

# Software Engineering

ECE444 (Fall2020)

Shurui Zhou

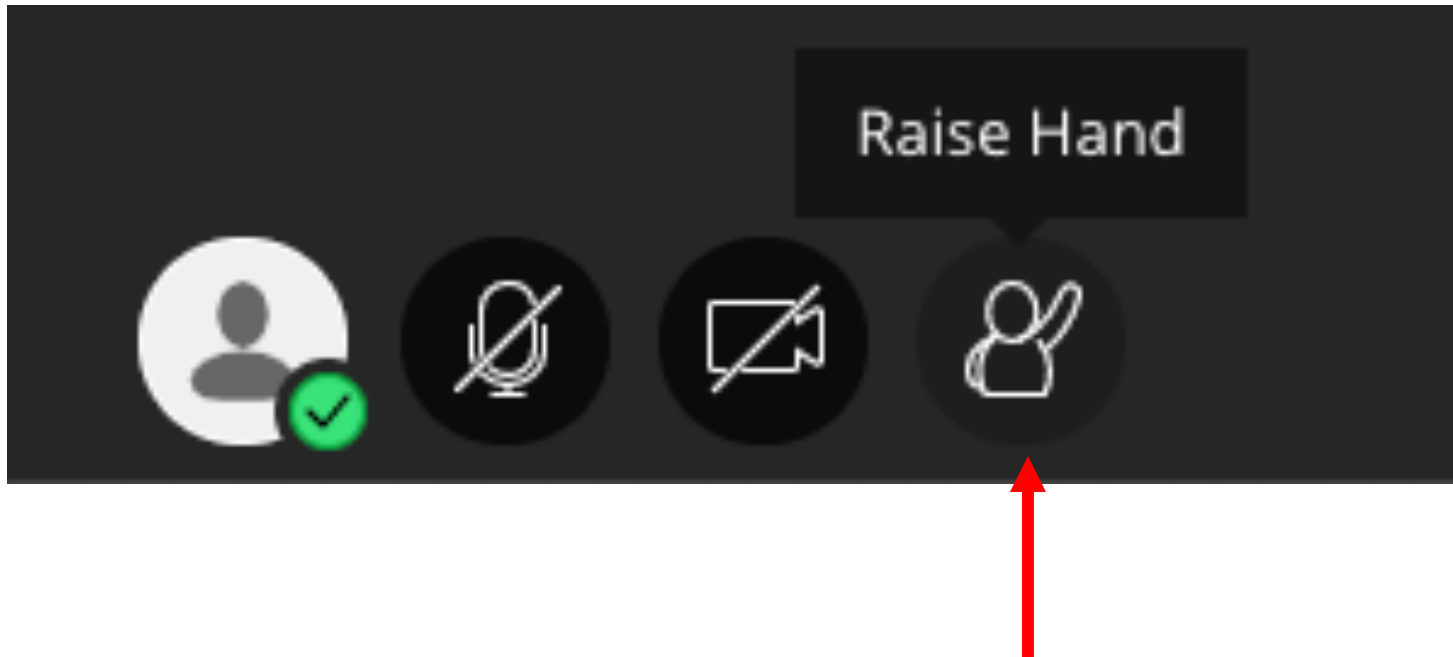
Assistant Professor



The Edward S. Rogers Sr. Department  
of Electrical & Computer Engineering  
**UNIVERSITY OF TORONTO**

# Lecture Logistics during a Pandemic

- If you can hear me in BB Collaborate, please click Raise Hand





# Shurui Zhou

<https://www.eecg.utoronto.ca/~shuruiz/>  
shuruiz@ece.utoronto.ca

**Carnegie  
Mellon  
University**

2014 - 2020 Ph.D.  
School of Computer Science  
Institute for Software Research



2020 Fall – Assistant Professor

## **Research Interests**

- Software Engineering (SE)
- SE for AI
- Collaborative Software Development
- Open Source

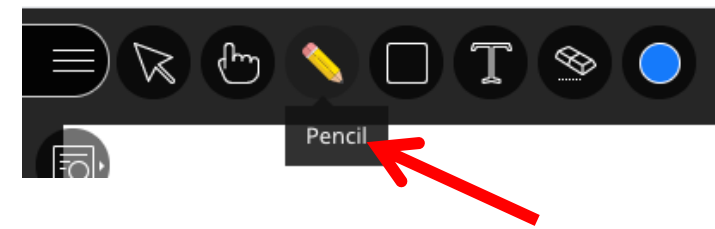


# Teaching Assistants

|                  |                                   |
|------------------|-----------------------------------|
| Akshay Kawlay    | ak.kawlay@mail.utoronto.ca        |
| Yunhao Mao       | yunhao.mao@mail.utoronto.ca       |
| Karthik Mohan    | karthik.mohan@mail.utoronto.ca    |
| Keerthi Nelaturu | keerthi.nelaturu@mail.utoronto.ca |
| Jiten Saini      | jiten.saini@mail.utoronto.ca      |
| Haotian Zhao     | haotian.zhao@mail.utoronto.ca     |

Where are you?

How are you feeling today?



First of all:

You are not alone!  
We are undertaking this  
new experience together.



# This is not normal. We understand.

- Expect:
  - Internet and bandwidth issues
  - Timezone issues
  - Distractions -- parents, siblings, pets
  - Feeling isolated, feeling overwhelmed
  - Many additional sources of stress
  - Hard time dealing with -gestures widely- *everything...*

*Talk to us about accommodations of any kind*

# Simulating in-class Experience

- Discussions and interactions are important. We'll have regular in-class discussions and exercises
- Use chat or "raise hand" feature
- Muted by default, keep camera on if possible
- Attend lecture and recitation live, recordings only as backup
- I may call on you
- Contact me for accommodations!

