.cs.umass.edu/~hconboy/class/2021Fall/CS520/
- Submission of assignments via Moodle: https://umass.moonami.com/course/view.php?id=916
Recap: Software Engineering

What is Software Engineering?
The complete process of specifying, designing, developing, analyzing, deploying, and maintaining a software system.

Why is it important?
- Software is everywhere and complex.
- Software defects are expensive and range from annoying to life threatening.

Goals
- Decompose a complex engineering problem.
- Organize processes and effort.
- Improve software reliability.
- Improve developer productivity.
Recap: Software Engineering

What is Software Engineering?
The complete process of specifying, designing, developing, analyzing, deploying, and maintaining a software system.

Why is it important?
- Software is everywhere and complex.
- Software defects are expensive and range from annoying to life threatening.

Goals
- Decompose a complex engineering problem.
- Organize processes and effort.
- Improve software reliability.
- Improve developer productivity.
Today

- Requirements engineering
- Architecture versus design
Today

- Requirements engineering
- Architecture versus design
Requirements Engineering: Stakeholders

- “individuals and organizations who are actively involved in the project, or whose interests may be positively or negatively affected as a result of project execution or successful project completion”

[Project Management Institute (PMI®), 1996]
Example: Electronic gradebook
Requirements Engineering: What is a software requirements specification?

- Documents the assumptions about, features requested, and behavior of a given software application excepted by the users

- Defines a set of requirements that must be satisfied by the software application
Requirements Engineering: What is a software requirements specification?

- Documents the assumptions about, features requested, and behavior of a given software application excepted by the users

- Defines a set of requirements that must be satisfied by the software application
Requirements Engineering: Two key types of requirements

- **Non-functional requirement**: A quality constraint on the software application (often called the ‘ilities’), e.g., understandability

- **Functional requirement**: An intended (or unintended) behavior of the software application, e.g., Initially, the electronic gradebook needs to allow registered users to login to it.

**NOTE** There are other types of requirements to describe assumptions, features, and usage scenarios (e.g., UML use cases).
Requirements Engineering:
Two key types of requirements

- **Non-functional requirement**: A quality constraint on the software application (often called the ‘ilities’), e.g., understandability
- **Functional requirement**: An intended (or unintended) behavior of the software application, e.g., Initially, the electronic gradebook needs to allow registered users to login to it.

*NOTE* There are other types of requirements to describe assumptions, features, and usage scenarios (e.g., UML use cases).
Requirements Engineering: Non-functional Requirements Example

Agility in a software application supports the following:

- Debuggability
- Extensibility
- Portability
- Scalability
- Securability
- Testability
- Understandability

Requirements Engineering: Understandability Example
Today

- Requirements engineering
- Architecture versus design
Today

- Requirements engineering
- Architecture versus design
Software development: the high-level problem
Software development: the high-level problem

One solution: “Hope for the best”
Software development: the high-level problem

Another solution: Modeling the architecture and design

Specification  ???  Source code
What is modeling?

Building an abstract representation of reality

- Ignoring (insignificant) details.
- Level of abstraction depends on viewpoint and purpose:
  - Communication
  - Verification
  - Code generation
- Focusing on the most important aspects/properties.
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 layers?
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 layers?
Architecture vs. design

Specification

Architecture

Design

Source code

Development process

Level of abstraction

What’s the difference?
Software architecture vs. design

Architecture (what components are developed?)
- Considers the system as a whole:
  - High-level view of the overall system.
  - What components exist?
  - What type of storage, database, communication, etc?

Design (how are the components developed?)
- Considers individual components:
  - Data representation
  - Interfaces, Class hierarchies
  - ...

A first example: Goal

Goal: group and count CS520 grades.
Software architecture: Pipe and Filter

```
B,CS520,Juan
B,CS320,Juan
A,CS520,Jane
A,CS520,Jon
```

```
grep CS520 grades.csv | cut -f 1 -d ',' | sort | uniq -c
```

```
2 A
1 B
```

Pipes represents as “|”

The 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

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

**Simple weather station**

<table>
<thead>
<tr>
<th>Current</th>
<th>30 day history</th>
</tr>
</thead>
<tbody>
<tr>
<td>25° F</td>
<td></td>
</tr>
<tr>
<td>-4° C</td>
<td>min: 20° F max: 35° F</td>
</tr>
</tbody>
</table>

- Reset

- Reset history button

01/01 -> 0
01/02 -> -5
01/03 -> -10
01/04 -> -4
...

Temp. sensor
Model View Controller: example

Simple weather station

Current | 30 day history
--- | ---
25° F | 
-4° C | min: 20° F max: 35° F

View

Controller

Model

Temp. sensor

Reset history button

Reset

01/01 -> 0
01/02 -> -5
01/03 -> -10
01/04 -> -4
...

01/01 -> 0
01/02 -> -5
01/03 -> -10
01/04 -> -4
Real World Example:
UMass Amherst CICS Weather Station

- Located on top of the CS building
- Here are two different User Interfaces (UIs)
Identifying architecture patterns (1)

1. A web browser (e.g., Chrome, Edge, Safari) applies which of the following architecture patterns?
   
   a) Client-Server (or n-tier)
   
   b) Model View Controller
   
   c) Pipe and Filter
Identifying architecture patterns (2)

2. Here is a common compiler architecture. Which architecture pattern is being applied?

a) Client-Server (or n-tier)

b) Model View Controller

c) Pipe and Filter
Identifying architecture patterns (3)

2. Here is the TCP/IP protocol architecture. Which architecture pattern is being applied?
   
   a) Client-Server (or n-tier)
   
   b) Model View Controller
   
   c) Pipe and Filter

https://www.w3.org/People/Frystyk/thesis/Tcplp.html
Summary: Software architecture vs. design

Architecture and design goals
- Lower complexity: separation of concerns, well defined interfaces
- Simplify communication
- Allow effort estimation and progress monitoring