CMPT 479/886/745
Software Engineering: Theory and Practice

Introduction

Nick Sumner
wsumner@sfu.ca
Introduction

Nick Sumner
wsumner@sfu.ca
CMPT 479/886/745
Software Engineering: Theory and Practice

Introduction

Nick Sumner
wsumner@sfu.ca
Introduction

Who am I?
- Nick Sumner (wsumner@sfu.ca)
- Research Faculty (Software Engineering, Compilers, Program Analysis)
Introduction

- Who am I?
  - Nick Sumner (wsumner@sfu.ca)
  - Research Faculty

- Who is your TA?
  - Golnaz Gharachorlu
Introduction

• Who am I?
  – Nick Sumner (wsumner@sfu.ca)
  – Research Faculty

• Who is your TA?
  – Golnaz Gharachorlu

• What is the course website?
Introduction

- **Who am I?**
  - Nick Sumner (wsumner@sfu.ca)
  - Research Faculty

- **Who is your TA?**
  - Golnaz Gharachorlu

- **What is the course website?**

- **Where can you discuss course issues?**
  - CourSys
    - [https://coursys.sfu.ca/2019fa-cmpt-479-x1/discussion/](https://coursys.sfu.ca/2019fa-cmpt-479-x1/discussion/)
What is this class?

- **Software engineering** (informally)
  - Systematic approaches for managing risk while producing or providing software.
What is this class?

- **Software engineering** (informally)
  - Systematic approaches for managing risk while producing or providing software.
  - How can we write code that is adaptable to changing requirements?
What is this class?

- **Software engineering** (informally)
  - Systematic approaches for managing risk while producing or providing software.
  - How can we write code that is adaptable to changing requirements?
  - How can you know that code is correct or discover its incorrectness?
What is this class?

- **Software engineering** (informally)
  - Systematic approaches for managing risk while producing or providing software.
  
  - How can we write code that is adaptable to changing requirements?
  - How can you know that code is correct or discover its incorrectness?
  - Can you defend against attackers?
What is this class?

- **Software engineering** (informally)
  - Systematic approaches for managing risk while producing or providing software.
  - How can we write code that is adaptable to changing requirements?
  - How can you know that code is correct or discover its incorrectness?
  - Can you defend against attackers?
  - Can you discover what attackers have done?
What is this class?

- **Software engineering** (informally)
  - Systematic approaches for managing risk while producing or providing software.
  
  - How can we write code that is adaptable to changing requirements?
  - How can you know that code is correct or discover its incorrectness?
  - Can you defend against attackers?
  - Can you discover what attackers have done?
  - Can you automatically generate software?
What is this class?

- **Software engineering** (informally)
  - Systematic approaches for managing risk while producing or providing software.
  
  - How can we write code that is adaptable to changing requirements?
  - How can you know that code is correct or discover its incorrectness?
  - Can you defend against attackers?
  - Can you discover what attackers have done?
  - Can you automatically generate software?

Programs are themselves data that you can construct, analyze, transform, synthesize, ...
What is this class?

- **Software engineering** (informally)
  - Systematic approaches for managing risk while producing or providing software.
  - How can we write code that is adaptable to changing requirements?
  - How can you know that code is correct or discover its incorrectness?
  - Can you defend against attackers?
  - Can you discover what attackers have done?
  - Can you automatically generate software?
  - Spans techniques from novel logics to rigorous empirical assessment.
  - Rich *interaction* between theory and practice matter.
What is this class?

- **Software engineering** (informally)
  - Systematic approaches for managing risk while producing or providing software.
  - How can we write code that is adaptable to changing requirements?
  - How can we know that code is correct or discover its incorrectness?
  - Can you defend against attackers?
  - Can you discover what attackers have done?
  - Can you automatically generate software?
  - Spans techniques from novel logics to rigorous empirical assessment.
  - Rich *interaction* between theory and practice matter.

I will expect you to
What is this class?

- **Software engineering** (informally)
  - Systematic approaches for managing risk while producing or providing software.
  - How can we write code that is adaptable to changing requirements?
  - How can we prove the correctness of our code?
  - Can you defend against attackers?
  - Can you discover what attackers have done?
  - Can you automatically generate software?

- Spans techniques from novel logics to rigorous empirical assessment.
- Rich *interaction* between theory and practice matters.

I will expect you to reason formally.
What is this class?

