Assuring the Future of Software Engineering & AI Engineering

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

Institute for Software
Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Starting Point: The SEI's Study on Future of Software Engineering

- CMU SEI's National Agenda Study (November 2021) was intended to catalyze the software engineering community by creating a research & development vision, strategy, & roadmap to engineer the next-generation of software-reliant systems

DARPA

NSF

Software Engineering Institute

Carnegie Mellon

ARCHITECTING THE FUTURE OF SOFTWARE ENGINEERING

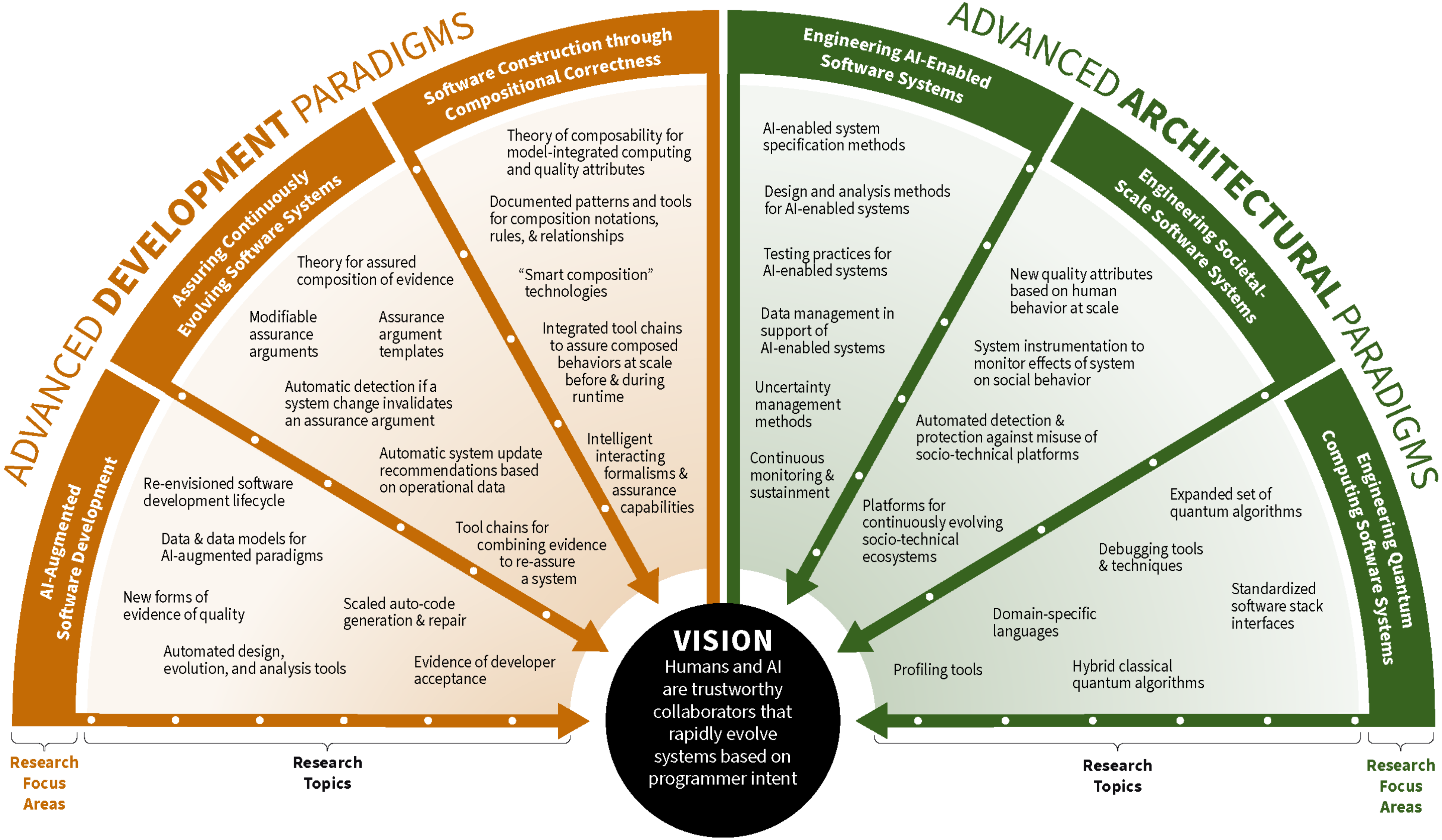
A National Agenda for Software Engineering Research & Development

Carnegie Mellon University
Software Engineering Institute

Study available at www.sei.cmu.edu/go/national-agenda

The Study Defined a Software Engineering Roadmap for 10-15 Years

- The software engineering roadmap codified research focus areas & research objectives



ADVANCED **DEVELOPMENT** PARADIGMS

AI-Augmented Software Development

Re-envisioned software development lifecycle

Data & data models for AI-augmented paradigms

New forms of evidence of quality

Automated design, evolution, and analysis tools

Scaled auto-code generation & repair

Evidence of developer acceptance

Assuring Continuously Evolving Software Systems

Theory for assured composition of evidence

Modifiable assurance arguments

Assurance argument templates

Automatic detection if a system change invalidates an assurance argument

Automatic system update recommendations based on operational data

Tool chains for combining evidence to re-assure a system

Software Construction through Compositional Correctness

Theory of composability for model-integrated computing and quality attributes

Documented patterns and tools for composition notations, rules, & relationships

"Smart composition" technologies

Integrated tool chains to assure composed behaviors at scale before & during runtime

Intelligent interacting formalisms & assurance capabilities

Research Focus Areas

Research Topics

VISION

Humans and AI are trustworthy collaborators that rapidly evolve systems based on programmer intent

ADVANCED **ARCHITECTURAL** PARADIGMS

Engineering AI-Enabled Software Systems

AI-enabled system specification methods

Design and analysis methods for AI-enabled systems

Testing practices for AI-enabled systems

Data management in support of AI-enabled systems

Uncertainty management methods

Continuous monitoring & sustainment

Engineering Societal-Scale Software Systems

New quality attributes based on human behavior at scale

System instrumentation to monitor effects of system on social behavior

Automated detection & protection against misuse of socio-technical platforms

Platforms for continuously evolving socio-technical ecosystems

Engineering Quantum Computing Software Systems

Expanded set of quantum algorithms

Debugging tools & techniques

Standardized software stack interfaces

Domain-specific languages

Profiling tools

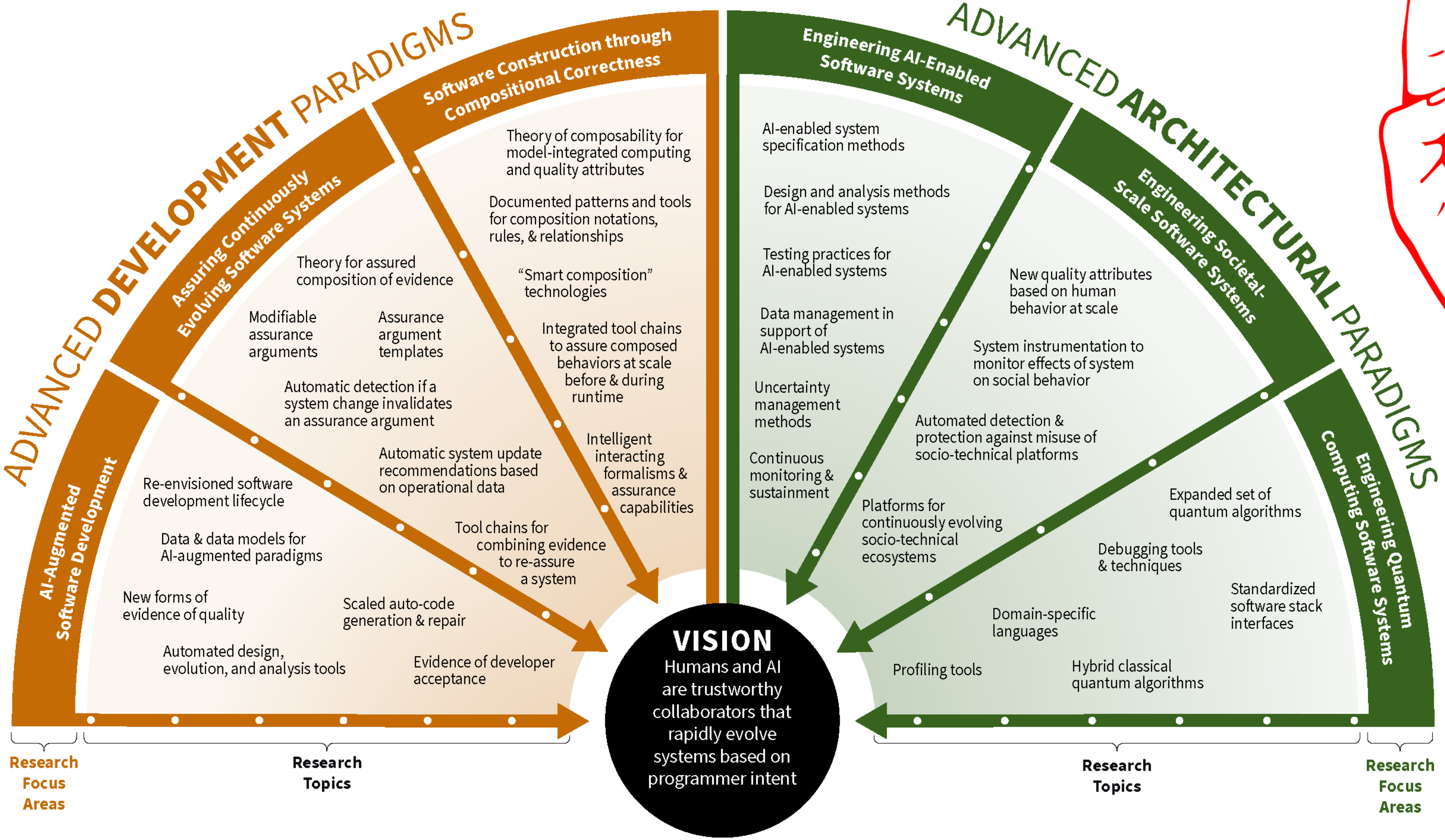
Hybrid classical quantum algorithms

Research Topics

Research Focus Areas

The Study Defined a Software Engineering Roadmap for 10-15 Years

- The software engineering roadmap codified research focus areas & researc objectives



ADVANCED DEVELOPMENT PARADIGMS

AI-Augmented Software Development

Re-envisioned software development lifecycle

Data & data models for AI-augmented paradigms

New forms of evidence of quality

Automated design, evolution, and analysis tools

Scaled auto-code generation & repair

Evidence of developer acceptance

Assuring Continuously Evolving Software Systems

Theory for assured composition of evidence

Modifiable assurance arguments

Assurance argument templates

Automatic detection if a system change invalidates an assurance argument

Automatic system update recommendations based on operational data

Tool chains for combining evidence to re-assure a system

Software Construction through Compositional Correctness

Theory of composability for model-integrated computing and quality attributes

Documented patterns and tools for composition notations, rules, & relationships

"Smart composition" technologies

Integrated tool chains to assure composed behaviors at scale before & during runtime

Intelligent interacting formalisms & assurance capabilities

ADVANCED ARCHITECTURAL PARADIGMS

Engineering AI-Enabled Software Systems

AI-enabled system specification methods

Design and analysis methods for AI-enabled systems

Testing practices for AI-enabled systems

Data management in support of AI-enabled systems

Uncertainty management methods

Continuous monitoring & sustainment

Engineering Societal-Scale Software Systems

New quality attributes based on human behavior at scale

System instrumentation to monitor effects of system on social behavior

Automated detection & protection against misuse of socio-technical platforms

Platforms for continuously evolving socio-technical ecosystems

Engineering Quantum Computing Software Systems

Expanded set of quantum algorithms

Debugging tools & techniques

Standardized software stack interfaces

Domain-specific languages

Hybrid classical quantum algorithms

Profiling tools

VISION
Humans and AI are trustworthy collaborators that rapidly evolve systems based on programmer intent

Research Focus Areas

Research Topics

Research Topics

Research Focus Areas

The Study's Emerging Vision of the Future of Software Engineering

- "The current notion of software development will be replaced by one where ***the software pipeline consists of humans & AI as trustworthy collaborators that rapidly evolve systems based on programmer intent***"

Then Something Unexpected Happened..

- Almost exactly one year later (November 2022) ChatGPT “escaped from the lab”!

See en.wikipedia.org/wiki/ChatGPT

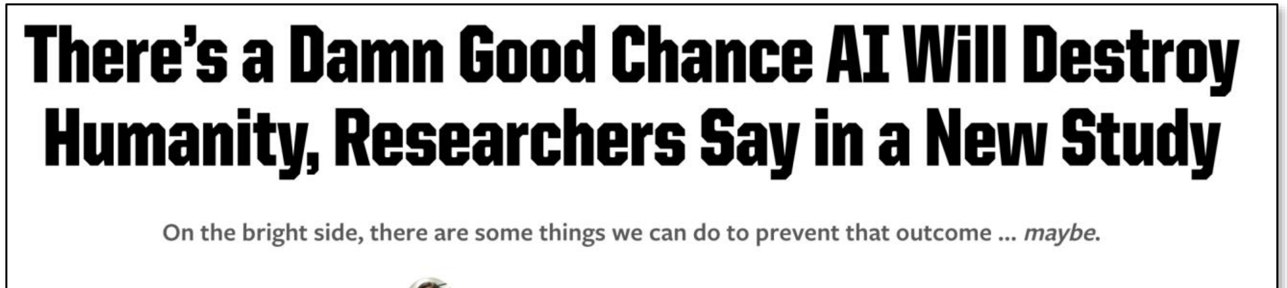
Then Something Unexpected Happened..

- Almost exactly one year later (November 2022) ChatGPT "escaped from the lab"!

"Generative AI" creates models or systems that can synthesize new & original content

See en.wikipedia.org/wiki/Generative_artificial_intelligence

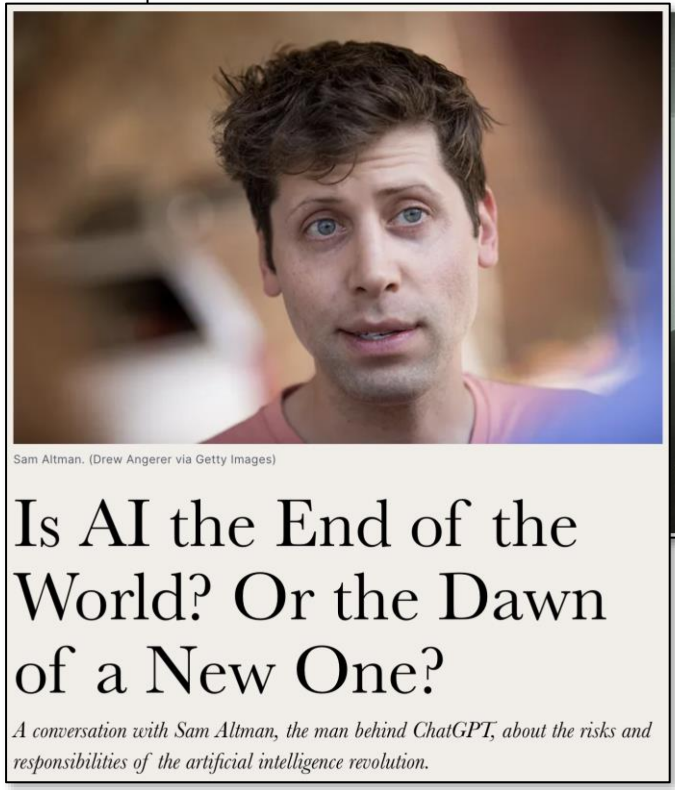
& Suddenly the World is Coming to an End..



There's a Damn Good Chance AI Will Destroy Humanity, Researchers Say in a New Study

On the bright side, there are some things we can do to prevent that outcome … *maybe*.

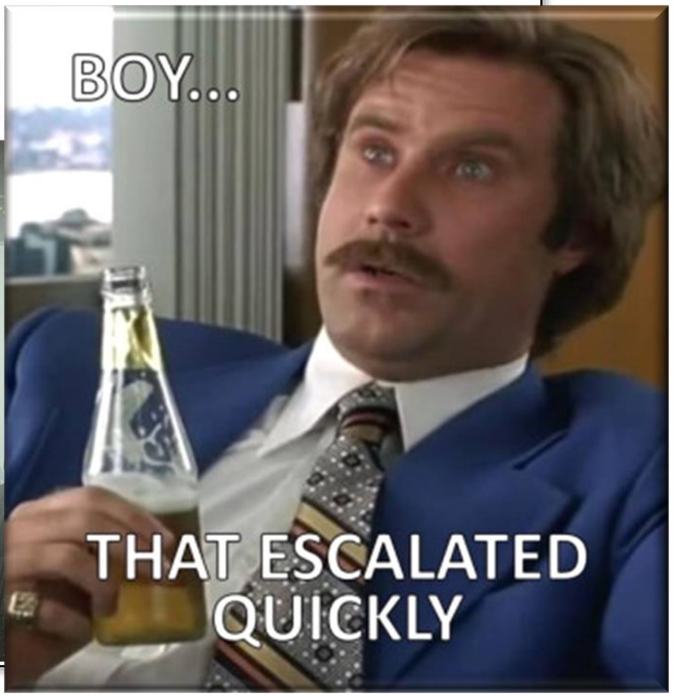
BY CAROLINE DELBERT PUBLISHED:

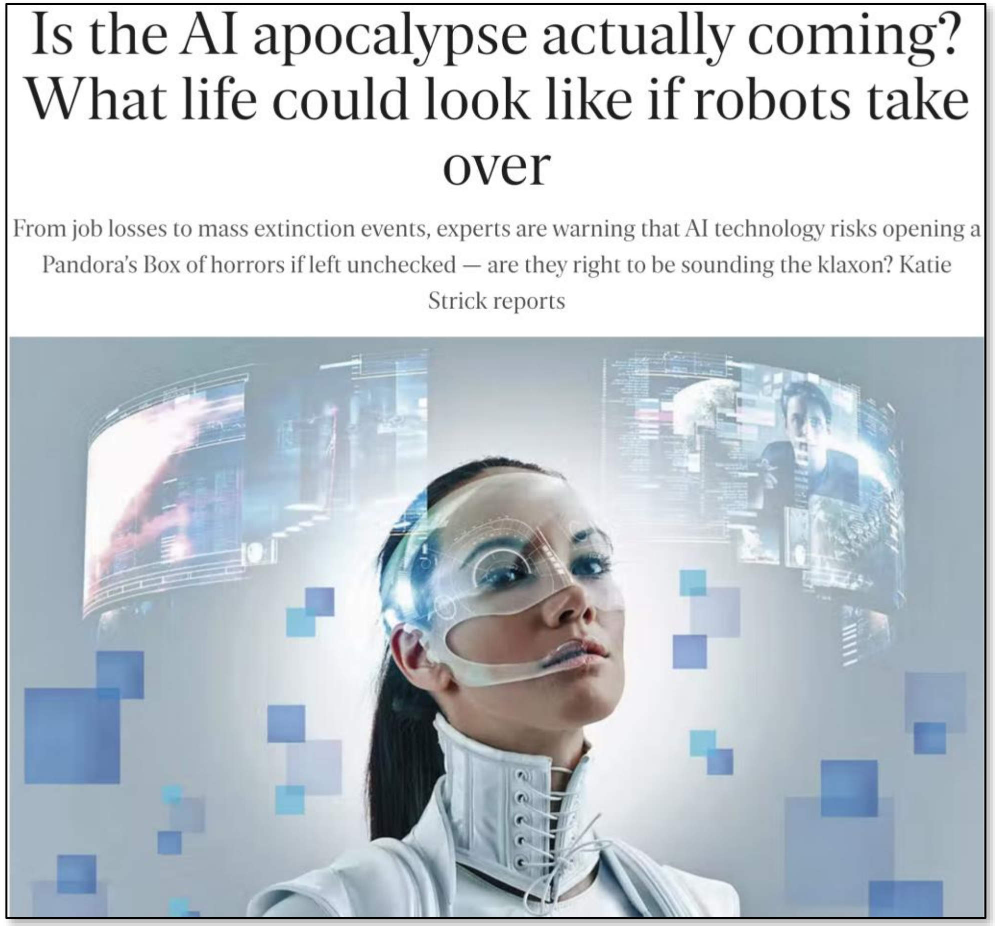


Sam Altman. (Drew Angerer via Getty Images)

Is AI the End of the World? Or the Dawn of a New One?

