CS 520
Theory and Practice of Software Engineering
Fall 2022

Software architecture

September 13, 2022
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: User story

- An informal, natural language description of a feature of a software system written from the perspective of an end user.

- This description should generally specify:
  - who wants the given feature
  - what that feature needs to do
  - why the feature is wanted

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
Requirement Engineering:
Two kinds 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).
Example: Set of non-functional requirements
Requirements Engineering: Phases

1. Elicitation
2. Specification
3. Analysis
4. Management

Requirements → Architecture & design → Implementation
Requirements Engineering: Phases

1. Elicitation
2. Specification
3. Analysis
4. Management

Requirements \rightarrow Architecture & design \rightarrow Implementation

Requirements \leftarrow Architecture & design \leftarrow Implementation
Requirements Engineering: Phases

1. Elicitation
2. Specification
3. Analysis
4. Management

Requirements -> Architecture & design -> Implementation

Requirements <-> Architecture & design <-> Implementation
Software development: the high-level problem

Specification → ??? → Source code
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 $\rightarrow$ ??? $\rightarrow$ 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

Example: Linux Kernel
- 16 million Lines of Code!
- What does the code do?
- Are there dependencies?
- Are there different layers?
Architecture vs. design

What's the difference?

Development process

Specification

Architecture

Source code

Design

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?
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.

grades.csv
A first example: Approach

```
grep CS520 grades.csv
```

```
B,CS520,Juan
B,CS320,Juan
A,CS520,Jane
A,CS520,Jo
```
A first example: Approach

```
grep CS520 grades.csv
```

- B,CS520,Juan
- B,CS320,Juan
- A,CS520,Jane
- A,CS520,Jo

grades.csv

2 A
1 B
A first example: Approach

grep CS520 grades.csv | cut -f 1 -d ','
A first example: Approach

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

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

```
2 A
1 B
```
A first example: Approach

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

grades.csv

B,CS520,Juan
B,CS320,Juan
A,CS520,Jane
A,CS520,Jo

2 A
1 B
A first example: Approach

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

grades.csv

<table>
<thead>
<tr>
<th>B,CS520,Juan</th>
<th>B</th>
<th>A</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>B,CS320,Juan</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A,CS520,Jane</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A,CS520,Jo</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2 A
1 B
A first example: Approach

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

2 A
1 B
Architecture or design pattern?

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

```
2 A
1 B
```
Software architecture: Pipe and Filter

The architecture doesn’t specify the design or implementation details of the individual components (filters)!
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>min: 20° F</td>
</tr>
<tr>
<td></td>
<td>max: 35° F</td>
</tr>
<tr>
<td>-4° C</td>
<td></td>
</tr>
</tbody>
</table>

Current temperature: 25° F

Historical temperatures:
- 01/01 -> 0
- 01/02 -> -5
- 01/03 -> -10
- 01/04 -> -4

Reset history button

Temp. sensor
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