# Learning Goals

---

- Learn how software is developed in a systematic way
- Learn by doing - 2 main group projects
  - Web application development
  - Open source excursion
- Learn the state-of-the-art research topics in software engineering
  - Reading papers

# Disclaimer

- First time teaching + First time online teaching
- Many experiments for online teaching
- Significant redesign (course structure/homework)
  - ECE444 (Fall2019-UofT)
  - + 17-313 Software Engineering (CMU)
  - + 17-214 Principles of Software Construction (CMU)
  - + 17-652 Requirement Engineering (CMU)



# Software is everywhere

# Agenda for today

- Case studies
- Introduction
- Syllabus
- Logistics

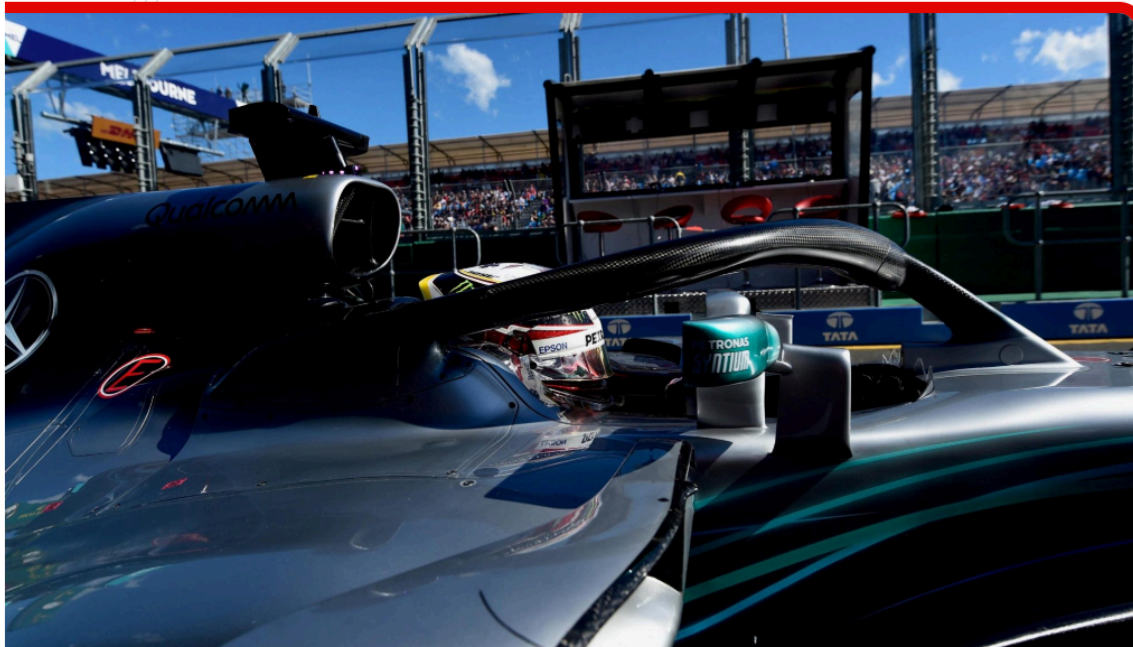
# Software glitch cost Hamilton victory - Mercedes

25 March 2018

MERCEDES

AUSTRALIA

HAMILTON



The software the team has used for five years to simulate such scenarios had generated the incorrect figures, consigning Hamilton to a second-place finish behind Vettel's Ferrari.

"Lewis did nothing wrong - it was down to a software bug or an algorithm that was simply wrong"

Toto Wolff

<https://www.formula1.com/en/latest/article/software-glitch-cost-hamilton-victory-mercedes.6VzyCYpEpaulYsOWYCqYS.html#:~:text=A%20software%20glitch.,season%20opening%20race%20in%20Australia.&text=The%20world%20champion%20immediately%20asked,time%20Mercedes%20had%20given%20him.>

## Toyota Case: Single Bit Flip That Killed

By Junko Yoshida 10.25.2013 0

During the trial, embedded systems experts who reviewed Toyota's electronic throttle source code testified that they found Toyota's source code defective, and that it contains bugs — including bugs that can cause unintended acceleration.

"We did a few things that NASA apparently did not have time to do," Barr said. For one thing, by looking within the real-time operating system, the experts identified "unprotected critical variables." They obtained and reviewed the source code for the "sub-CPU," and they "uncovered gaps and defects in the throttle fail safes."

The experts demonstrated that "the defects we found were linked to unintended acceleration through vehicle testing," Barr said. "We also obtained and reviewed the source code for the black box and found that it can record false information about the driver's actions in the final seconds before a crash."

Stack overflow and software bugs led to memory corruption, he said. And it turns out that the crux of the issue was these memory corruptions, which acted "like ricocheting bullets."

<https://www.eetimes.com/toyota-case-single-bit-flip-that-killed/>

When asked if the whole case for unintended acceleration could be pinned on the task X death, Barr replied, "The task X death in combination with other task deaths." There are

Barr also said more than half the dozens of tasks' deaths studied by the experts in their experiments "were not detected by any fail-safe."



- One pilot said it was “unconscionable that a manufacturer, the FAA, and the airlines would have pilots flying an airplane without adequately training, or even providing available resources and sufficient documentation to understand the highly complex systems that differentiate this aircraft from prior models”

<https://www.theverge.com/2019/5/2/18518176/boeing-737-max-crash-problems-human-error-mcas-faa>





FBI traitor Robert Hanssen

In the early 1990s, Russian mobsters partnered with Italian Mafia families in Newark, N.J., to skim millions of dollars in federal and New Jersey state gasoline and diesel taxes. Special Agent Larry Depew set up an undercover sting operation under the direction of Robert J. Chiaradio, a supervisor at the Federal Bureau of Investigation's Washington, D.C., headquarters.

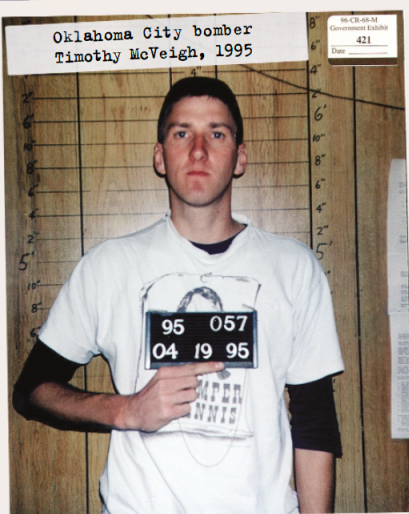
## Who Killed the

By Harry Golaststein

9/11



Oklahoma City bomber  
Timothy McVeigh, 1995



CLOCKWISE FROM TOP: FBI/AP PHOTO; AP PHOTO; CHAD SCHUCHMAN/AP PHOTO



D.C. sniper  
John Allen Muhammad, 2002



Anthrax attack, 2001

## Virtual Case File?

How the FBI blew more than  
\$100 million on case-management  
software it will never use

Depew collected reams of evidence from wiretaps, interviews, and financial transactions over the course of two and a half years. Unfortunately, the FBI couldn't provide him with a database program that would help organize the information, so Depew wrote one himself. He used it to trace relationships between telephone calls, meetings, surveillance, and interviews, but he could not import information from other investigations that might shed light on his own. So it wasn't until Depew mentioned the name of a suspect to a colleague that he obtained a briefcase that his friend had been holding since 1989.

