

CS 3300-A Introduction to Software Engineering

Lecture 01: Introduction; Project Overview; Prompt Engineering

Dr. Nimisha Roy ▶ nroy9@gatech.edu

Contents

- Course Overview
 - Introduction and Big Picture
- Course Logistics
 - Schedule, Class Organization, Projects, Team Formation, Policies
- Project Overview
- Prompt Engineering

Your Instruction Team

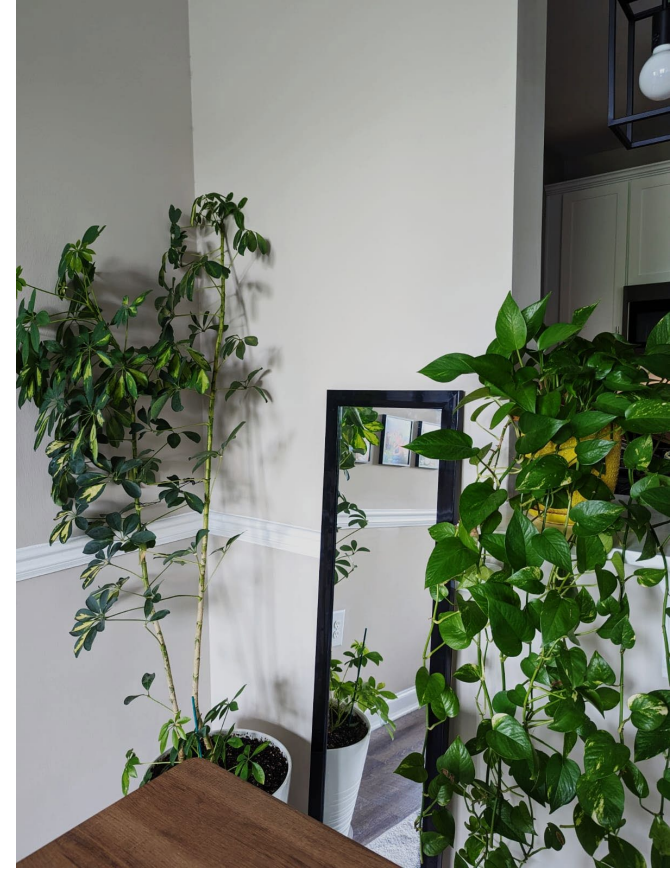


Instructor: Dr. Nimisha Roy

- 5th year teaching this course
- Lecturer in the College of Computing (SCI/OMSA/OMSCS)
- PhD in Computational Science & Engineering from Georgia Tech

Research Interests: Computing Education, Gen AI in education, Sustainability in CS education, Intelligent infrastructure, Software Development

Hobbies: singing (jamming), interior designing, caring for my plants



TA Team

- Nitin Chawla: Head TA
- Dax Delvira: GTA
- Nitya Yamjala: UTA



Course Overview

What is Software Engineering?

What do software engineers do?

- Software Engineering – The discipline allows systematic application of methods to build and manage high-quality programs/software.
- Software engineers are engineering or computer science professionals who combine engineering principles and programming.
 - End-to-end software life cycle planning and execution
 - Maintaining Disciplined Product Control
 - Use modern programming practices
 - Perform Continuous Validation and Improvement of the process

Industry Roles

- **Front-End Engineer/Developer:** Creating user interfaces and focus on the visual elements, fixing bugs and user experience (UX).
- **Back-End Engineer/Developer:** Focus on the core logic and performance of the application while taking scalability and integration into account.
- **Full Stack Engineer/Developer:** Handle both front-end and back-end, which means that they create a full application on their own.
- **QA Engineer/Analyst:** Writes the software that tests for quality. QA ensures the product or processes run as expected.
- **DevOps Engineer:** Build and maintain back-end software and distributed systems, such as database servers and application infrastructure.
- **Security Engineer:** Use their ethical hacking skills to test the security of software systems for vulnerabilities that need to be fixed.

What do software engineers do? **Current landscape - What has changed?**

- Introduction of AI tools like GPT and Copilot into software development processes.
 - AI tools are being integrated into coding, design, and testing phases, automating tasks that were traditionally manual.
- AI is reshaping roles and tasks within the software industry
 - emergence of new roles such as AI Integration Specialist, AI-assisted testing engineer, Prompt engineer, and Ethical AI Advisor, reflecting the need for expertise in navigating the complexities introduced by AI technologies.

The course CS 3300 this semester is a partial CURE (Course Undergraduate Research Experience)

We will be integrating Gen AI into the Software Engineering process throughout the semester.

How do you benefit from it?

- CURES are great to have an experience with within your curriculum
- Can help you explore/reinforce your research interests
- Helps you gain practical skills gained from using AI in real-world software projects.
- Helps you get a taste of software engineering industry relevant skills of the future – prepares you for current and future industry demands.

What is Software Engineering?

Why is it Important?



What is this ?

- ☐ 4th of July fireworks
- ☐ Flare gun in action
- ☒ Explosion of Ariane 5 rocket due to Software errors

What is Software Engineering?

Why is it Important?



Your PC ran into a problem and needs to restart. We're just collecting some error info, and then we'll restart for you.

20% complete



For more information about this issue and possible fixes, visit <https://www.windows.com/stopcode>

What about This?

CrowdStrike's Falcon Sensor
Software Flawed Update
\$10- \$15 billion economic
damage

What is Software Engineering?

Why is it Important?

Why is it so difficult to build good software??

