CS 520
Theory and Practice of Software Engineering
Fall 2021

Best and worst software development practices

September 9, 2021
Recap: Software requirements and architecture

- Specification
- Architecture
- Design
- Source code
- Development process
- Level of abstraction
Recap: 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]
Recap: US online banking app example
Recap: 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).
Recap: software architecture examples

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

- **N-tier / Client-Server**

- **MVC (Model-View-Controller)**
Recap: software architecture and design goals

Architecture and design goals
- Lower complexity: separation of concerns, well defined interfaces
- Simplify communication
- Allow effort estimation and progress monitoring
An in-class discussion on best and worst software development practices.
Setup and goals

● 4- or 5- person teams
● Examples
● 2 rounds
  ○ First phase
    ■ For each of 4 examples, decide whether it represents good or bad practice.
    ■ Goal: discuss and reach consensus on good or bad practice.
  ○ Second phase (known solutions)
    ■ For each example, try to understand why it is good or bad practice.
    ■ Goal: come up with one or more explanations or a counter argument.

and then repeat with 4 more examples
Round 1: Good or bad?
Example 1.1: good or bad?

src/Model-exp.java
src/Model-old.java
src/Model.java
src/View.java
src/GraphicalView.java
src/GraphicalView-opt.java
src/TextualView.java
Example 1.2: good or bad?

src/build.xml
src/Model.java
src/View.java
src/GraphicalView.java
src/TextualView.java
src/README.txt
Example 1.3: good or bad?

```java
public enum PaymentType {DEBIT, CREDIT}

public void doTransaction(double amount, PaymentType payType) {
    switch (payType) {
    case DEBIT:
        ... // process debit card
        break;
    case CREDIT:
        ... // process credit card
        break;
    default:
        throw new IllegalArgumentException("Unexpected payment type");
    }
}
```
public File[] getAllLogs(Directory dir) {
    if (dir == null || !dir.exists() || dir.isEmpty()) {
        return null;
    } else {
        int numLogs = ... // determine number of log files
        File[] allLogs = new File[numLogs];
        for (int i=0; i<numLogs; ++i) {
            allLogs[i] = ... // populate the array
        }
        return allLogs;
    }
}
Round 1: why is it good or bad?
Example 1.1: this is bad! why?

```
src/Model-exp.java
src/Model-old.java
src/Model.java
src/View.java
src/GraphicalView.java
src/GraphicalView-opt.java
src/TextualView.java
```
Example 1.2: this is good! why?

- src/build.xml
- src/Model.java
- src/View.java
- src/GraphicalView.java
- src/TextualView.java
- src/README.txt
public enum PaymentType {DEBIT, CREDIT}

public void doTransaction(double amount, PaymentType payType) {
    switch (payType) {
    case DEBIT:
        // process debit card
        break;
    case CREDIT:
        // process credit card
        break;
    default:
        throw new IllegalArgumentException("Unexpected payment type");
    }
}
Snippet 1.4: this is bad! why?

```java
public File[] getAllLogs(Directory dir) {
    if (dir == null || !dir.exists() || dir.isEmpty()) {
        return null;
    } else {
        int numLogs = ... // determine number of log files
        File[] allLogs = new File[numLogs];
        for (int i=0; i<numLogs; ++i) {
            allLogs[i] = ... // populate the array
        }
        return allLogs;
    }
}
```
Round 2: Good or bad?
Example 2.1: good or bad?

```
src/Mod.java
src/View.java
src/GView.java
src/TView.java
```
Example 2.2: good or bad?

```java
/**
 * Sets the width of this rectangle to the given positive number.
 * 
 * @param width The new width
 * 
 * @throws IllegalArgumentException when the width is a negative number
 */

public void setWidth(int width) { ... }
```
public BitSet(int size, boolean initialValue) {
    this.bitSet = new boolean[size];

    for (int i = 0; i < this.bitSet.length; i++) {
        this.bitSet[i] = initialValue;
    }
}
Example 2.4: good or bad?

```java
public void addStudent(Student student, String course) {
    if (course.equals("CS520")) {
        cs520Students.add(student);
    }
    allStudents.add(student);
}
```
Solutions

- Example 2.1:
- Example 2.2:
- Example 2.3:
- Example 2.4:
Round 2: why is it good or bad?
Example 2.1: this is bad! why?

csrc/Mod.java
csrc/View.java
csrc/GView.java
csrc/TView.java
Example 2.2: good or bad?

```java
/**
 * Sets the width of this rectangle to the given positive number.
 * 
 * @param width The new width
 * 
 * @throws IllegalArgumentException when the width is a negative number
 */
public void setWidth(int width) {
    ...
}
```
Snippet 2.3: this is bad! huh?

```java
public BitSet(int size, boolean initialValue) {
    this.bitSet = new boolean[size];

    for (int i = 0; i < this.bitSet.length; i++) {
        this.bitSet[i] = initialValue;
    }
}
```
Snippet 2.4: short but also bad! why?

```java
public void addStudent(Student student, String course) {
    if (course.equals("CS520")) {
        cs520Students.add(student);
    } else {
        allStudents.add(student);
    }
}
```
Code reviewing
A Closer Look at 12 Powerful Code Review Tools

In this section, we review the most popular static code review tools.

- Review Board
- Crucible
- GitHub
- Phabricator
- Collaborator
- CodeScene
- Visual Expert
- Gerrit
- Rhodecode
- Veracode
- Reviewable
- Peer Review for Trac

https://kinsta.com/blog/code-review-tools/
Final project description

- Each team of 4 will carry out **one** of the following projects:
  - MSR Mining Challenge
  - Replication Study (e.g., Automated Program Repair)
  - ML Development Toolkits (e.g., Weights & Biases)
  - EleNa: Elevation-based Navigation
- The key phases of the project are: topic selection, mid-point presentation, final presentation (and submission)
- More details available here: https://people.cs.umass.edu/~hconboy/class/2021Fall/CS520/finalProject.pdf