"When I opened it up, it was a treasure trove of information about who's involved in the conspiracy, including the Gambino family, the Genovese family, and the Russian components. It listed percentages of who got what, when people were supposed to pay, the number of gallons. It became a central piece of evidence," Depew recalled during an interview at the FBI's New Jersey Regional Computer Forensic Laboratory, in Hamilton, where he is the director. "Had I not just picked up the phone and called that agent, I never would have gotten it."

A decade later, Depew's need to share information com-

bined with his do-it-yourself database skills and connection to his old supervisor, Chiaradio, would land him a job managing his first IT project—the FBI's Virtual Case File.

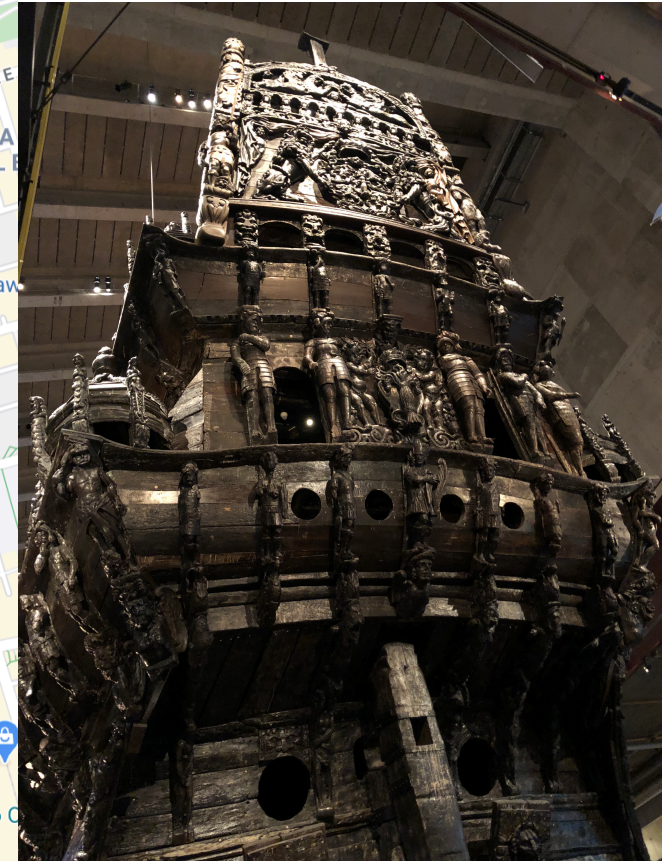
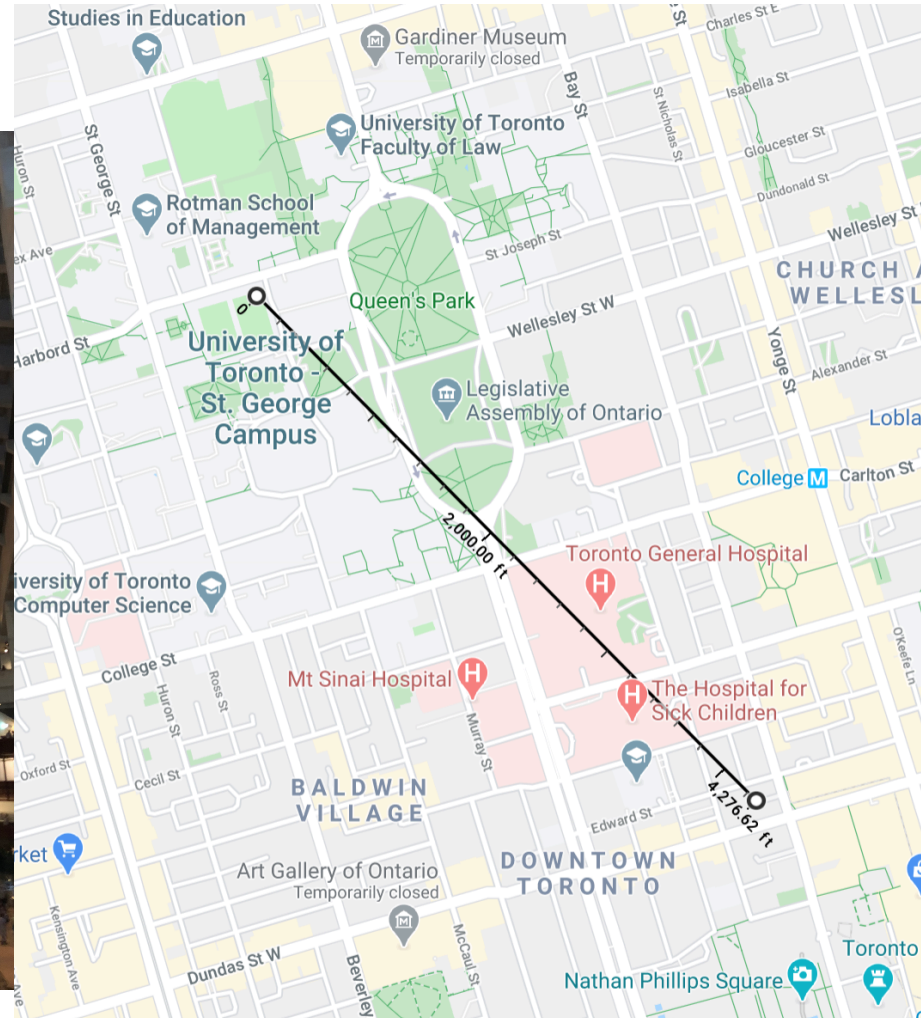
Depew's appointment to the FBI's VCF team was an auspicious start to what would become the most highly publicized software failure in history. The VCF was supposed to automate the FBI's paper-based work environment, allow agents and intelligence analysts to share vital investigative information, and replace the obsolete Automated Case Support (ACS) system. Instead, the FBI claims, the VCF's contractor, Science Applications International Corp. (SAIC), in San Diego, delivered 700,000 lines of code so bug-ridden and functionally off target that this past April, the bureau had to scrap the US \$170 million project, including \$105 million worth of unusable code. However, various government and independent reports show that the FBI—lacking IT management and technical expertise—shares the blame for the project's failure.