Topic of this course and why Software Engineering is an important course in Computer Science

CRASH



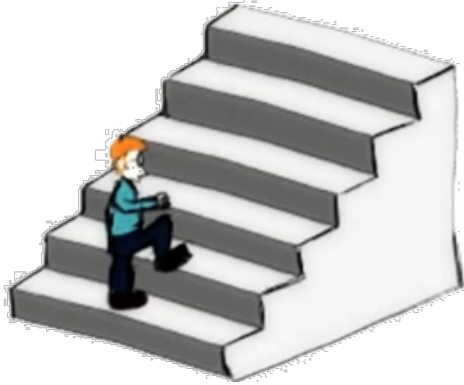
What are attributes of a good software?

- Functionality
 - The software should deliver the required functionality and performance to the user
- Usability
 - Should be easy to use. Not unnecessarily complex
- Maintainability
 - Software must be evolvable to meet changing needs
- Dependability
 - Software must be trustworthy (reliability, security, and safety)
- Efficiency/Productivity
 - Time & Cost effective

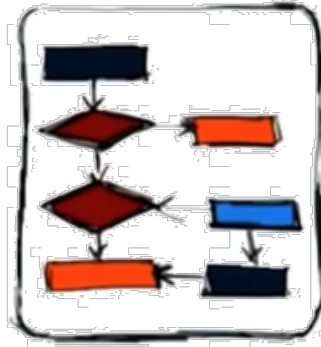
Some questions

- What is the largest software system on which you have worked?
- How many LOCs/day were you producing?
- How many LOCs/day professional software engineers produce?
 < 25? 25-50? 50-100? 100-1000? > 1000?
- But what are they doing with the rest of their time?
- How do large systems get built?
- What process should be followed?
 - No one size fits all
 - We will see several