- **Software engineering** (informally)
  - Systematic approaches for managing risk while producing or providing software.
  - How can we write code that is adaptable to changing requirements?
  - How can you know that code is correct or discover its incorrectness?
  - Can you defend against attackers?
  - Can you discover what attackers have done?
  - Can you automatically generate software?
  - Spans techniques from novel logics to rigorous empirical assessment.
  - Rich *interaction* between theory and practice matter.

I will expect you to reason formally. reason practically.
What is this class?

- **Software engineering** (informally)
  - Systematic approaches for managing risk while producing or providing software.
  - How can we write code that is adaptable to changing requirements?
  - How can we know that code is correct or discover its incorrectness?
  - Can you defend against attackers?
  - Can you discover what attackers have done?
  - Can you automatically generate software?

- Spans techniques from novel logics to rigorous empirical assessment.
- Rich *interaction* between theory and practice matter.

I will expect you to

reason formally.

reason practically.

*apply formalism to solve practical problems.*
What is this class?

- **Software engineering** (informally)
  - Systematic approaches for managing risk while producing or providing software.
  
  - How can we write code that is adaptable to changing requirements?
  - How can we know that code is correct or discover its incorrectness?
  - Can you defend against attackers?
  - Can you discover what attackers have done?
  - Can you automatically generate software?
  
  - Spans techniques from novel logics to rigorous empirical assessment.
  - Rich *interaction* between theory and practice matter.

I will expect you to

reason formally.

reason practically.

apply formalism to solve practical problems.

recognize that practice may differ from formalism.
What is this class?

Software Engineering
What is this class?

Software Engineering

This Class
What is this class?

Software Engineering

This Class

There is too much breadth.
There is too much depth.
What is this class?

- Important things we will *not* cover (nonexhaustive)
What is this class?

- Important things we will not cover (nonexhaustive)
  - Social aspects of software engineering
  - Project planning and management (Agile vs agile vs ...)
  - Requirements management
  - SLOs, SLA, and most SRE
  - Monoliths vs Services vs Microservices
  - Middleware management
  - ...
What is this class?

- **Important things we will *not* cover (nonexhaustive)**
  - Social aspects of software engineering
  - Project planning and management (Agile vs agile vs ...)
  - Requirements management
  - SLOs, SLA, and most SRE
  - Monoliths vs Services vs Microservices
  - Middleware management
  - ...

These are worthwhile topics. Seek them elsewhere.
What is this class?

• What we will (likely) cover
  – Foundations of software design
  – Performance & bottleneck analysis
  – Testing
  – Formal models of programs
  – Symbolic execution and automated test generation
  – Dynamic analysis
  – Static analysis
  – Parallelism & concurrency
  – Software security
  – Program synthesis
What is this class?

- What we will (likely) cover
  - Foundations of software design
  - Performance & bottleneck analysis
  - Testing
  - Formal models of programs
  - Symbolic execution and automated test generation
  - Dynamic analysis
  - Static analysis
  - Parallelism & concurrency
  - Software security
  - Program synthesis
What is this class?

- What we will (likely) cover
  - Foundations of software design
  - Performance & bottleneck analysis
  - Testing
  - Formal models of programs
  - Symbolic execution and automated test generation
  - Dynamic analysis
  - Static analysis
  - Parallelism & concurrency
  - Software security
  - Program synthesis
What is this class?

What we will (likely) cover
- Foundations of software design
- Performance & bottleneck analysis
- Testing
- Formal models of programs
- **Symbolic execution and automated test generation**
- **Dynamic analysis**
- **Static analysis**
- Parallelism & concurrency
- Software security
- Program synthesis
What is this class?

- What we will (likely) cover
  - Foundations of software design
  - Performance & bottleneck analysis
  - Testing
  - Formal models of programs
  - Symbolic execution and automated test generation
  - Dynamic analysis
  - Static analysis
  - **Parallelism & concurrency**
  - **Software security**
  - **Program synthesis**
What is this class?

- What we will (likely) cover
  - Foundations of software design
  - Performance & bottleneck analysis
  - Testing
  - Formal models of programs
  - Symbolic execution and automated test generation
  - Dynamic analysis
  - Static analysis
  - Parallelism & concurrency
  - Software security
  - Program synthesis
What is this class?

- What we will (likely) cover
  - Foundations of software design
  - Performance & bottleneck analysis
  - Testing
  - Formal models of programs
  - Symbolic execution and automated test generation
  - Dynamic analysis
  - Static analysis
  - Parallelism & concurrency
  - Software security
  - Program synthesis

- https://www.cs.sfu.ca/~wsumner/teaching/745/19/schedule.html
What is this class?