In a devastating 81-page audit, released in 2005, Glenn A. Fine, the U.S. Department of Justice's inspector general, described eight factors that contributed to the VCF's

FROM LEFT: STEVE HARRIS/AP PHOTO; KENNETH LAMBERT/AP PHOTO



# Vasa Syndrome (Swedish 17th-century warship)



# What happened? (Vasa Syndrome)

- Changing shipbuilding orders
- No specifications for modified keel
- Shifting armaments requirements

*Requirements*

- Shipwright's death

*Teams*

- No way to calculate stability, stiffness, or sailing characteristics

*Metrics*

- Failed pre-launch stability tests

*QA*

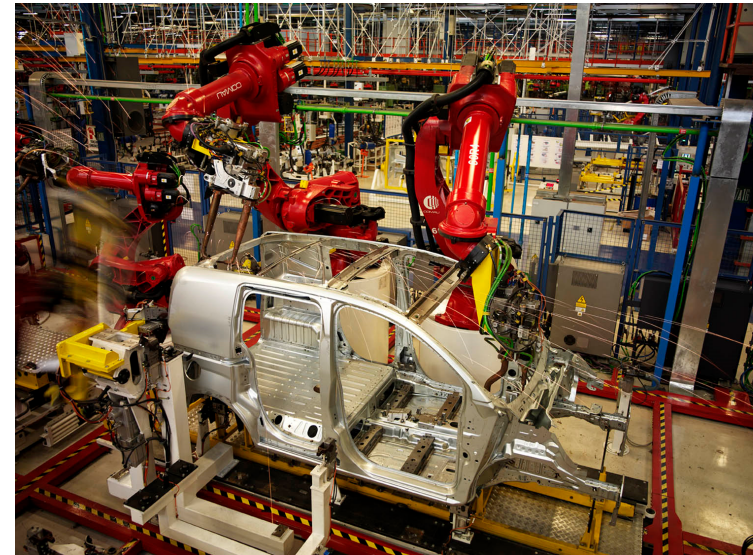


# Software Engineering

What is **engineering**? And how is it different from  
**hacking/programming**?

# Producing a car/bridge

- Estimable costs and risks
- Expected results
- High quality
- Separation between plan and production
- Simulation before construction
- Quality assurance through measurement
- Potential for automation



# Software Engineering?

*„The Establishment and use of sound  
**engineering principles** in order to obtain  
**economically** software that is **reliable**  
and works **efficiently** on **real** machines.”*

*[Bauer 1975, S. 524]*

“**Software engineering** is the branch of computer science that creates practical, cost-effective solutions to computing and information processing problems, preferentially by applying scientific knowledge, developing software systems in the service of mankind.

Software engineering entails making **decisions** under constraints of limited time, knowledge, and resources. [...]

Engineering quality resides in engineering judgment. [...]

Quality of the software product depends on the engineer's faithfulness to the engineered artifact. [...]

Engineering requires reconciling conflicting constraints. [...]

Engineering skills improve as a result of careful systematic reflection on experience. [...]

Costs and time constraints matter, not just capability. [...]

Software Engineering for the 21st Century: A basis for rethinking the curriculum Manifesto, CMU-ISRI-05-108

# 1968 NATO Conference on Software Engineering

- international experts on computer software who agreed on defining best practices for software grounded in the application of engineering.