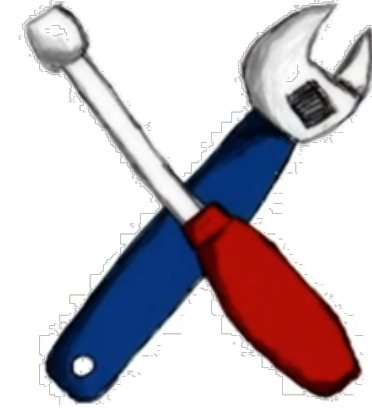
Software Engineering



Methodologies



Techniques



Tools



To build software
of high quality



That works



And fits into
Budget

History

The 60s

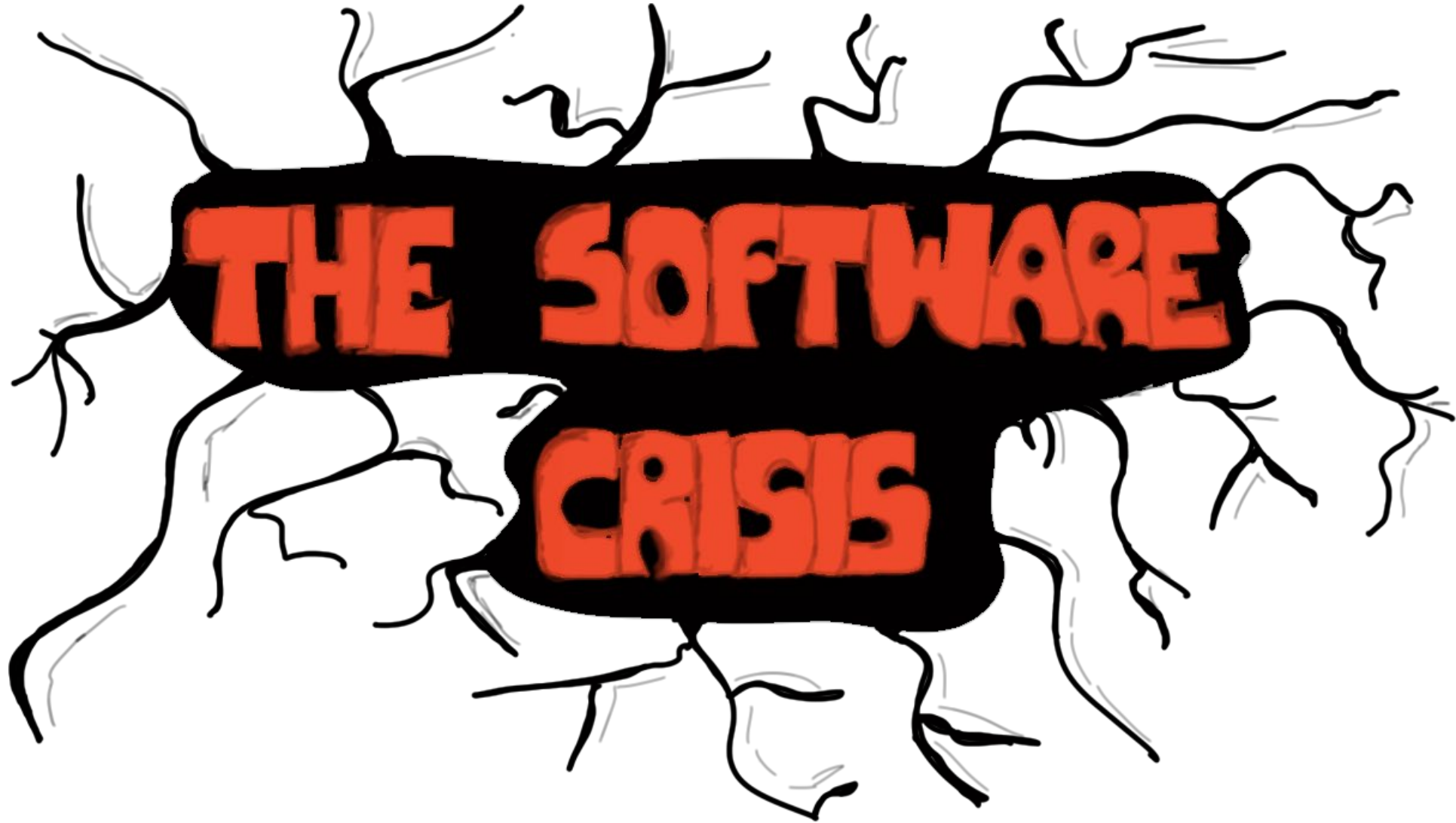


Man on Moon



Polaroid

History

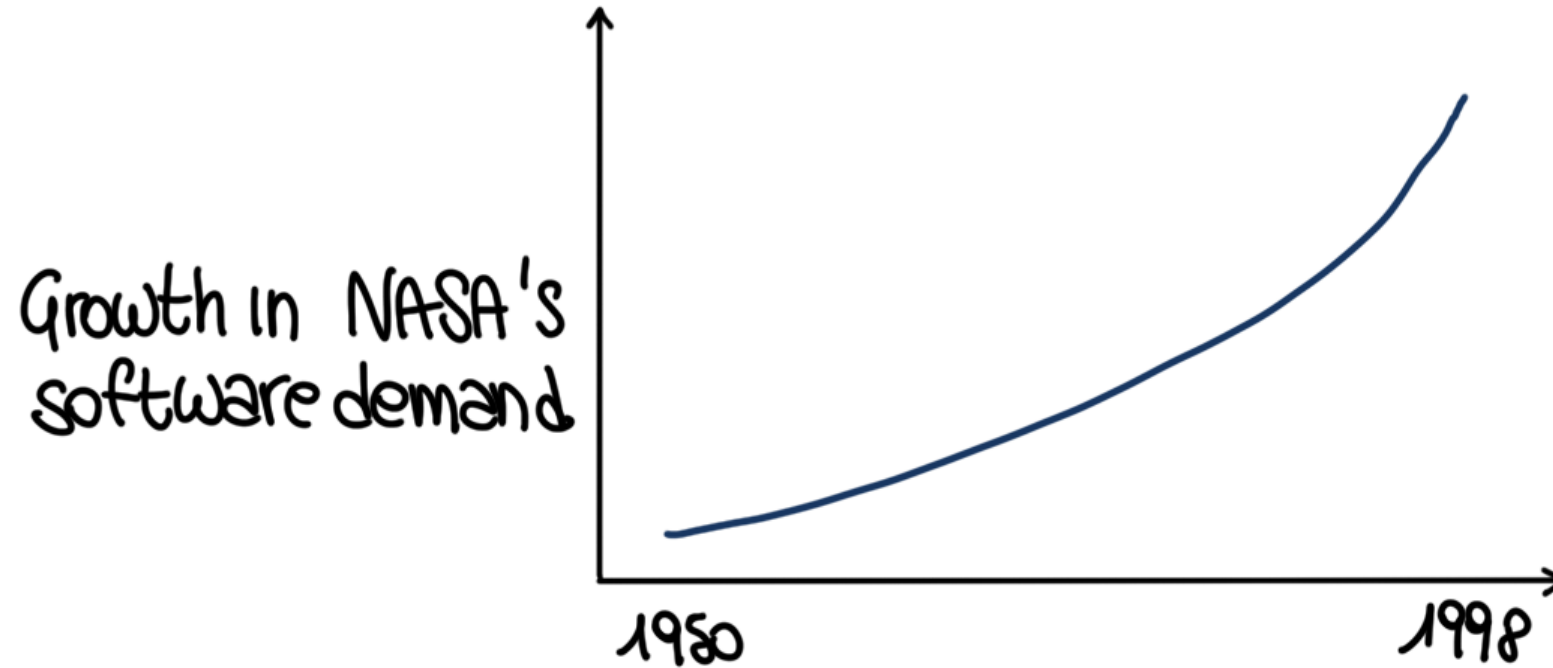


https://en.wikipedia.org/wiki/Software_crisis

History

Reason 1: Rising Demand for Software

HW \Rightarrow SW



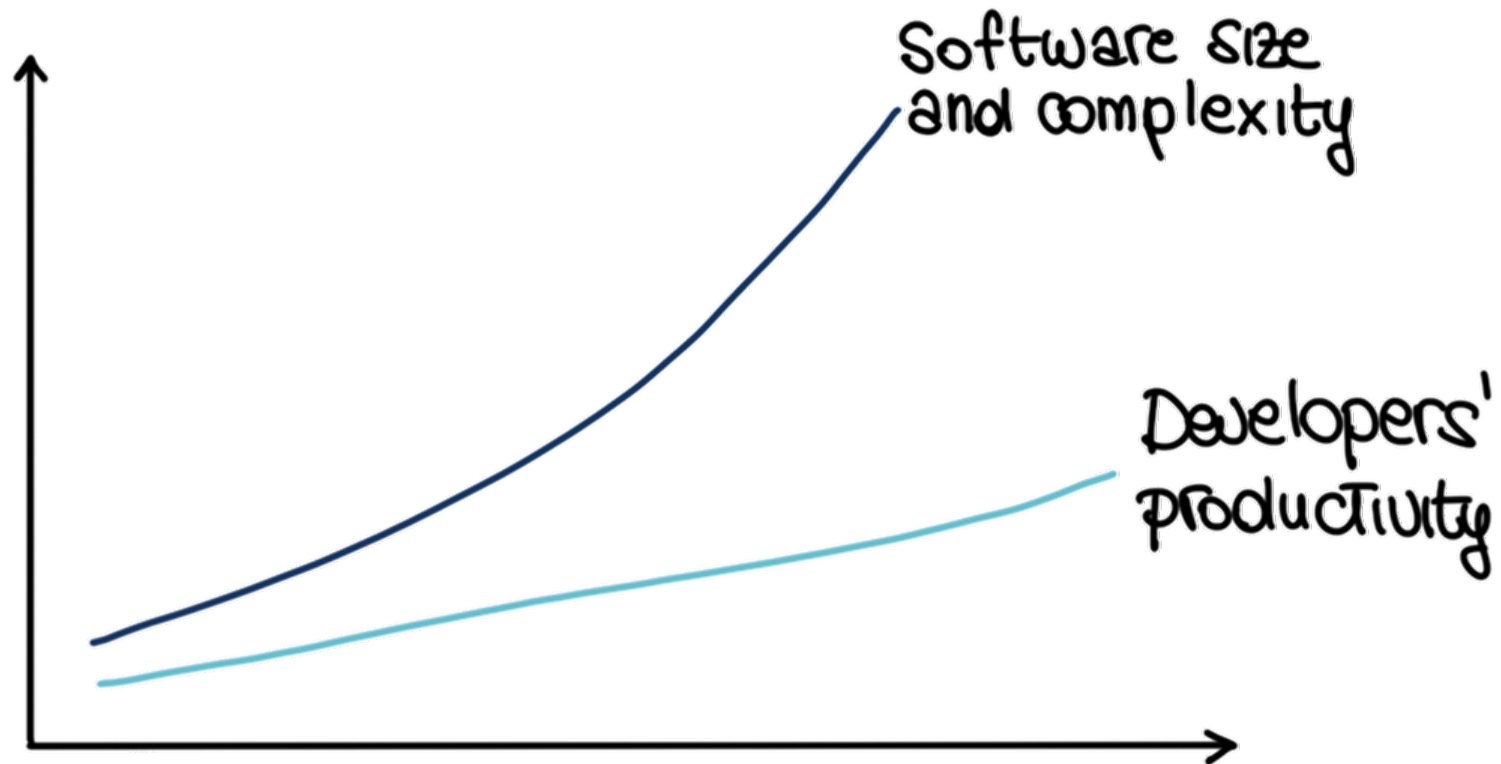
