CS520/620

Instructor
● Prof. René Just
● Office: CS358
● Office hours: Wednesdays 11am -- 1pm or by appointment
● rjust@cs.umass.edu

Teaching assistants
● Kristina Fedorenko
● Office hours: TBA
● kfedoren@cs.umass.edu
● Tommy Nguyen
● tdnguyen@umass.edu
Today

- My background
- What is Software Engineering?
- Your expectations
- Course overview
- My expectations
- Logistics
My background
My background
My background
My background
My background

My research areas
- Software testing and debugging
- Static program analysis
- Software security
- Mining software repositories
- Empirical software engineering
What is Software Engineering?
What is Software Engineering?
What is Software Engineering?
What is Software Engineering?
What is Software Engineering?

All of the above and much more!
It’s more than just writing code.
What is Software Engineering?

More than just programming

● The complete process of specifying, designing, developing, analyzing, and maintaining a software system.

● Common Software Engineering tasks:
  ○ Requirements engineering
  ○ Specification writing
  ○ Software architecture and design
  ○ Programming
  ○ Software testing and debugging
What is Software Engineering?

More than just programming

- The complete process of specifying, designing, developing, analyzing, and maintaining a software system.
- Common Software Engineering tasks:
  - Requirements engineering
  - Specification writing
  - Software architecture and design
  - Programming
  - Software testing and debugging

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?

Experimental infrastructure is software, too!

Example: Design/configuration space exploration

- 150 configurations
- 85 hours per execution
- 25,000+ CPU hours (~3 CPU years)
- $10k in elastic computing credits
Why is Software Engineering important?

Software development: Ad-hoc or systematic?

Pros: Ad-hoc

- No formal process.  
  “Brain to keyboard”
- Easy, quick, and flexible.
Why is Software Engineering important?

Software development: Ad-hoc or systematic?

Pros: Ad-hoc
● No formal process.  
  “Brain to keyboard”
● Easy, quick, and flexible.

Cons: Ad-hoc
● Might lack important tasks such as design or testing.
● Doesn’t scale to multiple developers.
● How to measure effort and progress?
Summary: Software Engineering

What is Software Engineering?
● The complete process of specifying, designing, developing, analyzing, and maintaining a software system.

Why is it important?
● Decomposes a complex engineering problem.
● Organizes processes and effort.
● Improves software reliability.
● Improves developer productivity.
Your expectations

Introduction and a brief (5 minute) survey

- **Position:** What are you looking for (industry vs. academia)?
- **Top-2 tasks:** What do you think your tasks related to SE will be?
- **Top-2 expectations:** What do you expect from this course?
Course overview: the big picture

● **Software architecture and design**
  ○ Software modelling 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 testing and debugging**
  ○ Learning about cutting-edge research.
  ○ Hands-on experience, using testing and debugging techniques.

● **Class project**
  ○ Empirical study, development of a research prototype, etc.
Course overview: the big picture

- **Software architecture and design** 2 assignments
  - Software modelling and UML crash course.
  - Best practices and OO design principles.
  - Architecture and Design patterns.

- **Empirical Software Engineering** 1 paper review
  - Reasoning about experimental designs and studies.
  - Understanding and reasoning about threats to validity.

- **Software testing and debugging** 1 paper review & 4 lab sessions
  - Learning about cutting-edge research.
  - Hands-on experience, using testing and debugging techniques.

- **Class project** 2-month project
  - Empirical study, development of a research prototype, etc.
Course overview: timeline

September
● Software architecture and design

October
● Empirical Software Engineering
● Software testing

November
● Midterm exam: 11/08
  ○ Recap and discussion of example questions: 11/01
  ○ Q&A session: 11/03

December
● Software debugging
● Project presentations
Course overview: grading

520 & 620
- 30% Homeworks: 2 HWs + 4 lab sessions (in-class exercises)
- 30% Midterm exam
- 5% Participation

520
- 20% Class project
- 15% Paper reviews (2 papers)

620
- 15% Class project
- 10% Paper reviews (2 papers)
- 10% Paper presentation (1 paper)
Course overview: grading

520 & 620
- 30% Homeworks: 2 HWs + 4 lab sessions (in-class exercises)
- 30% Midterm exam
- 5% Participation

520
- 20% Class project
- 15% Paper reviews (2 papers)

620
- 15% Class project
- 10% Paper reviews (2 papers)
- 10% Paper presentation (1 paper)

Questions?
Expectations

- Programming experience.
- Familiarity with one OO programming language (Java, C++, ...).
- Reading and reviewing 2 research papers.
- 620 students: presenting 1 research paper.
Logistics

- Hasbrouck Lab 134, Tu/Th, 4:00pm – 5:15pm.
- Lectures, lab session, and presentations.
- Course material, policies, and schedule on web site: http://people.cs.umass.edu/~rjust/courses/2016Fall/CS520.620
- Submission of assignments via Moodle: https://moodle.umass.edu
- Discussions on Piazza: https://piazza.com