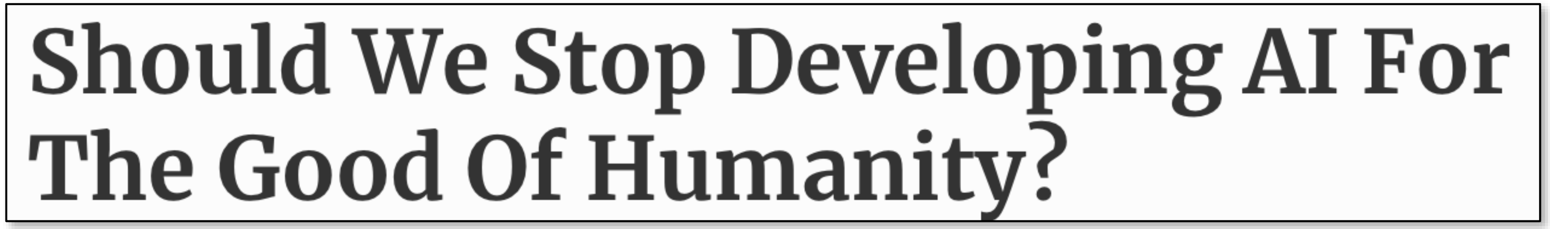
A conversation with Sam Altman, the man behind ChatGPT, about the risks and responsibilities of the artificial intelligence revolution.





Is the AI apocalypse actually coming? What life could look like if robots take over

From job losses to mass extinction events, experts are warning that AI technology risks opening a Pandora's Box of horrors if left unchecked — are they right to be sounding the klaxon? Katie Strick reports

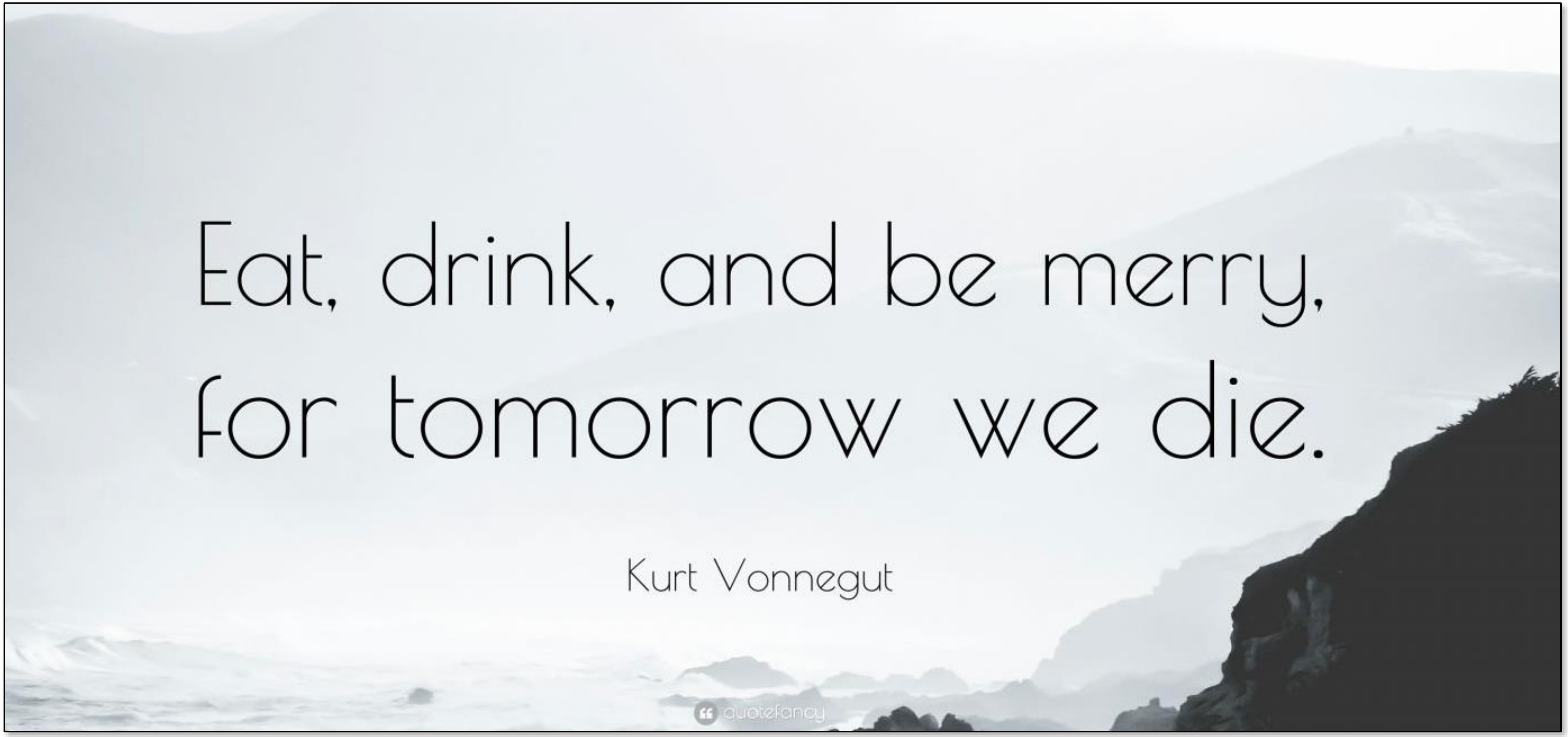


Should We Stop Developing AI For The Good Of Humanity?

See www.google.com/search?q=is+AI+the+end+of+the+world

& Suddenly the World is Coming to an End

- Of course, if generative AI ends humanity as we know it would certainly make it easier to predict the future of software engineering & AI engineering 🙃



Eat, drink, and be merry, for tomorrow we die.

Kurt Vonnegut

quotefancy

See www.magicalquote.com/bookquotes/eat-drink-and-be-merry-for-tomorrow-we-die

& Suddenly the World is Coming to an End

- Of course, if generative AI ends humanity as we know it would certainly make it easier to predict the future of software engineering & AI engineering 🙃

Eat, drink, and be merry, for tomorrow we die.

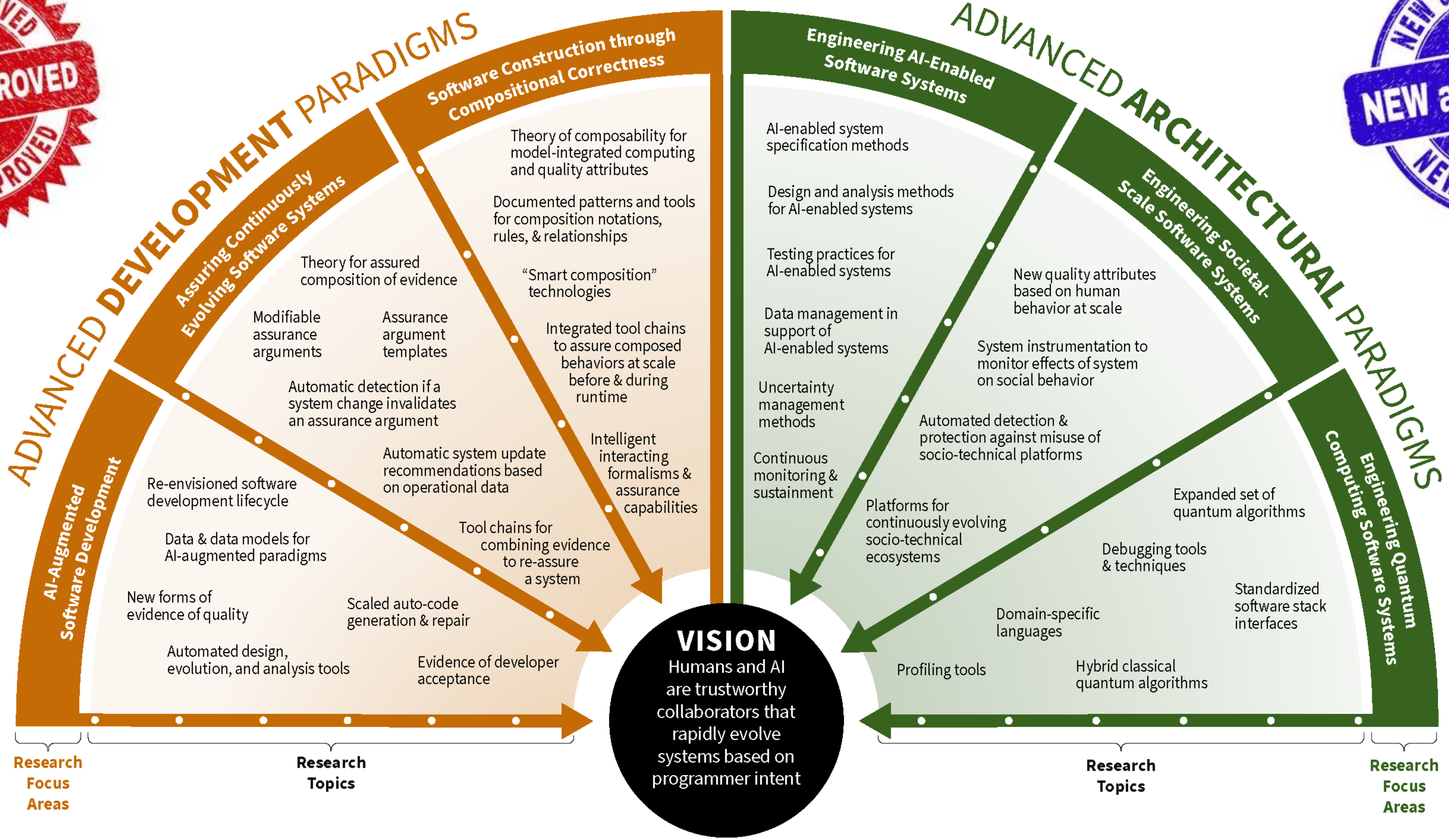
Kurt Vonnegut

but alas we never do

See hellopoetry.com/poem/10170/the-flaw-in-paganism

How Advances in Generative AI May Affect Our Study Findings

NEW and IMPROVED
NEW and IMPROVED
NEW and IMPROVED



ADVANCED **DEVELOPMENT** PARADIGMS

AI-Augmented Software Development

Re-envisioned software development lifecycle
Data & data models for AI-augmented paradigms
New forms of evidence of quality
Automated design, evolution, and analysis tools
Scaled auto-code generation & repair
Evidence of developer acceptance

Assuring Continuously Evolving Software Systems

Theory for assured composition of evidence
Modifiable assurance arguments
Assurance argument templates
Automatic detection if a system change invalidates an assurance argument
Automatic system update recommendations based on operational data
Tool chains for combining evidence to re-assure a system

Software Construction through Compositional Correctness

Theory of composability for model-integrated computing and quality attributes
Documented patterns and tools for composition notations, rules, & relationships
"Smart composition" technologies
Integrated tool chains to assure composed behaviors at scale before & during runtime
Intelligent interacting formalisms & assurance capabilities

VISION
Humans and AI are trustworthy collaborators that rapidly evolve systems based on programmer intent

ADVANCED **ARCHITECTURAL** PARADIGMS

Engineering AI-Enabled Software Systems

AI-enabled system specification methods
Design and analysis methods for AI-enabled systems
Testing practices for AI-enabled systems
Data management in support of AI-enabled systems
Uncertainty management methods
Continuous monitoring & sustainment

Engineering Societal-Scale Software Systems

New quality attributes based on human behavior at scale
System instrumentation to monitor effects of system on social behavior
Automated detection & protection against misuse of socio-technical platforms
Platforms for continuously evolving socio-technical ecosystems

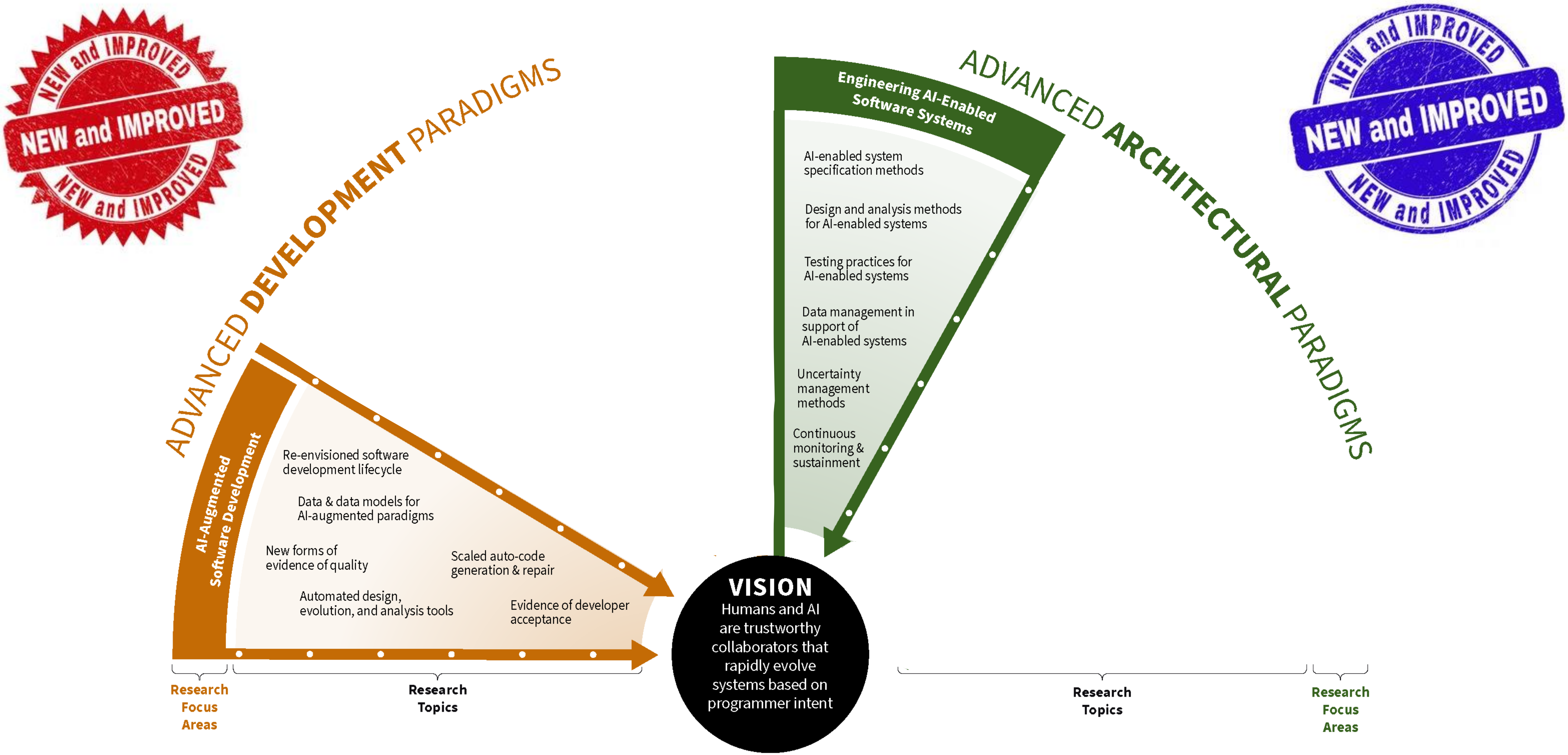
Engineering Quantum Computing Software Systems

Expanded set of quantum algorithms
Debugging tools & techniques
Standardized software stack interfaces
Domain-specific languages
Hybrid classical quantum algorithms
Profiling tools

Research Focus Areas
Research Topics
Research Topics
Research Focus Areas

NEW and IMPROVED
NEW and IMPROVED
NEW and IMPROVED

How Advances in Generative AI May Affect Our Study Findings



ADVANCED **DEVELOPMENT** PARADIGMS

AI-Augmented Software Development

- Re-envisioned software development lifecycle
- Data & data models for AI-augmented paradigms
- New forms of evidence of quality
- Scaled auto-code generation & repair
- Automated design, evolution, and analysis tools
- Evidence of developer acceptance

Research Focus Areas

Research Topics

VISION
Humans and AI are trustworthy collaborators that rapidly evolve systems based on programmer intent

ADVANCED **ARCHITECTURAL** PARADIGMS

Engineering AI-Enabled Software Systems

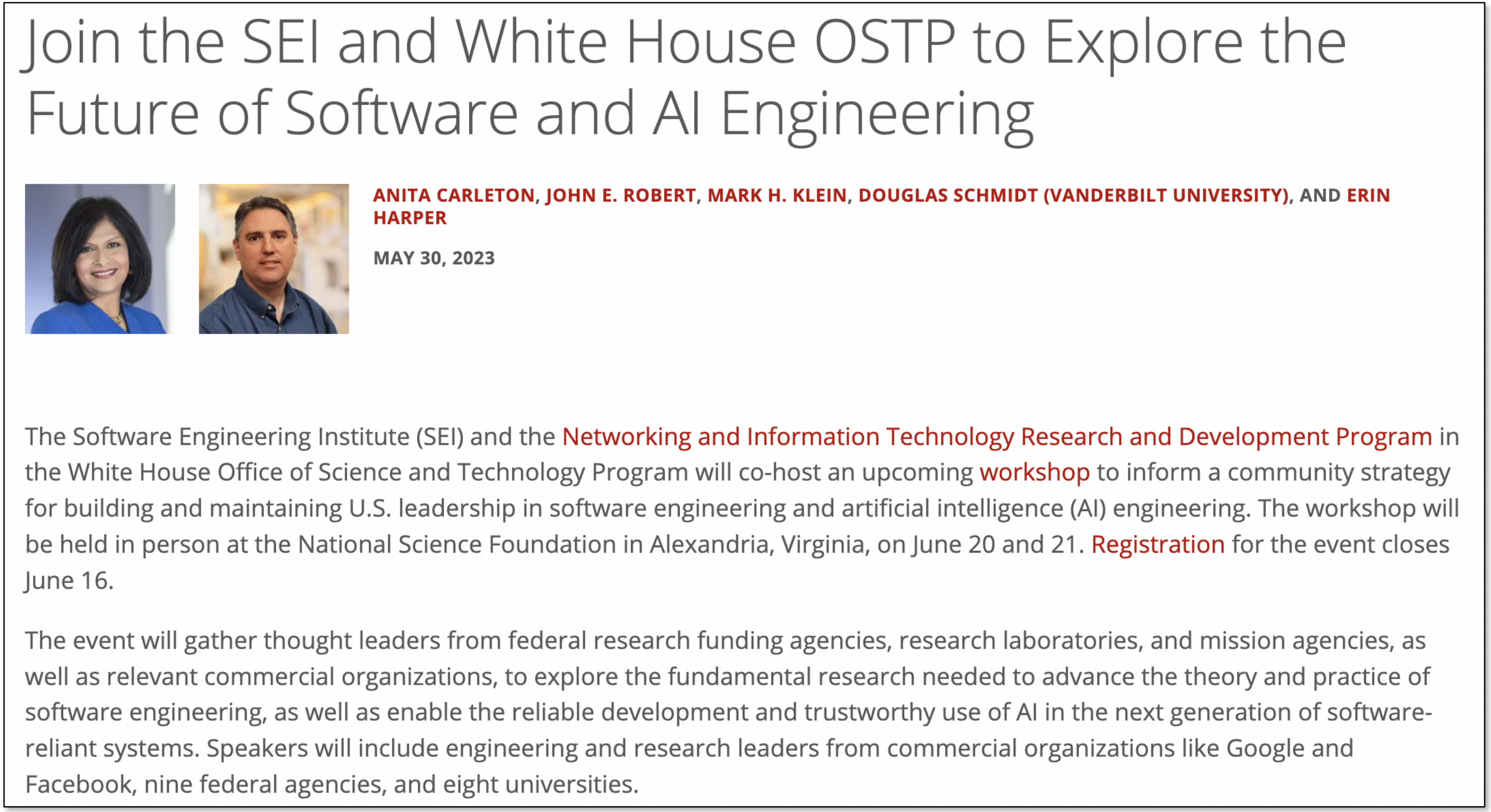
- AI-enabled system specification methods
- Design and analysis methods for AI-enabled systems
- Testing practices for AI-enabled systems
- Data management in support of AI-enabled systems
- Uncertainty management methods
- Continuous monitoring & sustainment

Research Topics

Research Focus Areas

We'll focus mostly on two of the six research focus areas in this presentation

How Advances in Generative AI May Affect Our Study Findings



Join the SEI and White House OSTP to Explore the Future of Software and AI Engineering

ANITA CARLETON, JOHN E. ROBERT, MARK H. KLEIN, DOUGLAS SCHMIDT (VANDERBILT UNIVERSITY), AND ERIN HARPER

MAY 30, 2023

The Software Engineering Institute (SEI) and the Networking and Information Technology Research and Development Program in the White House Office of Science and Technology Program will co-host an upcoming workshop to inform a community strategy for building and maintaining U.S. leadership in software engineering and artificial intelligence (AI) engineering. The workshop will be held in person at the National Science Foundation in Alexandria, Virginia, on June 20 and 21. Registration for the event closes June 16.

