CS450 Modern Software Engineering

with Chiké Abuah

chike.abuah@wallawalla.edu

open to feedback about the course at any point

Syllabus

Welcome to CS450!

I am Chiké Abuah, your instructor for CS450 Modern Software Engineering.

Cheesecake!

I know stuff

Just in general :)

Also in particular regarding *computer science* and *software engineering*

PhD in cybersecurity/privacy & 5+ years of professional software engineering, including @ Amazon


MY CS PROFESSOR

Generative AI is a useful tool

ai.cs.wallawalla.edu

setup!

say hi and get a response back

Fall 2025 Course Context

- Administrative initiative to include GenAI tools and content in courses
- Particularly in CS since it comes from here
- GenAI is a very popular and established tool in 2025!

A collegial relationship is **a cooperative and respectful partnership between colleagues who are peers, sharing knowledge and working toward a common purpose with mutual trust, support, and a sense of shared responsibility**. It involves respectful, considerate conduct among colleagues, fostering a positive, productive, and supportive environment characterized by open communication, a sense of equality, and collective problem-solving.

Discussion: Attitudes around GenAI?

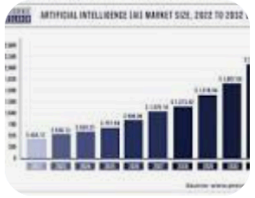
- The Good and the Bad
- Please remain collegial
- Introduce yourself and offer an opinion :)

Google

generative ai use in software statistics

AI Mode | All | Images | Videos | News | Short videos | Forums | More | Tools

AI Overview



Generative AI is being rapidly adopted, with some surveys showing over 70% of enterprises using it and 94% embracing it in software development. In software development, GenAI accelerates tasks like code generation (up to 47% faster), documentation (50% faster), and refinement (63% faster). Adoption is high, with 82% of users leveraging it in multiple SDLC phases, particularly for design, prototyping, and ideation, though lagging in maintenance. Market growth is substantial, with projections indicating billions in annual economic value and a potential 7% increase in global GDP.

Software Development Specifics

- **Accelerated Tasks:** Fortune Business Insights found that AI-powered tools can make writing new code 47% faster, documenting functionality 50% faster, and refining existing code 63% faster.
- **Broad SDLC Use:** According to one report, 82% of users are implementing GenAI across at least two phases of the Software Development Life Cycle (SDLC), with 26% using it across four or more phases.
- **Key SDLC Areas:** Design and prototyping (65% usage), ideation and requirement development (59% usage), and testing and QA (45% usage) are major areas of adoption.
- **Areas for Improvement:** The use of GenAI for software maintenance and updates is lower, with only 14% of users reporting involvement in this phase.

Economic potential of generative AI - McKinsey

Jun 14, 2023 — We identified 63 generative AI use cases spanning 16 business functions tha…

McKinsey & Company

100+ Generative AI Statistics [August 2025] - Master of Code Global

Generative AI Adoption Statistics: What the Data Says About 2025 Trends * Overall enterprise…

Master of Code Global

100+ AI Statistics Shaping Business in 2025 - Vena Solutions

May 27, 2025 — Workers' throughput of realistic daily tasks increased by 66% when using AI…

Vena Solutions

Show all

Why include GenAI in this course?

- I want you to be competitive
- You will always have value as long as your skills match the times you live in
- This requires us all to constantly learn new skills and evolve

Related Work

Prompt Problems: A New Programming Exercise for the Generative AI Era

Paul Denny
University of Auckland
Auckland, New Zealand
paul@cs.auckland.ac.nz

Juho Leinonen
University of Auckland
Auckland, New Zealand
juho.leinonen@auckland.ac.nz

James Prather
Abilene Christian University
Abilene, TX, USA
james.prather@acu.edu

Andrew Luxton-Reilly
University of Auckland
Auckland, New Zealand
a.luxton-reilly@auckland.ac.nz

Thezyrie Amarouche
University of Toronto Scarborough
Toronto, ON, Canada
thezyrie.amarouche@mail.utoronto.ca

Brett A. Becker
University College Dublin
Dublin, Ireland
brett.becker@ucd.ie

Brent N. Reeves
Abilene Christian University
Abilene, Texas, USA
brent.reeves@acu.edu

messages [19]. Many of these explorations also revealed that LLMs are not infallible and can produce solutions that do not align with best programming practice [5], struggle with longer and higher-level specifications [13], and cause students to become confused when reading code that they did not write themselves [16, 26]. Babe et al. even found that LLMs can mislead students, causing them to believe that their own prompts are more (or less) effective than they are in reality [2].

