CASE WESTERN RESERVE
UNIVERSITY

Introduction

CSDS 393/493: Software Engineering

Spring 2025

Agenda

- Course information
  - Structure
  - Policy
- What is software engineering
  - History
  - Contents

About Me

Sumon Biswas

Assistant Professor

Department of Computer and Data Sciences

Case Western Reserve University

Research Interest:

- Software Engineering and AI
- Responsible AI Engineering, Software Safety and Fairness

https://sumonbis.github.io

Course Overview

Learning objective

- Understanding of the **software development lifecycle**
- Understanding of the issues and challenges
- Familiarity with the **SE techniques and tools** for each phase
- Experience working as part of a team on a significant software development **project**
- Cultivation of critical thinking skills needed by professional software engineers

Course Overview

Main activities

- **Class participation**
- **Quizzes and assignments**
- **Group project**
  - Milestones
    - Project proposal
    - Software requirement specification
    - Design document
    - Functional testing
  - Three "live" demonstrations
  - **Final** presentation

Grade Breakdown

- Quizzes and assignments (60%)
- Project (40%)

Course Registrations

- Ensure that you satisfy the prerequisite
  - CSDS 233 or ECSE 233 or Data Structures with a grade of C or higher
  - Intermediate to advanced **programming** skills in at least one modern programming language, such as Java, C++, C#, Python, Rust, Kotlin
- CSDS 393
  - Computer Science Major/Minor or Data Science Major
  - Others can register for 393N
- CSDS 493
  - Graduate students
  - Students will develop an additional feature for the project and submit an extra assignment

Team Project

- Teams
  - 4 students (in special case 3-5 students that needs approval)
  - Assign a team leader (mostly for communications)
- Project will be proposed by the teams and approved by the instructor
- Essential components
  - Front-end
    - Example: android app, or webpages, etc.
  - Back-end
    - Example: databases, etc.
  - Multi-user features

Teamwork

- Teamwork is an essential part of this course
- Projects have components that are graded for the entire group and components that are graded individually

We expect significant efforts for productive teamwork and addressing the team issues

- Managing the team, time, and risks

Start forming the teams

Forming the Team

- Form groups based on schedule availability
  - This is ridiculously important
- Consider experience and working styles
- Share your background and interests
  - Posting will be available on Canvas

Use of Generative AI

- Must not be used in Quizzes
- Can be used for the project. However, the models will sometimes hallucinate and generate superficial, bogus output
- It's your responsibility to check the quality of the output from an LLM
- In your submissions, clearly document how you've used these tools
  - Models used for generation
  - Prompts used for generation
  - Other methods