The event will gather thought leaders from federal research funding agencies, research laboratories, and mission agencies, as well as relevant commercial organizations, to explore the fundamental research needed to advance the theory and practice of software engineering, as well as enable the reliable development and trustworthy use of AI in the next generation of software-reliant systems. Speakers will include engineering and research leaders from commercial organizations like Google and Facebook, nine federal agencies, and eight universities.

See insights.sei.cmu.edu/blog/join-the-sei-and-white-house-ostp-to-explore-the-future-of-software-and-ai-engineering

Impact on AI-Augmented Software Development

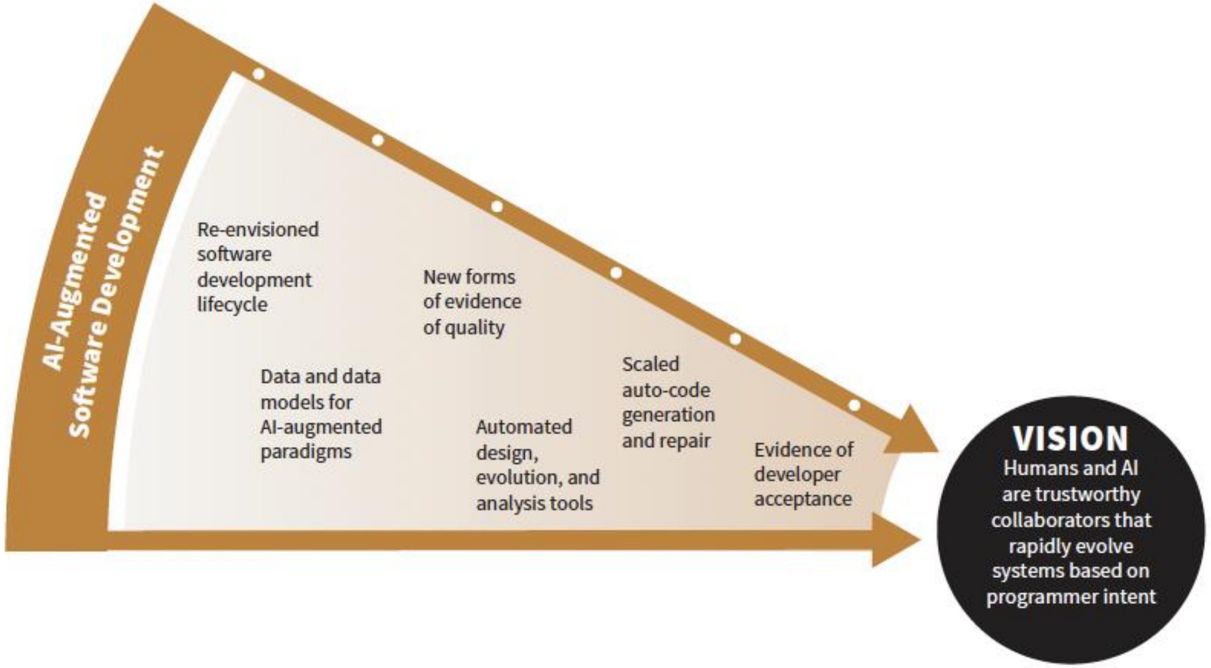
Impact on AI-Augmented Software Development

- We'll start out with a "high-percentage" predication:

See www.basketballnetwork.net/old-school/julius-erving-explained-why-he-didnt-attempt-dunks-that-have-good-chances-of-missing

Impact on AI-Augmented Software Development

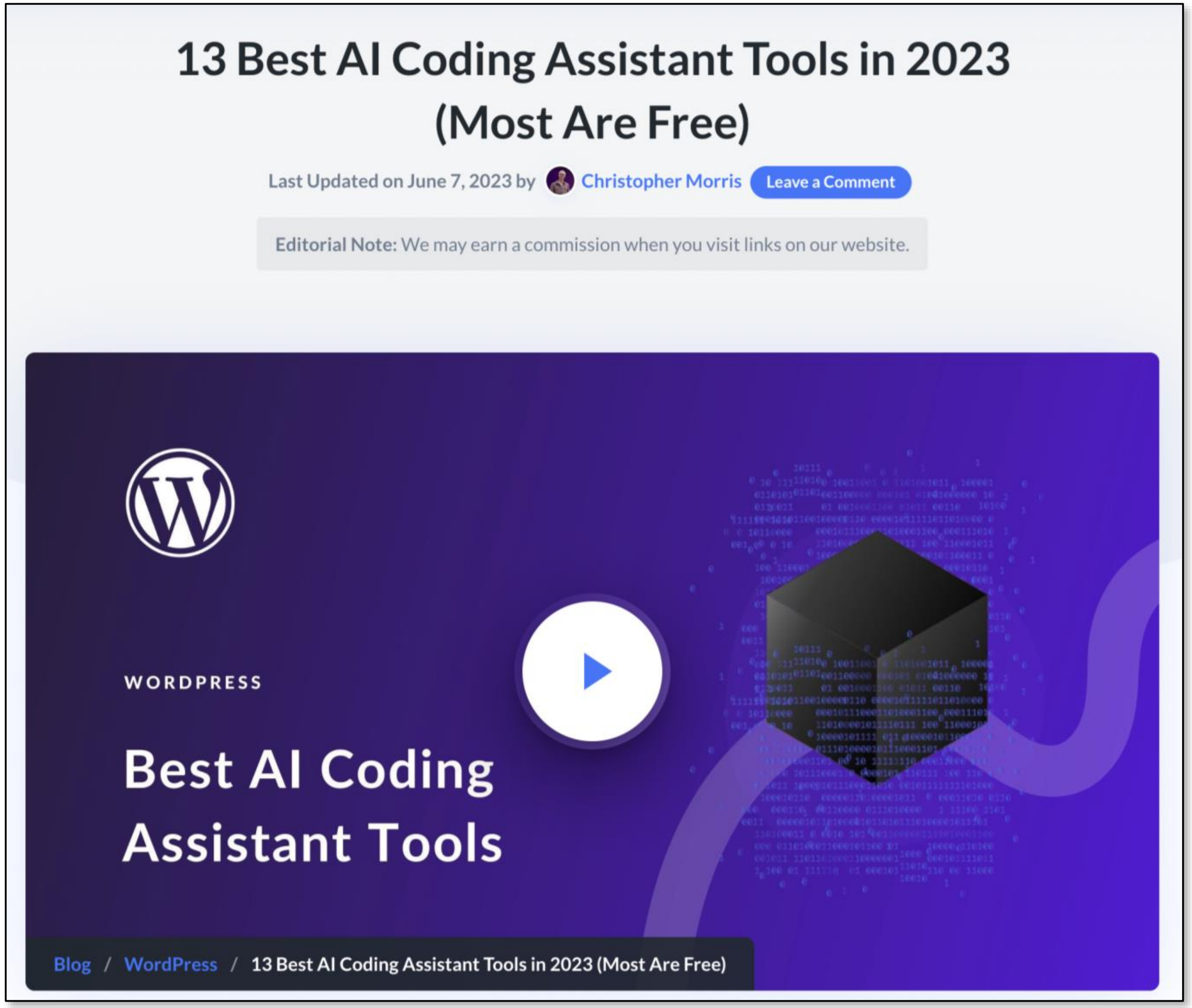
- We'll start out with a "high-percentage" predication: Generative AI is/will have a transformative impact on the practice of software development


AI-Augmented Software Development
Re-envisioned software development lifecycle
New forms of evidence of quality
Data and data models for AI-augmented paradigms
Automated design, evolution, and analysis tools
Scaled auto-code generation and repair
Evidence of developer acceptance
VISION
Humans and AI are trustworthy collaborators that rapidly evolve systems based on programmer intent

See dev.to/wesen/llms-will-fundamentally-change-software-engineering-3oj8

Impact on AI-Augmented Software Development

- We'll start out with a "high-percentage" predication: Generative AI is/will have a transformative impact on the practice of software development
    - AI-based tools are increasingly applied to improve the efficiency/quality of software engineers & reduce their cognitive load



13 Best AI Coding Assistant Tools in 2023 (Most Are Free)

Last Updated on June 7, 2023 by Christopher Morris

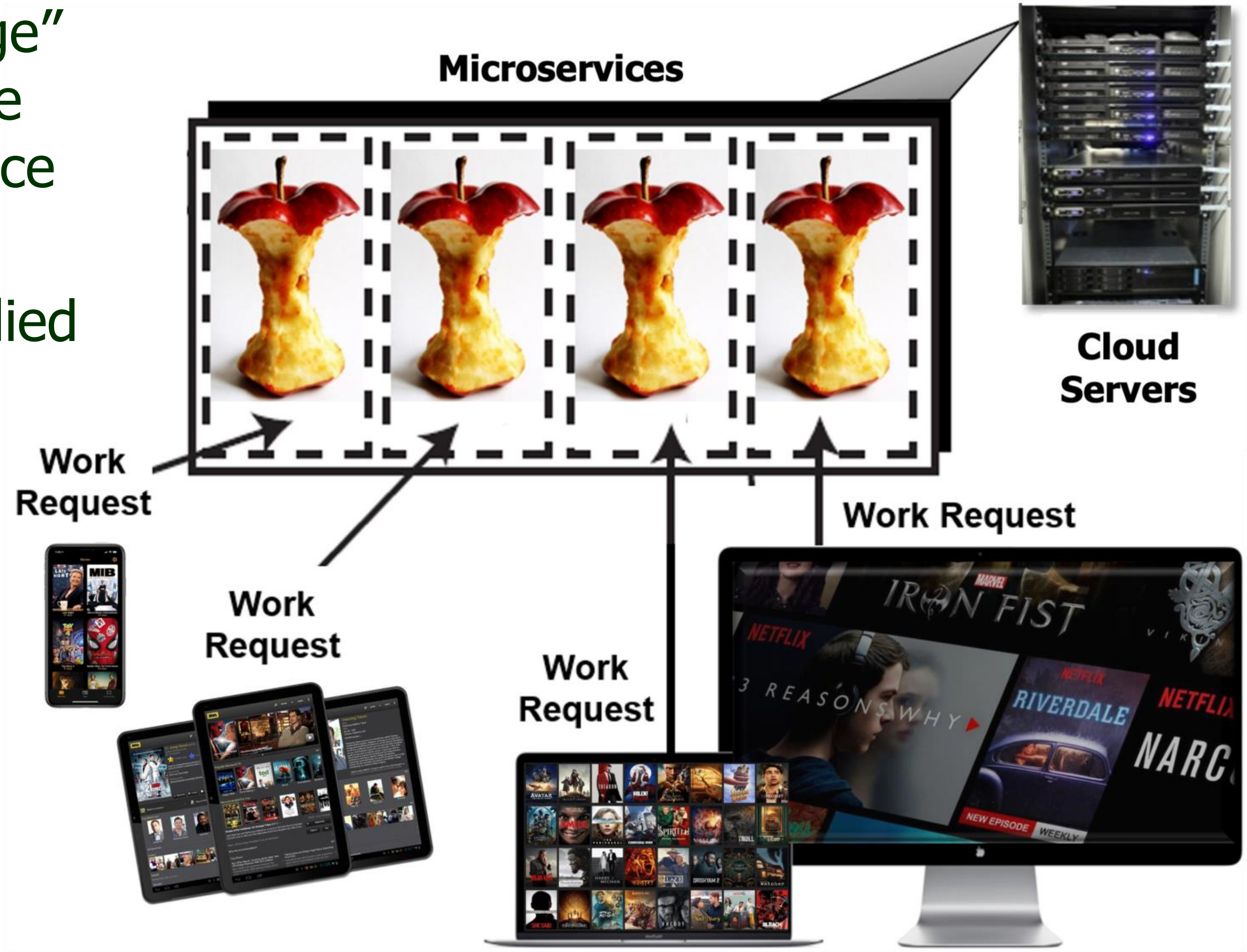
Editorial Note: We may earn a commission when you visit links on our website.

Blog / WordPress / 13 Best AI Coding Assistant Tools in 2023 (Most Are Free)

See www.elegantthemes.com/blog/wordpress/best-ai-coding-assistant

Impact on AI-Augmented Software Development

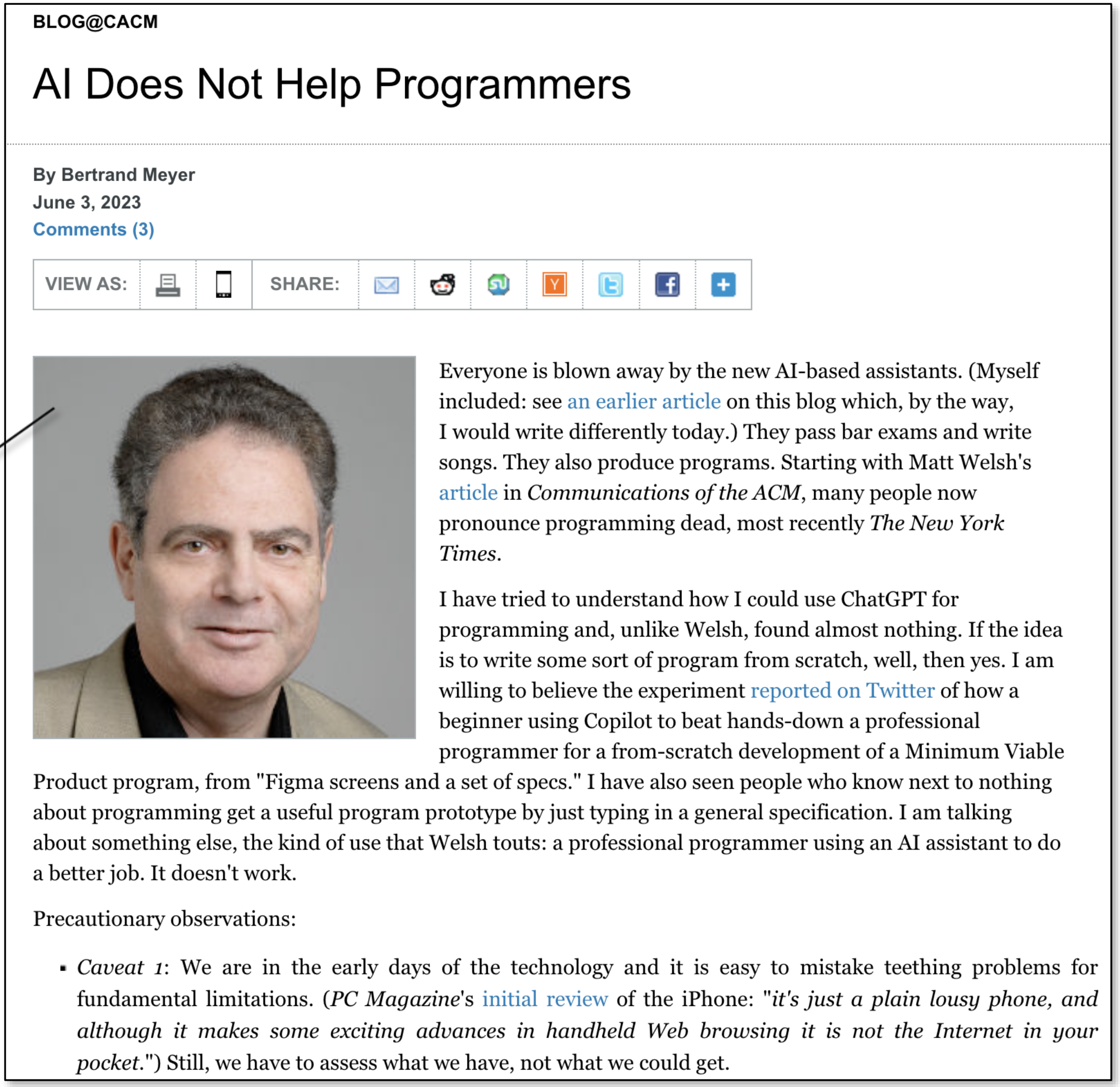
- We’ll start out with a “high-percentage” predication: Generative AI is/will have a transformative impact on the practice of software development
  - AI-based tools are increasingly applied to improve the efficiency/quality of software engineers & reduce their cognitive load



I’ll provide more evidence for my perspective in this presentation shortly

Impact on AI-Augmented Software Development

- We'll start out with a "high-percentage" predication: Generative AI is/will have a transformative impact on the practice of software development
  - AI-based tools are increasingly applied to improve the efficiency/quality of software engineers & reduce their cognitive load



BLOG@CACM

AI Does Not Help Programmers

By Bertrand Meyer
June 3, 2023
Comments (3)

Everyone is blown away by the new AI-based assistants. (Myself included: see an earlier article on this blog which, by the way, I would write differently today.) They pass bar exams and write songs. They also produce programs. Starting with Matt Welsh's article in *Communications of the ACM*, many people now pronounce programming dead, most recently *The New York Times*.

I have tried to understand how I could use ChatGPT for programming and, unlike Welsh, found almost nothing. If the idea is to write some sort of program from scratch, well, then yes. I am willing to believe the experiment reported on Twitter of how a beginner using Copilot to beat hands-down a professional programmer for a from-scratch development of a Minimum Viable Product program, from "Figma screens and a set of specs." I have also seen people who know next to nothing about programming get a useful program prototype by just typing in a general specification. I am talking about something else, the kind of use that Welsh touts: a professional programmer using an AI assistant to do a better job. It doesn't work.

Precautionary observations:

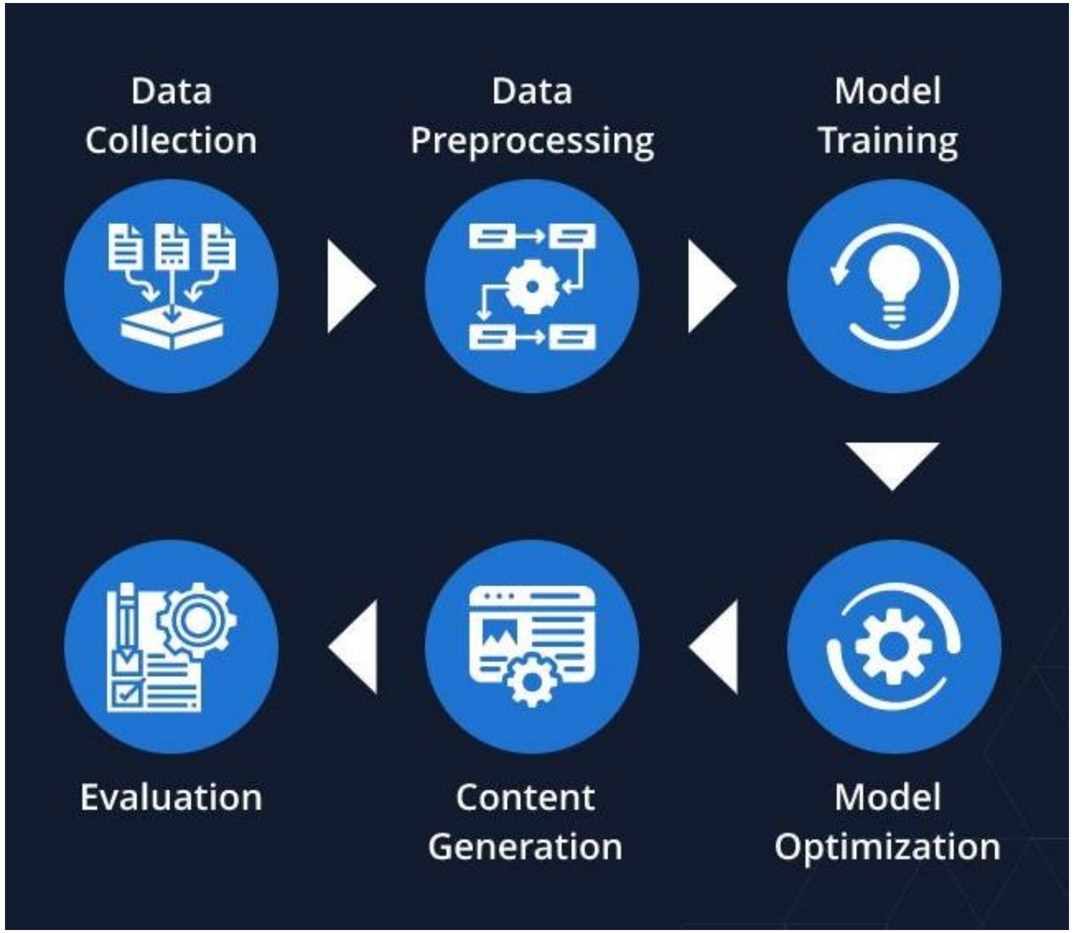
- *Caveat 1*: We are in the early days of the technology and it is easy to mistake teething problems for fundamental limitations. (*PC Magazine*'s initial review of the iPhone: "*it's just a plain lousy phone, and although it makes some exciting advances in handheld Web browsing it is not the Internet in your pocket.*") Still, we have to assess what we have, not what we could get.