5.2.1 Exposure to new coding constructs. As our evaluation was conducted early in both courses, the generated code would sometimes contain features that were unfamiliar to students. For the most part, students commented positively on this aspect, and a theme emerged around how these problems would introduce students to new programming constructs and techniques. As one CS1 student

commented: *"These exercises introduced me to new functions... so this method of writing code could help increase my programming vocabulary"*. Similar feedback was provided by students in the CS2 course, even though they had prior programming experience: *"[Promptly] could find condensed ways to solve them using Python3's inbuilt functions, some even we have not been taught yet."*

One student commented on the value of seeing both the structure and syntax of the code generated by the LLM: *"The main benefit I gained ... was observing the logical structure of the programs that it created. In all three cases it used functions that I was previously unaware of, allowing me to gain an understanding of how they could be used and the correct syntax for implementing them."*

5.2.2 Enhancing computational thinking. Constructing prompts that clearly describe the steps needed to solve a problem draws on computational thinking skills. This was noted in the student reflections, as illustrated by the following quote from a CS2 student: *"I do think that writing prompts for code is a good way of developing analytical and problem-solving thinking and skills as it forces you to think through the steps needed to take the input through to the output".*

Several participants found that writing prompts helped them improve their problem-solving skills, as they could focus on the logic required rather than low-level syntax: *"I think while writing prompts for AI, we actually have to have a clear logic to break down the question and explain in plain words"* and *"Gaining experience from writing prompts can help me become a more effective programmer by allowing me to generate the necessary code while focusing solely on the logic of the code I want to create".*

The Evolution of SWE

- Not too long ago, there was...

No Compilers

No High-Level PLs

No Frameworks

No Google

Any others?

No Stack OverFlow

No Cloud

No static analysis

No code profilers

No IDEs

- GenAI is a productivity tool
- Think of yourself as the architect
- Verify Reasoning
- Reviewing Code
- Verifying facts

6 DISCUSSION

In contrast to other tools students use, such as compilers, learning to use LLMs presents unique challenges. For example, we do not need to worry about teaching students that compilers might sometimes make a mistake, and yet the literature documents the difficulty students have with compiler error messages [4, 19]. In contrast, identical input prompts to an LLM can produce different outputs, and these can sometimes be both syntactically and semantically incorrect. Deliberate exposure to the inconsistencies of LLMs, such as through practice with Prompt Problems, can serve to highlight the importance of a "critical eye" in evaluating generated code and may help to moderate potential over-reliance on these tools.

- Moderation against over-reliance
- Understanding limitations and use cases
- Understanding your value as a software engineer that is creative and can adapt to using any tool

Prompt Problems: A New Programming Exercise for the Generative AI Era

https://dl.acm.org/doi/pdf/10.1145/3626252.3630909

LLMs have flaws

- Garbage-In, Garbage-Out
- AI Hallucinations
- AI Slop

The ultimate copy machine

But lacking in originality (and creativity?)

I have to know what I'm looking for

I have to know the right questions to ask

I have to guide the AI

I have to review the AI

I have to know what "correct" looks like.

I have to make sure everything works and makes sense.

AI Overview

Training a foundational AI model can cost millions to hundreds of millions of dollars, with frontier models costing over a billion dollars by 2027, primarily due to the immense computational power and specialized hardware required. Smaller, specialized models for specific tasks can cost anywhere from thousands to tens of thousands of dollars, and fine-tuning pre-trained models can cost significantly less, ranging from tens of thousands to a few million dollars.

Will AIs replace us?

No

No, AI will not replace programmers entirely but will transform the role by automating routine tasks and enhancing productivity, requiring programmers to adapt by focusing on complex problem-solving, system design, and continuous learning to work collaboratively with AI tools. Developers who embrace AI as a powerful partner to enhance their skills will remain in demand, while those who don't adapt risk becoming obsolete.

