CS 4530: Fundamentals of Software Engineering

Module 1.1 Course Introduction

Adeel Bhutta, Rob Simmons, and Mitch Wand

Khoury College of Computer Sciences

Instructors

Adeel Bhutta

Section 1,2,3

Mitch Wand

Section 4

Rob Simmons

Section 5,6

Teaching Assistants

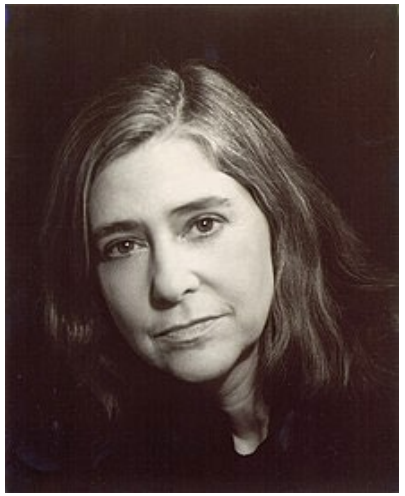
- We have around 360 students and 18 teaching assistants.
- Their contact info and pictures are on the website https://neu-se.github.io/CS4530-Spring-2026/staff/

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

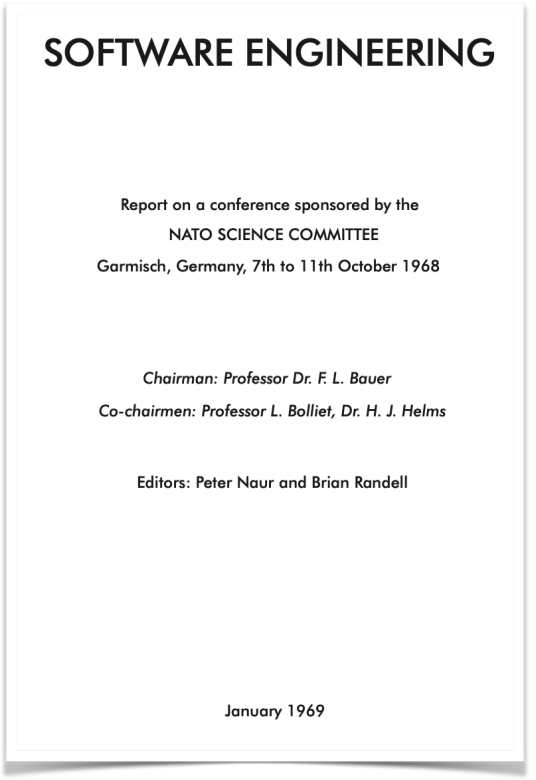
Origins of "software engineering"

Margaret Hamilton @ NASA, around 1963



The Apollo Guidance Computer's software, basically

1968 NATO conference on Software Engineering + Outcomes



SOFTWARE ENGINEERING

Report on a conference sponsored by the
NATO SCIENCE COMMITTEE
Garmisch, Germany, 7th to 11th October 1968

Chairman: Professor Dr. F. L. Bauer
Co-chairmen: Professor L. Bolliet, Dr. H. J. Helms

Editors: Peter Naur and Brian Randell

January 1969

Friedrich Bauer

Barry Boehm

Anthony Oettinger, ACM President

Comm. of the ACM, August 1966 (page 546)

ing nature. We must recognize ourselves—not necessarily all of us, and not necessarily any one of us all the time—as members of an *engineering* profession, be it hardware engineering or software engineering, a profession without artificial and irrelevant boundaries like that between "scientific" and "business" applications.

(all pictures from Wikipedia)