History

Reason 2: Increasing product complexity

SIZE	EXAMPLE	
10^2 LOC	Class exercise	} Programming effort
10^3 LOC	Small project	
10^4 LOC	Term project	
10^5 LOC	Word processor	} Software engineering effort
10^6 LOC	Operating system	
10^7 LOC	Distributed system	
...	...	

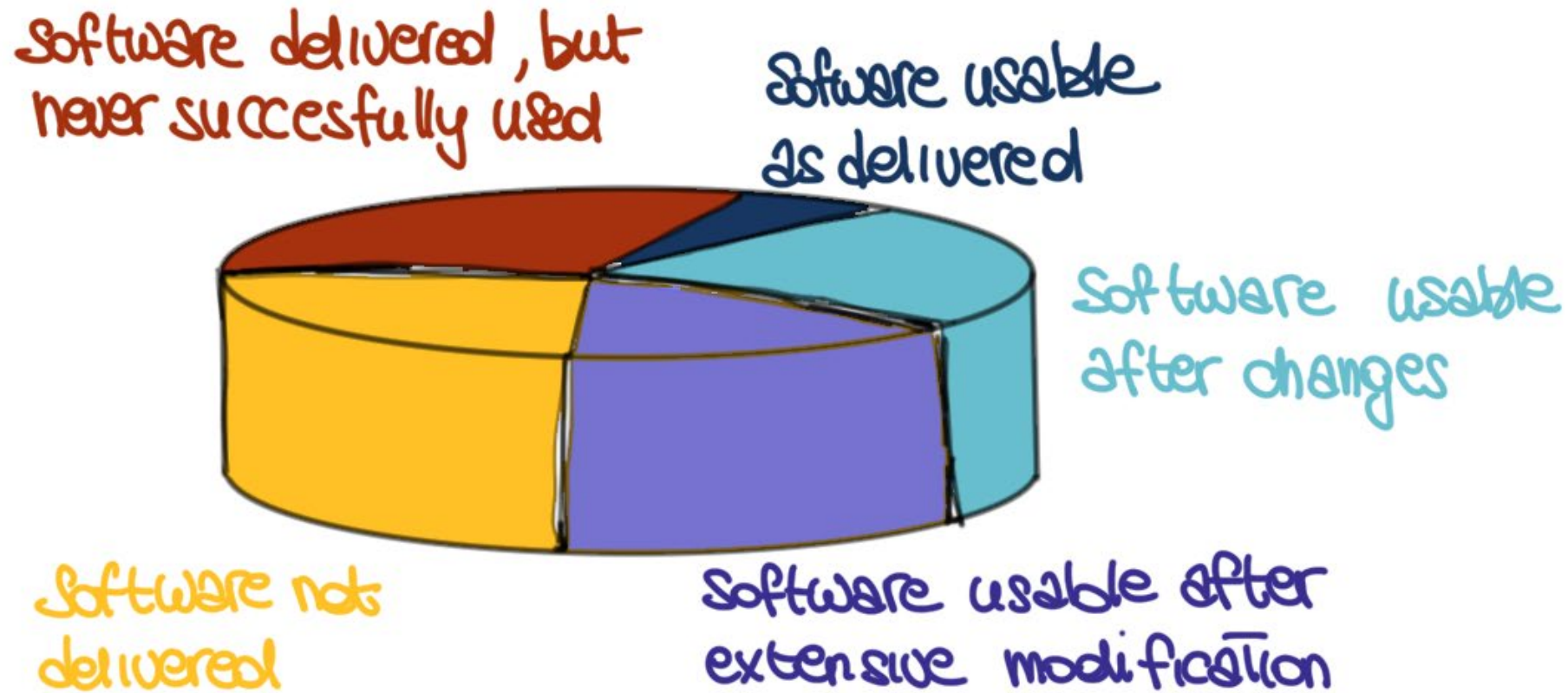
History

Reason 3: Developers Productivity Growth



History

Study of 9 Development Contracts totalling \$7 million (Davis, 1990)