How AI will change programming:

- **Automation of Repetitive Tasks:** AI-powered tools can handle mundane coding tasks, freeing up programmers to concentrate on more complex and creative aspects of software development.
- **Increased Productivity:** AI can help developers write code faster and create prototypes in a fraction of the time it would take to do it manually.
- **Shift in Skill Requirements:** The demand for basic coding skills will decrease, while the value of higher-level expertise in system design, business logic, and innovation will increase.
- **Emergence of New Roles:** Generative AI is expected to create new roles in software engineering and operations, necessitating a significant upskilling of the current workforce.

Why programmers won't be replaced:

- **Human Expertise is Still Needed:** Writing code requires domain-specific institutional knowledge, critical thinking, and intuition that AI currently lacks.
- **Focus on Higher-Value Work:** AI will allow programmers to focus on the more strategic and creative elements of software engineering that require human intelligence and understanding.
- **AI as a Tool, Not a Replacement:** AI tools are powerful aids that can augment a

Will AI Replace Programmers? Navigating the Future of Coding

Mar 22, 2024 — The short answer is no. The future of programming is not a battle between…

UC San Diego Extended Studies

Will AI Make Software Engineers Obsolete? Here's the Reality

Mar 4, 2025 — Tech. March 04, 2025. Artificial intelligence (AI) is rapidly transforming softwar…

Carnegie Mellon University

Is AI going to replace coding jobs by 2030, or will it just change how ...

Jun 6, 2025 — Will AI replace programmers in the next 10 years? Unlikely. Look, AI is a very powerful tool. It has saved…

Quora

Show all

r/computerscience • 2 yr. ago
[deleted]

Will AI replace software engineers?

Sorry, this post was deleted by the person who originally posted it.

5 · 453

trowgundam • 2y ago

The first rule of being a software engineer is to be adaptable. Computers and technology are always changing, usually at a breakneck pace. If you can't adapt and learn to utilize new technology, you are not suited to this field. No, AI isn't gonna replace us. Anyone that thinks that is delusional and has no understanding how technology works. AI will just be another tool in our toolbelt. What you can do is start experimenting with it now. Get CoPilot and start learning how you can integrate it with your workflow in order to make your job easier. Don't fear change, instead always be looking for how you can better exploit that change for your own benefit.

65

Instructor Responsibility

- I will do my best to focus the material on SWE tools and practices that are currently in popular use today.
- I will do my best to make the course fun and engaging.

Students' Responsibility

Positive Attitude (also see syllabus)

“Mistakes happen all the time, particularly in software development”

“Gerald Weinberg coined the term *egoless programming* in his book The Psychology of Computer Programming” (1971)