Goal: to make software an engineering discipline

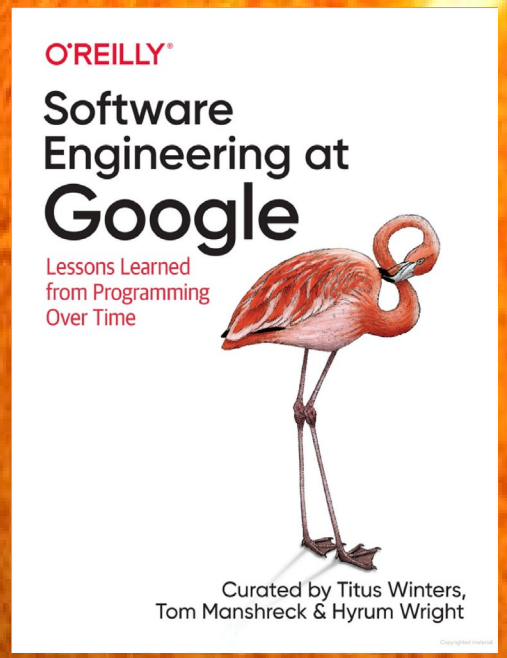
Software engineering concerns the

- 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

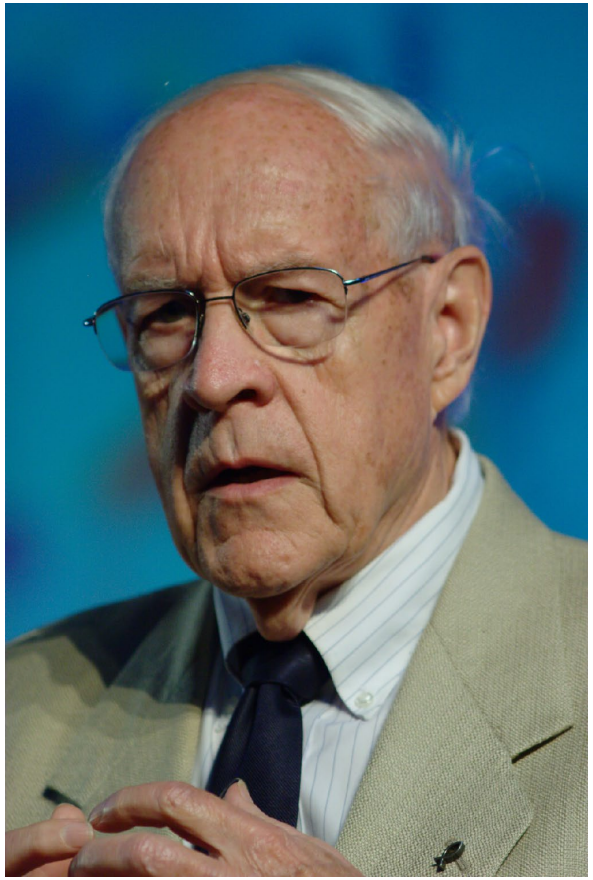
"Any fool can write code that a computer can understand. Good programmers write code that humans can understand"

- Martin Fowler

Problem #2: People need to talk to each other

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

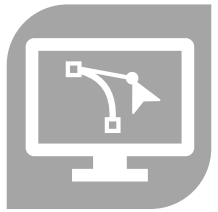
Fred Brooks, 1975



So, software engineering must encompass:

PEOPLE,

PROCESSES,



& PROGRAMS

The course will cover

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

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

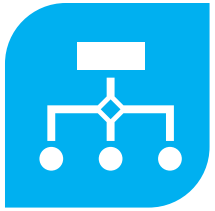
- key questions: What are people's needs? How do we design software artifacts that meets those needs?

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:

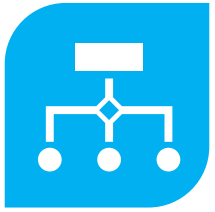
PEOPLE

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.
  - …demonstrate the ability to ***use*** these key processes
  - …demonstrate the application of key technologies and major tools used in elementary software engineering tasks.
  - …design and implement a portfolio-worthy software engineering project in a small team environment that can be showcased to recruiters.

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

This is maybe a little different

The course will be delivered through:

- Lectures and in-class discussion
  - In person (most sections) or via zoom (only for online section)
  - Slides will be available on course website
- Practice activities
  - these will give you practice with the technologies we will use
  - we will often start these during class, sometimes in groups
  - 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
  - Primary “reading” for the course in lieu of a textbook

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.
- 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 3 deliverable projects
  - 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 March 11 - 13 during Week 10). There will not be a final exam. Check Calendars
- Participation / Completion of activities (10%)

Course Requirements

To pass, you must also:

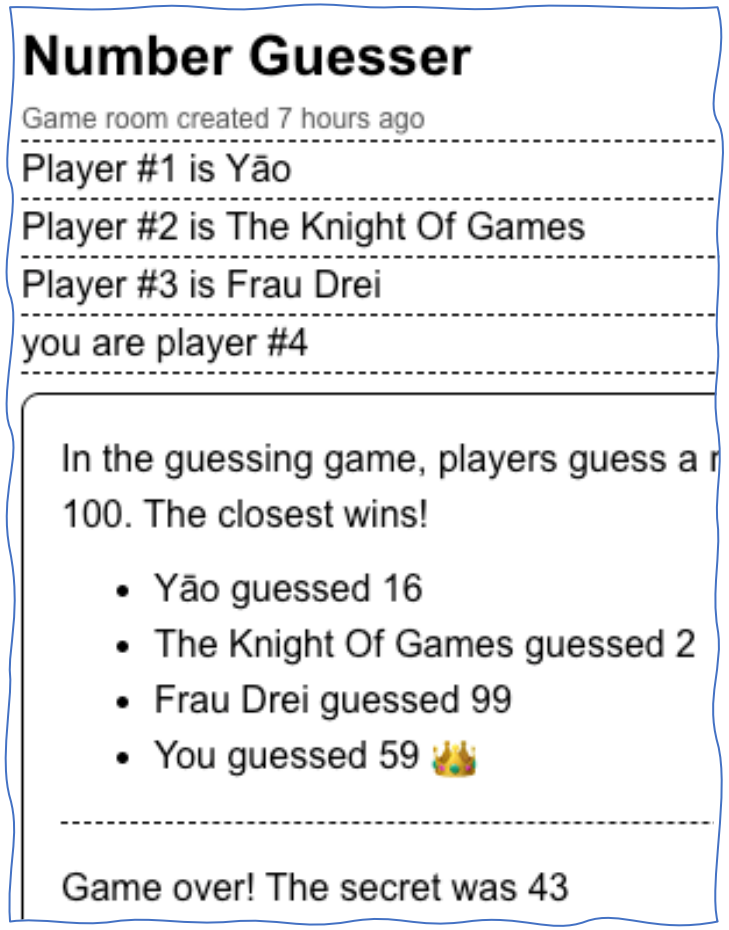
- Do most of the individual projects
- Demonstrate understanding of concepts in midterm
- Demonstrate ability to do software development & engineering in group project
- Present your final project

Technology

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

Welcome to the Game Nite team!

- We want to be the #1 spot for people who want the social experience of watching Twitch™ but for turn-based games
- CS 4530 is part traditional academic course, part “having a new software job” simulation
- The individual projects are like “onboarding” projects, and the group project is a scaled-down version of actual product delivery
- Course staff have a secondary role as part of this simulation



Welcome to the Game Nite team!

- 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 (usually Gradescope appeals)
  - 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

- Individual projects & activities are **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.
  - If you have an accommodation from Disability Access Services (previously DRC), you must request it from the instructors separately for each assignment or exam.
- No late submissions allowed for any **group project** submissions
  - DAS Accommodations are usually NOT available for Group Assignments (please work with instructor, we'll make something work!)

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. Copying any part of another student's assignment is strictly prohibited.
- Small snippets of publicly posted code are allowed **with attribution**
- 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.

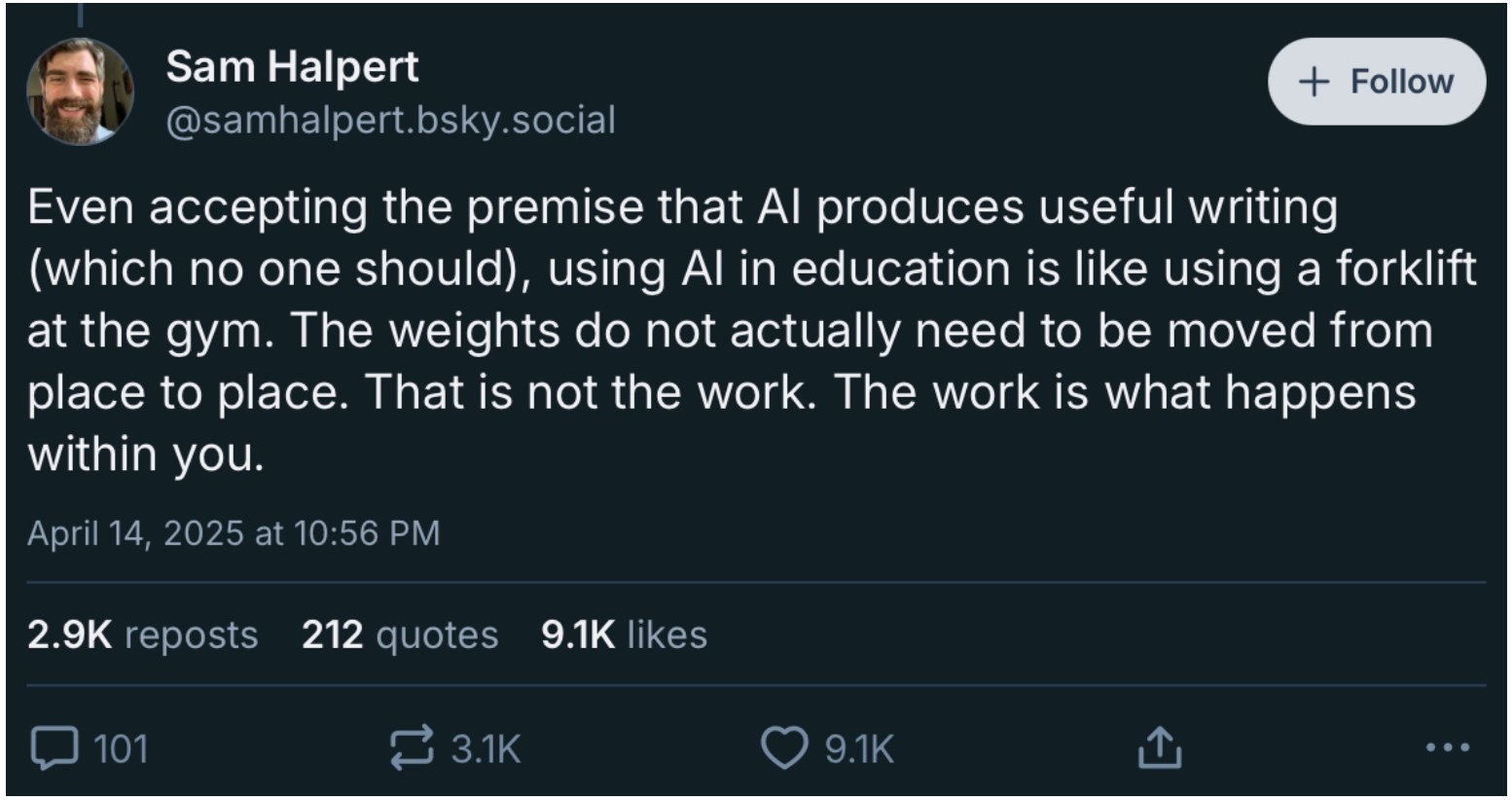
Academic Integrity (2)