History

Birth of Software Engineering



History

Birth of Software Engineering



Link to proceedings:

<http://homepages.cs.ncl.ac.uk/brian.randell/NATO/>

What are common causes of SW failures?

- No standard procedures for development
- Inadequate understanding of requirements
- Sheer complexity of software (e.g., concurrency, distribution)
- Size of project (too large for a single manager)
- Difficult to match technical knowledge of staff with project needs
- Poor design/implementation/testing methodology
- Requirements change during project
- Poor documentation
- Force fitting software components to applications
- Changing/reusing code without understanding it
- Poor management: lack of communication, poor cost/schedule estimates
- Unrealistic expectations
- Lack of measurement
- Lack of teamwork
- Performance differences among staff
- ...

Software importance today

- More and more systems are software-controlled
- The economies of ALL developed nations are dependent on software
- Expenditure on software represents a significant fraction of GNP in all developed countries
- AI technologies integrated into every stage of the software development lifecycle.
- AI-Driven Innovations: Examples include AI-assisted code completion, code generation, test case generation, design diagram generation, predictive analytics in software design, etc.
- Impact on Productivity and Efficiency: AI tools streamline processes, reduce development time, and enhance software quality and accuracy.

What are the key challenges facing SE?

- How can we build high-quality systems?
- How can we do it in a reasonable time?
- How can we do it at a reasonable cost?

- Effective Integration of AI: Overcoming the complexity of integrating AI tools into established software engineering workflows.
- Educational Adaptation: Updating curricula to include AI-focused software engineering practices to prepare graduates for the evolving industry demands.
- Quality Assurance with AI: Ensuring that AI-generated code or designs maintain high quality and adhere to industry standards.

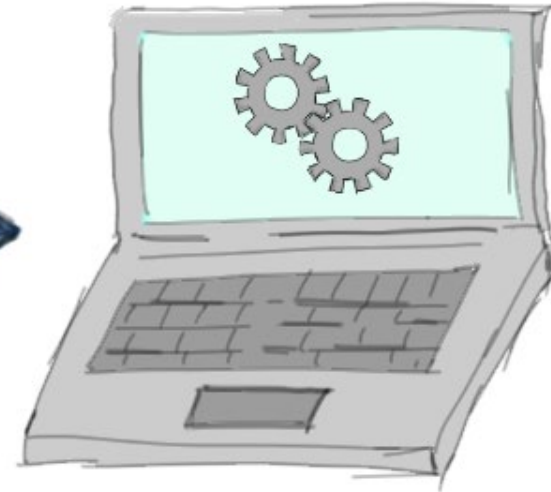
Software Process, Phases, Tools of the Trade

Abstract Idea

System Implementation



Very Complex
Process



Software Process

- Systematic
- Formal

Software Process, Phases, Tools of the Trade

Software Phases



Requirements

Engineering



Design



Implementation



Maintenance



Verification &
Validation

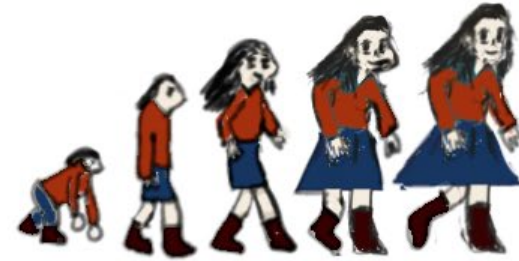
Software Process, Phases, Tools of the Trade

Software Process [Methodologies]

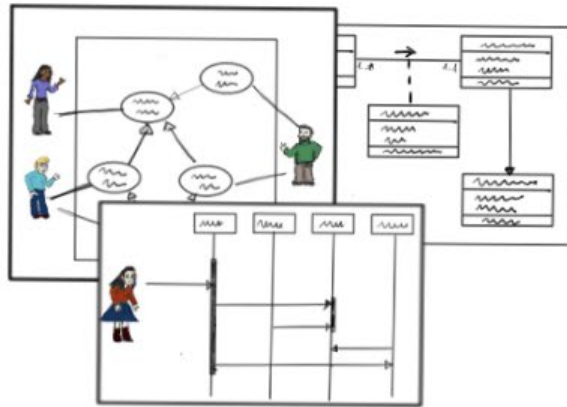
Waterfall



Evolutionary
Prototyping



RUP/USP



Agile

Course Logistics

Class organization

- Class Website and Canvas
- Ed Discussion
- In-class Lectures
- Attendance is required
- Tools of the Trade Demonstrations – Bring Laptops to class.
- Team-work
- Some research of gen-AI tools

Refer to:

Canvas

For anything (syllabus, lectures, helpful links, logistics etc.) related to this class

Ed Discussion

For any communication with the instruction team. No Emails please!