## ESEC/FSE 2020

Sun 8 - Fri 13 November 2020 Sacramento, California, United States

Attending ▾ Program ▾ Tracks ▾ Organization ▾ Search Series ▾

Sign i

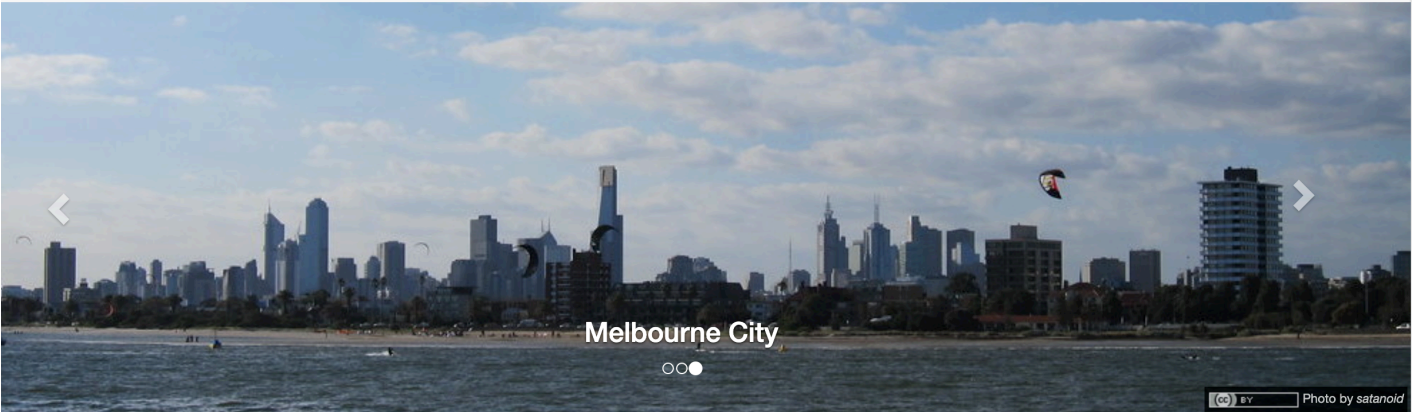


Sacramento Skyline at Night

## ASE 2020

Mon 21 - Fri 25 September 2020 Melbourne, Australia

Attending ▾ Tracks ▾ Organization ▾ Search Series ▾

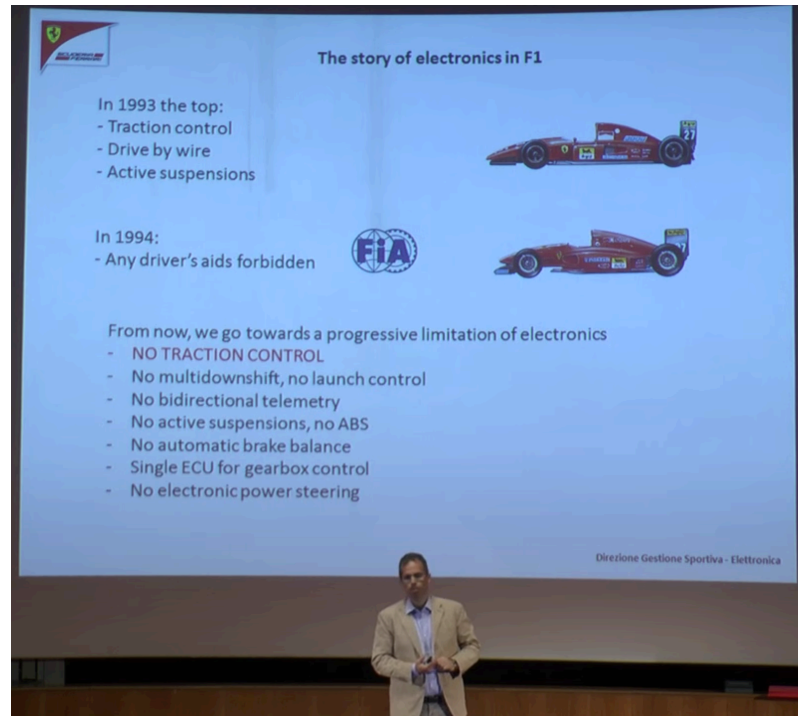


Melbourne City

○○●

Photo by satanoid

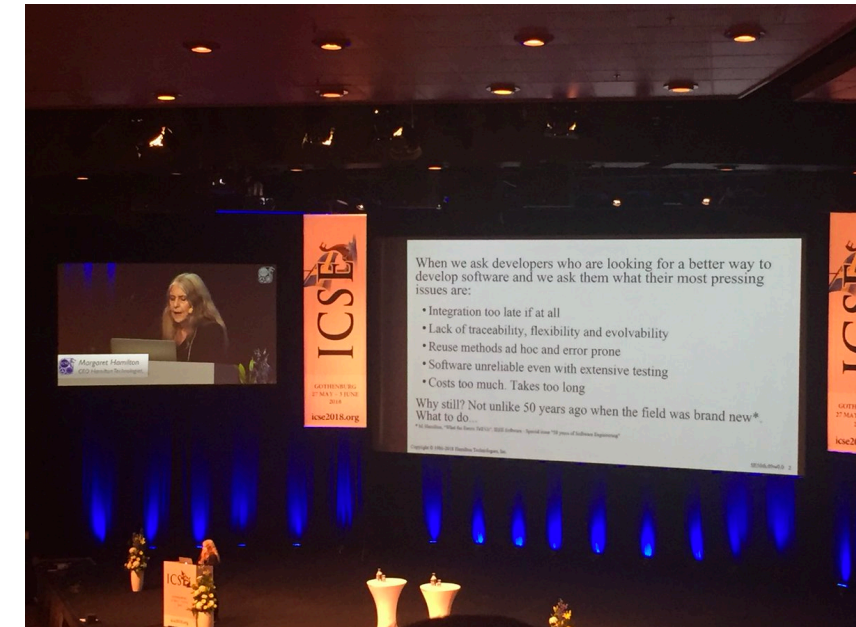
# International Conference in Software Engineering



ICSE 2015 'Software Engineering in Ferrari F1'



ICSE 2018 'The Language as a Software Engineer' (Margaret Hamilton)



# Learning Goals

---

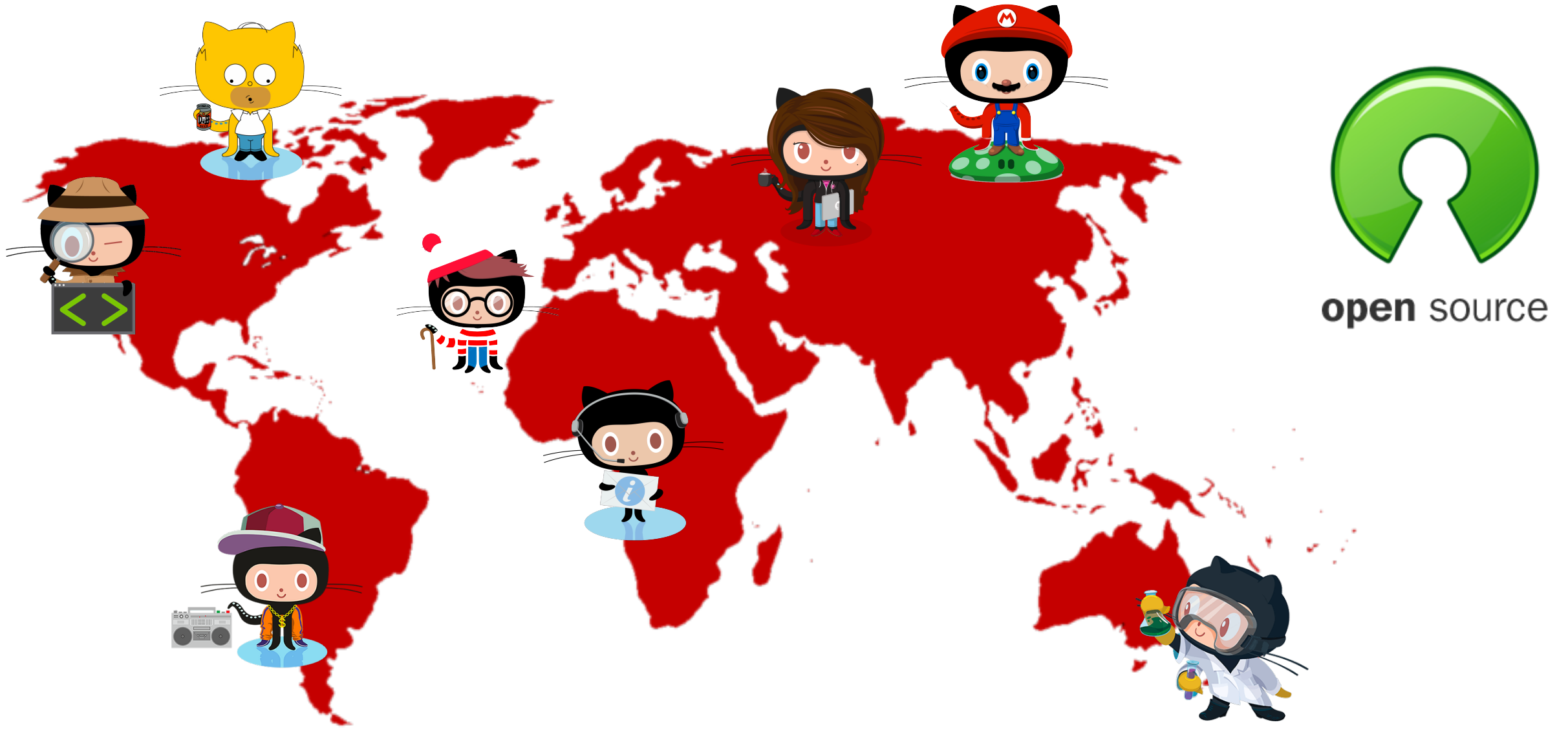
- Learn how software is developed in a systematic way
- Learn by doing - 2 main group projects
  - Web application development
  - Open source excursion
- Learn the state-of-the-art research topics in software engineering
  - Reading papers