Not everyone is equally bullish about the benefits of generative AI for programmers, of course!!!

See cacm.acm.org/blogs/blog-cacm/273577-ai-does-not-help-programmers/fulltext

Impact on AI-Augmented Software Development

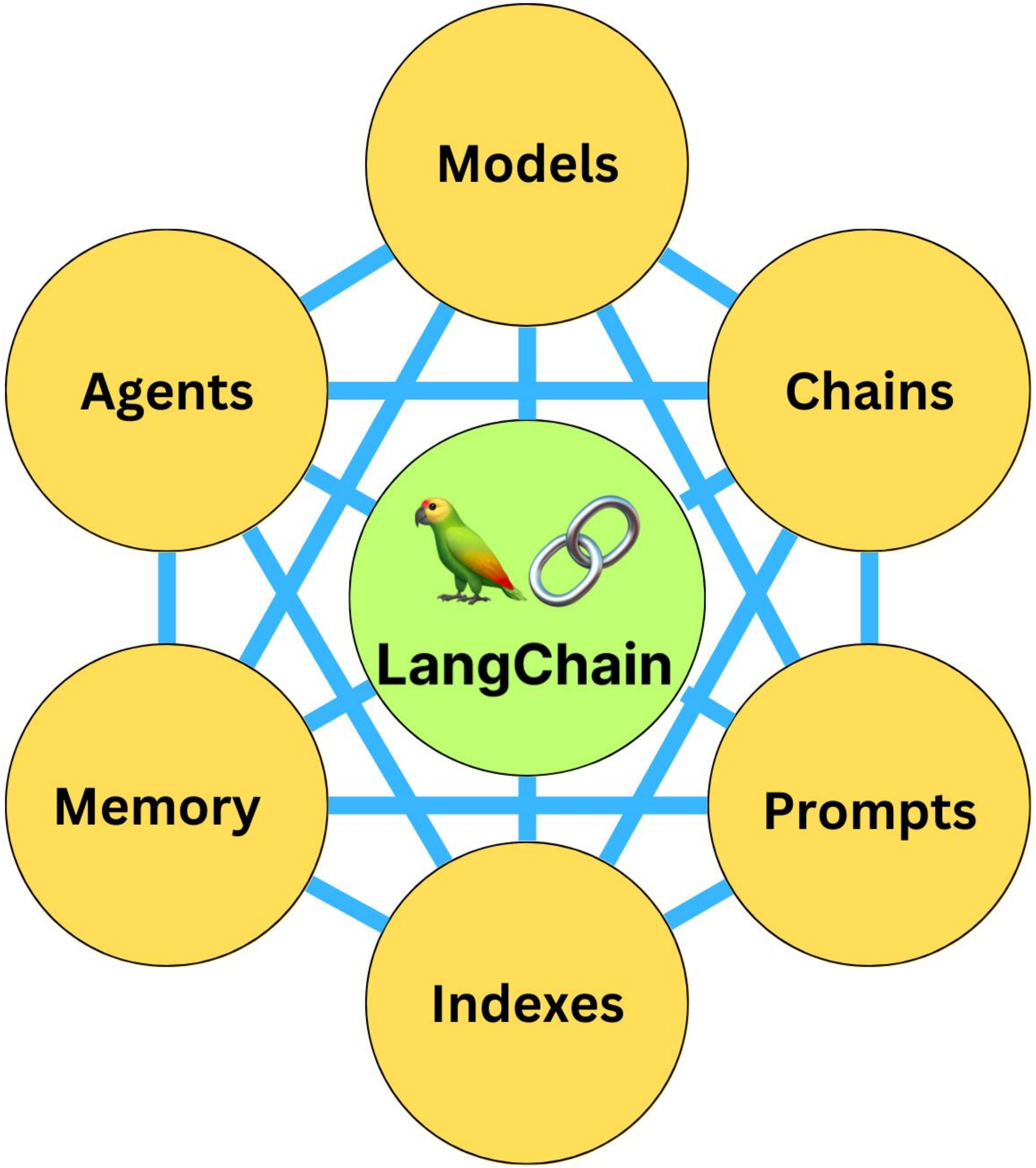
- Some key R&D challenges include
  - Training large-language models on better vetted & robust code bases



See neal-lathia.medium.com/evaluating-llms-trained-on-code-bb2bdab3cb37

Impact on AI-Augmented Software Development

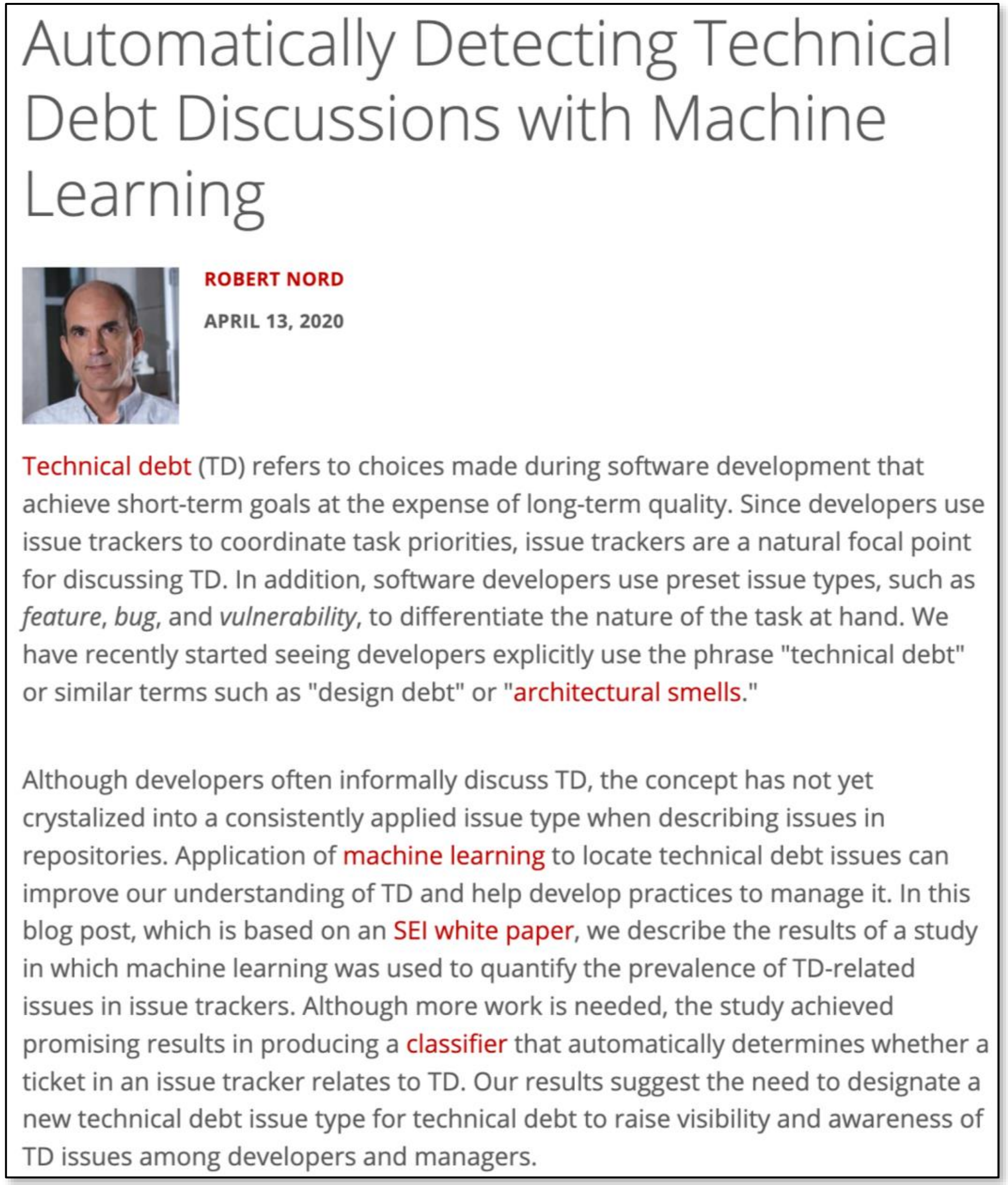
- Some key R&D challenges include
  - Training large-language models on better vetted & robust code bases
  - Re-envisioning the software development lifecycle (SDLC)
    - i.e., increased AI & automation tool support for developers & other stake holders beyond programming



See docs.langchain.com/docs

Impact on AI-Augmented Software Development

- Some key R&D challenges include
  - Training large-language models on better vetted & robust code bases
  - Re-envisioning the software development lifecycle (SDLC)
  - Effectively capturing & leveraging data generated throughout the SDLC



Automatically Detecting Technical Debt Discussions with Machine Learning

ROBERT NORD

APRIL 13, 2020

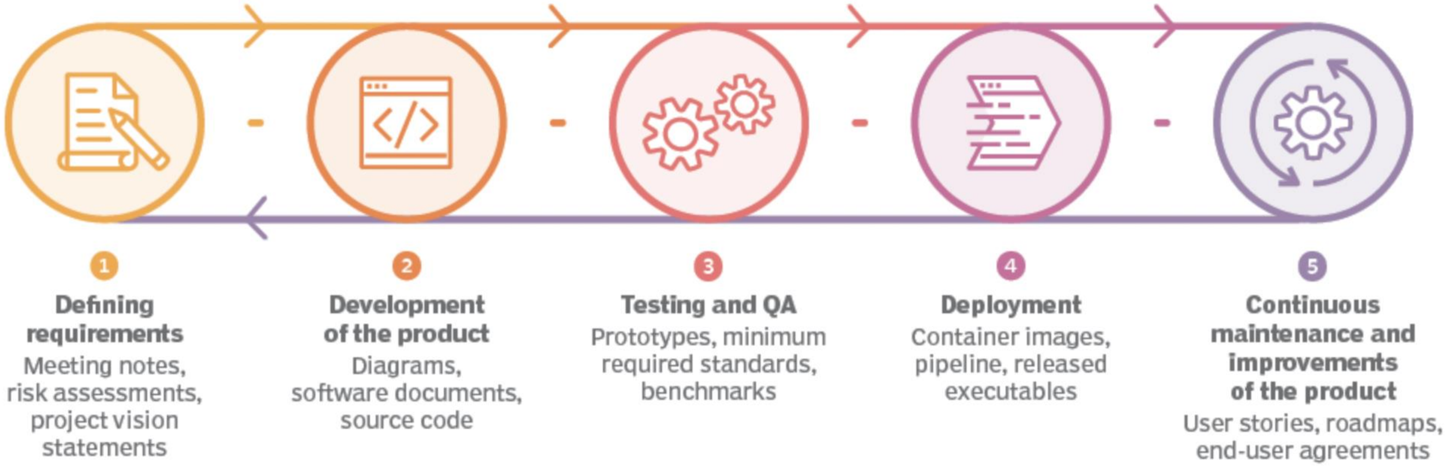
Technical debt (TD) refers to choices made during software development that achieve short-term goals at the expense of long-term quality. Since developers use issue trackers to coordinate task priorities, issue trackers are a natural focal point for discussing TD. In addition, software developers use preset issue types, such as *feature*, *bug*, and *vulnerability*, to differentiate the nature of the task at hand. We have recently started seeing developers explicitly use the phrase "technical debt" or similar terms such as "design debt" or "architectural smells."

Although developers often informally discuss TD, the concept has not yet crystalized into a consistently applied issue type when describing issues in repositories. Application of machine learning to locate technical debt issues can improve our understanding of TD and help develop practices to manage it. In this blog post, which is based on an SEI white paper, we describe the results of a study in which machine learning was used to quantify the prevalence of TD-related issues in issue trackers. Although more work is needed, the study achieved promising results in producing a classifier that automatically determines whether a ticket in an issue tracker relates to TD. Our results suggest the need to designate a new technical debt issue type for technical debt to raise visibility and awareness of TD issues among developers and managers.

See insights.sei.cmu.edu/blog/automatically-detecting-technical-debt-discussions-with-machine-learning

Impact on AI-Augmented Software Development

- Some key R&D challenges include
  - Training large-language models on better vetted & robust code bases
  - Re-envisioning the software development lifecycle (SDLC)
  - Effectively capturing & leveraging data generated throughout the SDLC
    - e.g., many non-code artifacts can be analyzed at scale by AI tools much better/faster/cheaper than by humans



① **Defining requirements**
Meeting notes, risk assessments, project vision statements

② **Development of the product**
Diagrams, software documents, source code

③ **Testing and QA**
Prototypes, minimum required standards, benchmarks

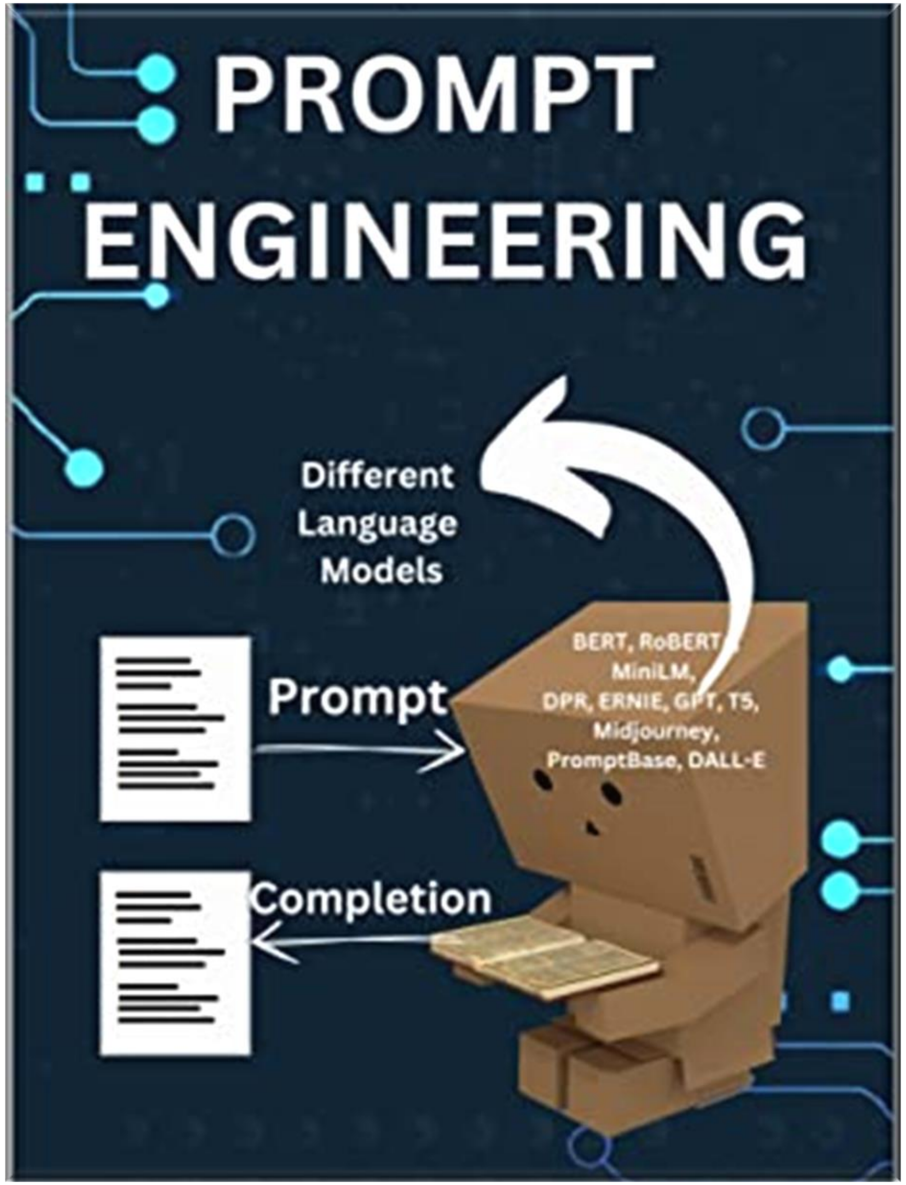
④ **Deployment**
Container images, pipeline, released executables

⑤ **Continuous maintenance and improvements of the product**
User stories, roadmaps, end-user agreements

See aiperspectives.springeropen.com/articles/10.1186/s42467-020-00005-4

Impact on AI-Augmented Software Development

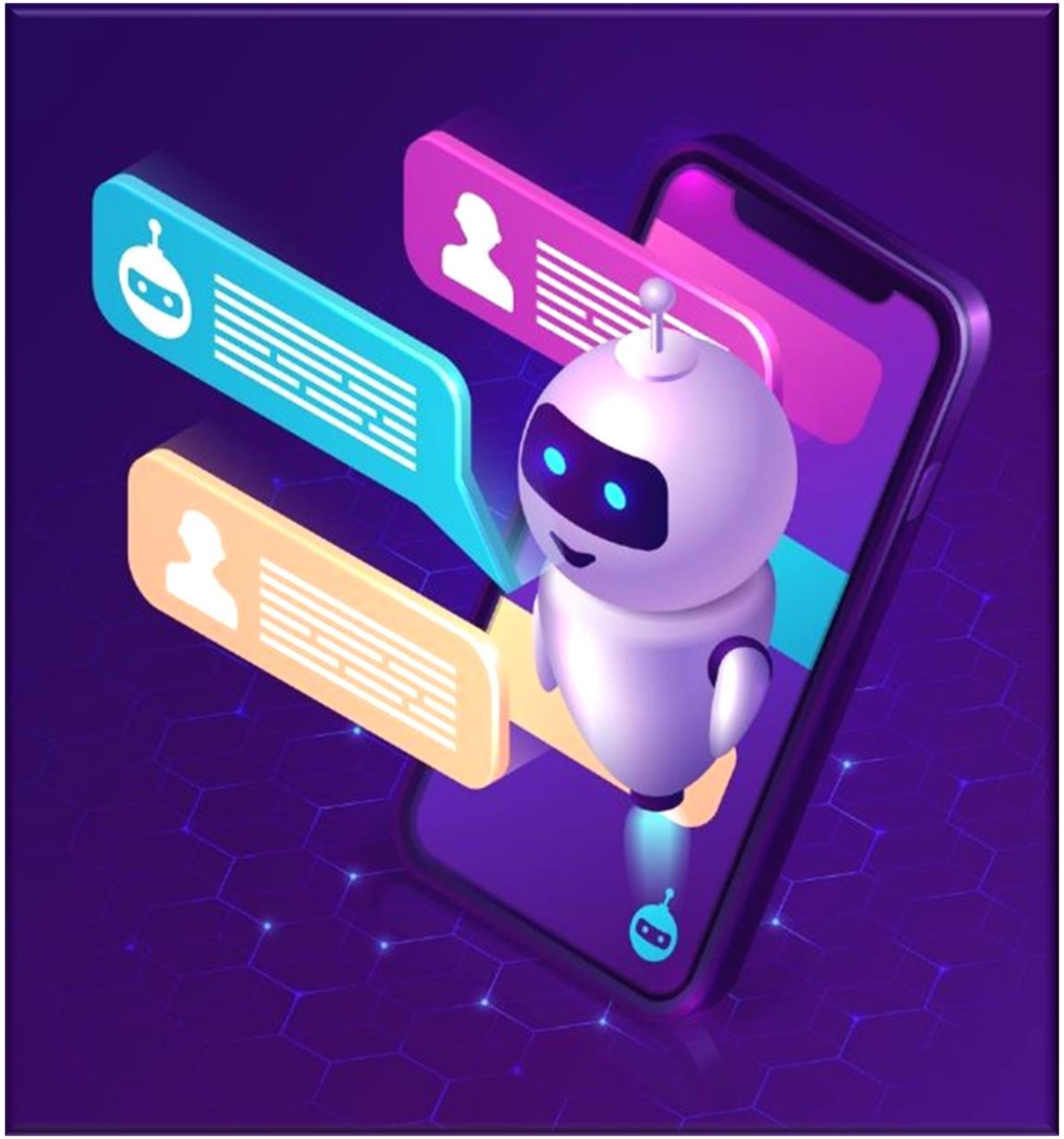
- Some key R&D challenges include
  - Training large-language models on better vetted & robust code bases
  - Re-envisioning the software development lifecycle (SDLC)
  - Effectively capturing & leveraging data generated throughout the SDLC
  - Formalizing the discipline of "Prompt Engineering"


