17-423/723: Software System Design

Designing with AI

Feb 25, 2026

Leaning Goals

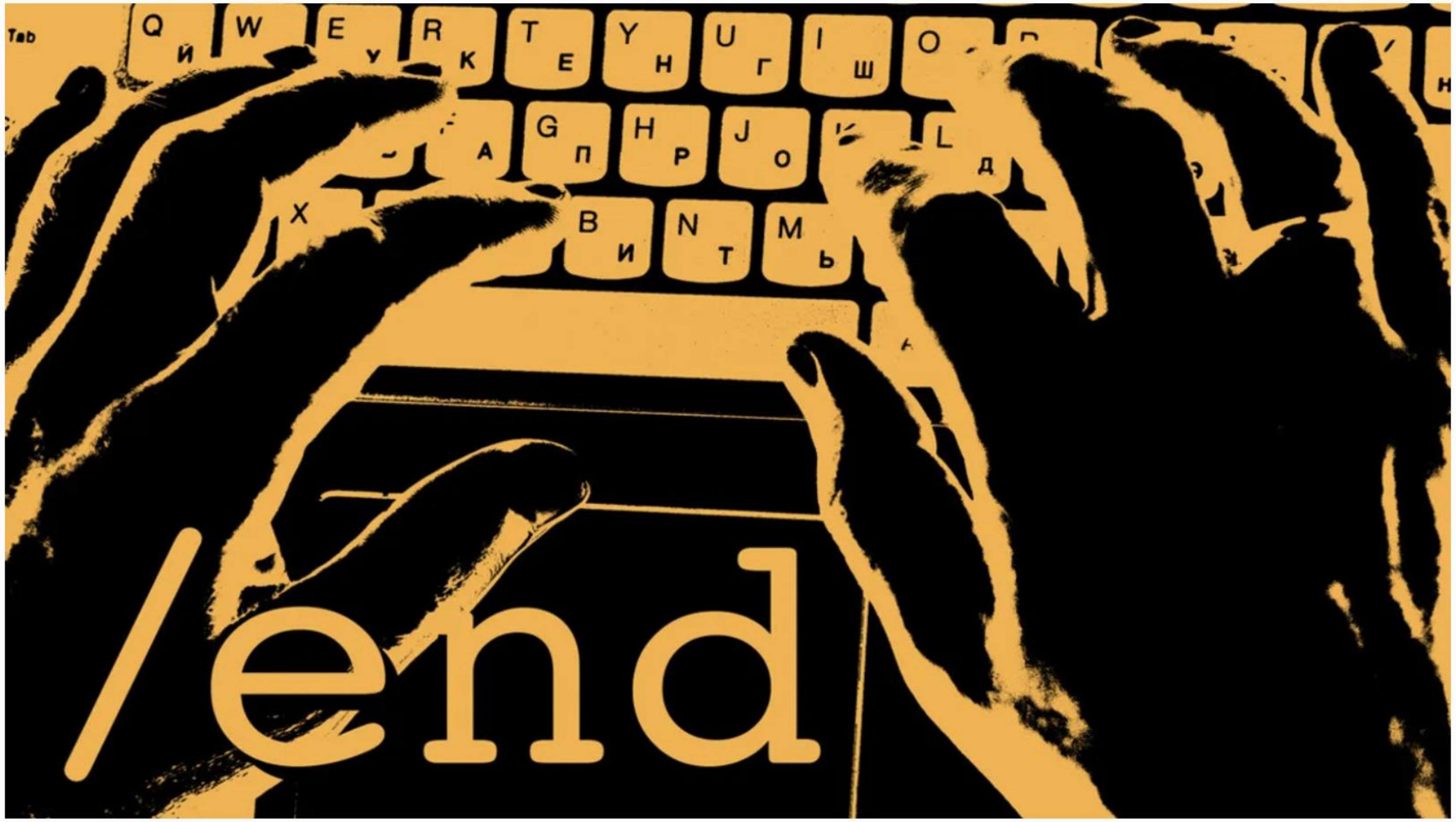
- Describe ways in which generative AI can assist with or automate design tasks
- Apply different prompting strategies to improve the effectiveness of gen AI on design tasks
- Critically evaluate the output of gen AI on design tasks to identify potential flaws and unsubstantiated claims