# Globally Distributed Software Development





# Globally Distributed Software Development

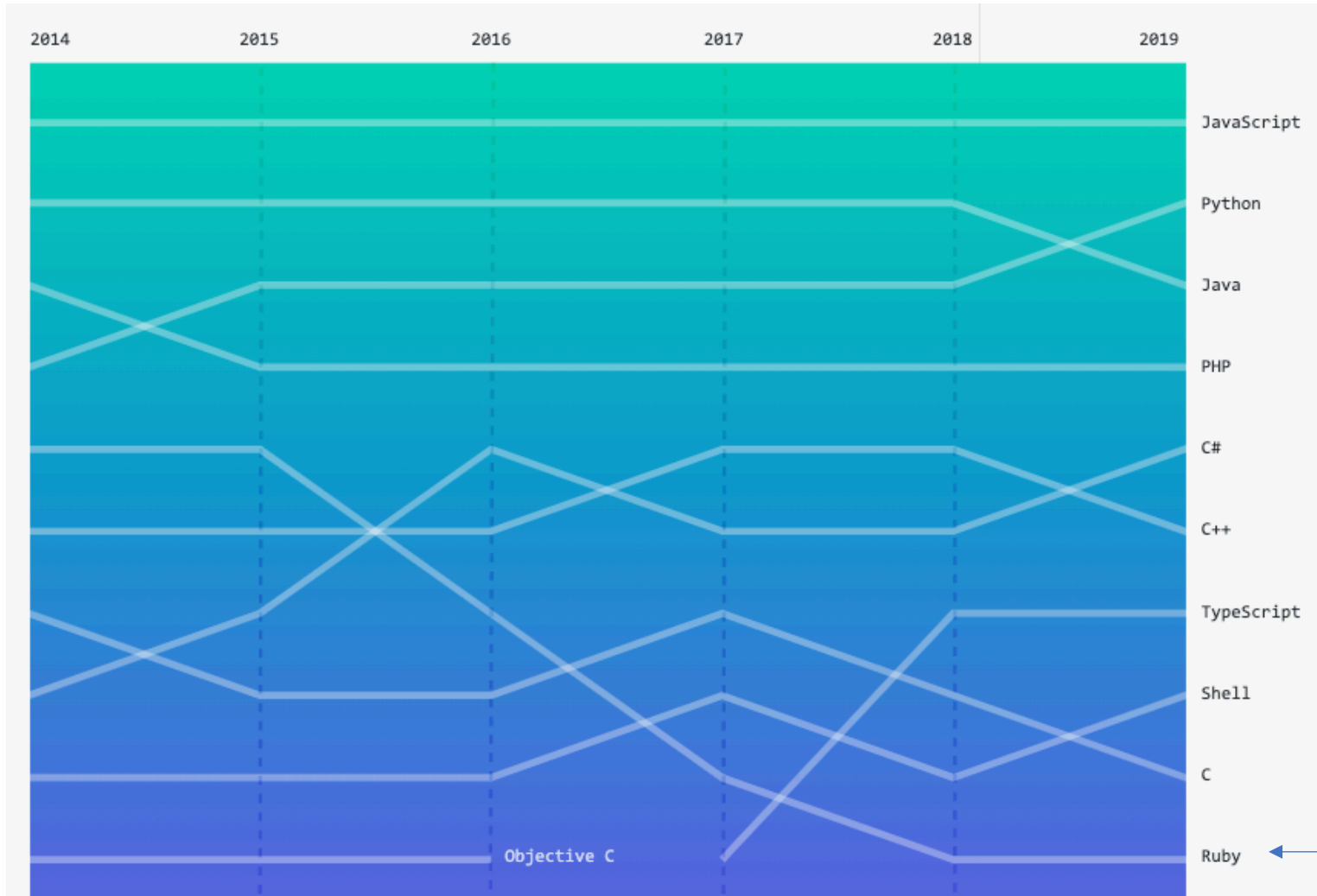




# ECE444

## 2019 vs 2020

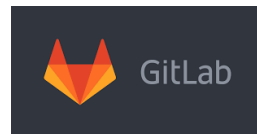
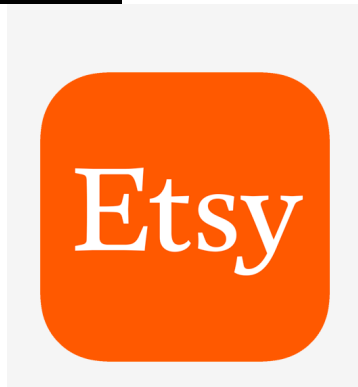
# Top Languages