PROMPT
ENGINEERING
Different Language Models
Prompt
Completion
BERT, RoBERTa, MiniLM, DPR, ERNIE, GPT, T5, Midjourney, PromptBase, DALL-E

See en.wikipedia.org/wiki/Prompt_engineering

Impact on AI-Augmented Software Development

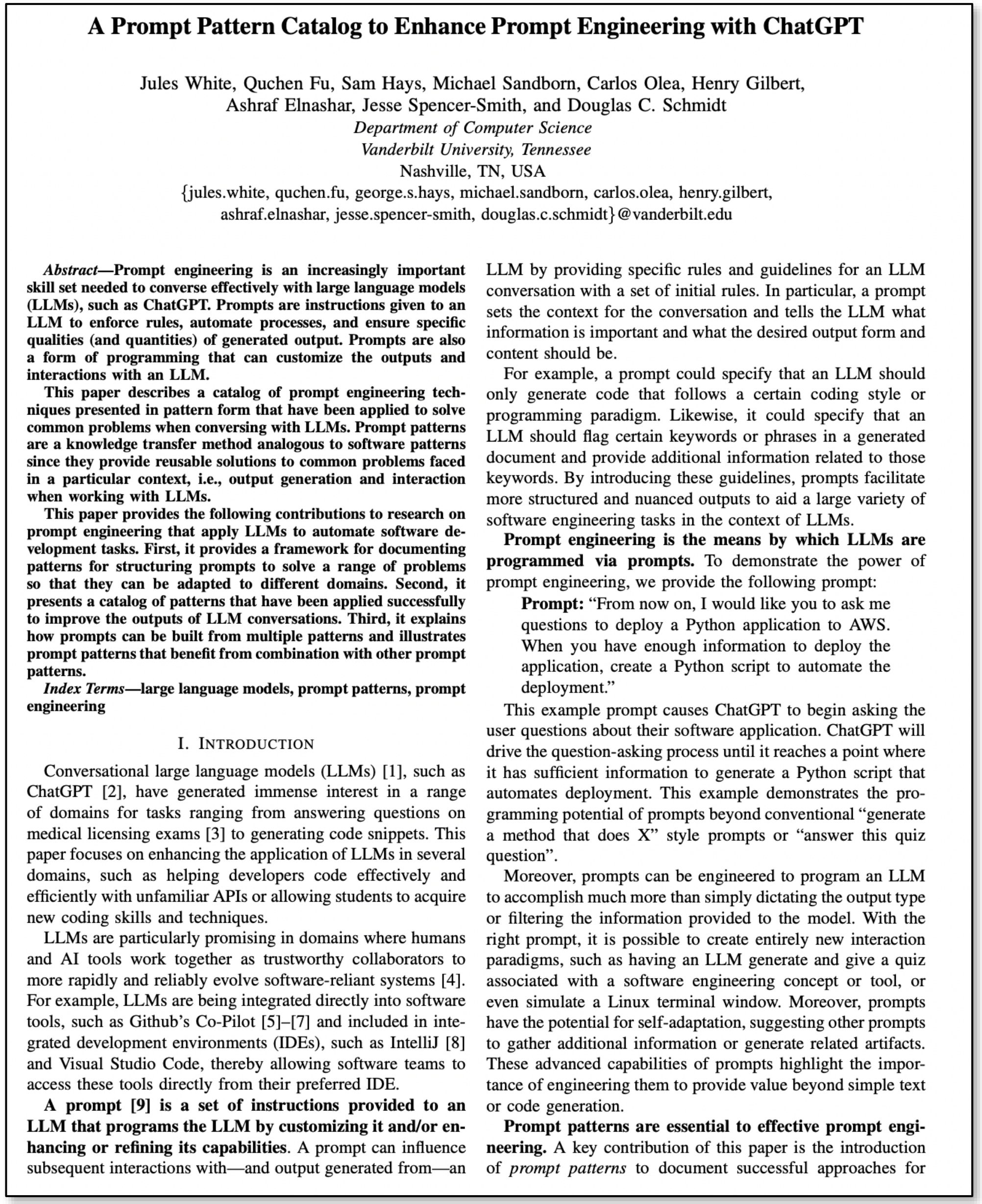
- Some key R&D challenges include
  - Training large-language models on better vetted & robust code bases
  - Re-envisioning the software development lifecycle (SDLC)
  - Effectively capturing & leveraging data generated throughout the SDLC
  - Formalizing the discipline of "Prompt Engineering", e.g.
    - Learning to "program" using natural language



See en.wikipedia.org/wiki/Prompt_engineering

Impact on AI-Augmented Software Development

- Some key R&D challenges include
  - Training large-language models on better vetted & robust code bases
  - Re-envisioning the software development lifecycle (SDLC)
  - Effectively capturing & leveraging data generated throughout the SDLC
  - Formalizing the discipline of "Prompt Engineering", e.g.
    - Learning to "program" using natural language
    - Codifying "prompt patterns"
      - A knowledge transfer method for interacting w/large-language models (LLMs)



A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT

Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert,
Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt
Department of Computer Science
Vanderbilt University, Tennessee
Nashville, TN, USA
{jules.white, quchen.fu, george.s.hays, michael.sandborn, carlos.olea, henry.gilbert,
ashraf.elnashar, jesse.spencer-smith, douglas.c.schmidt}@vanderbilt.edu

***Abstract*—Prompt engineering is an increasingly important skill set needed to converse effectively with large language models (LLMs), such as ChatGPT. Prompts are instructions given to an LLM to enforce rules, automate processes, and ensure specific qualities (and quantities) of generated output. Prompts are also a form of programming that can customize the outputs and interactions with an LLM.**

This paper describes a catalog of prompt engineering techniques presented in pattern form that have been applied to solve common problems when conversing with LLMs. Prompt patterns are a knowledge transfer method analogous to software patterns since they provide reusable solutions to common problems faced in a particular context, i.e., output generation and interaction when working with LLMs.

This paper provides the following contributions to research on prompt engineering that apply LLMs to automate software development tasks. First, it provides a framework for documenting patterns for structuring prompts to solve a range of problems so that they can be adapted to different domains. Second, it presents a catalog of patterns that have been applied successfully to improve the outputs of LLM conversations. Third, it explains how prompts can be built from multiple patterns and illustrates prompt patterns that benefit from combination with other prompt patterns.

***Index Terms*—large language models, prompt patterns, prompt engineering**

I. INTRODUCTION

Conversational large language models (LLMs) [1], such as ChatGPT [2], have generated immense interest in a range of domains for tasks ranging from answering questions on medical licensing exams [3] to generating code snippets. This paper focuses on enhancing the application of LLMs in several domains, such as helping developers code effectively and efficiently with unfamiliar APIs or allowing students to acquire new coding skills and techniques.

LLMs are particularly promising in domains where humans and AI tools work together as trustworthy collaborators to more rapidly and reliably evolve software-reliant systems [4]. For example, LLMs are being integrated directly into software tools, such as Github's Co-Pilot [5]–[7] and included in integrated development environments (IDEs), such as IntelliJ [8] and Visual Studio Code, thereby allowing software teams to access these tools directly from their preferred IDE.

A prompt [9] is a set of instructions provided to an LLM that programs the LLM by customizing it and/or enhancing or refining its capabilities. A prompt can influence subsequent interactions with—and output generated from—an LLM by providing specific rules and guidelines for an LLM conversation with a set of initial rules. In particular, a prompt sets the context for the conversation and tells the LLM what information is important and what the desired output form and content should be.

For example, a prompt could specify that an LLM should only generate code that follows a certain coding style or programming paradigm. Likewise, it could specify that an LLM should flag certain keywords or phrases in a generated document and provide additional information related to those keywords. By introducing these guidelines, prompts facilitate more structured and nuanced outputs to aid a large variety of software engineering tasks in the context of LLMs.

Prompt engineering is the means by which LLMs are programmed via prompts. To demonstrate the power of prompt engineering, we provide the following prompt:

> **Prompt:** "From now on, I would like you to ask me questions to deploy a Python application to AWS. When you have enough information to deploy the application, create a Python script to automate the deployment."

This example prompt causes ChatGPT to begin asking the user questions about their software application. ChatGPT will drive the question-asking process until it reaches a point where it has sufficient information to generate a Python script that automates deployment. This example demonstrates the programming potential of prompts beyond conventional "generate a method that does X" style prompts or "answer this quiz question".

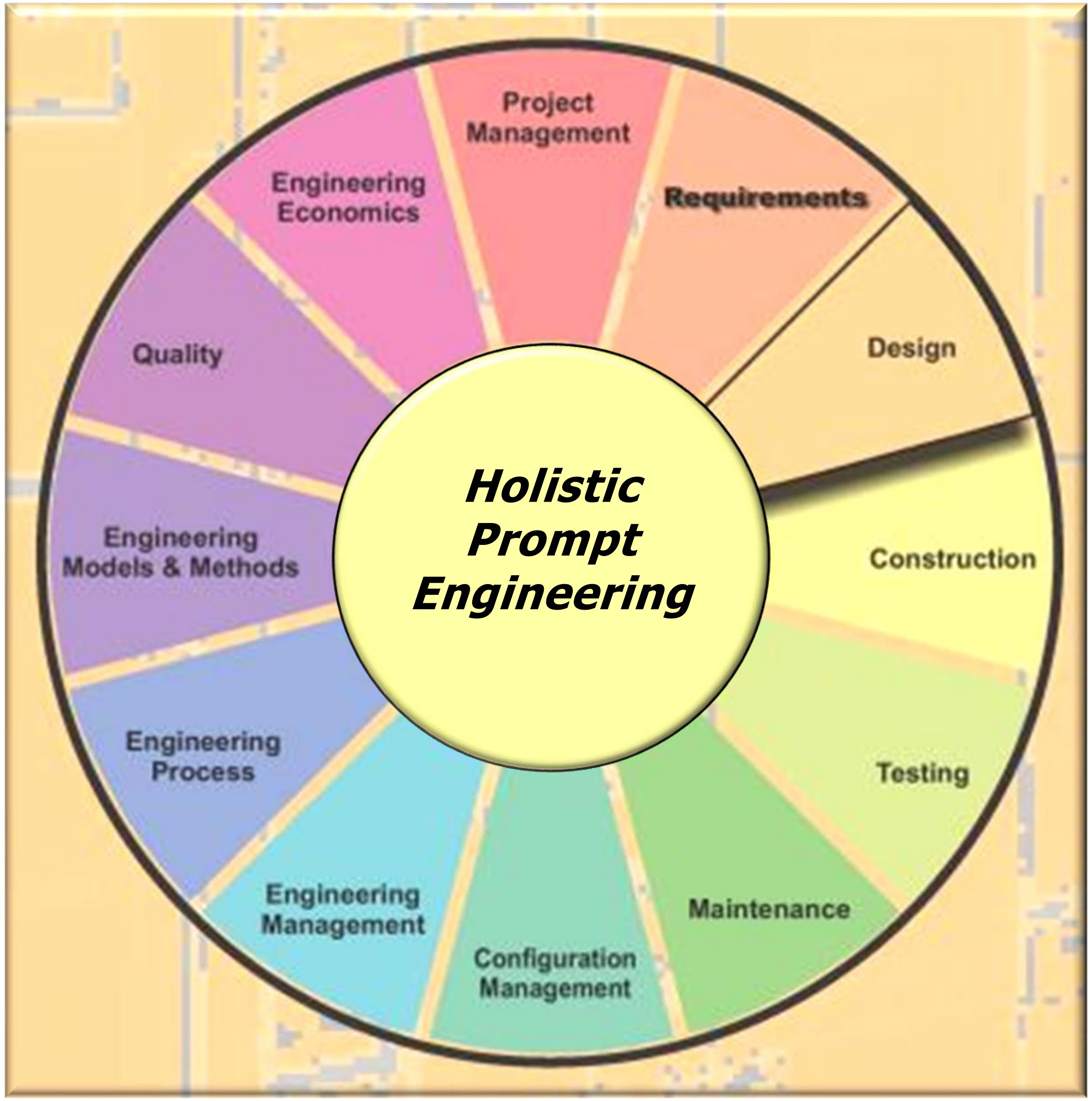
Moreover, prompts can be engineered to program an LLM to accomplish much more than simply dictating the output type or filtering the information provided to the model. With the right prompt, it is possible to create entirely new interaction paradigms, such as having an LLM generate and give a quiz associated with a software engineering concept or tool, or even simulate a Linux terminal window. Moreover, prompts have the potential for self-adaptation, suggesting other prompts to gather additional information or generate related artifacts. These advanced capabilities of prompts highlight the importance of engineering them to provide value beyond simple text or code generation.

Prompt patterns are essential to effective prompt engineering. A key contribution of this paper is the introduction of *prompt patterns* to document successful approaches for

See arxiv.org/abs/2302.11382

Impact on AI-Augmented Software Development

- Some key R&D challenges include
  - Training large-language models on better vetted & robust code bases
  - Re-envisioning the software development lifecycle (SDLC)
  - Effectively capturing & leveraging data generated throughout the SDLC
  - Formalizing the discipline of "Prompt Engineering", e.g.
    - Learning to "program" using natural language
    - Codifying "prompt patterns"
    - Integrating canonical quality attributes associated with software engineering

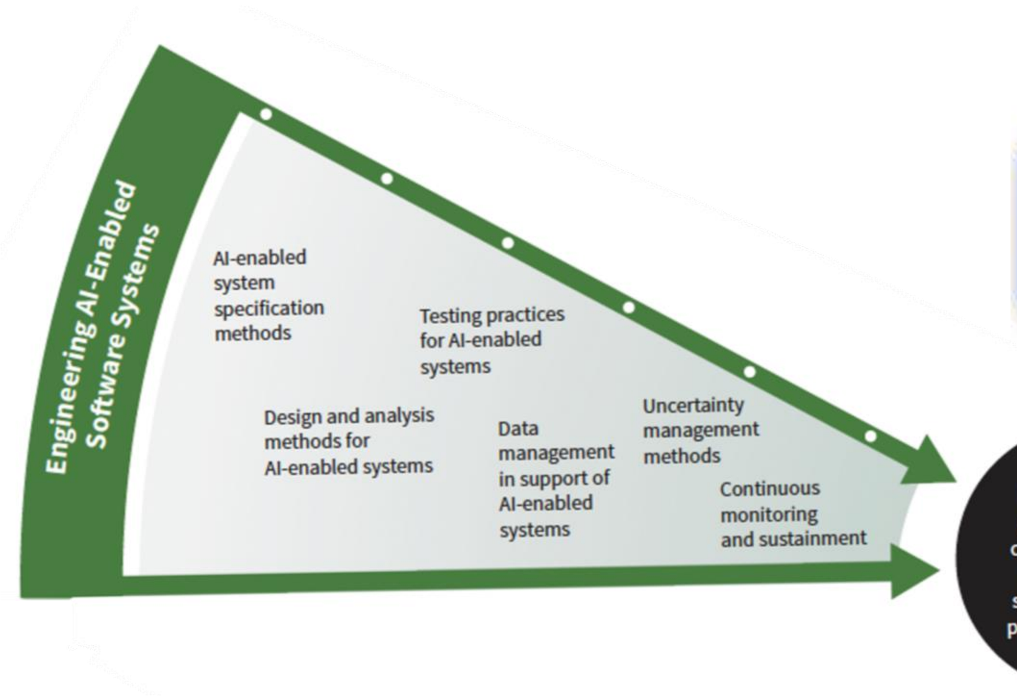


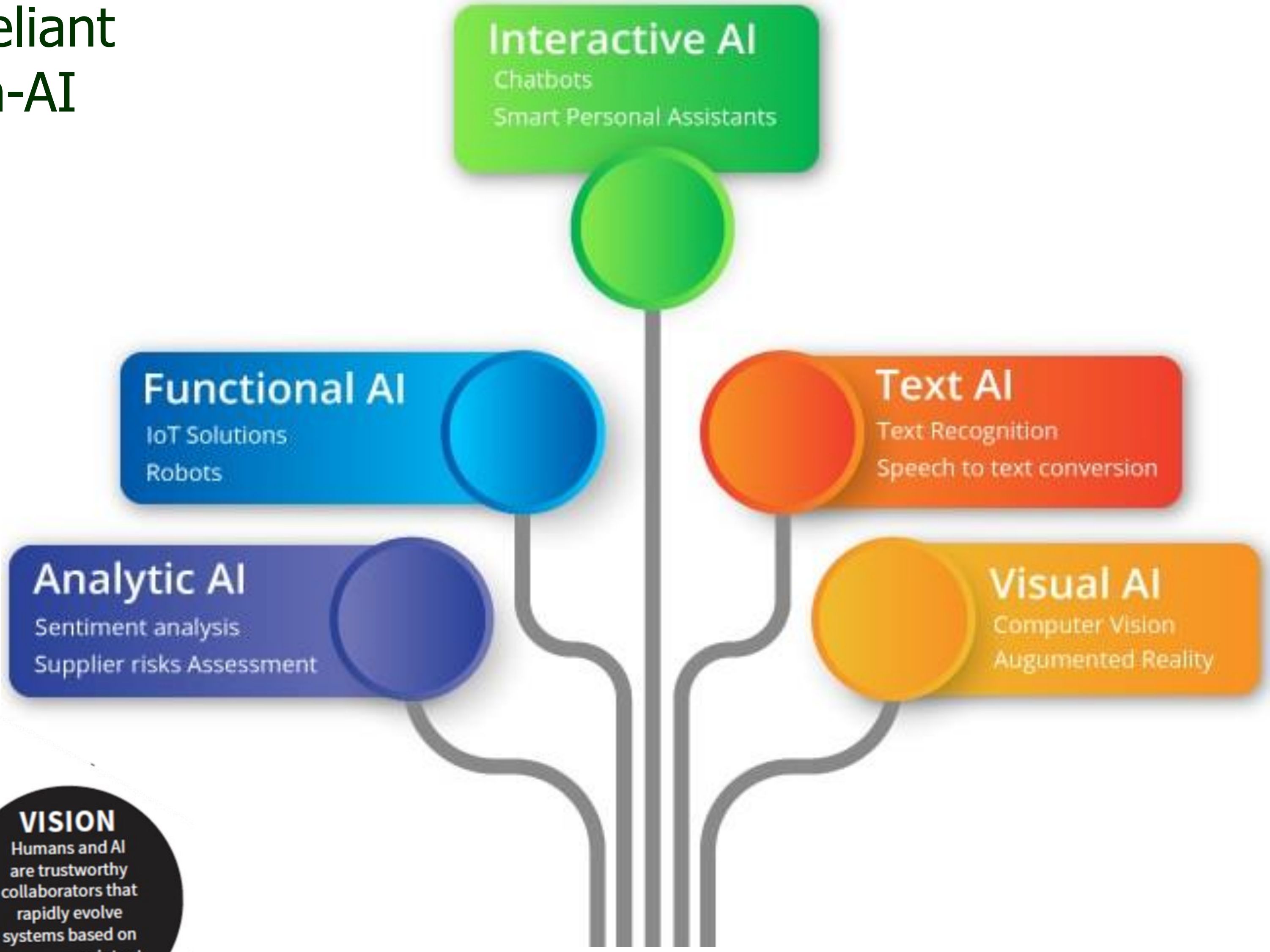
See hbr.org/2023/06/ai-prompt-engineering-isnt-the-future