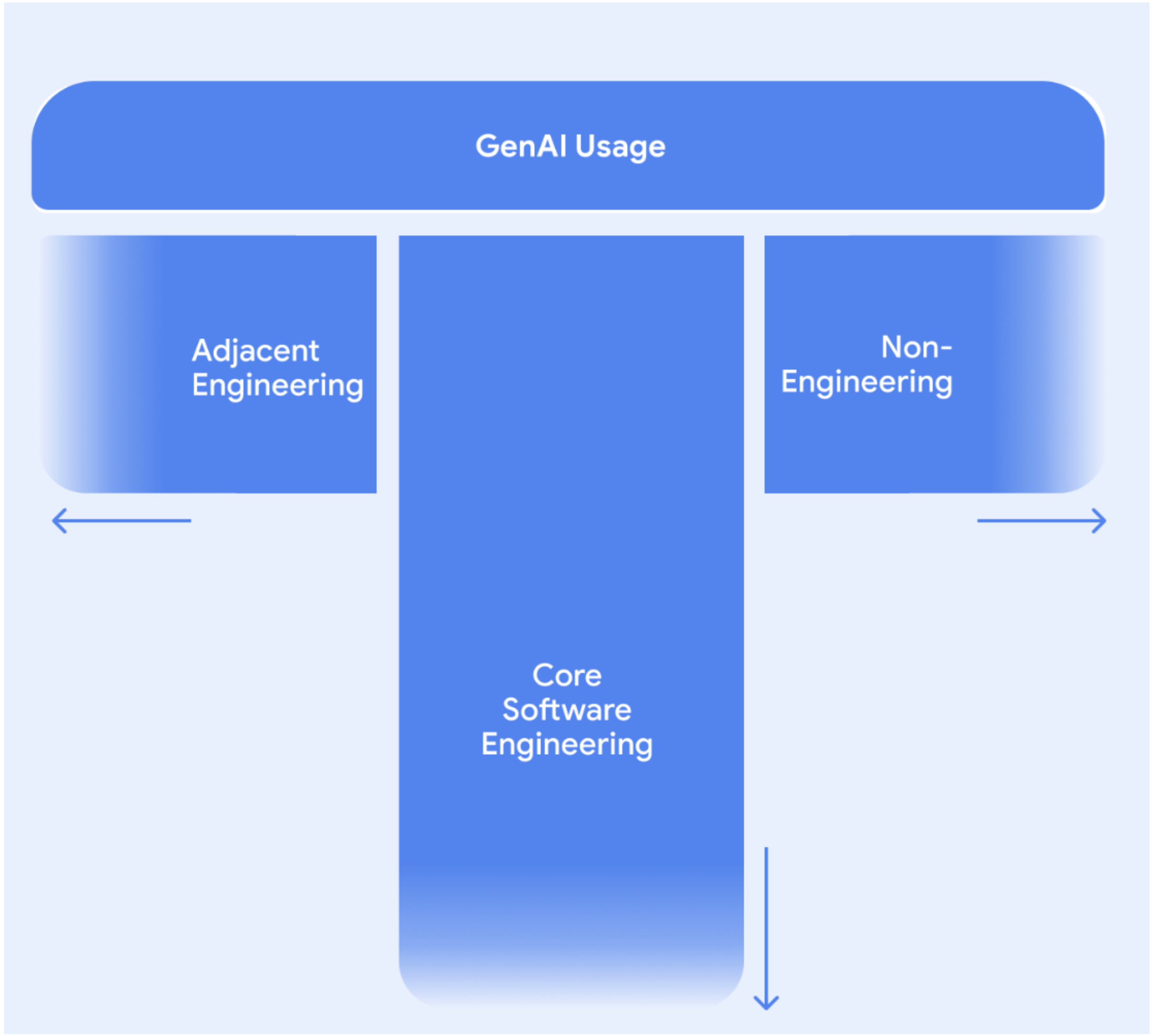
The End of Programming

The end of classical computer science is coming, and most of us are dinosaurs waiting for the meteor to hit.