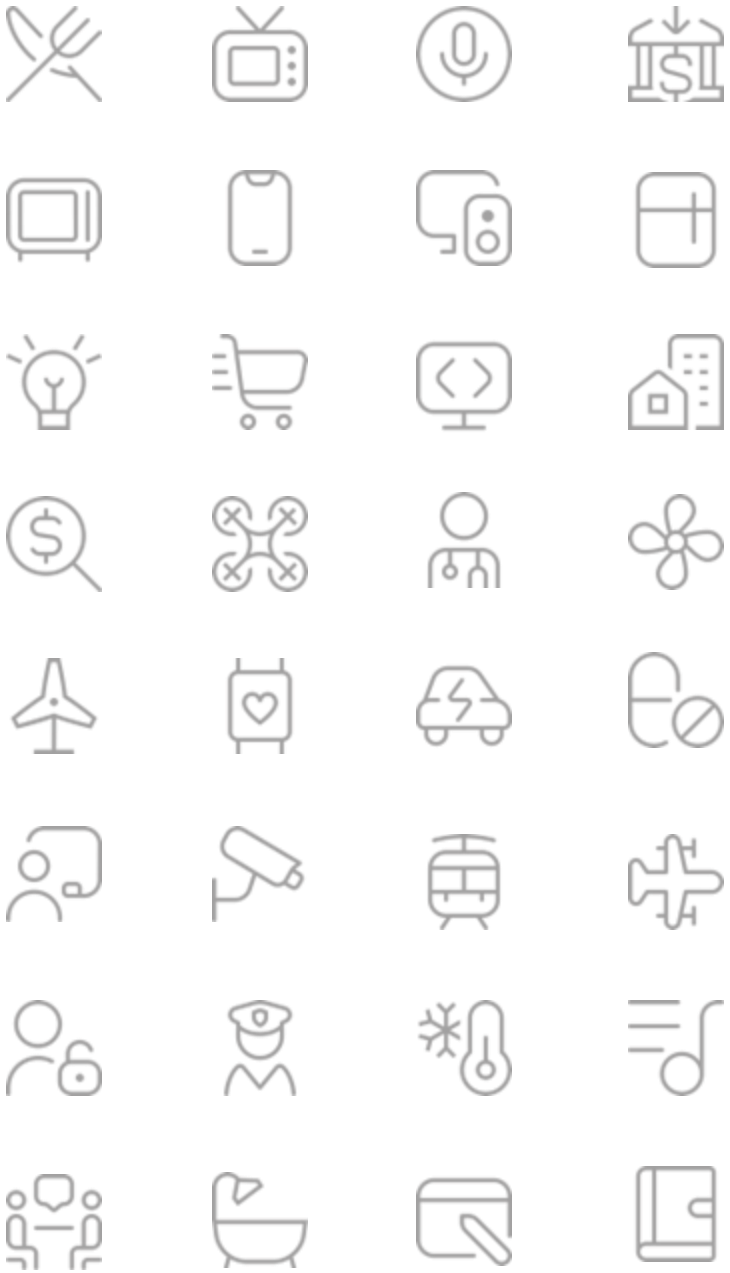
Late Day Policy

- Assignments: No late days
  - Simply doesn't work with team assignments
  - Plan for unexpected delays ahead of time (not just before deadline)

- No makeup quiz or late assignment unless approved for the following:
  - You request at least 48 hours before the quiz or due date (giving a valid reason)
  - Health reasons

Course Philosophy

- Hands-on Experience in a collaborative project
- Growth mindset & learning from failures
  - Learning from real-world case examples
  - Learning from your own mistakes in the project
- Active student participation