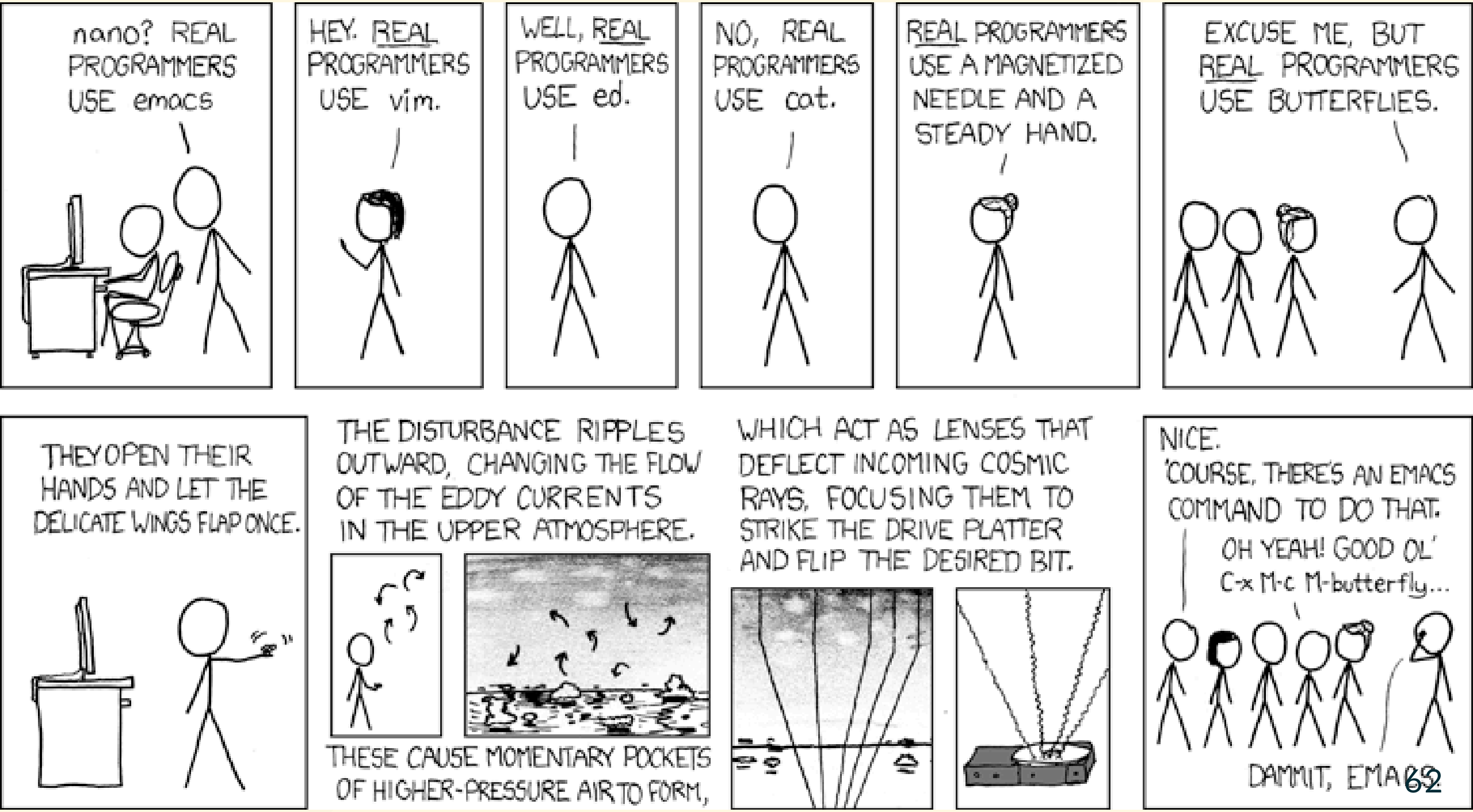
THE RULES OF EGOLESS PROGRAMMING (slightly modified)

- Understand and accept that you will make mistakes.
- “You are not your code.”
- “No matter how much “karate” you know, someone else will always know more.”
- “Treat people with respect, deference, and patience.”
- “Critique code instead of people.”


MY CODE WORKS
I DON'T KNOW WHY

TBH not that great but it's a start

Are you a real programmer?


nano? REAL PROGRAMMERS USE emacs
HEY. REAL PROGRAMMERS USE vim.
WELL, REAL PROGRAMMERS USE ed.
NO, REAL PROGRAMMERS USE cat.
REAL PROGRAMMERS USE A MAGNETIZED NEEDLE AND A STEADY HAND.
EXCUSE ME, BUT REAL PROGRAMMERS USE BUTTERFLIES.
THEY OPEN THEIR HANDS AND LET THE DELICATE WINGS FLAP ONCE.
"
THE DISTURBANCE RIPPLES OUTWARD, CHANGING THE FLOW OF THE EDDY CURRENTS IN THE UPPER ATMOSPHERE.
THESE CAUSE MOMENTARY POCKETS OF HIGHER-PRESSURE AIR TO FORM,
WHICH ACT AS LENSES THAT DEFLECT INCOMING COSMIC RAYS, FOCUSING THEM TO STRIKE THE DRIVE PLATTER AND FLIP THE DESIRED BIT.
NICE.
'COURSE, THERE'S AN EMACS COMMAND TO DO THAT.
OH YEAH! GOOD OL' C-x M-c M-butterfly...
DAMMIT, EMA

what are the most important things for modern software engineering students to know/have?

a strong foundation in core computer science principles

modern development practices like **AI-assisted workflows** and **DevOps**

interpersonal and strategic skills

A *real software engineer* is both a skilled programmer and an effective collaborator who understands the business context of their work

Core technical fundamentals

- Data structures and algorithms
- Programming languages
- Databases
- Object-Oriented Programming (OOP)

Modern development practices

Agile methodologies

Most development teams operate using agile methods. A strong grasp of practices like **Scrum and Kanban**, including sprint planning and code reviews, is essential for working effectively in a team environment.

AI literacy and AI-assisted development

AI is now an integral part of the development process. Students need to know how to effectively use tools like **GitHub Copilot** for code generation and how to **debug, review, and productionalize AI-generated code**. Familiarity with **AI model APIs and prompt engineering** is also highly valuable.

