CS 520
Theory and Practice of Software Engineering
Spring 2022

Course introduction

January 25, 2022
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

- Kimberley Faria
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

Just one out of many important tasks!
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!
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 $\Rightarrow$ 6.5M pages
- 1K pages/ream $\Rightarrow$ 6.5K reams
- 2 inches/ream $\Rightarrow$ 13K inches
- 13K inches $\approx$ four times the height of the CS building

- 5 words/LoC @ 50 wpm $\Rightarrow$ 32M min $\approx$ 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

Infrastructure
Parallel executions of all possible configurations

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.34</td>
<td>0.81</td>
</tr>
<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>
</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 requirements, architecture, and design**
  ○ Requirements engineering.
  ○ 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 requirements, architecture, and design**
  ○ Requirements engineering.
  ○ Software modeling and UML crash course.
  ○ Best practices and OO design principles.
  ○ Architecture and design patterns.

● **Software verification & validation (including manual reviews, testing, model checking, theorem proving) as well as debugging**
  ○ Learning about cutting-edge research.
  ○ Hands-on experience, using V&V and debugging techniques.

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

January/February
- Software requirements, architecture, and design
- Best programming practices
- Final project: Topic selection

March
- Software development processes
- Testing
- Debugging
- Final project: Mid-point report

April/May
- Verification of programs (data flow analysis, model checking, theorem proving)
- Reasoning about programs
- Final 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.
Assignments

- 3 homeworks [Individual]
- 4 in-class exercises [Group]
- Final project [Group]
- Participation questionnaires [Individual]
Course overview: grading

- 35% Homeworks [Individual]
- 30% In-class exercises [Group]
- 25% Final 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, final project presentations

- Course schedule and policies on web site: 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

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

- Submission of assignments via Gradescope // Will provide the URL later