  - We encourage you to ask questions and participate in class discussions
  - Wrong answers support learning!



Software is everywhere

NEWS

Software glitch cost Hamilton victory - Mercedes

25 March 2018

2018 | MERCEDES | AUSTRALIA | HAMILTON

"We calculated the VSC gap which was needed [if one was activated]. Our computer said 15 seconds was the necessary time in order to jump us."

DESIGNLINES | AUTOMOTIVE DESIGNLINE

Toyota Case: Single Bit Flip That Killed

By Junko Yoshida 10.25.2013 0

Share Post Share on Facebook Share on Twitter

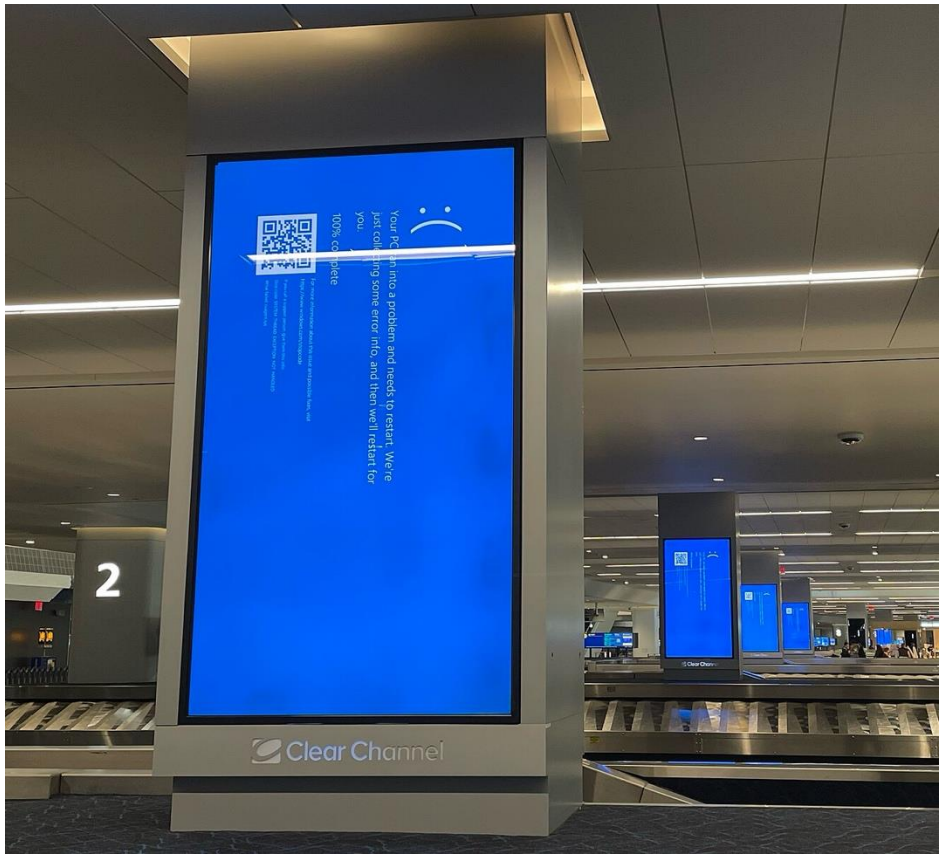
MADISON, Wis. — Could bad code kill a person? It could, and it apparently did.