- **What we will (likely) cover**
  - Foundations of software design
  - Performance & bottleneck analysis
  - Testing
  - Formal models of programs
  - Symbolic execution and automated test generation
  - Dynamic analysis
  - Static analysis
  - Parallelism & concurrency
  - Software security
  - Program synthesis

There is still far too much! We will focus on *breadth* over *depth*.

- [https://www.cs.sfu.ca/~wsumner/teaching/745/19/schedule.html](https://www.cs.sfu.ca/~wsumner/teaching/745/19/schedule.html)
What is this class?

- What we will (likely) cover

This Course
What is this class?

- What we will (likely) cover

- CMPT 373
- CMPT 473
- My Seminar

This Course
How will the class be structured?

- **Grading:**
  - Assignments (weekly): 50%
  - Exams: 25%
  - Term Project: 25%
How will the class be structured?

- **Grading:**
  - Assignments (weekly): 50%
  - Exams: 25%
  - Term Project: 25%

- **Assignments**
  - A short programming and/or written assignment each week
  - Demonstrate understanding & application of in class material
  - Will expect you to think critically & independently
How will the class be structured?

- Grading:
  - Assignments (weekly): 50%
  - Exams: 25%
  - Term Project: 25%

- Assignments
  - A short programming and/or written assignment each week
  - Demonstrate understanding & application of in class material
  - Will expect you to think critically & independently

- Exams
  - Just the final
  - Demonstrate competence & application of course material
How will the class be structured?

- Term Projects:
  - An open ended project that demonstrates competence
How will the class be structured?

• **Term Projects:**
  – An open ended project that demonstrates competence
  – Address real world problems in software engineering
How will the class be structured?

- **Term Projects:**
  - An open ended project that demonstrates competence
  - Address real world problems in software engineering
  - Develop a new tool or technique to address the problem
How will the class be structured?

- **Term Projects:**
  - An open ended project that demonstrates competence
  - Address real world problems in software engineering
  - Develop a new tool or technique to address the problem
  - For undergrads, I have preplanned projects if you want them
  - Grads should come up with a proposal of their own
How will the class be structured?

- **Term Projects:**
  - An open ended project that demonstrates competence
  - Address real world problems in software engineering
  - Develop a new tool or technique to address the problem
  - For undergrads, I have preplanned projects if you want them
  - Grads should come up with a proposal of their own
  - Discussing with me in advance is recommended
  - Initial proposals due on Oct 16. Meetings w/me on 16th & 17th.
How will the class be structured?

- **Term Projects:**
  - An open ended project that demonstrates competence
  - Address real world problems in software engineering
  - Develop a new tool or technique to address the problem
  
  - For undergrads, I have preplanned projects if you want them
  - Grad students should come up with a proposal of their own
  - Discussing with me in advance is recommended
  - Initial proposals due on Oct 16. Meetings w/me on 16th & 17th.

I want you to walk away with a project you are proud of.
It may lead to a paper.
It may to a business.
Policies & Expectations

- Late Submissions
  - None accepted in general (3 late days to spend throughout semester)
Policies & Expectations

- **Late Submissions**
  - None accepted in general (3 late days to spend throughout semester)

- **Cheating**
  - Any instance results in a score of 0 for the entire assignment involved.
  - Repeat offenders will be reported and recommended for immediate failure in the course.
Policies & Expectations

- Late Submissions
  - None accepted in general (3 late days to spend throughout semester)

- Cheating
  - Any instance results in a score of 0 for the entire assignment involved.
  - Repeat offenders will be reported and recommended for immediate failure in the course.

It is better to get 0 credit than to cheat!
Policies & Expectations

- Late Submissions
  - None accepted in general (3 late days to spend throughout semester)

- Cheating
  - Any instance results in a score of 0 for the entire assignment involved.
  - Repeat offenders will be reported and recommended for immediate failure in the course.

- Expected Workload
  - Strong should expect to spend 9-10 hours outside of class per week.
  - If you are missing some skills, you should expect to spend more.
  - This is not a required class.
    If you are only here for credit, it is better to leave.
Policies & Expectations

- Late Submissions
  - None accepted in general (3 late days to spend throughout semester)

- Cheating
  - Any instance results in a score of 0 for the entire assignment involved.
  - Repeat offenders will be reported and recommended for immediate failure in the course.

- Expected Workload
  - Strong should expect to spend 9-10 hours outside of class per week.
  - If you are missing some skills, you should expect to spend more.
  - This is not a required class.
    If you are only here for credit, it is better to leave.

- Attendance
  - You don’t have to attend, but all in class materials are your responsibility
Let’s get started