Impact on Engineering AI-enabled Software Systems

Impact on Engineering AI-enabled Software Systems

- AI-enabled systems are software-reliant systems that include both AI & non-AI components


Engineering AI-Enabled Software Systems
AI-enabled system specification methods
Testing practices for AI-enabled systems
Design and analysis methods for AI-enabled systems
Data management in support of AI-enabled systems
Uncertainty management methods
Continuous monitoring and sustainment
VISION
Humans and AI are trustworthy collaborators that rapidly evolve systems based on programmer intent

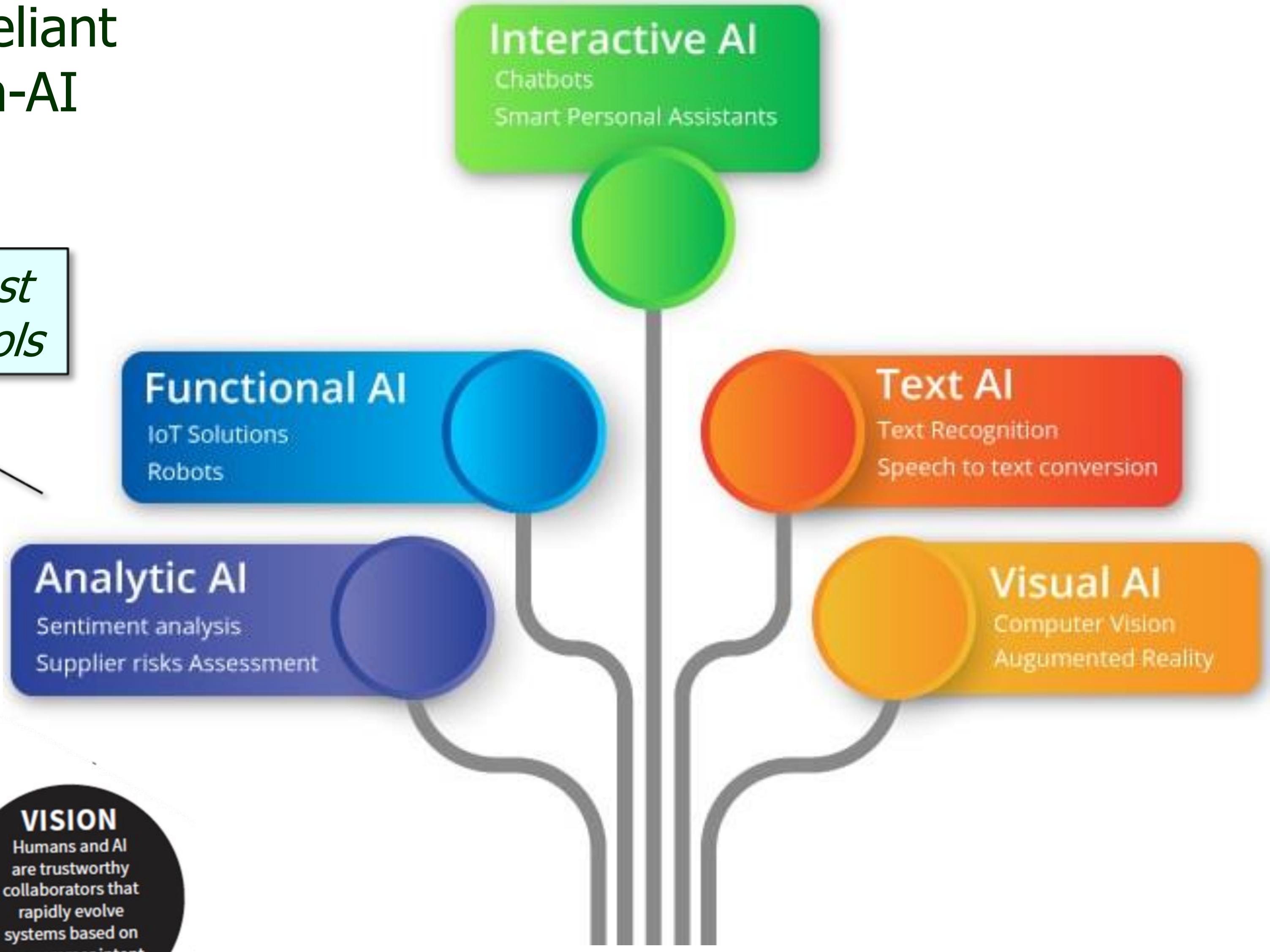


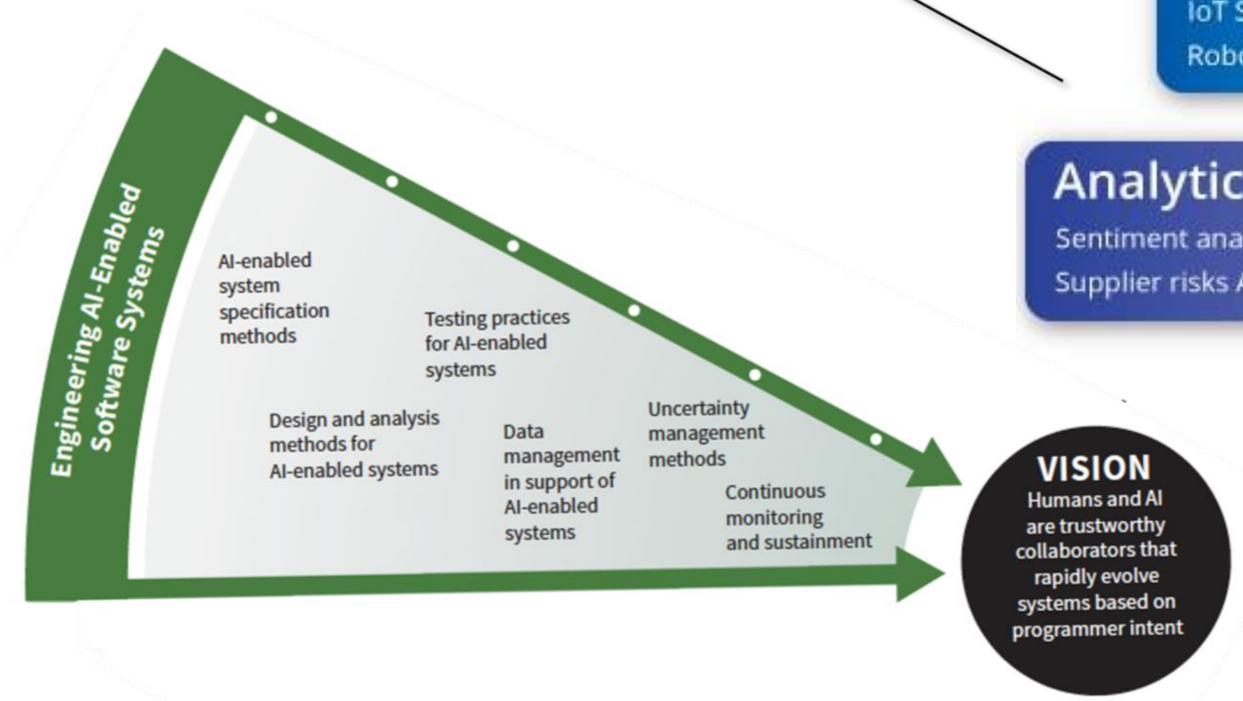
See readitquik.com/articles/ai/7-different-types-of-artificial-intelligence-explained

Impact on Engineering AI-enabled Software Systems

- AI-enabled systems are software-reliant systems that include both AI & non-AI components

These AI components go well beyond just generative AI techniques, methods, & tools




Engineering AI-Enabled Software Systems
AI-enabled system specification methods
Testing practices for AI-enabled systems
Design and analysis methods for AI-enabled systems
Data management in support of AI-enabled systems
Uncertainty management methods
Continuous monitoring and sustainment
VISION
Humans and AI are trustworthy collaborators that rapidly evolve systems based on programmer intent

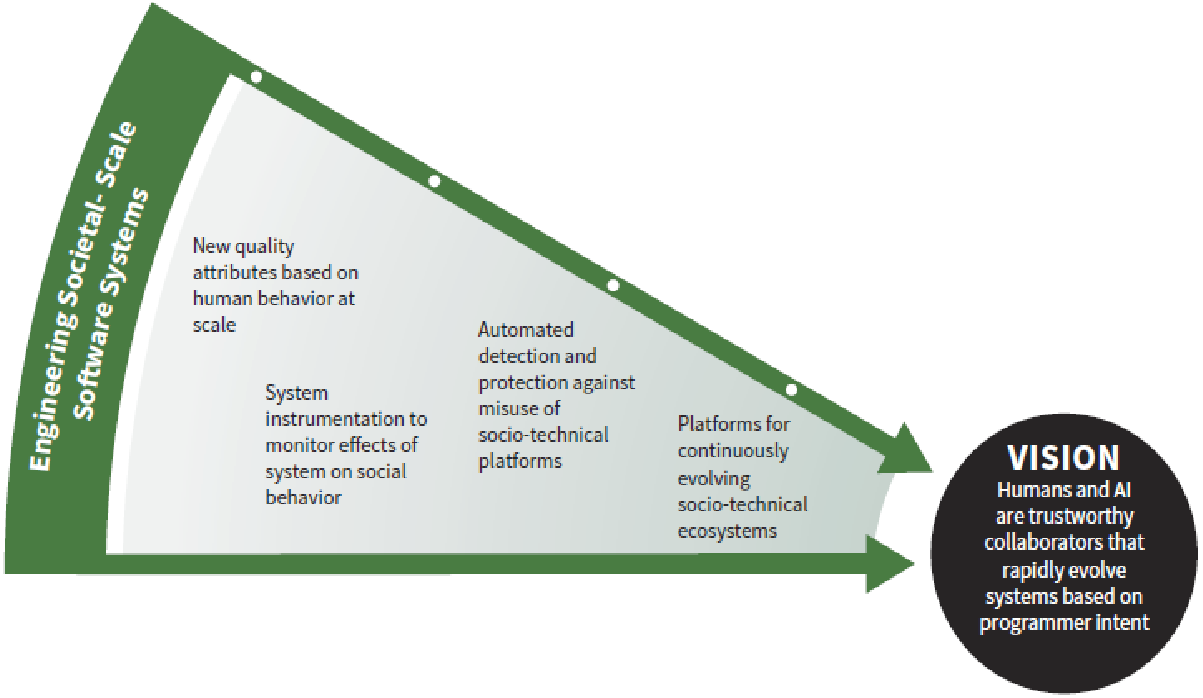
See readitquik.com/articles/ai/7-different-types-of-artificial-intelligence-explained

Impact on Engineering AI-enabled Software Systems

- AI-enabled systems are software-reliant systems that include both AI & non-AI components, e.g.,
  - Infotainment systems

Impact on Engineering AI-enabled Software Systems

- AI-enabled systems are software-reliant systems that include both AI & non-AI components, e.g.,
  - Infotainment systems



New quality attributes based on human behavior at scale

System instrumentation to monitor effects of system on social behavior

Automated detection and protection against misuse of socio-technical platforms

Platforms for continuously evolving socio-technical ecosystems

VISION
Humans and AI are trustworthy collaborators that rapidly evolve systems based on programmer intent

These types of systems are covered in other parts of the SEI’s National Agenda Study

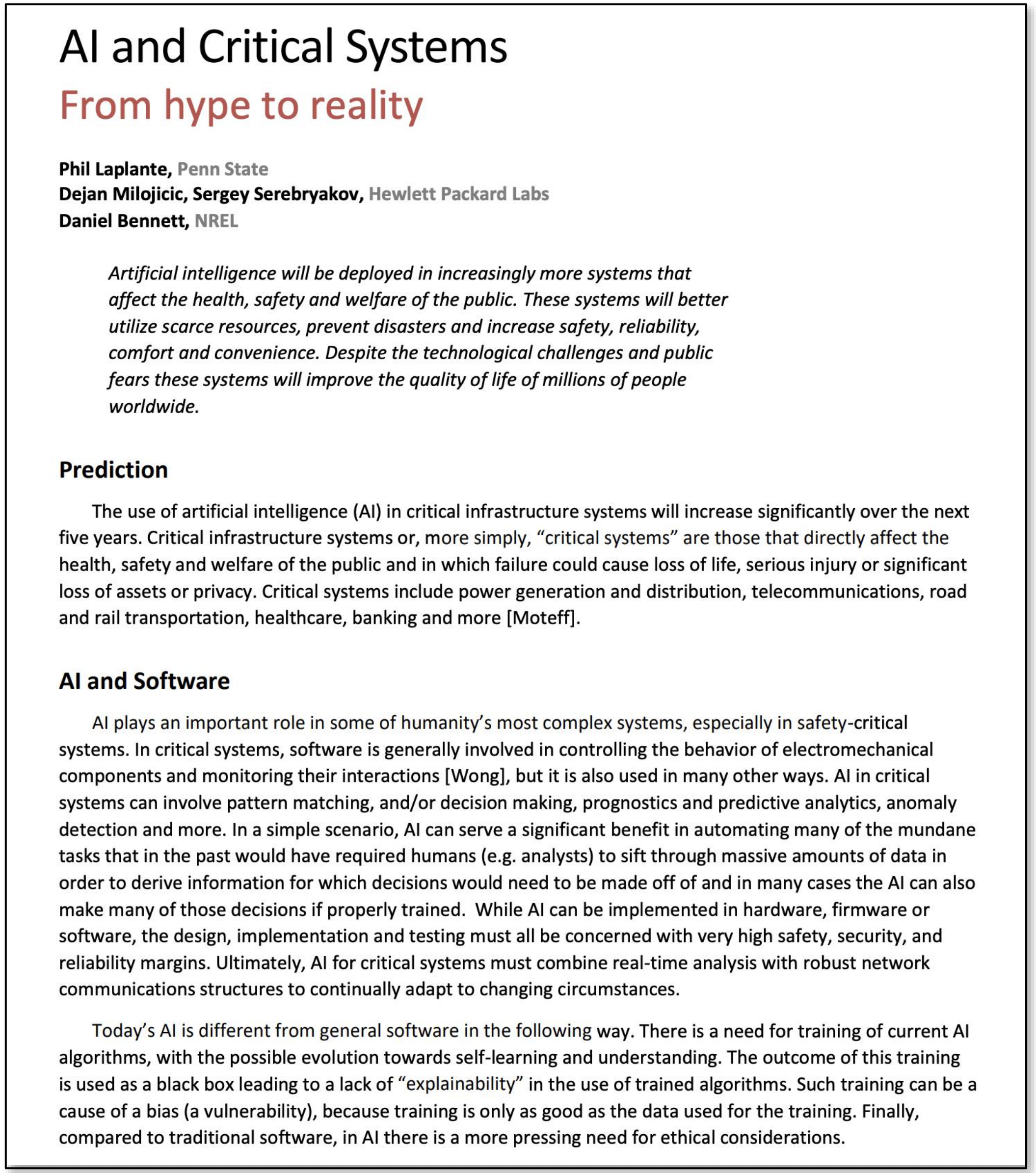
Impact on Engineering AI-enabled Software Systems

- AI-enabled systems are software-reliant systems that include both AI & non-AI components, e.g.,
  - Infotainment systems
    - e.g., YouTube is a multimedia content delivery platform that uses AI for recommendations, captioning, moderation, analysis, advertising, etc.

See influencermarketinghub.com/how-does-the-youtube-algorithm-work

Impact on Engineering AI-enabled Software Systems

- AI-enabled systems are software-reliant systems that include both AI & non-AI components, e.g.,
  - Infotainment systems
  - Mission-critical systems



AI and Critical Systems

From hype to reality

Phil Laplante, Penn State
Dejan Milojicic, Sergey Serebryakov, Hewlett Packard Labs
Daniel Bennett, NREL

Artificial intelligence will be deployed in increasingly more systems that affect the health, safety and welfare of the public. These systems will better utilize scarce resources, prevent disasters and increase safety, reliability, comfort and convenience. Despite the technological challenges and public fears these systems will improve the quality of life of millions of people worldwide.

Prediction

The use of artificial intelligence (AI) in critical infrastructure systems will increase significantly over the next five years. Critical infrastructure systems or, more simply, "critical systems" are those that directly affect the health, safety and welfare of the public and in which failure could cause loss of life, serious injury or significant loss of assets or privacy. Critical systems include power generation and distribution, telecommunications, road and rail transportation, healthcare, banking and more [Moteff].

AI and Software

AI plays an important role in some of humanity's most complex systems, especially in safety-critical systems. In critical systems, software is generally involved in controlling the behavior of electromechanical components and monitoring their interactions [Wong], but it is also used in many other ways. AI in critical systems can involve pattern matching, and/or decision making, prognostics and predictive analytics, anomaly detection and more. In a simple scenario, AI can serve a significant benefit in automating many of the mundane tasks that in the past would have required humans (e.g. analysts) to sift through massive amounts of data in order to derive information for which decisions would need to be made off of and in many cases the AI can also make many of those decisions if properly trained. While AI can be implemented in hardware, firmware or software, the design, implementation and testing must all be concerned with very high safety, security, and reliability margins. Ultimately, AI for critical systems must combine real-time analysis with robust network communications structures to continually adapt to changing circumstances.

Today's AI is different from general software in the following way. There is a need for training of current AI algorithms, with the possible evolution towards self-learning and understanding. The outcome of this training is used as a black box leading to a lack of "explainability" in the use of trained algorithms. Such training can be a cause of a bias (a vulnerability), because training is only as good as the data used for the training. Finally, compared to traditional software, in AI there is a more pressing need for ethical considerations.

See www.osti.gov/servlets/purl/1713282

Impact on Engineering AI-enabled Software Systems

- AI-enabled systems are software-reliant systems that include both AI & non-AI components, e.g.,
  - Infotainment systems
  - Mission-critical systems
    - e.g., Palantir’s AI Platform (AIP) applies generative AI models to support battle management decision-making processes

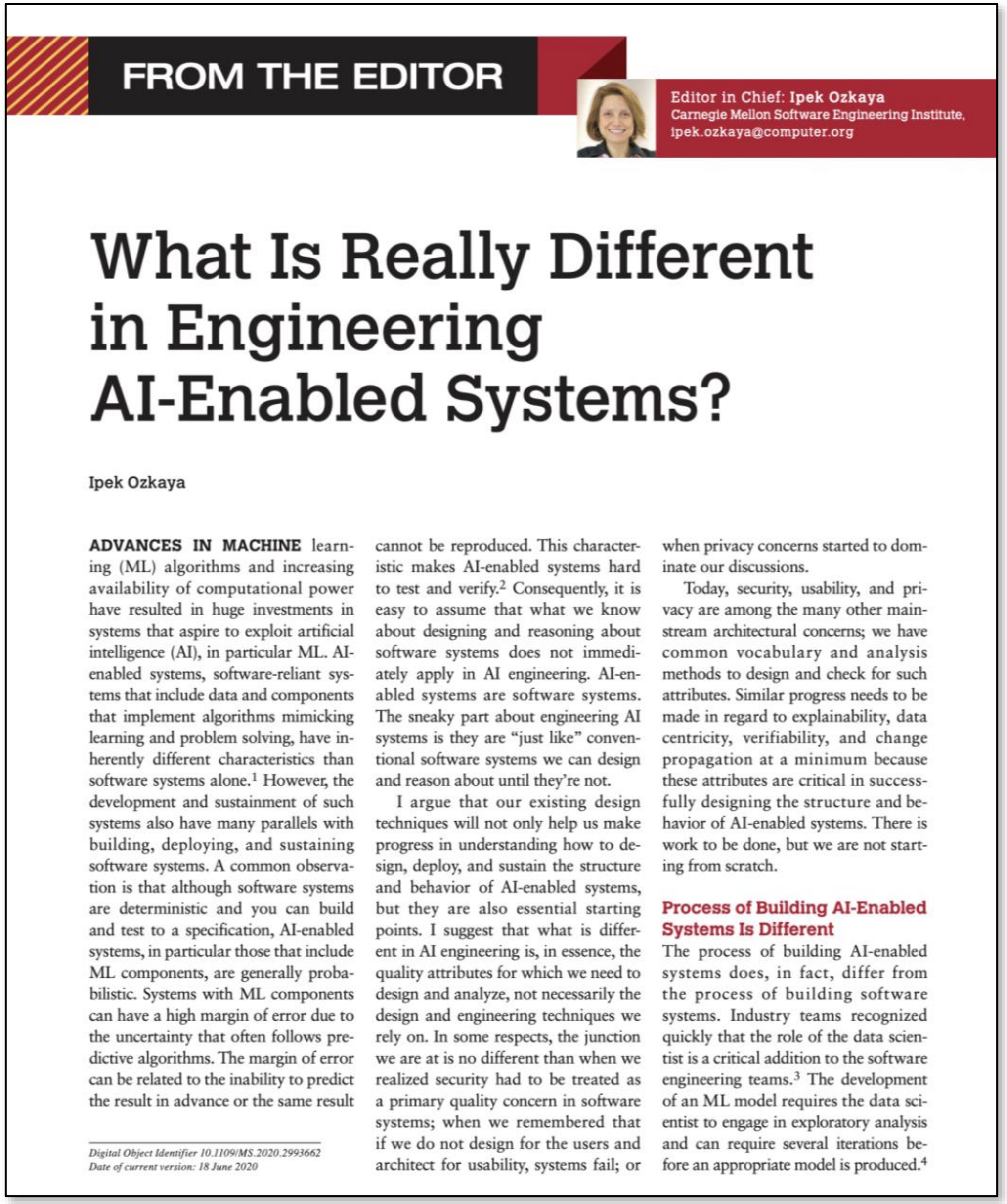
Palantir shows off an AI that can go to war

The system aims to serve as an 'ethical' and 'legal' veneer for future military automation efforts.