Python

Ruby

# Companies using Ruby on Rails



<https://www.ideamotive.co/blog/40-best-ruby-on-rails-companies-websites>

# Companies using Python



Instagram



Quora



<https://realpython.com/world-class-companies-using-python/>

| Point of Comparison | Ruby on Rails                                                                                                                                                                     | Django                                                                                                                                                                        |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Language            | Ruby                                                                                                                                                                              | Python                                                                                                                                                                        |
| Known for           | Rapid development                                                                                                                                                                 | Dynamic applications                                                                                                                                                          |
| Main benefit        | Powerful RubyGems                                                                                                                                                                 | AI & ML apps                                                                                                                                                                  |
| Syntax              | Flexible coding                                                                                                                                                                   | One Obvious way                                                                                                                                                               |
| Popularity          | Vibrant community                                                                                                                                                                 | Academic teaching                                                                                                                                                             |
| Unique features     | COC, DRY                                                                                                                                                                          | Data science                                                                                                                                                                  |
| Pros                | <ul style="list-style-type: none"> <li>• Easy migration</li> <li>• Quick development</li> <li>• Diverse tools</li> <li>• Automated testing</li> <li>• Active community</li> </ul> | <ul style="list-style-type: none"> <li>• Scalable apps</li> <li>• Highly configurable</li> <li>• REST API</li> <li>• MVC programming</li> <li>• High compatibility</li> </ul> |
| Cons                | <ul style="list-style-type: none"> <li>• Tricky API creation</li> <li>• Low flexibility</li> <li>• Poor runtime</li> <li>• Poor documentation</li> </ul>                          | <ul style="list-style-type: none"> <li>• No multiple requests</li> <li>• Based on ORM</li> <li>• Standalone</li> <li>• Too tight knit</li> </ul>                              |



Let's go around the "room" for introductions:



Hello  
my name is



Hello  
my name is

Name (preferred name)

Interesting software development experience?  
One topic you are particularly interested in, if any?





## ACTIVITY: Introduce Yourself

Shurui Zhou

[All Sections](#)

Hi all,

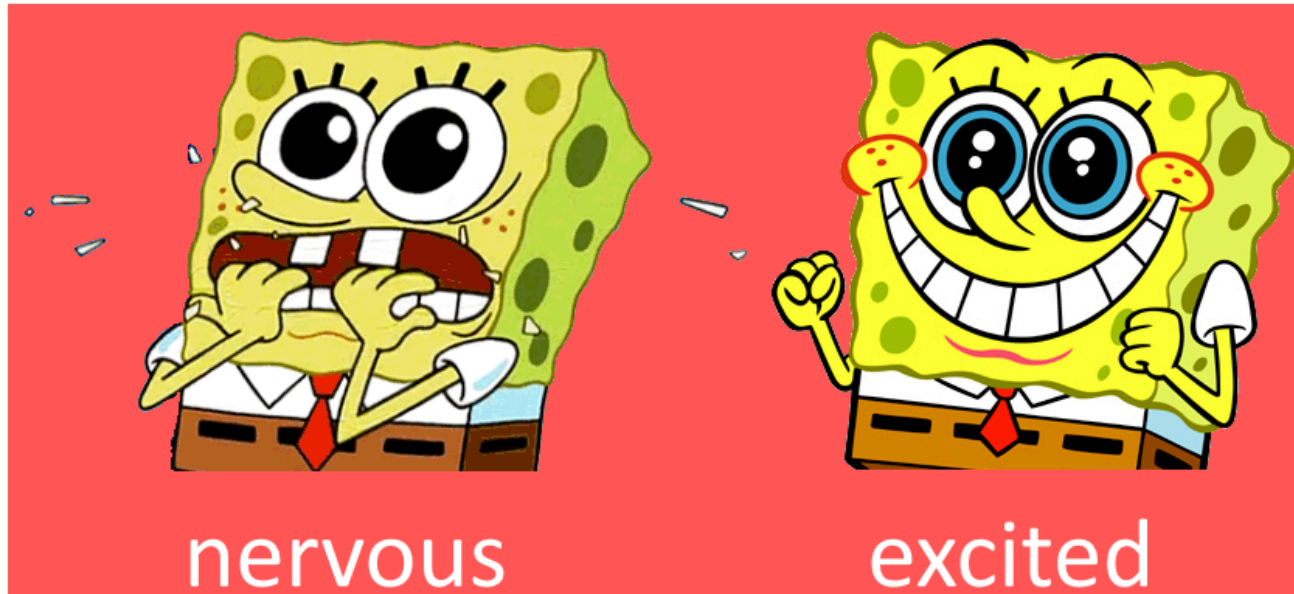
[Social presence and a feeling of community](#) are key drivers of student success in online learning. Please post below the fo to build rapport between students in this course. Please keep answers BRIEF and use only short phrases or bullet points.

1. name, program, which grade you are
2. one topic you would like to learn (will need a detailed answer in Assignment\_1)
3. if you like, post an image or a link, share a quotation or jot a few words that indicate **how you are feeling about this cc**

Here is an example below to get the discussion started.

### EXAMPLE

1. Shurui Zhou, Instructor of ECE444, 1st year faculty
2. Open source



Teachers in normal classes



**Stop talking**

Teachers now



**please guys say  
something**

# Active Lecture

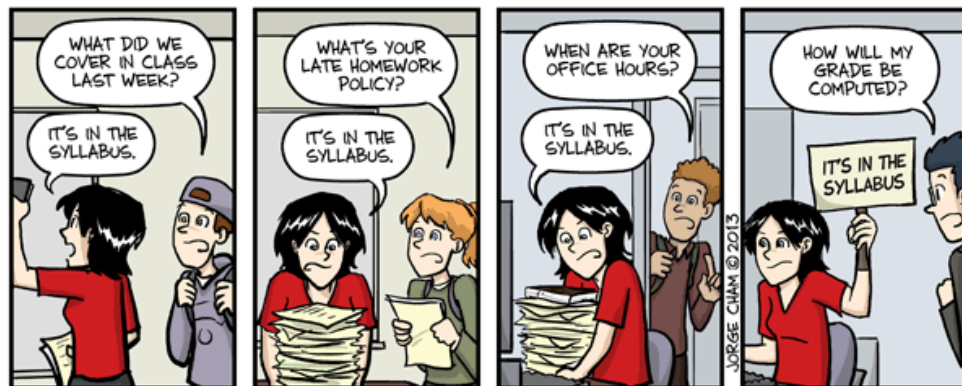
- Case study driven
- Discussion highly encouraged
- Contribute own experience
- Regular active in-class exercises
- In-class presentation
- Discussions over definitions



# Syllabus and course mechanics

- <https://shuiblue.github.io/UofT-ECE444/>
- Mixed with SE concept and Python&Flask
- Tools
  - Quercus: Assignment distribution, hand-in, and grades
  - Git, GitHub: code management
  - Piazza: Discussion board

Please post your questions on Piazza, not to TA's personal email



**IT'S IN THE SYLLABUS**

This message brought to you by every instructor that ever lived.

WWW.PHDCOMICS.COM



## Software Engineering

### Questions & Learning Goals

- When is a program good enough to ship? Have you built what the customer wanted?
- You can write code. Can you build software?
- How to efficiently collaborate with teammates in a distributed setting?
- How do you get a patch accepted into an open-source project?
- What are the research topics in the Software Engineering area?

(This course is a significant redesign of previous ECE444 and heavily inspired by the CMU Software Engineering courses [15-313](#) and [15-214](#). Many thanks to Prof. [Michael Hilton](#), Prof. [Christian Kästner](#), and Prof. [Claire Le Goues](#) for generously sharing the course materials and providing valuable advices.)

<https://shuiblue.github.io/UofT-ECE444/>

# Logistics - Lectures

**Lectures:** Mon/Wed/Fri 14:00-15:00 EST

- Bb Collaborate (zoom as a backup)
- 'U of T time' — classes start at 10 minutes past the hour, and wrap at the top of the hour.
- 20 min after each class to be available to talk or answer questions

# Logistics - Labs

- **Lab-1:** Thur 12:00-15:00  
**Lab-2:** Wed 09:00-12:00
- Bb Collaborate (zoom as a backup)
- Lab tasks (participation)

- Next week: Git & GitHub

## How flexible are you to attend Labs?

Lab1 (Thur 12:00-15:00)

Lab2 (Wed 09:00-12:00)

- ☐ I want to stick to what I've been assigned
- ☐ I am ok with either one
- ☐ None of these works for me

# Assignment 1 – fill in the survey

- To help us to tailor class and form teams

## Assignment 1 - First Class Survey

The survey contains **two part**:

- **Microsoft Survey** ([link](#)) -- to be used only for the purposes of our course to help the facilitator
- **Availability Survey** on When2Meet ([link](#)) -- to be used for scheduling virtual office hours

\* Please use your UofT email (the one linked on Quercus) as your name

\* Please fill in your availability in Eastern Standard Time

-----When2Meet screenshot-----

## ECE444 students availability

# Forming Teams

*“Instructors should form teams rather than allowing students to self-select.”*

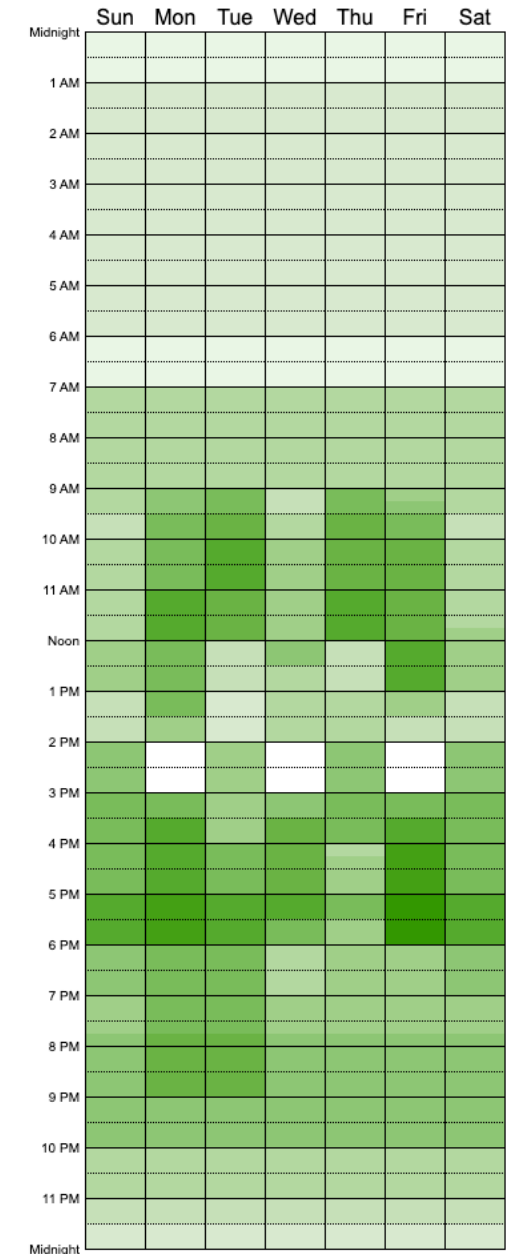
- 3-5 students per team
- Criteria: Form teams whose members are diverse in ability levels and who have common blocks of time to meet outside class [1].
- Gaining experience on generating and comparing alternative solutions and resolving conflicts
- We will send out the list of formed teams before Monday (9/14)

[1] Oakley, Barbara, et al. "Turning student groups into effective teams." *Journal of student centered learning* 2.1 (2004): 9-34.



# Logistics -- Office Hours

- 3 office hours times spanning the day/evening to cover Asia / Europe+Africa / America



# Reading and Quizzes

- Reading assignments for some lectures
  - Preparing in-class discussions
  - Background material, case descriptions, possibly also podcast, video, wikipedia
- Short and easy online quizzes on readings, due by start of lecture

# Evaluation

- Web application development (65%)
- Contribute to an open source project (20%)
- Participation in reading quizzes and lab tasks (15 %)

# Participation

Both quality and quantity are important,  
quality more than quantity

# Professionalism

- Being a professional means you should work well with others
- The best professionals are those who make those around them better
- If you feel someone is not treating you or someone else in a professional manner, you have two options:
  - If you feel you have the standing to do so, speak up!
  - Reach out to the course staff, and we will meet with you privately to discuss it, as well as preserve your anonymity

# Academic Honesty

- See web page
- In a nutshell: do not copy, do not lie, do not share or publicly release your solutions
- In group work, be honest about contributions of team members, do not cover for others
- If you feel overwhelmed or stressed, please come and talk to us (see syllabus for other support opportunities)



# Todos

- Surveys!
- Think about project ideas
- Check notification/emails regularly