Introduce yourselves on Ed 😊

Grading Distribution

- Assignments (Individual) 20%
- Project (group) 65%
- Team Management/ CATME Peer Review (Individual) 6%
- Quizzes/Lecture Activities (Individual) 6%
- Survey Submissions (Individual) 3%

Tools of the Trade Lectures

Demonstration in Lectures (lets check the living schedule) – Bring your Laptops

- Version Control System (Look at course prerequisite knowledge: You should know basic Git and code review from 2340)
- IDEs
- Front-End Development
- Back-End Development
- Google Cloud Platform
- Basics of how to use AI tools

Information about projects

- 1 project
- You will choose the project. It can be web- or mobile-app. You will use AI tools for implementation, design, and testing.
- Team based
 - Different teams
 - Grades will be adjusted based on peer review
- Tools/environments
 - Deployment on GCP is mandatory. Springboot is encouraged but tech stack is upto you. AI tools (Lucidchart, copilot, GPT demoed)

AI Tool Integration

- You will choose and experiment with AI tools for implementation, design, and testing.
- You will document your findings according to the metrics provided in the assignments
- Focus is on the basic implementation of software features in projects and more points allocated to your AI integration/reflection aspects
- We are providing primer videos on how to use AI for:
 - Design – lucid chart video on the website (it's free). Many design AI tools are free.
 - Implementation – GPT 4o API credits video on website
 - GPT 4o API credits will be provided to each team - \$40 credit
 - Implementation – copilot video on the website (it is free)
 - Testing – Video on the website
- Each individual will be allocated \$50 (+ possible \$50) worth of Google Cloud credits
- I will be covering basics of prompt engineering this class

Team Formation

- Team Formation Survey:
 - To be completed by *everyone*
 - **Released yesterday; Due: 5/16 at 11:59 PM**
- Team Preferences:
 - Groups of 2+ students: Sign up using Canvas Groups
 - Use the *existing* available group
 - ***Do NOT create your own group***
 - No preferences:
 - Will be matched via CATME
 - Everyone needs to fill out CATME

**Teams will be
formed and
mentors by
5/17.**

Summary

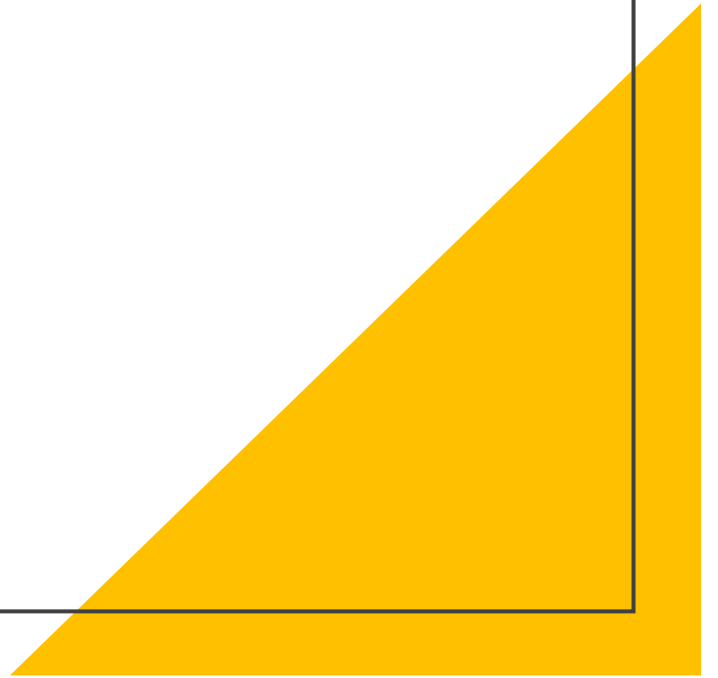
Summary

- SE important/critical discipline
 - Concerned with cost-effective software development (all aspects!)
 - Based on a systematic approach that uses appropriate tools and techniques, follows a process, and operates under specific development constraints
- Goal of SE is to deliver high-quality products that provide the expected functionality, meet projected time estimates, and have a reasonable cost



5 min break

Project Description



Topic of your choice: Web or Mobile Application

Required Features

- Define **six MMFs** that are:
 - User-facing
 - Independently valuable
- **Valid MMF examples:**
 - "Users can manage shared task boards with real-time collaboration and version control."
 - "Users can upload documents and receive editable AI-generated summaries."
 - "The system recommends personalized goals based on user activity using a trained model."
 - "Users receive contextual grammar feedback on short-form responses using LLMs."
 - "An analytics dashboard shows weekly engagement trends using charts and statistics."
 - "Team leads can generate auto-scored quiz questions using AI and assign them to users."

Required Features

- **Invalid MMF examples:**

- "Login/logout functionality" – **Too basic**
- "Use Firebase for authentication" – **Not a user-facing feature**
- "Deploy to GCP" – **An operational requirement, not a marketable feature**
- "Build a landing page" – **Not independently valuable**

- **Example User Flow (for MMF):**

MMF: "Users can upload documents and receive editable AI-generated summaries."

User Flow:

1. User logs into the application.
2. User navigates to the "Summarize" section.
3. User uploads a PDF or Word document.
4. AI processes the file and generates a summary.
5. Summary appears in a rich-text editor.
6. User can modify, copy, or export the summary.

Technology Suggestions

GOOGLE CLOUD PLATFORM*

Offers services for compute, storage, networking, big data, machine learning and the internet of things (IoT), as well as cloud management, security and developer tools.