See www.engadget.com/palantir-shows-off-an-ai-that-can-go-to-war-180513781.html

Impact on Engineering AI-enabled Software Systems

- AI-enabled software systems have inherently different characteristics than those without AI



FROM THE EDITOR

Editor in Chief: **Ipek Ozkaya**
Carnegie Mellon Software Engineering Institute,
ipek.ozkaya@computer.org

What Is Really Different in Engineering AI-Enabled Systems?

Ipek Ozkaya

ADVANCES IN MACHINE learning (ML) algorithms and increasing availability of computational power have resulted in huge investments in systems that aspire to exploit artificial intelligence (AI), in particular ML. AI-enabled systems, software-reliant systems that include data and components that implement algorithms mimicking learning and problem solving, have inherently different characteristics than software systems alone.[1] However, the development and sustainment of such systems also have many parallels with building, deploying, and sustaining software systems. A common observation is that although software systems are deterministic and you can build and test to a specification, AI-enabled systems, in particular those that include ML components, are generally probabilistic. Systems with ML components can have a high margin of error due to the uncertainty that often follows predictive algorithms. The margin of error can be related to the inability to predict the result in advance or the same result cannot be reproduced. This characteristic makes AI-enabled systems hard to test and verify.[2] Consequently, it is easy to assume that what we know about designing and reasoning about software systems does not immediately apply in AI engineering. AI-enabled systems are software systems. The sneaky part about engineering AI systems is they are “just like” conventional software systems we can design and reason about until they’re not.

I argue that our existing design techniques will not only help us make progress in understanding how to design, deploy, and sustain the structure and behavior of AI-enabled systems, but they are also essential starting points. I suggest that what is different in AI engineering is, in essence, the quality attributes for which we need to design and analyze, not necessarily the design and engineering techniques we rely on. In some respects, the junction we are at is no different than when we realized security had to be treated as a primary quality concern in software systems; when we remembered that if we do not design for the users and architect for usability, systems fail; or when privacy concerns started to dominate our discussions.

Today, security, usability, and privacy are among the many other mainstream architectural concerns; we have common vocabulary and analysis methods to design and check for such attributes. Similar progress needs to be made in regard to explainability, data centricity, verifiability, and change propagation at a minimum because these attributes are critical in successfully designing the structure and behavior of AI-enabled systems. There is work to be done, but we are not starting from scratch.

Process of Building AI-Enabled Systems Is Different

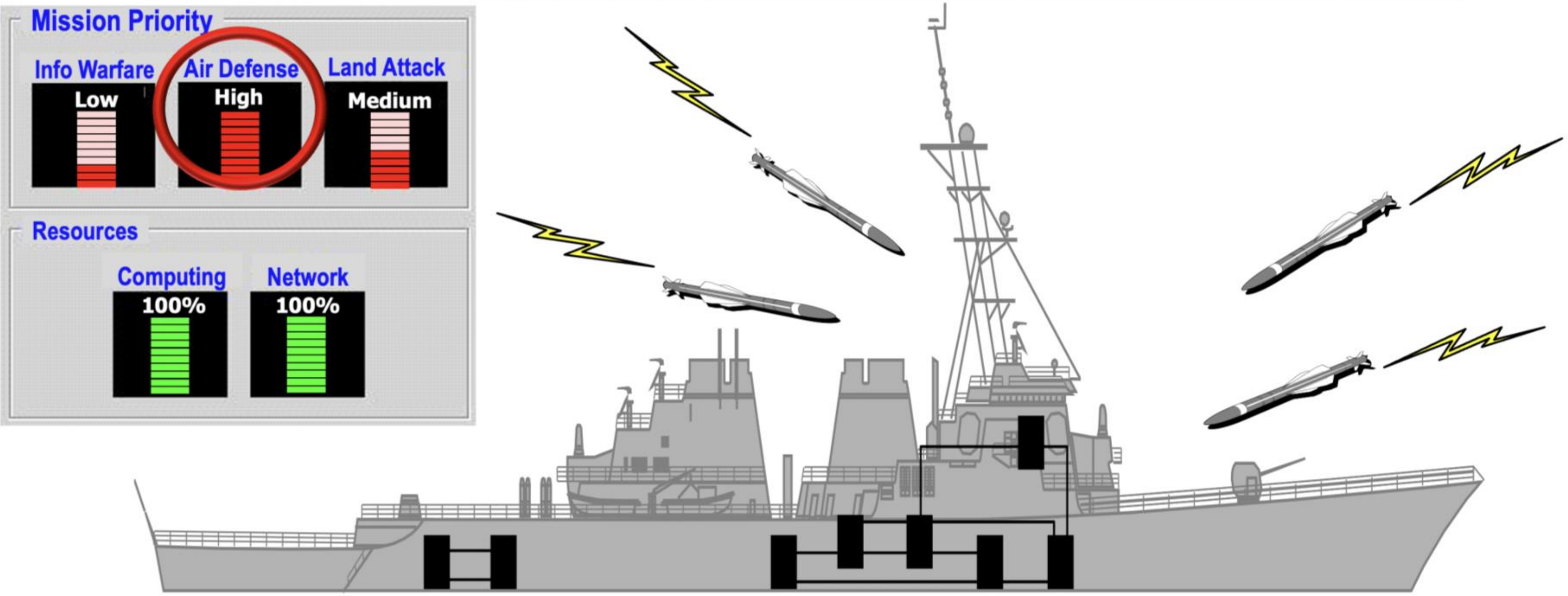
The process of building AI-enabled systems does, in fact, differ from the process of building software systems. Industry teams recognized quickly that the role of the data scientist is a critical addition to the software engineering teams.[3] The development of an ML model requires the data scientist to engage in exploratory analysis and can require several iterations before an appropriate model is produced.[4]

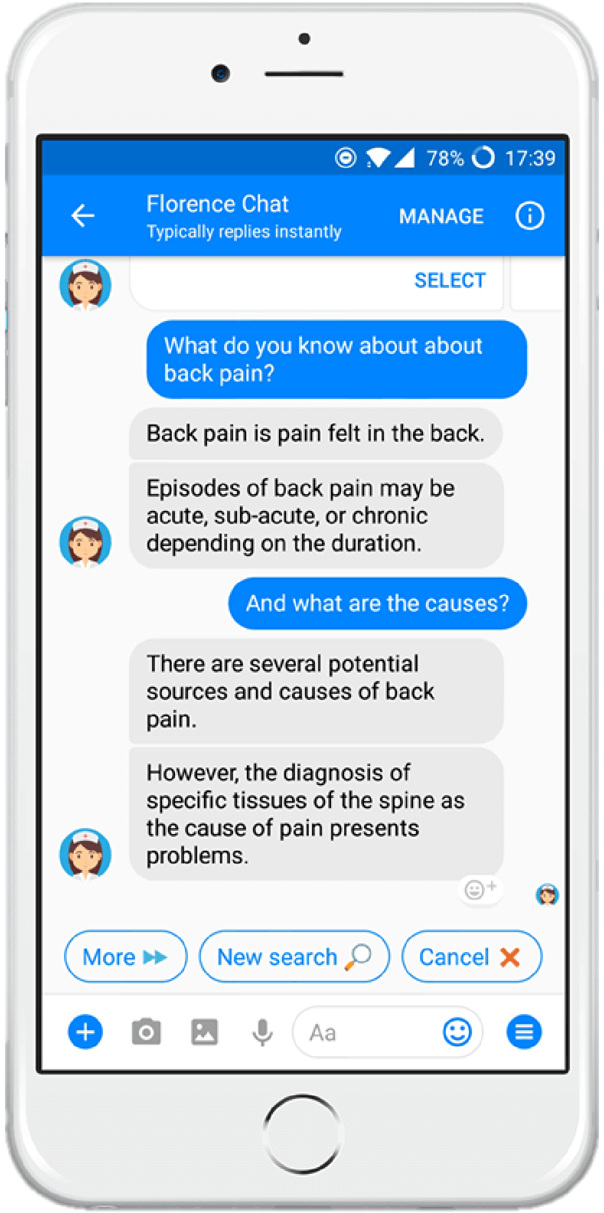
Digital Object Identifier 10.1109/MS.2020.2993662
Date of current version: 18 June 2020

See ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9121629

Impact on Engineering AI-enabled Software Systems

- AI-enabled software systems have inherently different characteristics than those without AI, e.g.
  - Non-deterministic
    - Sometimes intentionally & sometimes stochastically





What do you know about about back pain?

Back pain is pain felt in the back.

Episodes of back pain may be acute, sub-acute, or chronic depending on the duration.

And what are the causes?

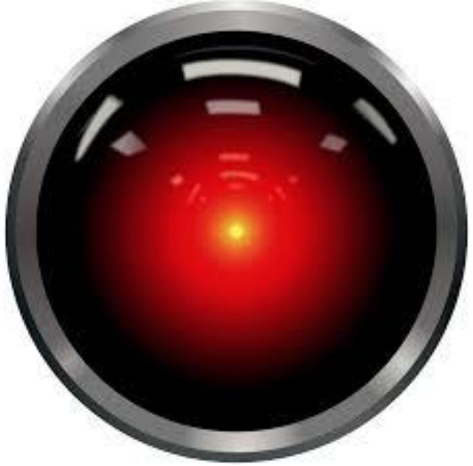
There are several potential sources and causes of back pain.

However, the diagnosis of specific tissues of the spine as the cause of pain presents problems.

See www.dre.vanderbilt.edu/~schmidt/PDF/autonomic-journal.pdfs

Impact on Engineering AI-enabled Software Systems

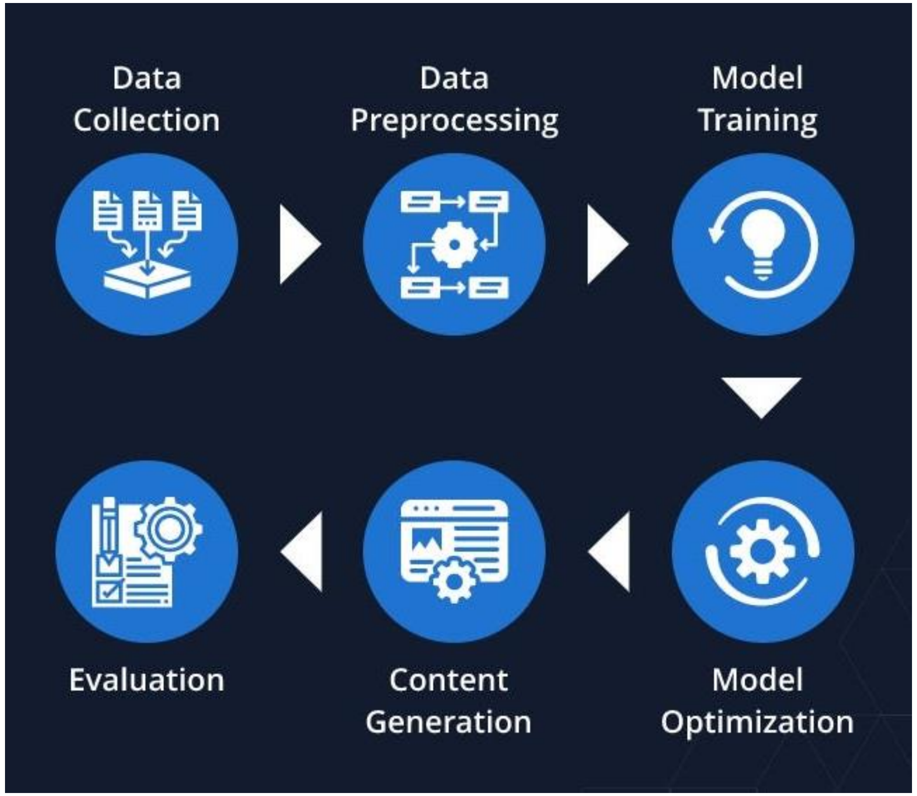
- AI-enabled software systems have inherently different characteristics than those without AI, e.g.
  - Non-deterministic
  - Emergent behavior
    - Complex patterns that arise spontaneously from the interactions of simpler elements or systems



See www.digitaltrends.com/cool-tech/recognizing-self-aware-sentient-ai

Impact on Engineering AI-enabled Software Systems

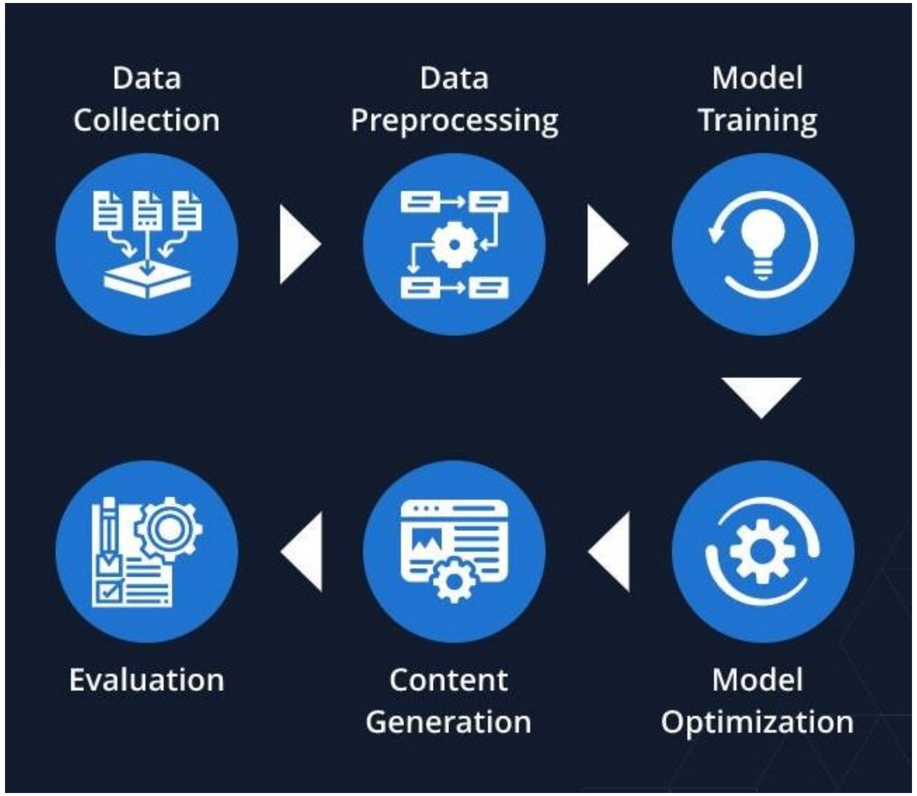
- AI-enabled software systems have inherently different characteristics than those without AI, e.g.
  - Non-deterministic
  - Emergent behavior
  - Extreme dependence on data
    - The quality, quantity, & depth of data determine the capabilities of AI applications that can be achieved



See developers.google.com/machine-learning/crash-course/data-dependencies/video-lecture

Impact on Engineering AI-enabled Software Systems

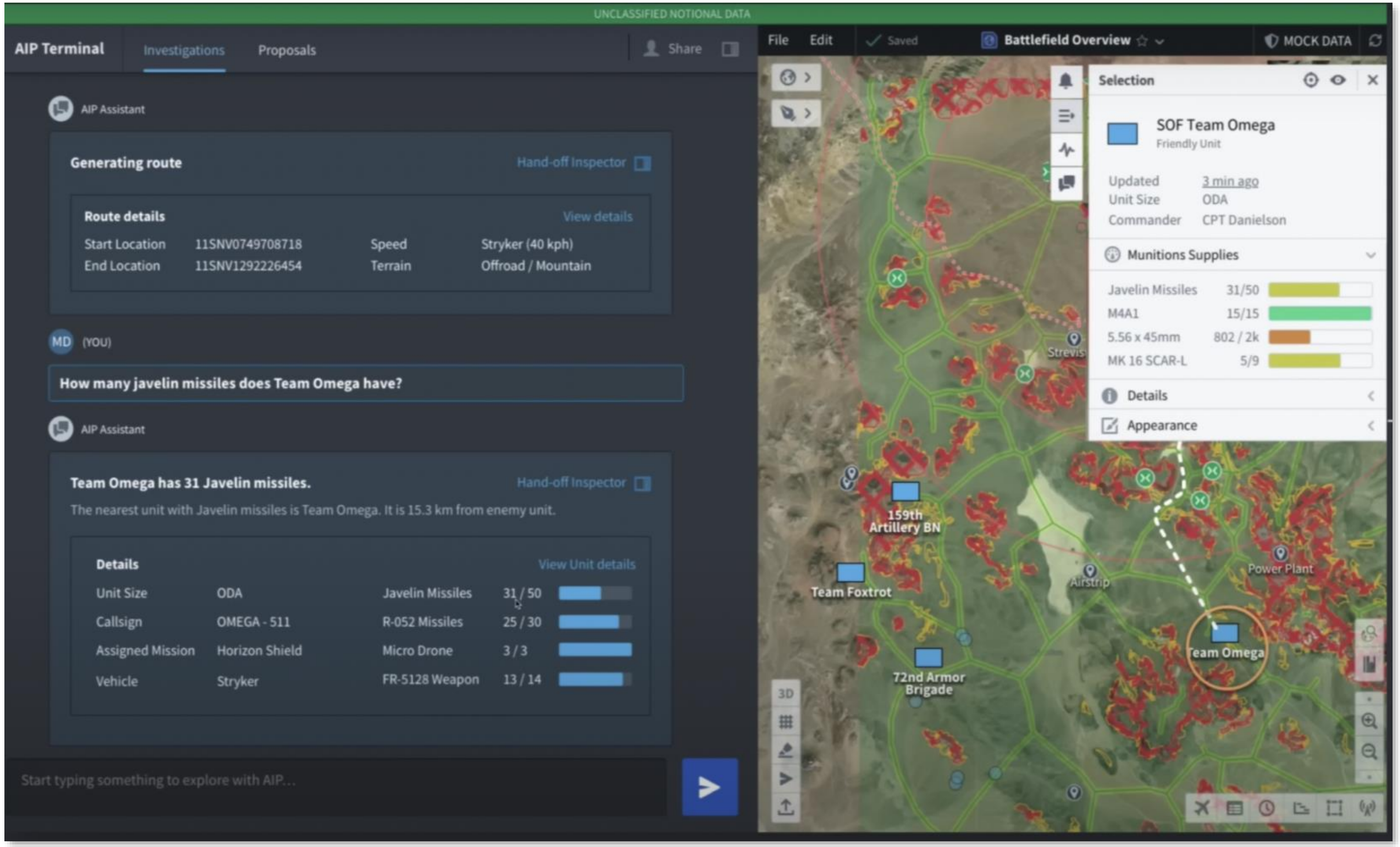
- AI-enabled software systems have inherently different characteristics than those without AI, e.g.
  - Non-deterministic
  - Emergent behavior
  - Extreme dependence on data
    - The quality, quantity, & depth of data determine the capabilities of AI applications that can be achieved
      - Testing of data thus has become as important as testing of code



Impact on Engineering AI-enabled Software Systems

Palantir

- AI-enabled software systems have inherently different characteristics than those without AI, e.g.
  - Non-deterministic
  - Emergent behavior
  - Extreme dependence on data
  - Designing for explainability & fairness
    - e.g., show traceability from recommended courses of actions to mission goals, resources, evidence, rules of engagement, policies, etc.