The Bookout v Toyota Motor Corp. case, which blamed sudden acceleration in a Toyota Camry for wrongful death, touches the issue directly.

This case — one of several hundred contending that Toyota's vehicles inadvertently accelerated — was the first in which a jury heard the plaintiffs' attorneys supporting their argument with extensive testimony from embedded systems experts. That testimony focused on Toyota's electronic throttle control system — specifically, its source code.

The plaintiffs' attorneys closed their argument by saying that the electronic throttle control system caused the sudden acceleration of a 2005 Camry in a September 2007 accident that killed one woman and seriously injured another on an Oklahoma highway off-ramp. It wasn't loose floor mats, a sticky pedal, or driver error.

NTSC revealed that a new software function in the flight control system caused the aircraft to nose down.



Multiple blue screens of death caused by a faulty software update at LaGuardia Airport, NY

Chaos and Confusion: Tech Outage Causes Disruptions Worldwide

July 19, 2024 at 7:49 am | *Updated July 19, 2024 at 12:28 pm*

Travelers wait in Terminal 1 for check-in at Hamburg Airport, in Germany on Friday July 19, 2024, as a widespread Microsoft outage disrupted flights, banks, media outlets and companies around the world on Friday. (Bodo Marks/dpa via AP) **Less**

Software Engineering

The term software engineering has multiple meanings

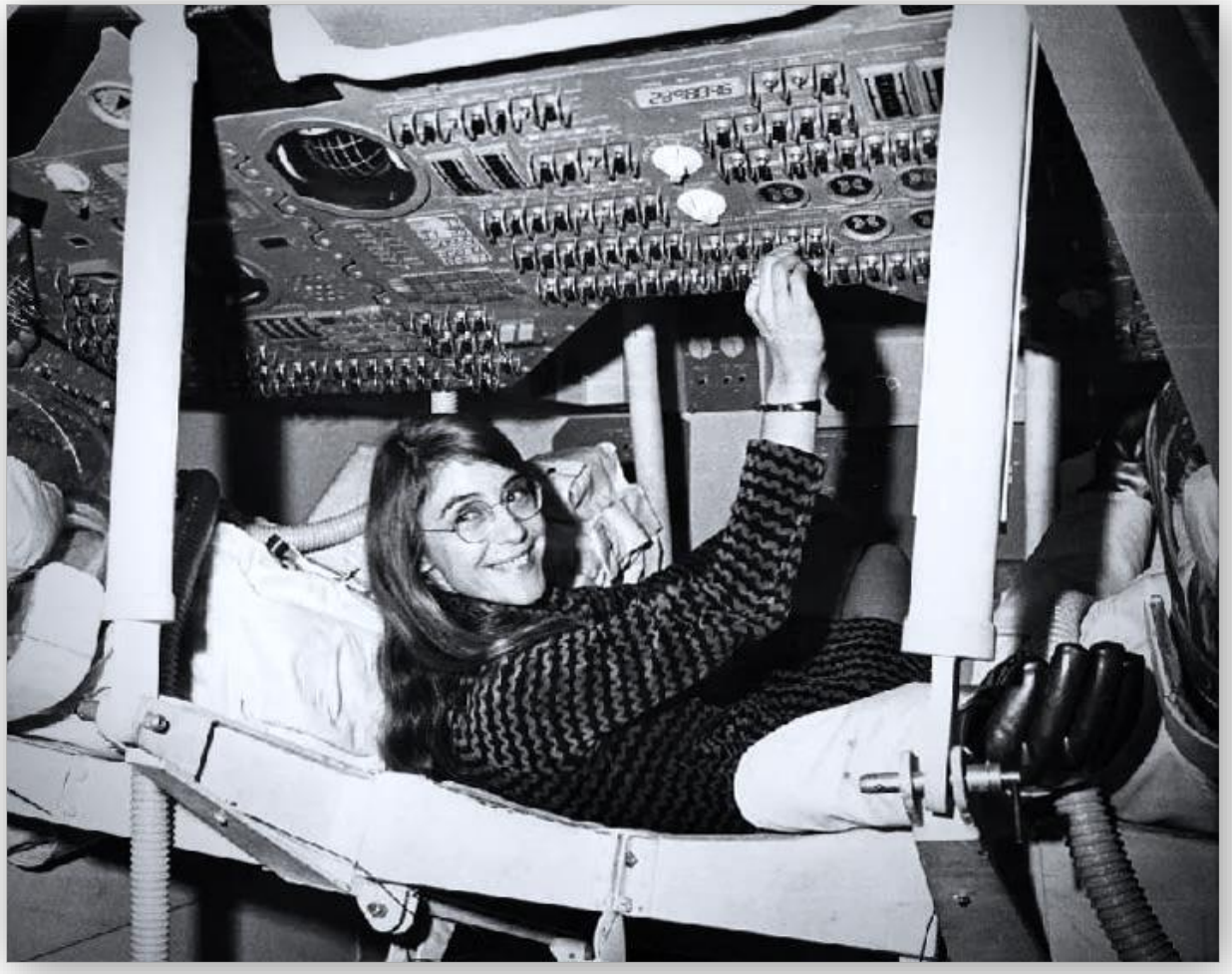
- The application of "engineering principles" to software development
- The software development profession
- The field of computing research that aims to improve methods and tools for software development

