CSE 503
Software Engineering

Course introduction
Today

- Logistics
- Brief introduction
- Course overview
- Why program analysis?
Logistics

- Tue/Thu, 11:30am – 12:50pm.
- Course material, schedule, etc. on course website: https://courses.cs.washington.edu/courses/cse503/22sp/ All slides are posted before class.
- Assignment submission and discussions via Canvas: https://canvas.uw.edu/courses/1545328 Linked from webpage.
The CSE 503 team

Instructor
- Michael Ernst
- Office hours: After class and by appointment
- mernst@cs.washington.edu

Teaching assistant
- Martin Kellogg
- Office hours: TBD
- kelloggm@cs.washington.edu
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
- Course overview
- Why program analysis?
What is Software Engineering?
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?
What is Software Engineering?

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

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

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
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
  - 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**: Invariants
- **Week 7**: Program Repair
- **Week 8**: Empirical Software Engineering
- **Week 9**: ML for Software Engineering
- **Week 10**: Wrap up
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**: Invariants
- **Week 7**: Program Repair
- **Week 8**: Empirical Software Engineering
- **Week 9**: ML for Software Engineering
- **Week 10**: Wrap up  
  Project presentation

Questions?
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
  - **Please** start right away!
Course overview: the project

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

Timeline
- **Week 3/4**: Project proposal and revision
- **Week 6**: Related work and methodology
- **Week 8**: Coding completed and initial results
- **Week 10**: Presentation and final report
Course overview: the project

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

Timeline
- **Week 3/4**: Project proposal and revision
- **Week 6**: Related work and methodology
- **Week 8**: Coding completed and initial results
- **Week 10**: Presentation and final report

Types of projects (non-exhaustive)
- proposing and evaluating a 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

Questions?
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**: Invariants
- **Week 7**: Program Repair
- **Week 8**: Empirical Software Engineering
- **Week 9**: ML for Software Engineering
- **Week 10**: Wrap up

HW 1

HW 2

In-class exercise

Project presentation
And there is more...

Special topics:

- **504: AI meets Software engineering**
  (ML and statistical methods for SE/program analysis)

- **599: Research methods**
  (Research design and statistics in R)
### Course Overview: The Big Picture

<table>
<thead>
<tr>
<th>Week 1: Introduction &amp; static vs. dynamic analysis</th>
<th>HW 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Week 2:</strong> Abstract Interpretation</td>
<td></td>
</tr>
<tr>
<td><strong>Week 3:</strong> Abstract Interpretation</td>
<td>HW 2</td>
</tr>
<tr>
<td><strong>Week 4:</strong> Testing</td>
<td></td>
</tr>
<tr>
<td><strong>Week 5:</strong> Delta Debugging</td>
<td>In-class exercise</td>
</tr>
<tr>
<td><strong>Week 6:</strong> Invariants</td>
<td></td>
</tr>
<tr>
<td><strong>Week 7:</strong> Program Repair</td>
<td></td>
</tr>
<tr>
<td><strong>Week 8:</strong> Empirical Software Engineering</td>
<td></td>
</tr>
<tr>
<td><strong>Week 9:</strong> ML for Software Engineering</td>
<td></td>
</tr>
<tr>
<td><strong>Week 10:</strong> Wrap up</td>
<td>Project presentation</td>
</tr>
</tbody>
</table>
Course overview: grading

- 50% Class project
- 35% HWs, in-class exercise, reading questions
- 15% Participation

Questions?
Course overview: expectations

- Conducting a quarter-long research project
- Some programming experience
- Reading and actively discussing research papers
- Have fun!
Today

- Logistics
- Brief introduction
- Your background
- Course overview

- Why program analysis?
Who cares about program analysis?
Who cares about program analysis?

- ~15 million lines of code

Let’s say 50 lines per page (0.05 mm)
Who cares about program analysis?

- ~15 million lines of code

Let’s say 50 lines per page
  - 300000 pages
  - 15 m (49 ft)
Who cares about program analysis?
Who cares about program analysis?

Unfortunately, WhatsApp has stopped.

OK
Does my program implement its specification?

```java
double foo(double[] d) {
    int n = d.length;
    double s = 0;
    int i = 0;
    while (i < n) {
        s += d[i];
        i++;
    }
    double a = s / n;
    return a;
}
```
Program analysis: examples

Does my program implement its specification?

```java
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;
}
```

Example analyses

- Unit testing
- Solver-aided reasoning

\( (\forall x \text{ fsa}(x)) \Rightarrow (\exists y \text{ pda}(y) \land \text{equivalent}(x, y)) \)
Program analysis: examples

What does this program (binary) do?
Program analysis: examples

What does this program (binary) do?

Example analyses

- Fuzzing
- Statistical inference of invariants and models
Program analysis: examples

Autocompletion: which methods to suggest?
Program analysis: examples

Autocompletion: which methods to suggest?

Example analyses

- Context-sensitive type checking

- Heuristics and frequency analysis
Program analysis: examples

Semantics: how to name this method?

```java
void f(int[] array) {
    boolean swapped = true;
    for (int i = 0; i < array.length && swapped; i++) {
        swapped = false;
        for (int j = 0; j < array.length - 1 - i; j++) {
            if (array[j] > array[j+1]) {
                int temp = array[j];
                array[j] = array[j+1];
                array[j+1] = temp;
            }
        }
    }
}
```
Program analysis: examples

Semantics: how to name this method?

```java
void f(int[] array) {
    boolean swapped = true;
    for (int i = 0; i < array.length && swapped; i++) {
        swapped = false;
        for (int j = 0; j < array.length - 1 - i; j++) {
            if (array[j] > array[j+1]) {
                int temp = array[j];
                array[j] = array[j+1];
                array[j+1] = temp;
                swapped = true;
            }
        }
    }
}
```

Example analyses

- Statistical language models
  (bag of words, n-grams, etc.)

- Heuristics and frequency analysis
Next time: static vs. dynamic analysis

A **static analysis** analyzes program source code without running the program
- What are examples?

A **dynamic analysis** observes program executions
- What are examples?

Which one should a programmer prefer, and why?