BACKEND SERVER ARCHITECTURE

- **SpringBoot*** – Framework for developing the backend server architecture and to run the server application easily
- **Maven*** - Used for dependency injection and management as a Java package manager. Provided as a part of central repository in Spring
- Mockito – Test stub method for unit testing SpringBoot application
- VS Code/IntelliJ - IDE

DATABASE

- GCP DataStore
- MySQL
- Google Storage
- MongoDB

* **Must use**

* **Optional but Will be demoed in class**

Technology Suggestions

FRONTEND

- Thymeleaf - Java template engine for both web and standalone environments. Set of Spring integrations.
- Leaflet - Open-source JavaScript library for creating interactive maps
- **jQuery*** - JS Library for implementing HTML document traversal and manipulation, event handling, animation, and Ajax.
- Bootstrap – CSS framework to create modern websites and web apps
- React.JS - open-source JavaScript library used for building user interfaces for single-page applications

VERSION CONTROL

GitHub*

AI Tools for Implementation, Design, and Testing

GPT 4o API*

Copilot, Codeium, Cody

*** Must use**

*** Optional but Will be demoed in class**

Some Considerations/Functionalities

Password Security

- Salting & Hashing Passwords
- JWT – Validity of Tokens (optional)

Database

- Google DataStore is recommended.
- Google storage has some problem: 2 people simultaneous logging in may cause overwrite

Webjars

Use webjars - client-side web libraries packaged into JAR (Java Archive) files to easily manage web dependencies

Testing

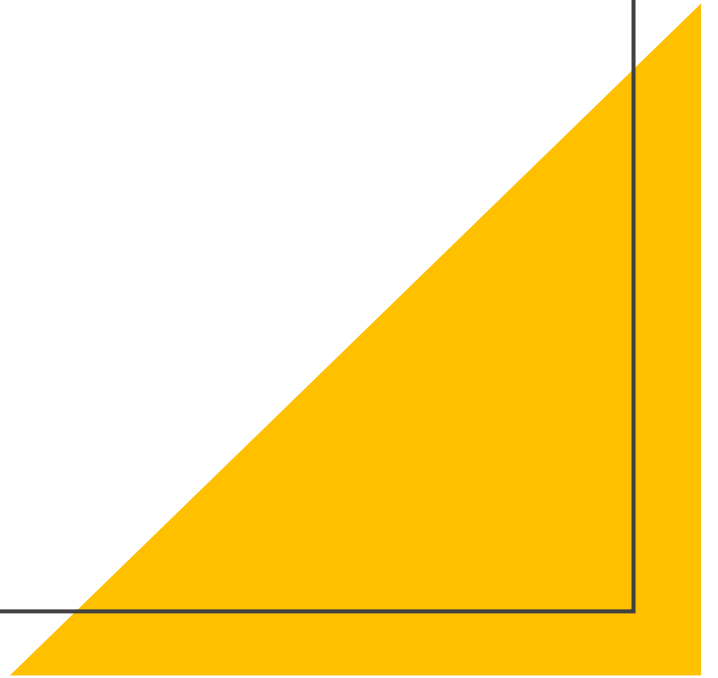
Important. Modules Included in Google App Engine.

Blackbox and Whitebox testing*

*** Must use**

*** Optional but Will be demoed in class**

Prompt Engineering





Introduction to Prompt Engineering in Software Engineering

- **Prompts** involve instructions and context passed to a language model to achieve a desired task.
- **Prompt engineering** is the practice of developing and optimizing prompts to efficiently use large language models (LLMs) for a variety of applications.
- **Importance in Software Engineering:** Accurate prompts can significantly enhance the utility of AI in code generation, debugging, requirement elicitation, and documentation, making them indispensable in modern software development.

Fundamentals of Prompt Engineering

Zero-shot Prompting: Instructing the AI without any context-specific training; based purely on general model training.

Few-shot Prompting: Providing examples to AI to specify the context or format of the desired answer; helps the AI understand domain-specific tasks.

Chain-of-Thought Prompting: Constructing prompts that simulate a logical reasoning path for the AI, helping it tackle complex software engineering problems step-by-step.

Prompt Chaining: Using the output of one prompt as the input for another, effectively chaining prompts together to achieve more complex tasks

Zero Shot Prompting

- Zero-shot prompting involves instructing the model to perform a task without providing examples, background, or additional information.
- Let's demonstrate zero-shot prompting via an example:

Query: Generate a python code snippet to get the average grade of students in homework 1

Output: Sure, here's a Python code snippet that calculates the average grade of students for homework 1. This assumes you have a list or array of grades for each student.

```
# Sample list of grades for homework 1
grades_homework1 = [85, 92, 78, 90, 88, 84, 70, 95, 82, 87]
# Calculate the average grade
average_grade = sum(grades_homework1) / len(grades_homework1)
print(f"The average grade for homework 1 is: {average_grade}")
```

Few Shot Prompting

- Few-shot prompting involves providing examples in prompts to guide the model towards better performance.
- **Example from Brown et al. 2020:**

Query: Correctly use a new word in a sentence.

Example 1: A "whatpu" is a small, furry animal native to Tanzania. An example of a sentence that uses the word whatpu is: "We were traveling in Africa, and we saw these very cute whatpus."

Example 2: To do a "farduddle" means to jump up and down really fast. What would be an example of a sentence that uses the word farduddle:

Output: When we won the game, we all started to farduddle in celebration.