By Matt Welsh

Posted Jan 1 2023





“1) coding and testing proficiency…2) risk assessment for production readiness…3) **good system design** (e.g., possessing a deep understanding of existing and alternative system architectures, design & system constraints, and carefully weighing the potential benefits and drawbacks of multiple design solutions to meet requirements).”

What do professional software developers need to know to succeed in an age of Artificial Intelligence? (2025)

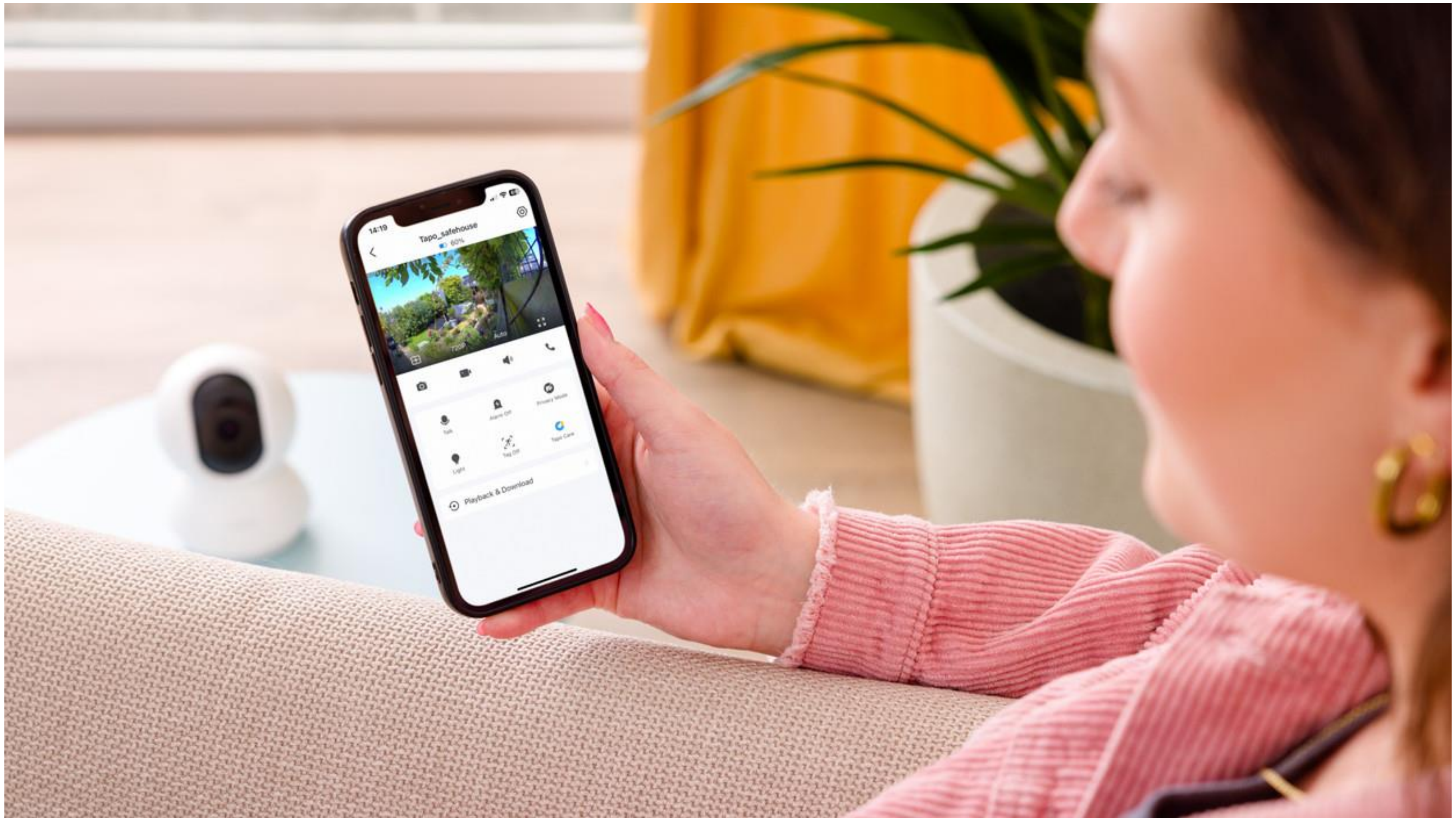
Q. Is software design “safe” from AI?

