The CSE 503 team

Instructor
- René Just
- Office: CSE2 338
- Office hours: After class and by appointment
- rjust@cs.washington.edu

Teaching assistant
- Martin Kellogg
- Office: CSE2 253
- Office hours: Wed, 2pm
- kelloggm@cs.washington.edu

Today
- Logistics
- Brief introduction
- Your background
- Course overview
- Why program analysis?

Logistics
- CSE 305, Wed/Fri, 11:30am – 12:50pm.
- Lectures, discussions, and presentations.
- Course material, schedule, etc. on website: https://homes.cs.washington.edu/~rjust/courses/2020Winter/CSE503
- Submission of assignments via Canvas: https://canvas.uw.edu
My background

My research areas
- Software testing and verification
- Software debugging
- Software security
- Empirical software engineering
- Data science / Applied ML
Your background

Introduction and a very brief survey
- What is your research area (or area of interest)?
- How long have you been in the program?
- What is your SE background (programming languages, etc.)?

Today
- Logistics
- Brief introduction
- Your background
  - Course overview
  - Why program analysis?

What is Software Engineering?

- Developing in an IDE and software ecosystem?
What is Software Engineering?

- Developing in an IDE and software ecosystem?
- Testing and debugging?
- Deploying and running a software system?
- Empirical evaluations?
- Modeling and designing?
What is Software Engineering?

● Developing in an IDE and software ecosystem?
● Testing and debugging?
● Deploying and running a software system?
● Empirical evaluations?
● Modeling and designing?

All of the above -- much more than just writing code!

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
○ Software testing and debugging
○ Refactoring

Just one out of many important tasks!

The Role of Software Engineering in Practice

(Development workflow at Microsoft, Big Code summit 2019)

The Role of Software Engineering in Research

Experimental infrastructure is software, too!

Example (automated debugging)

- 150 configurations, 1000+ benchmarks
- 1-85 hours per execution
- 200,000+ CPU hours (~23 CPU years)
Course overview: the big picture

- **Week 1**: Introduction & static vs. dynamic analysis
- **Week 2**: Abstract Interpretation
- **Week 3**: Abstract Interpretation
- **Week 4**: Testing
- **Week 5**: Delta Debugging
- **Week 6**: Slicing
- **Week 7**: Program Repair
- **Week 8**: Empirical Software Engineering
- **Week 9**: ML for Software Engineering
- **Week 10**: Wrap up

HW 1

Course overview: this week

- **Week 1**: Introduction & static vs. dynamic analysis
  - HW 1
- **Two high-level papers**
  - Static and dynamic analysis: synergy and duality
  - Lessons from Building Static Analysis Tools at Google
- **HW 1**
  - Brainstorming about software development difficulties

Course overview: the project

**Logistics**

- 2-4 team members
- Synergies with your work are welcomed! (Project ideas provided after HW 1)

**Timeline**

- **Week 2/3**: Project proposal and revision
- **Week 5**: Related work and methodology
- **Week 8**: Coding completed and initial results
- **Week 10**: Final report and presentation

Types of projects (non-exhaustive)

- proposing and evaluating a fundamental new technique
- developing and assessing new algorithms to replace currently-used ones
- translating a methodology to a new problem domain
- applying known techniques to new problem domains
- evaluation of existing techniques or tools (case studies or controlled experiment)
- implementation of a proposed but never implemented technique

Course overview: the project

**Logistics**

- 2-4 team members
- Synergies with your work are welcomed! (Project ideas provided after HW 1)

**Timeline**

- **Week 2/3**: Project proposal and revision
- **Week 5**: Related work and methodology
- **Week 8**: Coding completed and initial results
- **Week 10**: Final report and presentation

Types of projects (non-exhaustive)

- proposing and evaluating a fundamental new technique
- developing and assessing new algorithms to replace currently-used ones
- translating a methodology to a new problem domain
- applying known techniques to new problem domains
- evaluation of existing techniques or tools (case studies or controlled experiment)
- implementation of a proposed but never implemented technique
**Course overview: the big picture**

- **Week 1:** Introduction & static vs. dynamic analysis  
  HW 1
- **Week 2:** Abstract Interpretation
- **Week 3:** Abstract Interpretation  
  HW 2
- **Week 4:** Testing
- **Week 5:** Delta Debugging
  In-class exercise
- **Week 6:** Slicing
- **Week 7:** Program Repair
- **Week 8:** Empirical Software Engineering
- **Week 9:** ML for Software Engineering
- **Week 10:** Wrap up  
  Project presentation

**And there is more...**

**Special topics:**
- **504: AI meets Software engineering**  
  (ML and statistical methods for SE/program analysis)
- **599: Statistics (and R) done wrong**  
  (Research methods for SW systems research)

**Course overview: grading**

- **60%** Class project
- **30%** HWs and in-class exercise
- **10%** Reading questions and participation

**Course overview: expectations**

- Conducting a quarter-long group project
- Some programming experience
- Reading and discussing research papers
- Have fun!
Today
- Logistics
- Brief introduction
- Your background
- Course overview
- Why program analysis?

Why program analysis?
- ~15 million lines of code
  - Let’s say 50 lines per page (0.05 mm)
    - 300,000 pages
    - 15 m (49 ft)
Does my program implement its specification?

double foo(double[] d {
  int n = d.length;
  double s = 0;
  int i = 0;
  while (i<n)
  s = s + d[i];
  i = i + 1;
  double a = s / n;
  return a;
})

What does this program (binary) do?

Example analyses
- Unit testing
- Solver-aided reasoning
- Fuzzing
- Statistical inference of invariants and models
Program analysis: examples

Autocompletion: which methods to suggest?

Example analyses
- Context-sensitive type checking
- Heuristics and frequency analysis

Semantics: how to name this method?

Example analyses
- Statistical language models (bag of words, n-grams, etc.)
- Heuristics and frequency analysis