Software engineering methodology is the body of techniques used to develop software.

1968 NATO Conference on Software Engineering

- Provocative Title
- Call for Action
- “Software crisis”

https://isthisit.nz/posts/2022/1968-nato-software-engineering-conference/



She coined the term “software engineer” to describe her work and highlight the importance of this previously underrated area.

Margaret Hamilton, the lead of Apollo lunar mission software

Goals of Software Engineering

To produce, as quickly and inexpensively as possible, software that is:

- Easy to use and satisfies users' needs
- Reliable
- Efficient
- Straightforward to maintain, adapt, and enhance
- Secure

Aspects of Software Engineering

Technical aspects:

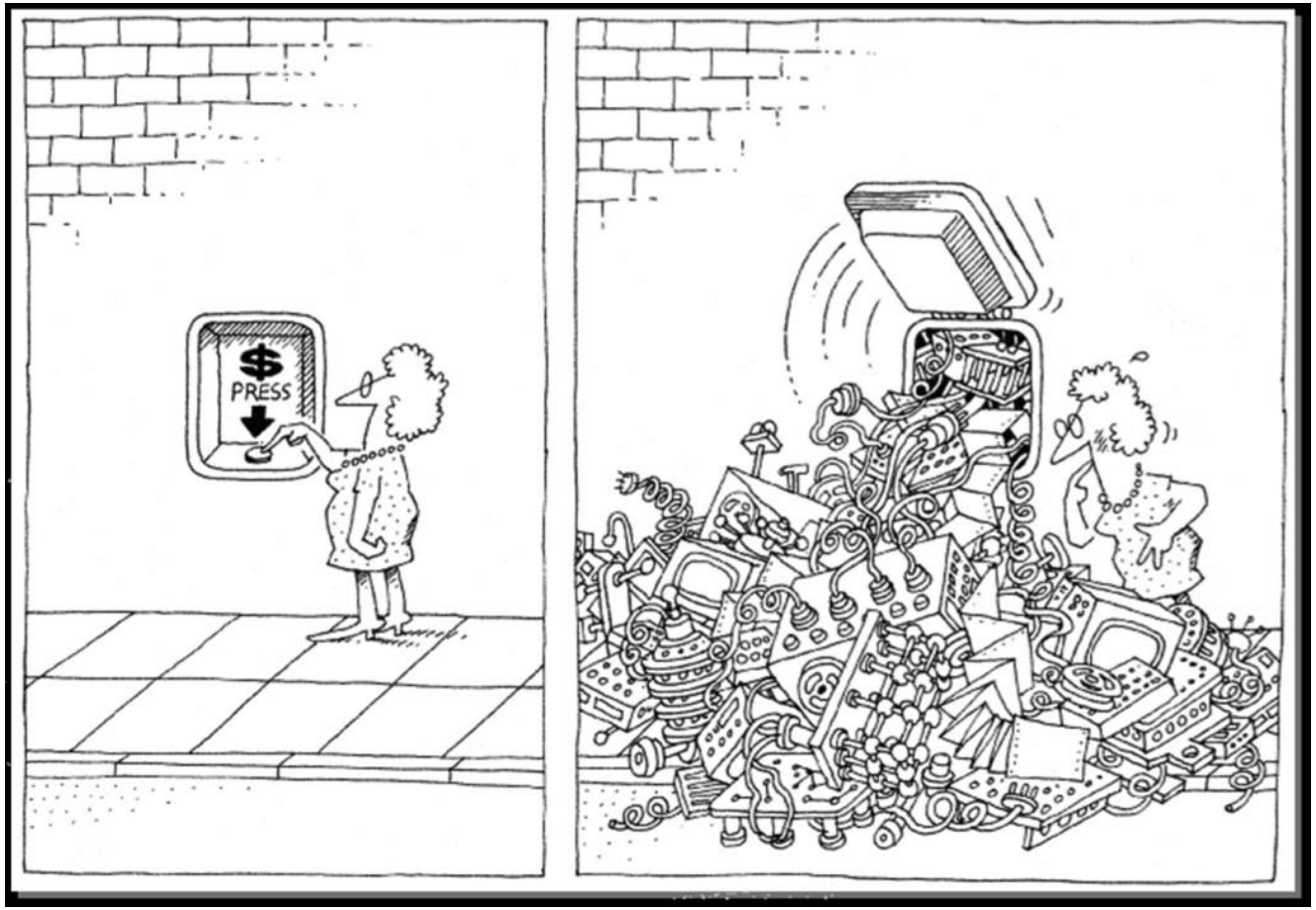
- Specification
- Design
- Programming
- Inspection and testing
- Static & dynamic analysis, "Software analytics"
- Debugging
- Maintenance and evolution
- Version control, integration, configuration management, and deployment

Aspects of Software Engineering

Non-technical aspects:

- Project management
  - Software is developed by teams
- Psychology
  - cognitive, behavioral, organizational
  - example: Behavior-Driven Software Design
- Law and ethics:
  - contracts, liability, intellectual property