Let’s find out!

Today’s Class: Designing with AI

- Part 1: Creating/improving design with AI
- Part 2: Evaluating the output of AI on design tasks

Case Study: IntelliGuard (from HW1)

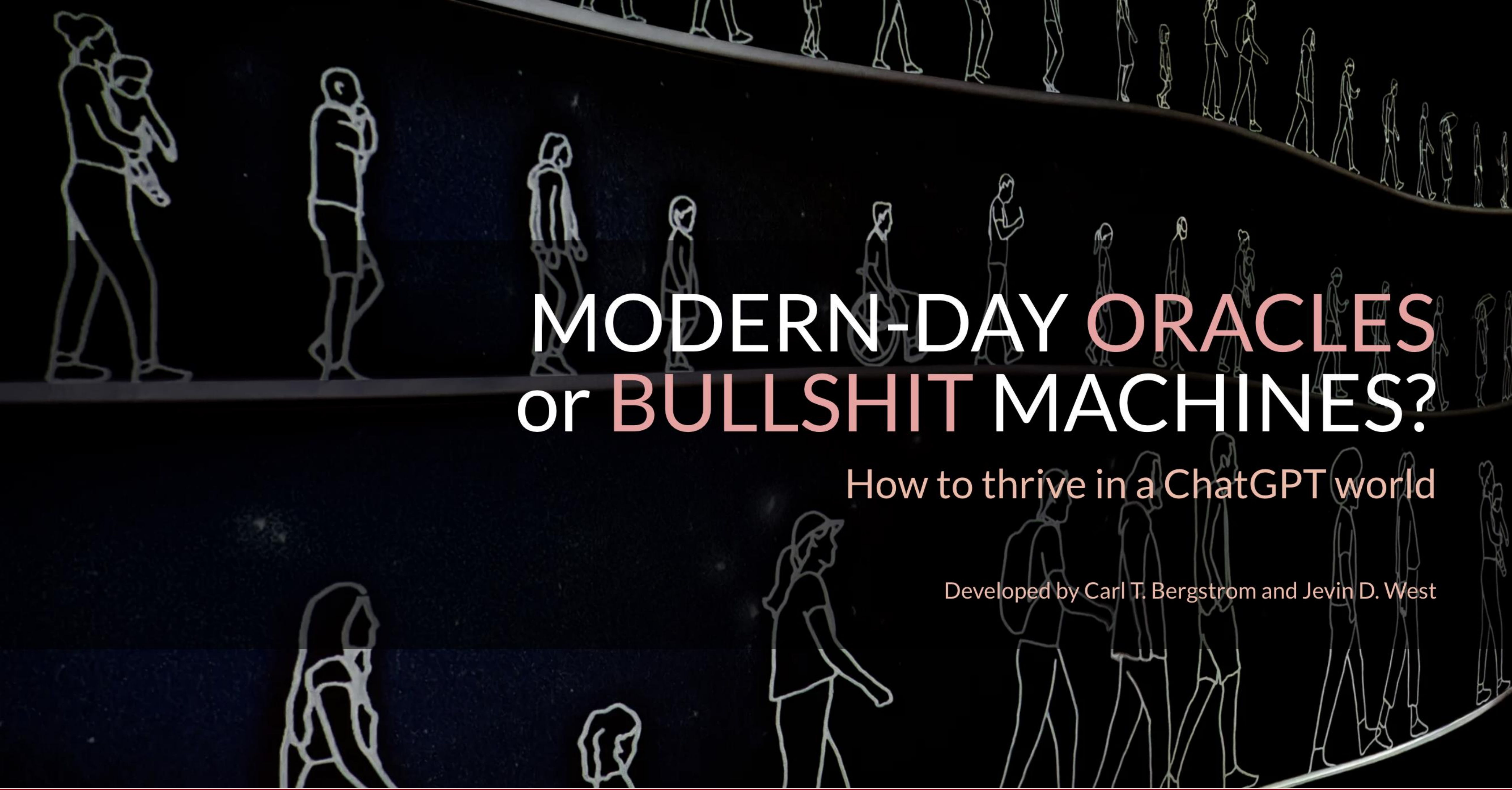


Activity Part 1: Designing with AI

- Break into a team of 3~4 people
- Pick one “representative” human design from HW1
- Each team will be assigned one of the “design tasks” from the following slide
- Use an LLM of your choice (Claude, ChatGPT, Gemini, etc.,) to generate a design for IntelliGuard, including:
  - Context model: Domain entities & assumptions
  - Component diagram: System components & responsibilities
- Compare the AI-generated design to the human design
  - Did AI find important assumptions/responsibilities not in the human design?
  - Did AI miss important assumptions/responsibilities?

Design Tasks

- **Team "Vibe designer"**
  - Provide a prompt describing the system description and ask the LLM to generate a design solution
- **Team "Design explorer"**
  - Provide a description of the system and ask the LLM to generate at least two alternative design solutions
  - Compare the candidates with respect to different quality attributes and select one based on a trade-off decision
- **Team "AI critic"**
  - Provide the human design (from HW1) to the LLM and ask it to critique and improve it



MODERN-DAY ORACLES or BULLSHIT MACHINES?

How to thrive in a ChatGPT world

Developed by Carl T. Bergstrom and Jevin D. West

LLMs: “Autocomplete in Overdrive”

- LLMs (and gen AI) are highly capable of generating **plausible content**; these are useful for many tasks (e.g., code generation, language translation)
- But **plausibility does not imply high quality**, and it’s often difficult to distinguish between the two
  - “It looks pretty good to me; let’s use it”
- As a system designer, we must apply critical judgement to extract actual substance from AI output
  - Unlike generated code, we can’t easily “test” a design by running it
  - But there are patterns of “bullshit” that we can systematically look for

Design Audit Checklist

- **Generic fluff**
  - **Ask**: Could this statement be written for any other system?