Cloud computing and DevOps

Modern applications are built and deployed in the cloud. Students must understand how **cloud platforms** (AWS, Azure, GCP) and DevOps practices (**CI/CD, containerization with Docker and Kubernetes**) work to deliver reliable and scalable software.

Version control

Expertise with version control systems, especially **Git and platforms like GitHub**, is non-negotiable. This enables effective **collaboration, code reviews** via pull requests, and the ability to track and manage changes over time.

Cybersecurity

Security must be a primary consideration, not an afterthought. Students should learn **secure coding practices, understand common vulnerabilities**, and be aware of cybersecurity risks.

Testing and debugging

Writing **automated tests (unit, integration, end-to-end)** is a fundamental skill. A developer's ability to **logically debug and troubleshoot** problems is a core measure of their technical depth.

In the age of AI, soft skills matter more than ever!

AI Overview

For modern software engineering students, the most important things to know include a strong foundation in technical skills, proficiency in contemporary tools and methodologies, and well-developed soft skills. Beyond writing code, a successful career requires continuous learning and adaptability in a dynamic field.

Soft skills and strategic thinking

- Communication
- Collaboration
- Leadership
- Critical Thinking
- Persuasion

- Charisma
- Presence
- Personality
- Aura


AURA

Communication and collaboration

Software engineering is a **team-based discipline**. You must be able to clearly **communicate technical concepts** to both technical and non-technical colleagues, **give and receive feedback** effectively, and **write clear documentation**.

Problem-solving and critical thinking

At its core, engineering is about solving problems. The ability to **logically break down complex issues**, analyze options, and **design optimal solutions** is more valuable than knowing any specific syntax.

Business-driven engineering

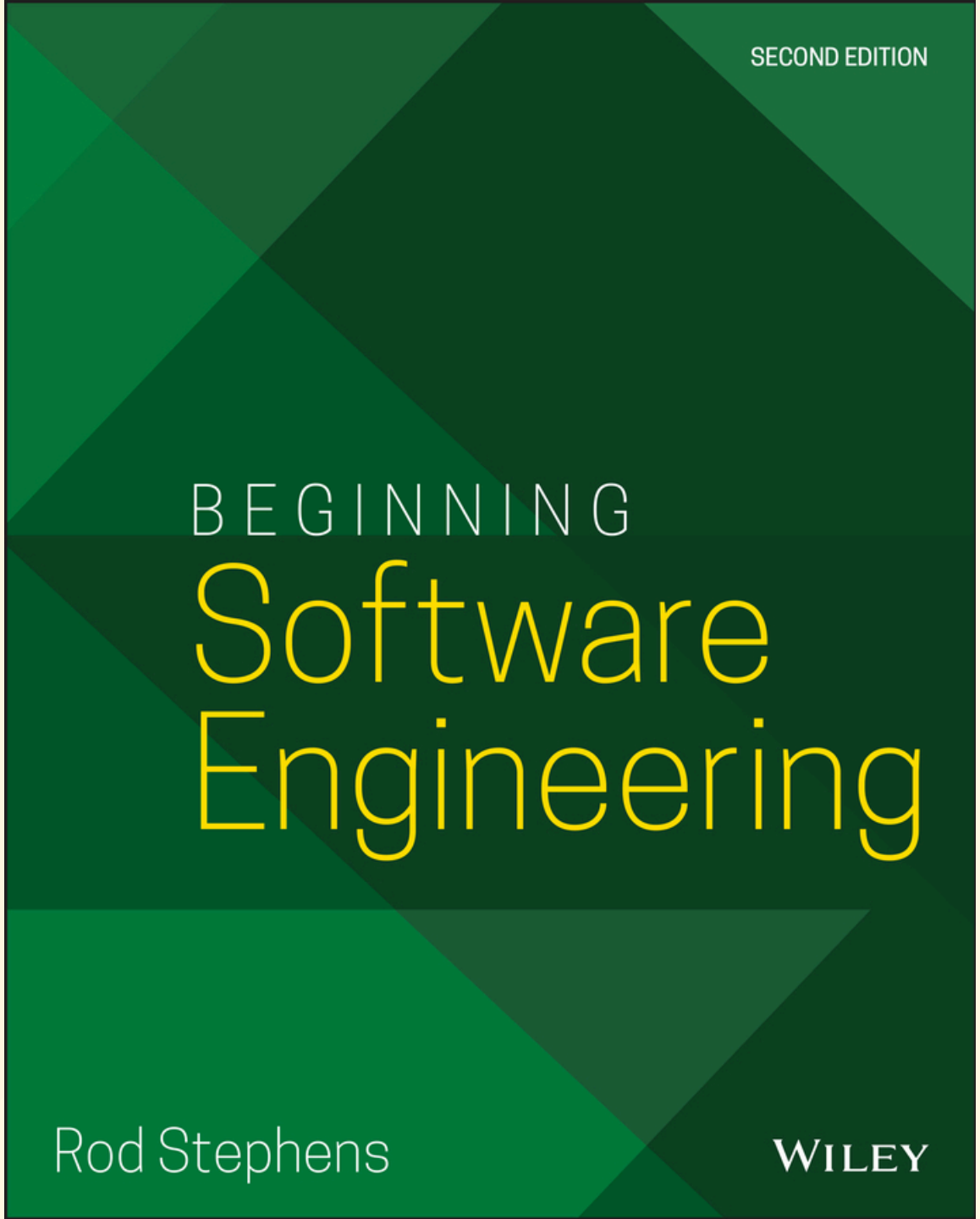
The best engineers understand the **business context and user needs** behind the features they build. They can manage **trade-offs between technical elegance and business goals**, and they can **articulate how their work delivers real value**.