Software Complexity

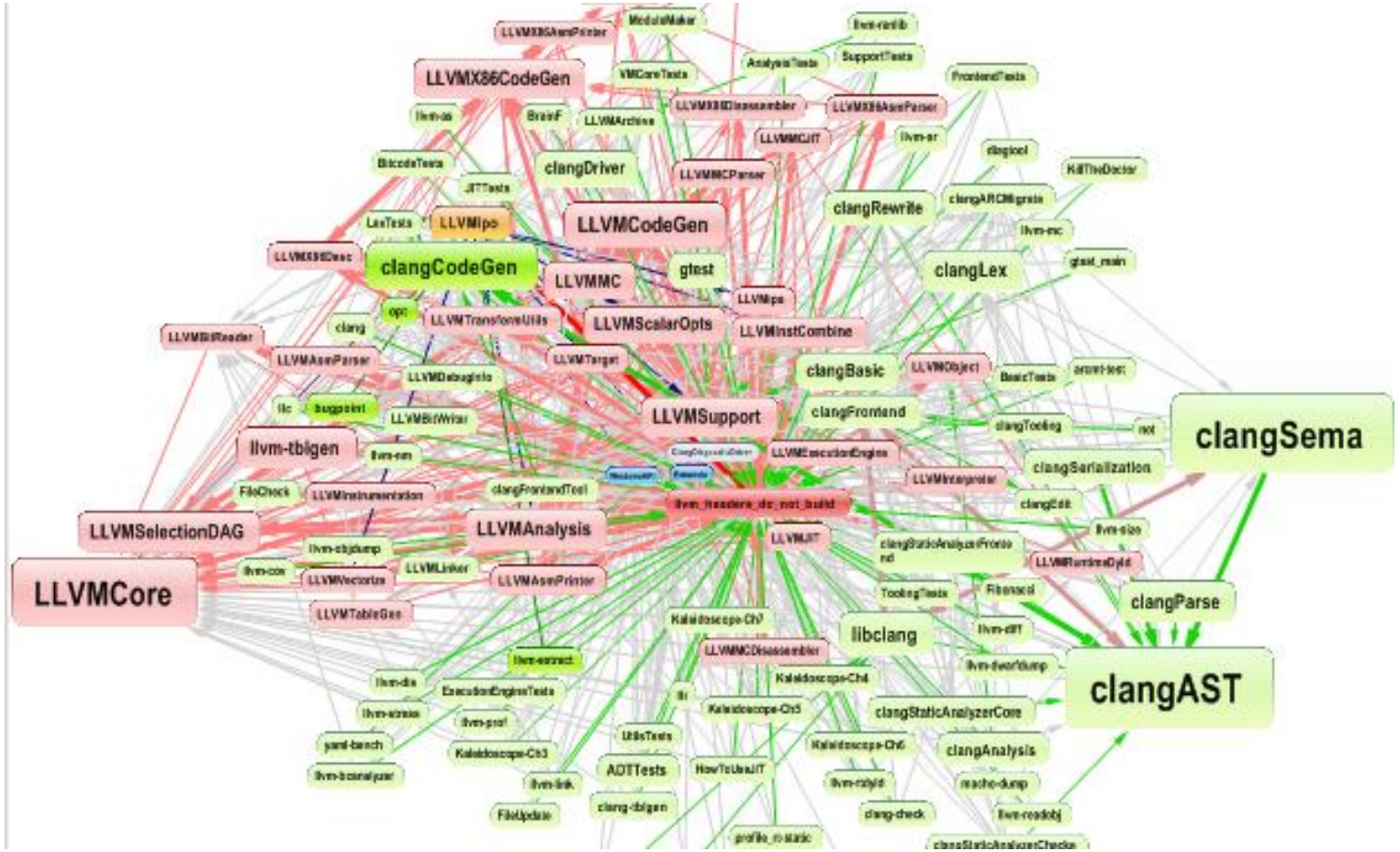


Software Complexity

Examples:

- Mac OS X 10.4: ~84 million SLOC.
- Linux kernel: ~10 million SLOC
- Google Suite (Gmail, Drive, Search, etc.): ~ 2 billion SLOC.
- Twitter/X: 10 million SLOC
- Zoom: 60,000 SLOC

Example: C-Language Family LLVM



The dependency graph for namespaces in LLVM library

Software Complexity

A primary issue confronting software engineers is complexity:

- Problem complexity
- Design/implementation complexity
- Platform and environment complexity
- Complexity due to change

Large software systems are among the most complex artifacts ever produced by human.

Consequences of Software Complexity

- Complex systems are difficult and time-consuming to produce and to maintain.
- They cannot be fully understood by any one person.
- They seldom satisfy all user needs and desires, which are highly changeable.
- They invariably contain residual defects.
- Many projects are completed late and exceed budget.
  - Some are never completed.

Software Engineering Methodology

- **Broad collection of techniques and tools addressing each phase of SDLC**
- **Continually evolving**
- **Specialized for particular subfields, e.g.,**
  - Web and mobile applications
  - Real-time systems
  - Health informatics
- **Some influential methods:**
  - Object-oriented programming
  - "Agile" methods
  - Design patterns
  - Test-driven design
  - A/B testing

Coarse Goal

To help you to understand the challenges, key ideas, and methods of large-scale software development

Reading Materials

There is no required textbook.

The presentation slides and reading notes will be shared on Canvas.

If you want to get one to supplement for the class notes, I recommend the latest edition of **Software Engineering by Ian Sommerville** (Pearson publishers)

Contact

Email: sumon@case.edu

Office: Olin Hall 608

Office hour: TBD (see canvas)

Guidance

- Course & HW structure may be different than what you are used to …
- Lecture topics are on ideas about software engineering; case studies and experiences
- Projects require applying these ideas to technical artifacts
- Technical aspects of project will not be taught in class
  - Explicit learning goal: learn new tools, languages, etc. on your own
  - TAs can help when needed
- Project requirements are often vague or under-specified (intentionally)
  - Feel free to ask for clarifications, but expect subjective responses

Recommended Readings

- Syllabus (posted on Canvas)
- 1968 NATO Software Engineering Conference