Few Shot Prompting: Another Example

- **Query:** Generate a function that can calculate and return a response status for an API call.
- **Example 1:** A function `getResponseStatus` returns "Success" if an API call response status is 200, and "Error" otherwise. An example of this function is

```
def getResponseStatus(response_code):  
    return "Success" if response_code == 200 else "Error"
```

- **Example 2:** The "checkApiCall" function is designed to make an API call and use `getResponseStatus` to determine and print the status.
- **Output:** Here's a Python code snippet that calculates and returns the response status for an API call:

Few Shot Prompting: Another Example

- **Output:** Here's a Python code snippet that calculates and returns the response status for an API call:

```
import requests

def getResponseStatus(response):
    if response.status_code == 200:
        return "Success"
    elif response.status_code == 404:
        return "Not Found"
    else:
        return "Error"

def checkApiCall(url):
    try:
        response = requests.get(url)
        status = getResponseStatus(response)
        return f"API Call Status: {status}"
    except requests.exceptions.RequestException as e:
        return f"Failed to make API call: {str(e)}"

# Example usage
url = "https://api.example.com/data"
result = checkApiCall(url)
print(result)
```



Chain of Thought (CoT) Prompting

- Prompting can be further improved by instructing the model to reason about the task when responding
- This is very useful for tasks that requiring reasoning
- CoT increases the likelihood of correct answers and makes the model's reasoning process transparent and easier to follow
- You can combine it with few-shot prompting to get better results
- You can also do zero-shot CoT where exemplars are not available

Chain of Thought (CoT) Prompt Example

Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27.



Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now? Explain the reasoning behind your answer.

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5+6=11$. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had $23-20=3$. They bought 6 more apples, so they have $3+6=9$. The answer is 9.



Prompt Chaining

- Prompt Chaining is a technique where the output of one prompt serves as the input or context for the next, creating a sequence of interlinked prompts that guide the AI towards a complex goal
- Enables the handling of multi-step problems within AI systems by maintaining a contextual flow, improving accuracy in tasks requiring deep contextual understanding
- This method allows for more nuanced interactions with AI, closely mimicking human-like reasoning and procedural task management
- **Example:**
 - **Initial Prompt:** "What is the first step in making a cake?"
 - **AI Output/Next Prompt:** "Gather all necessary ingredients."
 - **Continuation:** "What is the next step after gathering ingredients?"
 - **AI Output/Next Prompt:** "Preheat the oven to the appropriate temperature."
 - **Final Output:** "Mix the ingredients and place them in the oven."

Prompt Chaining Example

Initial Prompt:

- **Query:** "Generate a Python function to parse and summarize JSON data from a project management tool's API."
- **Chain Prompt 1:**

Output from Initial Prompt:

```
def parse_json(json_data):  
    """Extracts project details from JSON data."""  
    projects = json.loads(json_data)  
    summary = {project["id"]: project["name"] for project in projects}  
    return summary
```

Prompt Chaining Example

- **Follow-up Query:** "How would you use this function to display a formatted list of project names and their statuses?"
- **Chain Prompt 2:**
- **Output from Chain Prompt 1:**

```
def display_projects(json_data):  
    projects = parse_json(json_data)  
    for project_id, project_name in projects.items():  
        print(f"Project ID: {project_id}, Project Name: {project_name}")  
  
# Example JSON Data  
json_data = '''  
[  
    {"id": 1, "name": "Project Alpha", "status": "Active"},  
    {"id": 2, "name": "Project Beta", "status": "Completed"}  
]  
...  
  
display_projects(json_data)
```

Prompt Chaining Example

- **Follow-up Query:** "Now, create a function that updates the status of a specific project by project ID in the project list."
- **Chain Prompt 3:**
- **Output from Chain Prompt 2:**

```
def update_project_status(projects_json, project_id, new_status):  
    projects = json.loads(projects_json)  
    for project in projects:  
        if project['id'] == project_id:  
            project['status'] = new_status  
            break  
    return json.dumps(projects)  
  
# Updated usage  
updated_json = update_project_status(json_data, 1, 'Completed')  
print(updated_json)
```

Techniques for Effective Prompt Engineering

- **Clarity and Specificity:** Ensuring prompts are clear and detailed reduces ambiguity and improves AI accuracy. Example: Specify software frameworks, expected outputs, error handling needs.
- **Incorporation of Keywords:** Embedding technical terms specific to software development can help focus AI's responses on the relevant area. Example: "API", "async function", "state management".
- **Iterative Refinement:** Adjusting and evolving prompts based on the quality of AI's responses to hone in on the optimal solution.

Practical Applications of Prompt Engineering in SE

- **Code Generation/Completion:** Crafting prompts that describe the exact functionality and parameters of the code to be generated, potentially including compliance with coding standards and performance metrics.
- **Automated Testing:** Automatically generating unit tests, integration tests, and system tests that ensure all aspects of an application are thoroughly validated.
- **Debugging Assistance:** Formulating prompts that help identify bugs by describing the malfunction and the expected behavior, thus guiding AI to propose plausible fixes.
- **Requirements Specification:** Automating the generation of requirement specifications documents from meeting notes or user interviews.
- **UML Diagram Generation for Software Design:** creating class diagrams, sequence diagrams, or state diagrams from project documentation, meeting transcripts, and architect explanations

Navigating Challenges in Prompt Engineering

- Common Pitfalls: Overly broad prompts leading to generic outputs, under-specification causing irrelevant solutions.
- Best Practices:
 - Always start with a clear, concise objective; incorporate feedback loops to refine prompts with more detailed prompts; use structured prompt templates where possible.
 - Don't believe in all outputs. Use your coding knowledge to analyze and adapt the output.

To dos:

- Introduce yourself on Ed
- Submit team formation survey on CATME (due 5/16)
- Go through the class website to learn about the schedule
- Go through Project assignments on Canvas to learn the course expectation
- Go through the 5 AI videos on the class website in the readings column for Day 1
- 1 survey out on the class website for extra credit. It is called AI integration in software engineering. Take 10 minutes to complete the survey now and submit the extra credit assignment. (due 5/18)