CS 520
Theory and Practice of Software Engineering
Fall 2021

Course introduction

September 2, 2021
The CS 520 team

Instructor

- Heather Conboy
- Lectures: Tu/Th 10-11:15 AM will be recorded
- Office hours: TBD and by appointment
- hconboy@cs.umass.edu

Graders

- Sirisha Annamraju
- Supriya Shreekant Jahagirdar
- Mrinal Tak
Today

- What is Software Engineering?
- Why is Software Engineering important?
- Your expectations
- Course overview
- Our expectations
- Logistics
What is Software Engineering?
What is Software Engineering?

- Developing in an integrated development environment?
- Coding and debugging?
- Deploying and running a software system?
- Empirical evaluations?
- Modeling and designing?
What is Software Engineering?

- Developing in an integrated development environment?
- Coding and debugging?
- Deploying and running a software system?
- Empirical evaluations?
- Modeling and designing?

All of the above -- much more than just writing code!
What is Software Engineering?

**More than just writing code**
The complete process of specifying, designing, developing, analyzing, deploying, and maintaining a software system.
What is Software Engineering?

More than just writing code
The complete process of specifying, designing, developing, analyzing, deploying, and maintaining a software system.

- Common Software Engineering tasks include:
  - Requirements engineering
  - Specification writing and documentation
  - Software architecture and design
  - Programming
  - Verification (e.g., testing, model checking, theorem proving)
  - Software debugging and repair
What is Software Engineering?

More than just writing code
The complete process of specifying, designing, developing, analyzing, deploying, and maintaining a software system.

- Common Software Engineering tasks include:
  - Requirements engineering
  - Specification writing and documentation
  - Software architecture and design
  - Programming
  - Verification (e.g., testing, model checking, theorem proving)
  - Software debugging and repair
What is Software Engineering?

More than just writing code
The complete process of specifying, designing, developing, analyzing, deploying, and maintaining a software system.

- Common Software Engineering tasks include:
  - Requirements engineering
  - Specification writing and documentation
  - Software architecture and design
  - Programming
  - Verification (e.g., testing, model checking, theorem proving)
  - Software debugging and repair

Why is Software Engineering important?
Why is Software Engineering important?

Software is everywhere...
Why is Software Engineering important?

Software is everywhere...and buggy!
Why is Software Engineering important?

Software is everywhere...and buggy!

Unfortunately, WhatsApp has stopped.
Why is Software Engineering important?

Software is complex!

- Aircraft: ~15 million lines of code
How complex is software?

- Measures of complexity:
  - lines of code
  - number of classes
  - number of modules
  - module interconnections and dependencies
  - time to understand
  - # of authors
  - … many more
How complex is software?

- Measures of complexity:
  - lines of code
  - number of classes
  - number of modules
  - module interconnections and dependencies
  - time to understand
  - # of authors
  - … many more

Windows Server 2003: 50 MSLoC
Debian 5.0: 324 MSLoC
How big is 324 MSLoC?

- 50 lines/page ⇒ 6.5M pages
- 1K pages/ream ⇒ 6.5K reams
- 2 inches/ream ⇒ 13K inches
- 13K inches ≈ four times the height of the CS building

- 5 words/LoC @ 50 wpm ⇒ 32M min ≈ 61 years

And we don’t just want random words, we want compiling code!
Why is Software Engineering important?

Infrastructure is software, too!

Example: Design space exploration

<table>
<thead>
<tr>
<th></th>
<th>0.34</th>
<th>0.81</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.52</td>
<td>0.32</td>
</tr>
<tr>
<td>3</td>
<td>0.21</td>
<td>0.53</td>
</tr>
<tr>
<td>4</td>
<td>0.81</td>
<td>0.22</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Why is Software Engineering important?

Infrastructure is software, too!

Example: Design space exploration

- 150 configurations, 1000+ benchmarks
- 1-85 hours per execution
- 200,000+ CPU hours (~23 CPU years)
Summary: 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.
Your background and expectations

Introduction and a brief survey

- What is your background?
- What do you expect from this course?
- What are your learning goals (theory and practice)?
Course overview: the big picture

- **Software architecture and design**
  - Software modeling and UML crash course.
  - Best practices and OO design principles.
  - Architecture and Design patterns.

Goal: no more spaghetti code!
Course overview: the big picture

- **Software architecture and design**
  - Software modeling and UML crash course.
  - Best practices and OO design principles.
  - Architecture and Design patterns.

- **Empirical Software Engineering**
  - Reasoning about experimental designs and studies.
  - Understanding and reasoning about threats to validity.

Anything wrong with the following conclusions?
- **Not using Internet Explorer** makes the world a safer place/reduces murder rates.
- **Spending more time on learning** a programming language makes you a **worse programmer**.

Goal: properly reason about research studies and findings
Course overview: the big picture

● **Software architecture and design**
  ○ Software modeling and UML crash course.
  ○ Best practices and OO design principles.
  ○ Architecture and Design patterns.

● **Empirical Software Engineering**
  ○ Reasoning about experimental designs and studies.
  ○ Understanding and reasoning about threats to validity.

● **Software verification, debugging, and repair**
  ○ Learning about cutting-edge research.
  ○ Hands-on experience, using verification and debugging techniques.

● **Class project**
  ○ Development and evaluation of a research prototype, etc.
Course overview: rough timeline

September

- Software architecture and design
- Best programming practices
- Class project: Topic selection

October/November

- Verification of programs (e.g., testing, model checking)
- Program debugging and repair
- Class project: Mid-point report

November/December

- Empirical Software Engineering
- Reasoning about programs
- Class project: Completion
Our expectations

- Programming experience
- Familiarity with an OO programming language (e.g., Java, C++, etc.)
- Learning to apply new SE tools
- Reading technical papers and online documentation
- Active participation in discussions and group work
Gain experience applying SE tools and techniques

- Architecture and design patterns
- Specifications as UML diagrams (e.g., class diagrams)
- Program in an OO programming language (e.g., Java, javac, java)
- Document source code (e.g., javadoc)
- xUnit testing framework (e.g., JUnit)
- Debugging techniques
- Version Control system (e.g., git)
Exposure to cutting-edge research

● We will have 1 or more guest lectures on research
  ○ These will be held in class
  ○ Alternatively, these will be held out of class. Videos will be available.

● We might have 1 guest lecture on what it’s like to work in industry
Course overview: grading

Grading

- 35% Homework [Individual]
- 30% In-class exercises [Group]
- 25% Course project [Group]
- 10% Participation [Individual]
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/](https://people.cs.umass.edu/~hconboy/class/2021Fall/CS520/)

- Submission of assignments via Moodle: