CS 4530: Fundamentals of Software Engineering
Module 1.1 Course Introduction

Adeel Bhutta, Joydeep Mitra and Mitch Wand

Khoury College of Computer Sciences

Instructors

Adeel Bhutta

Section 1, 2, 5

Joydeep Mitra

Section 7, 8

Mitch Wand

Section 9

Teaching Assistants

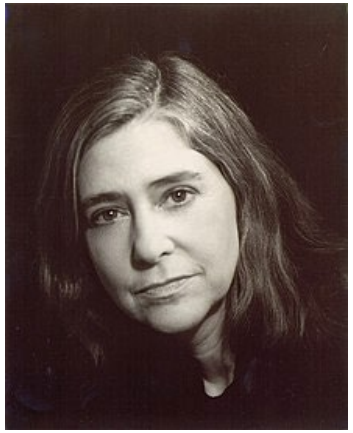
- We have around 300+ students and 20 teaching assistants.
- Their contact info and pictures are on the website https://neu-se.github.io/CS4530-Fall-2025/staff/

Learning Objectives for this Lesson

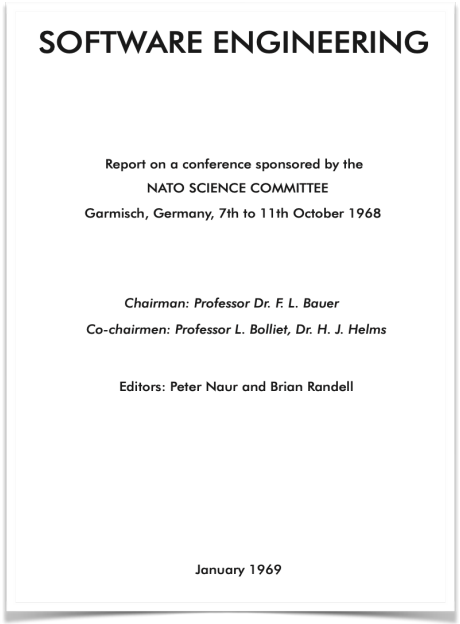
- By the end of this lesson, you should be able to:
  - Explain in general terms what software engineering is
  - List your weekly obligations as a student
  - List the requirements for completing the course

The idea of "software engineering" dates back to 1969

Margaret Hamilton
@ NASA, around 1963



The Apollo Guidance Computer's software



A call to action: We must study *how to build software*

NATO conference on Software Engineering + Outcomes

- Software was very inefficient
- Software was of low quality
- Software often did not meet requirements
- Projects were unmanageable and code difficult to maintain
- Software was never delivered

Goal: to make software an engineering discipline

Can we discover methods to build software that are as predictable in quality, cost, and time as, for instance, those used to build bridges in civil engineering?

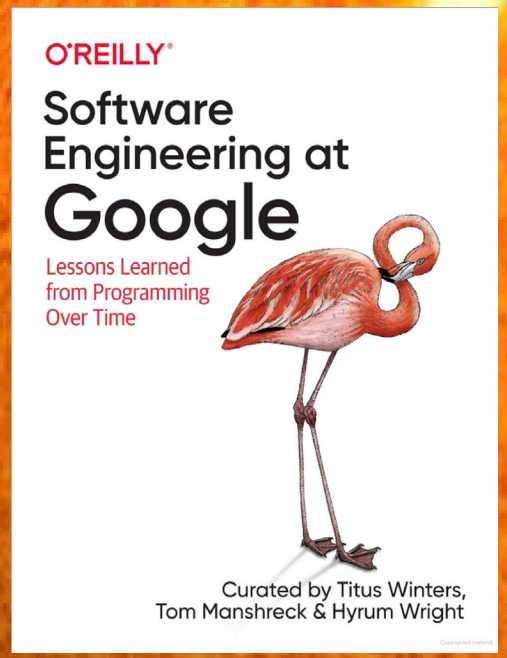
Software Engineering encompasses the entire software life cycle

- It should apply to the
  - design,
  - construction,
  - and maintenance

  => of large programs

  => over time.

Okay, what do you mean by "large"?



The Apollo Guidance Computer's software

Almost any pre-series-B startup

Your 4-person project in this class

(image from "The Scale of Space" on KWIT, March 2018)

Problem #1: Programs need to be read by people

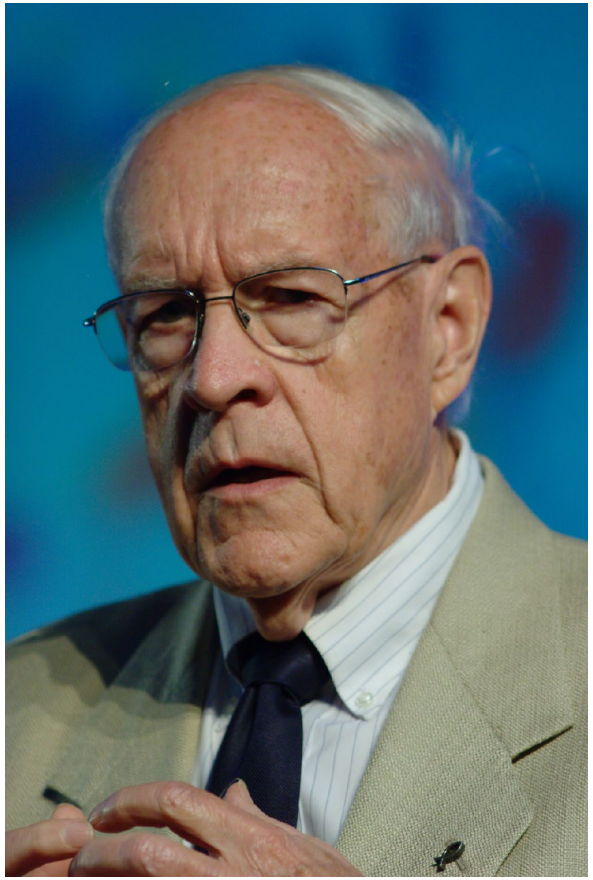
"Any fool can write code that a computer can understand. Good programmers write code that humans can understand"

- Martin Fowler

Problem #2: People need to talk to each other

"Adding manpower to a late software project makes it later"

Fred Brooks, 1975



So, software engineering must encompass:

PROGRAMS, PROCESSES, AND PEOPLE

The course will cover

- Programs
  - how to write programs that people can understand and maintain
  - in a particular domain (medium-sized web application)
- Processes
  - how to divide a large project into engineering tasks
  - how to coordinate the tasks to form a coherent whole
- People
  - how to organize teams and make them function effectively.

Think of {your software's} design at three scales

The Planning Scale

- key questions: How do we make software artifacts "good"? What does that mean? Who decides?

The Organizational Scale

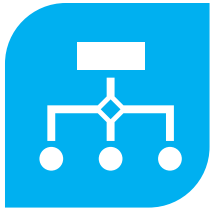
- key questions: What are people's needs? How do we design software artifacts that meets those needs?

The Implementation Scale

- key question: how to design software artifacts that are easy to test, understand, and modify?

So, software engineering must encompass:

PLANNING,



ORGANIZING,

& IMPLEMENTING

So, software engineering must encompass:

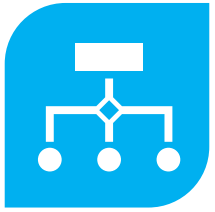
PEOPLE

PROCESSES

PROGRAMS

PLANNING

ORGANIZING



IMPLEMENTING

mostly out of scope

This class is here-ish

previous classes (OOD)

Learning Objectives for this course:

- By the end of this course, you will--
  - Be able to define and describe the phases of the software engineering lifecycle.
  - Be able to explain the role of key processes and technologies in modern software development.
  - Be able to productively apply instances of major tools used in elementary SE tasks.

These are normal learning goals for a course at a university!

- Design and implement a portfolio-worthy software engineering project in a small team environment that can be showcased to recruiters.

This is maybe a little different

The course will be delivered through:

- In-Class materials / lectures
  - in-person (most sections) or via zoom (online section)
  - slides (available on course website)
- Practice Activities
  - these will give you practice with the technologies we will use
  - we will often start these during class
  - these will be graded
- Tutorials
  - these will give you background on key processes and technologies {we will use}
  - at a greater level of detail than we can cover in class
  - much like good blog posts

Course Mechanics: Lectures and Attendance

- Classes will include both lectures and in-class activities.
- Be sure to bring your laptop
- Each instructor will use **individual approach** to grade the in-class activities.
- 100% attendance is expected for both on-the-ground and remote sections (especially when working on activities, “work on project” sessions and demos)
  - For excused absence, please contact the instructor by email.

Course Deliverables

- First, an individual project, which we will assign. This is to be done **individually**.
  - divided into 2 deliverables (not equally weighted).
  - this counts for 30% of course grade
- Then a **group** project, done in teams of about 4 people
  - this counts for 40% of course grade
- There will be an exam (worth 20%) on Oct 29 - 31 during Week 9). There will not be a final exam. Check Calendars
- Participation / Completion of activities (10%)

Technology

- We will use:
  - TypeScript as implementation language
  - Jest as Testing Framework
  - Visual Studio Code as our IDE
  - React for webapps
  - GitHub Projects for Project Management
  - GitHub Actions / Netlify /Heroku / Render for CI/CD
  - Also, other miscellaneous tools

Projects for You

- We will be using **Fake StackOverflow** as codebase
- The individual projects will help you become familiar with the codebase (worth 30% of course grade).
- The team project (worth 40%) will be a new feature that you will propose.
  - Instructors will form the teams **with** your input.
- Further breakdown of team project grade is:
  - Planning (worth 8% of course grade)
  - Process (worth 8%)
  - Product (worth 16%)
  - Reports (worth 8%)
- Peer evaluations (surveys) may be utilized, and individual contributions **WILL** impact your project grade (between 0-100%).

Grade Appeal Policy

- If you have concerns regarding the grading of your work, please let us know right away.
  - We provide mechanism for you to request regrades for all work submitted
  - Do not post on Piazza or email your TA or instructor
  - All regrade requests must be submitted within **7 days** from your receipt of the graded work.
  - If your regrade request is closed and you feel that the response was not satisfactory, you may appeal to the instructor via email within 48 hours

Late Policy

- Your work is **late** if it is not turned in by the deadline.
  - 10% will be deducted for late individual work turned in within 24 hours after the due date
  - Individual work submitted more than 24 hours late will receive a zero.
  - If you're worried about being busy around the time of a HW submission, please plan ahead and get started early.
  - No late submissions allowed for any **group work**
  - If you have an accommodation from Disability Access Services (previously DRC), you must request it from the instructors separately for each assignment or exam.
    - DAS or DRC Accommodations are usually NOT available for Group Assignments (please work with instructor)

Academic Integrity (1)

- Students must work individually on all homework assignments.
- We encourage you to have high-level discussions with other students in the class about the assignments, however, we require that when you turn in an assignment, it is only your work. That is, copying any part of another student's assignment is strictly prohibited.
- If you steal someone else's work, you fail the class.
- You are responsible for protecting your work. If someone uses your work, with or without your permission, you fail the class.

Academic Integrity (2)

- For Individual Projects and Activities.
  - The use of auto-complete tools is **permitted**
  - Use of Generative AI (eg Chat GPT, Claude, etc.) is **prohibited**
  - You may use LLM based AI tools like search engines, but you can **never** **copy-paste** code.
- For final projects
  - Use of AI tools is **permitted** as long as you understand what was submitted.
  - Use of **'vibe'** coding is strongly **discouraged**
- We reserve the right to "interview" you to gauge your understanding (with possible grade adjustments)

Academic Integrity (3)

- If you are concerned that by reusing and attributing that copied code it may appear that you didn't complete the assignment yourself, then please raise a discussion with the instructor.
- If you are in doubt whether using others' work is allowed, you should assume that it is NOT allowed unless the instructors confirm otherwise.

Communication

- Course web page (https://neu-se.github.io/CS4530-Fall-2025)
  - **Canvas** will mirror the course web site.
  - Assignments, important notices, etc., will appear in both places.
- Piazza (see Canvas for link)
  - Questions about content, policies, assignments, projects, etc. are better asked on Piazza, so everybody gets the same answers.
- Contacting the Instructor
  - For private questions about your individual situation, please email the instructor directly (do NOT use Canvas messages – sometimes they do not get through to the instructors)
  - Please put CS4530 in the subject line so your message does not get overlooked
  - We encourage all students to "meet" with the instructor at least once!
- Office Hours
  - Schedule is available at (https://neu-se.github.io/CS4530-Fall-2025/staff/)
  - TA Office Hours are held via **Khoury Office Hours App**

Review

- Now that you've studied this lesson, you should be able to:
  - Explain in general terms what software engineering is
  - List your weekly obligations as a student
  - List the requirements for completing the course