Adaptability and continuous learning

The tech landscape **evolves** rapidly. A modern software engineer must be a lifelong learner who is open to new tools, languages, and methodologies to remain relevant and effective.

"Software and cathedrals are much the same. First we build them, then we pray."

—Samuel Redwine


SECOND EDITION
BEGINNING
Software
Engineering
Rod Stephens
WILEY

In principle, software engineering is a simple two-step process:

(1) Write a best-selling program, and then

(2) buy expensive toys with the profits.

Unfortunately, the first step can be rather difficult.

To produce great software, you need to handle a huge number of complicated tasks

Any one of which can fail and sink the entire project

Over the years people have developed a multitude of approaches, methodologies and techniques to help keep software project tasks on track

- Waterfall
- Agile
- Extreme Programming

First we will discuss those basic tasks that any successful software project must handle in some way

Then we will discuss how different approaches such as Waterfall and Agile handle those tasks.

High Level → Low Level

Software Engineering from the Moon

"There are two ways of constructing a software design. One way is to make it so simple that there are obviously no deficiencies. The other way is to make it so complicated that there are no obvious deficiencies. The first method is far more difficult."

—C.A.R. Hoare

Learning Objectives:

- The basic steps required for successful software engineering
- Ways in which software engineering differs from other kinds of engineering
- How fixing one bug can lead to others
- Why it is important to detect mistakes as early as possible

In many ways, software engineering is a lot like other kinds of engineering.

Software Engineering from Pluto

- make a plan
- follow that plan
- heroically overcome unexpected obstacles 💪

The following describes the steps you need to take to keep a software engineering project on track

These are more or less the same for any large project although there are some important differences

REQUIREMENTS GATHERING

DESIGN

DEVELOPMENT

TESTING

DEPLOYMENT

MAINTENANCE

Switch Gears

Project Ideas!

AI tools for the community

Can we develop GenAI powered tools to improve the lives of our campus community?

Suggestions!

Career Path Advisor

Create an AI system that analyzes student skills, interests, and market trends to suggest career paths and required skill development.

Smart Flashcard Generator

Build an application that converts study materials into effective flashcards with spaced repetition algorithms and difficulty adjustment.

AI Research Paper Summarizer

Develop a system that takes academic papers and generates concise summaries, key findings, and relevance ratings for specific research topics.

AI-Powered Study Music Generator

Develop a system that creates personalized background music for studying based on the subject matter, time of day, and user preferences.

AI Story Collaboration Platform

Build a web application where users can co-write stories with AI, with features for branching narratives, character consistency, and style matching.

Custom Course Chatbot

Build a subject-specific teaching assistant that can answer questions about course materials, provide study guidance, and offer practice problems with explanations.

Brainstorm

Any project ideas?

Project Polling

HW1

Lab!

https://github.com/abuach/cs450students