CS 520
Theory and Practice of Software Engineering
Spring 2022

Software requirements and architecture

January 27, 2022
Recap: Logistics

- Will meet in person on Tuesday and Thursday, 10 AM – 11:15 AM and will also be recorded
  - Lectures, in-class exercises, final project presentations

- Course schedule and policies on web site: [https://people.cs.umass.edu/~hconboy/class/2022Spring/CS520/](https://people.cs.umass.edu/~hconboy/class/2022Spring/CS520/)

- Course materials (e.g., slides, recorded lectures, assignments) available through Moodle: [https://umass.moonami.com/course/view.php?id=30211](https://umass.moonami.com/course/view.php?id=30211)

- Q&A forums for assignments via Piazza: [https://piazza.com/umass/spring2022/CS520/home](https://piazza.com/umass/spring2022/CS520/home)

- Submission of assignments via Gradescope // Will provide the URL later
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
Example: Electronic gradebook

- **Users:** Graders, Instructors, Registrar’s Office
- **Developers:** Designers, programmers, testers, upper management, user support
- **UI experts:** Human Factors, HCI (Human-Computer Interaction)
- **Federal agencies:** ADA (Americans with Disabilities Act), FERPA (Family Educational Rights and Policy Act)
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
Requirements Engineering: Understandability Example

- Architecture diagram
- README file
- Build files (e.g., Eclipse .classpath, .project)
- Internal comments
- Naming conventions
- ...

16
Requirements Engineering: Phases

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

Requirements $\rightarrow$ Architecture & design $\rightarrow$ Implementation
Requirements Engineering: Phases

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

Requirements \[\rightarrow\] Architecture & design \[\rightarrow\] Implementation

\(\leftarrow\)
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”

Specification

???

Source code
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

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?
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

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

```
grep CS520 grades.csv
```

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

grep CS520 grades.csv

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 ','
```

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 ','`
A first example: Approach

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

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

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

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

grades.csv

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

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

2 A
1 B

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

B
A
A

A
A
B
A first example: Approach

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

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

```
B
A
A
B
```
Architecture or design pattern?

```
grades.csv

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

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)!

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

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

Pipes represents as “|”
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</td>
</tr>
<tr>
<td></td>
<td>max: 35° F</td>
</tr>
</tbody>
</table>

Reset history button

Temp. sensor

- 01/01 -> 0
- 01/02 -> -5
- 01/03 -> -10
- 01/04 -> -4
...
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</td>
</tr>
<tr>
<td></td>
<td>max: 35° F</td>
</tr>
</tbody>
</table>

Reset history button

Reset

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

Temp. sensor

Controller

Model
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 (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
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

NOTE) Could also be Client-Server where the Client submits the Program which the Server runs
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
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

NOTE) Could also be Pipe and Filter where the network packets are being passed among the layers

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