- 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.

Academic Integrity & LLM-based Tools

IP1 & IP2 are designed for you to do **without LLM-based tools.**

- No Copilot fancy auto-complete
- No natural language to code tools
- No chatbot-assisted code understanding



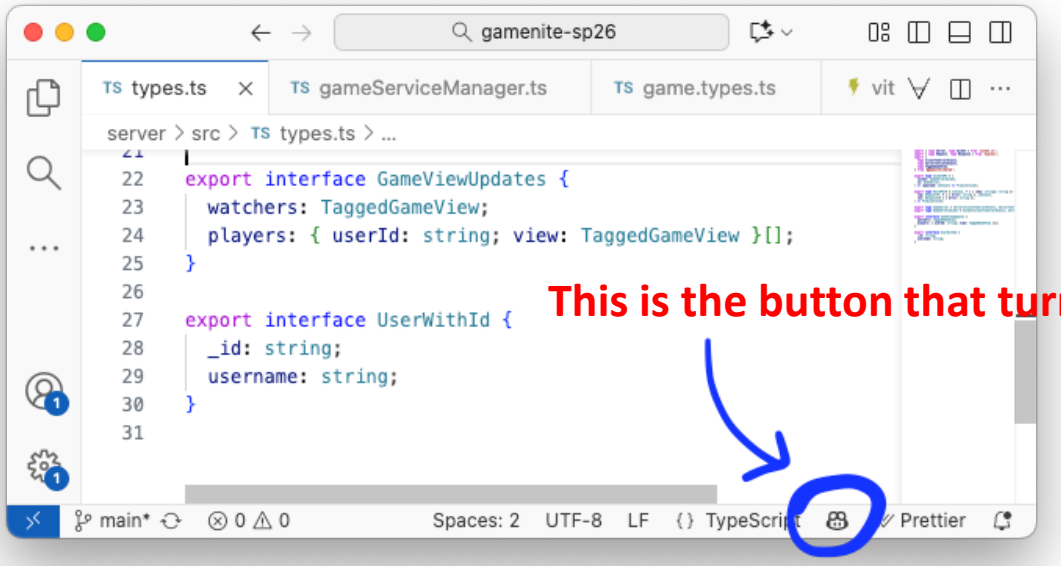
https://bsky.app/profile/samhalpert.bsky.social/post/3lmt3coqvqk2w

Using LLM-based Tools in This Class (1)

IP1 & IP2 are designed for you to do **without LLM-based tools.**

- No Copilot fancy auto-complete
- No natural language to code tools
- No chatbot-assisted code understanding

Why? Because there are sophisticated TypeScript-powered tools for navigating codebases in VS Code (which don't require a subscription). We want you to **<u>also</u>** have experience with these tools.



Using LLM-based Tools in This Class (2)

- For IP3, exploration of generative tools is **gently encouraged**
- For final projects
  - Use of AI tools is **permitted** — create a group policy on AI
- **You are responsible for your code and must understand it**
  - **You must document uses of LLM-based tools**
- We reserve the right to interview you to gauge your understanding (with possible grade adjustments)

No LLM Tools For Writing

- When we ask you to write English text, it's because we want you to write it and because we're going to read it.
- You're just disrespecting yourself and us by having us read LLM-generated text.

Communication

- Course web page (https://neu-se.github.io/CS4530-Spring-2026/)
  - **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-Spring-2026/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