  - e.g., “The system should use a scalable architecture with a cloud-based data pipeline and AI-driven analytics”
  - “This architecture provides a clean separation of concerns across deployment targets”
- **Responsibility gaps**
  - Components with **missing responsibilities**
  - **Vague description**: Sounds meaningful but doesn’t describe any specific behaviors (“The Security Module is responsible for managing the security of home”)
  - **Magic component**: A single component that does everything (“The AI Engine analyzes video, identifies strangers, and notifies users”)

Design Audit Checklist

- **Missing/unrealistic assumptions**
  - Design solution ignores domain assumptions or relies on assumptions that are unrealistic
  - “Video is streamed to the cloud where AI models detect intruders”
  - “The camera encrypts the video frames before sending them over the network”
- **Happy-path bias**
  - Design solution handles normal operation scenarios but does not address major failure/edge case scenarios
  - e.g., No mention of possible network signal loss; ML model producing a false positive on a family member; the user not responding to alert within a time window

Design Audit Checklist

- **False specificity**
  - Inverse of generic/vague statements
  - Specific-sounding technical choices; gives an impression of depth without being appropriate for the actual system
  - “Use a WebSocket connection for real-time alert push” or “store face embeddings in a vector database”
- **Overconfidence signal**
  - Claims in absolute terms without proper justifications
  - “This architecture guarantees security against DoS attacks”
  - “It minimizes the costs of retraining the face detection model”

Activity Part 2: Evaluating AI Output

- Review the AI-generated design from Part 1 using the checklist
- Identify potential issues/“bullshit” & record them in the Google doc (to be shared with the rest of the class!)

Discussion

- In what ways can AI be potentially useful for design tasks?
- What are possible risks of using AI for software design?
- Will software design remain “safe” from AI, or will it be eventually automated along with other development tasks?

Summary

- No ticket today!