AIP Assistant

Generating route

Route details

Start Location	11SNV0749708718	Speed	Stryker (40 kph)
End Location	11SNV1292226454	Terrain	Offroad / Mountain

MD (YOU)

How many javelin missiles does Team Omega have?

AIP Assistant

Team Omega has 31 Javelin missiles.

The nearest unit with Javelin missiles is Team Omega. It is 15.3 km from enemy unit.

Details

Unit Size	ODA	Javelin Missiles	31 / 50
Callsign	OMEGA - 511	R-052 Missiles	25 / 30
Assigned Mission	Horizon Shield	Micro Drone	3 / 3
Vehicle	Stryker	FR-5128 Weapon	13 / 14

Start typing something to explore with AIP...

Selection

SOF Team Omega
Friendly Unit

Updated	3 min ago
Unit Size	ODA
Commander	CPT Danielson

Munitions Supplies

Javelin Missiles	31/50
M4A1	15/15
5.56 x 45mm	802 / 2k
MK 16 SCAR-L	5/9

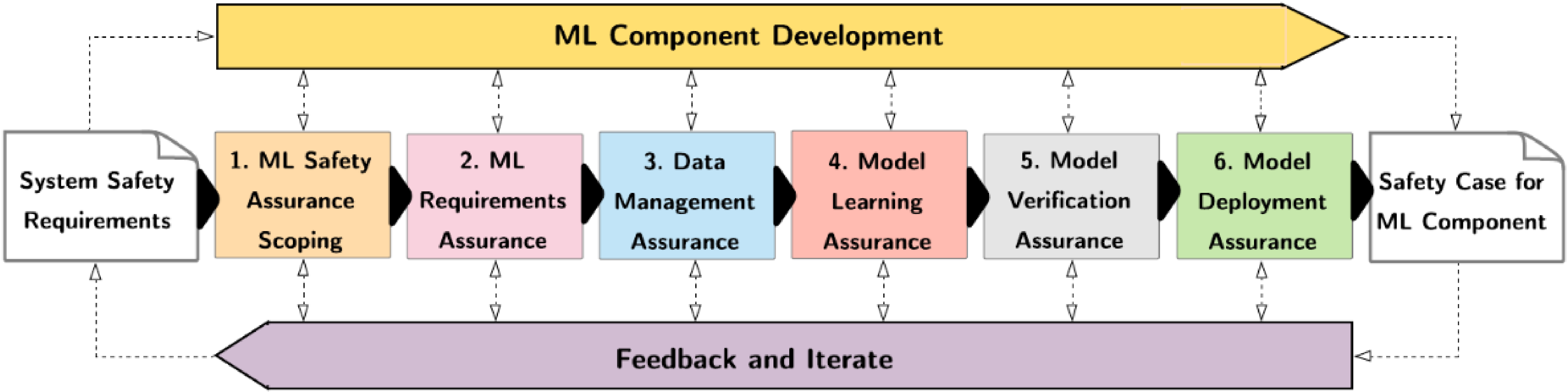
Details

Appearance

See www.youtube.com/watch?v=XEM5qz__HOU

Impact on Engineering AI-enabled Software Systems

- The R&D challenges include identifying & augmenting software engineering techniques for specifying, architecting, designing, analyzing, deploying, & sustaining AI-enabled software systems


ML Component Development
System Safety Assurance Requirements
1. ML Safety Assurance Scoping
2. ML Requirements Assurance
3. Data Management Assurance
4. Model Learning Assurance
5. Model Verification Assurance
6. Model Deployment Assurance
Safety Case for ML Component
Feedback and Iterate

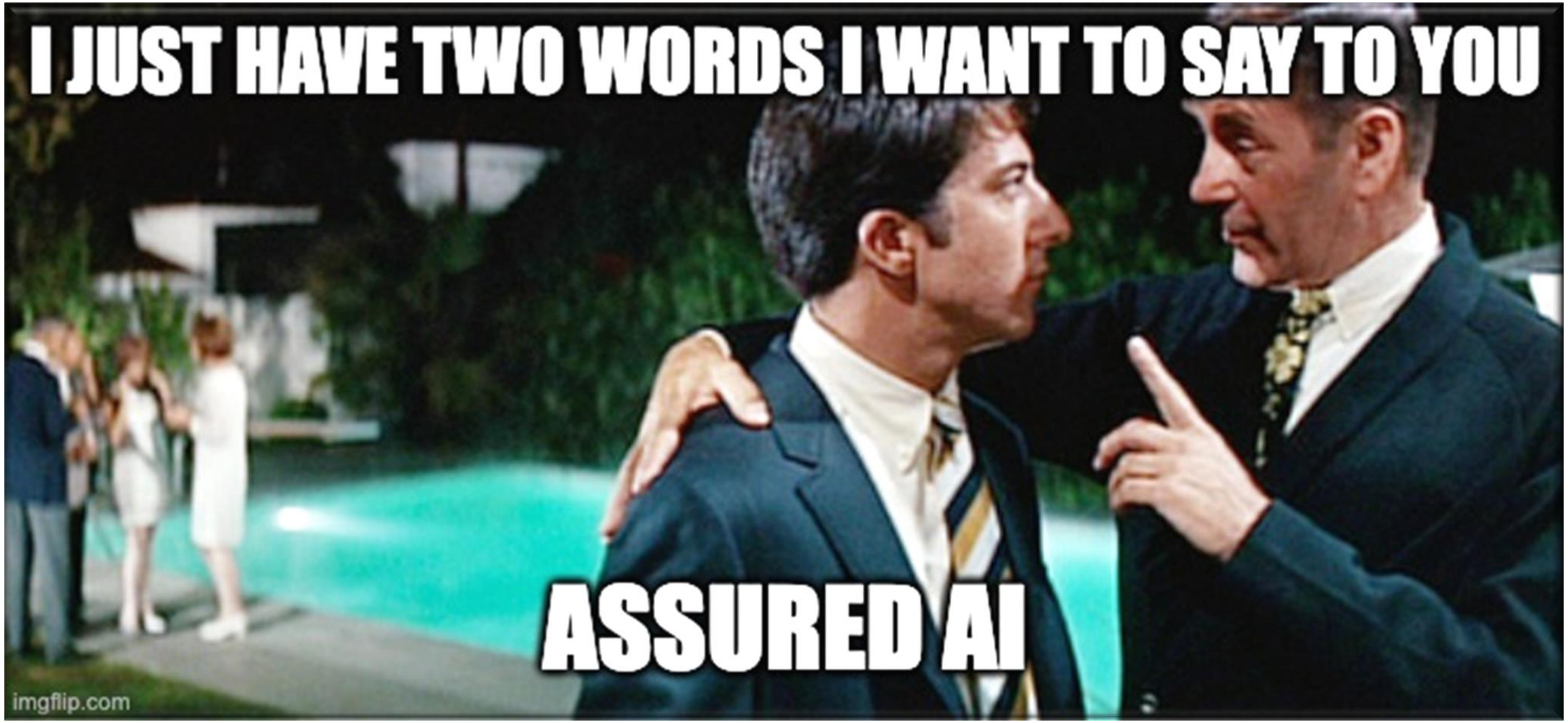
See also:

- Grace A. Lewis, Stephany Bellomo, & Ipek Ozkaya, "Characterizing and Detecting Mismatch in Machine-Learning-Enabled Systems," WAIN@ICSE 2021: 133-140
- Grace A. Lewis, Ipek Ozkaya, & Xiwei Xu, "Software Architecture Challenges for ML Systems," ICSME 2021: 634-638

See ira.lib.polyu.edu.hk/bitstream/10397/94631/1/Wang_Artificial_Intelligence_Safety-Critical.pdf

Impact on Engineering AI-enabled Software Systems

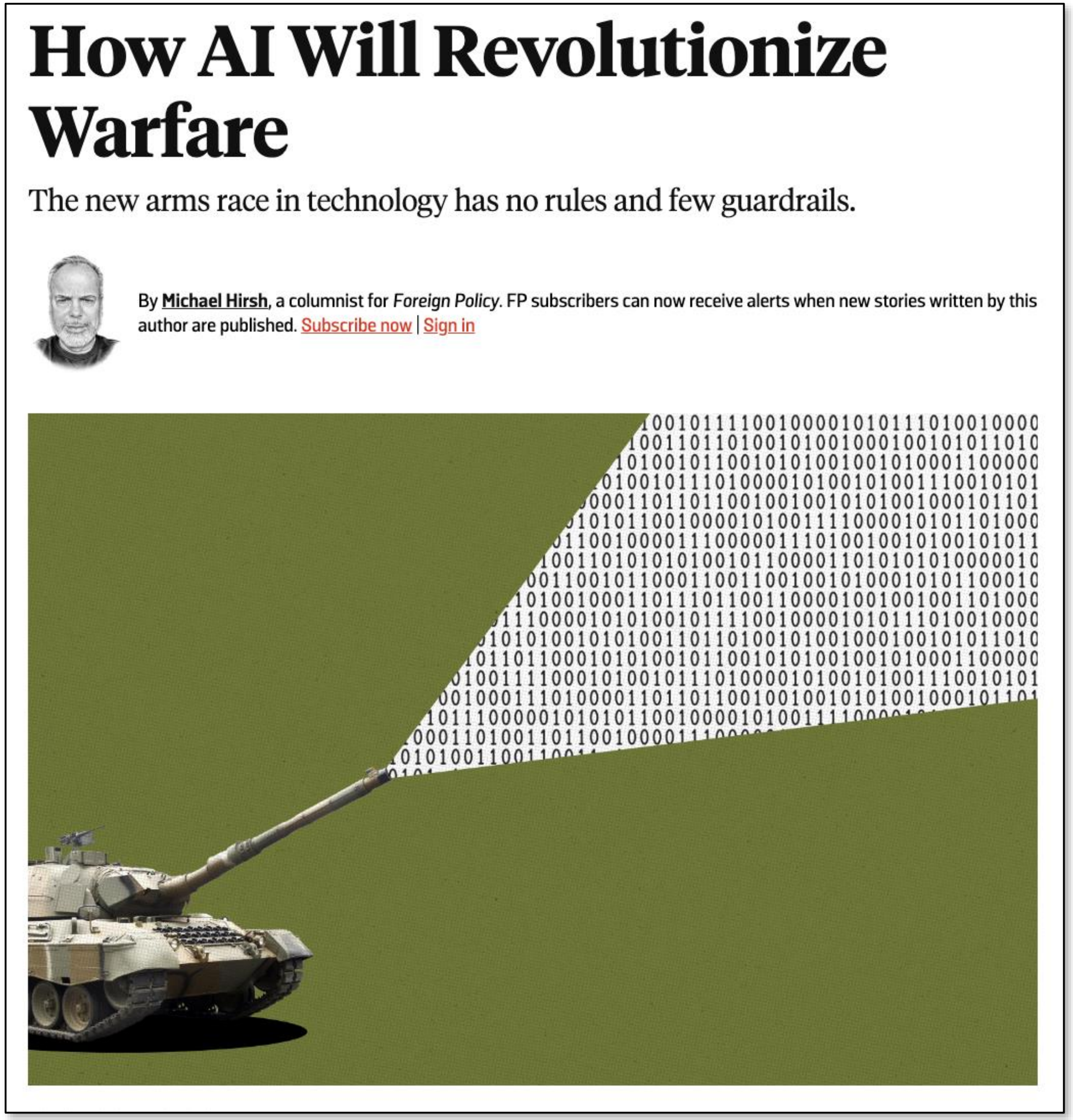
- The R&D challenges include identifying & augmenting software engineering techniques for specifying, architecting, designing, analyzing, deploying, & sustaining AI-enabled software systems



See www.newyorker.com/news/news-desk/plastics

Impact on Engineering AI-enabled Software Systems

- Addressing these R&D challenges is particularly important for the DoD



How AI Will Revolutionize Warfare

The new arms race in technology has no rules and few guardrails.

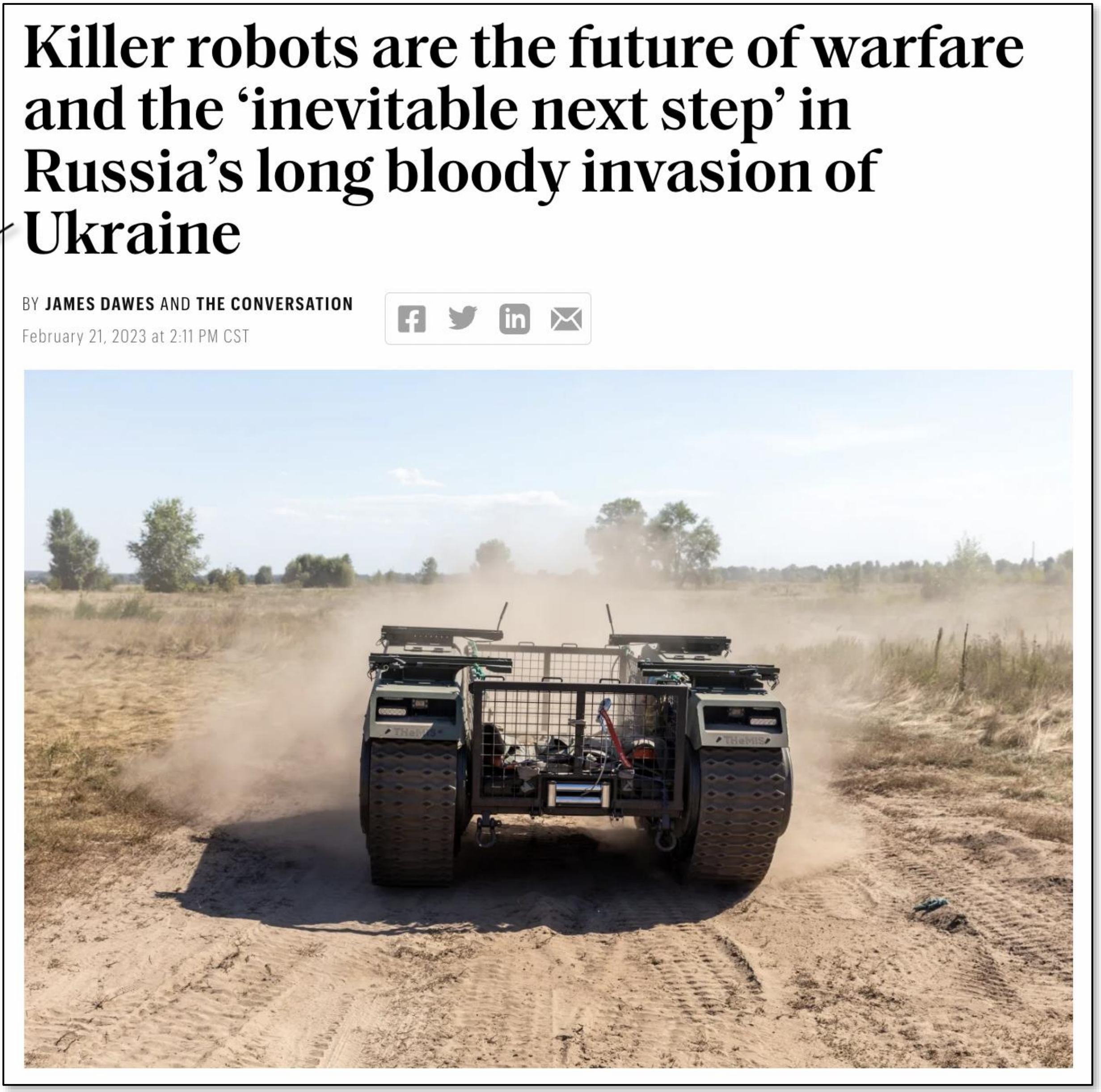
By **Michael Hirsh**, a columnist for *Foreign Policy*. FP subscribers can now receive alerts when new stories written by this author are published. Subscribe now | Sign in

See foreignpolicy.com/2023/04/11/ai-arms-race-artificial-intelligence-chatgpt-military-technology

Impact on Engineering AI-enabled Software Systems

- Addressing these R&D challenges is particularly important for the DoD

Not everyone is equally concerned with software-reliant system safety & avoiding mishaps & collateral damage..



Killer robots are the future of warfare and the 'inevitable next step' in Russia's long bloody invasion of Ukraine

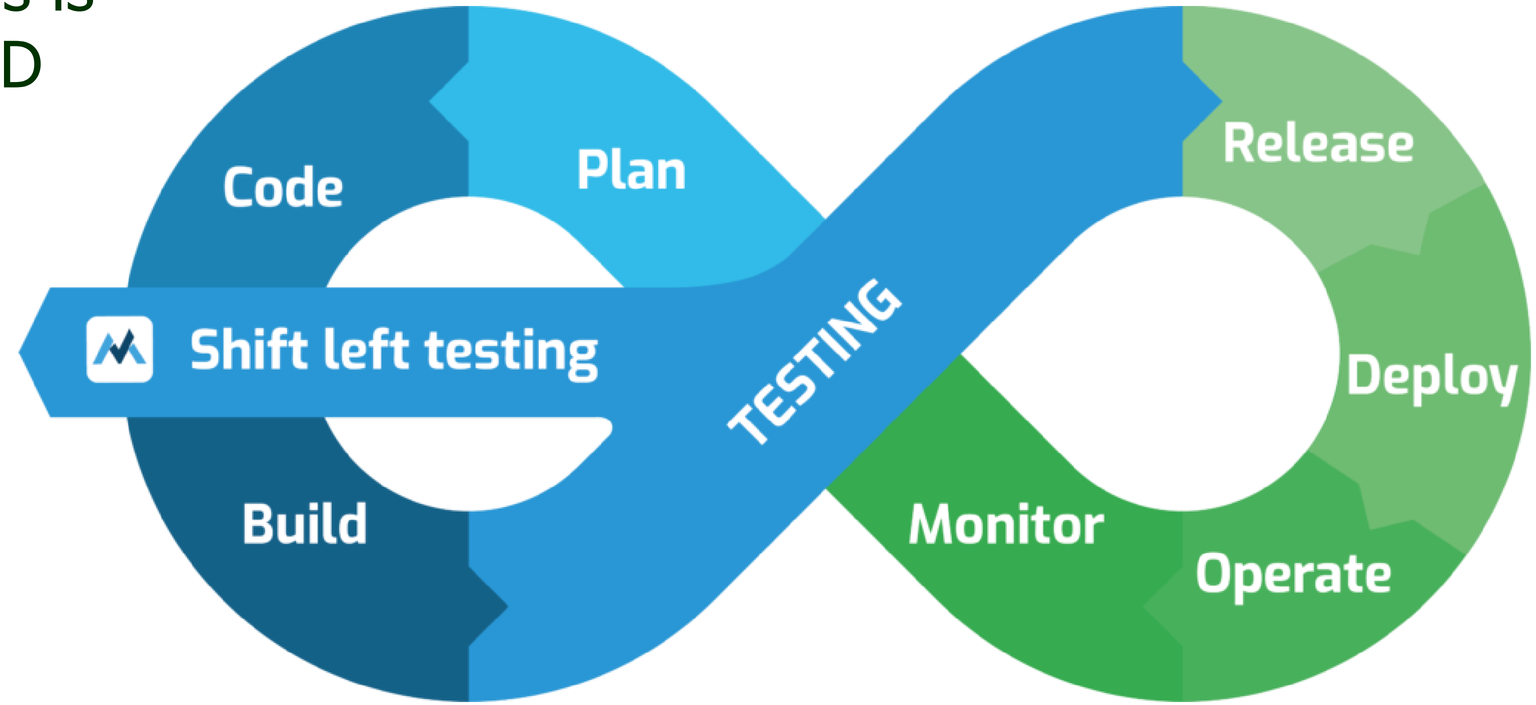
BY **JAMES DAWES** AND **THE CONVERSATION**

February 21, 2023 at 2:11 PM CST

See fortune.com/2023/02/21/killer-robots-a-i-future-warfare-russia-ukraine-invasion

Impact on Engineering AI-enabled Software Systems

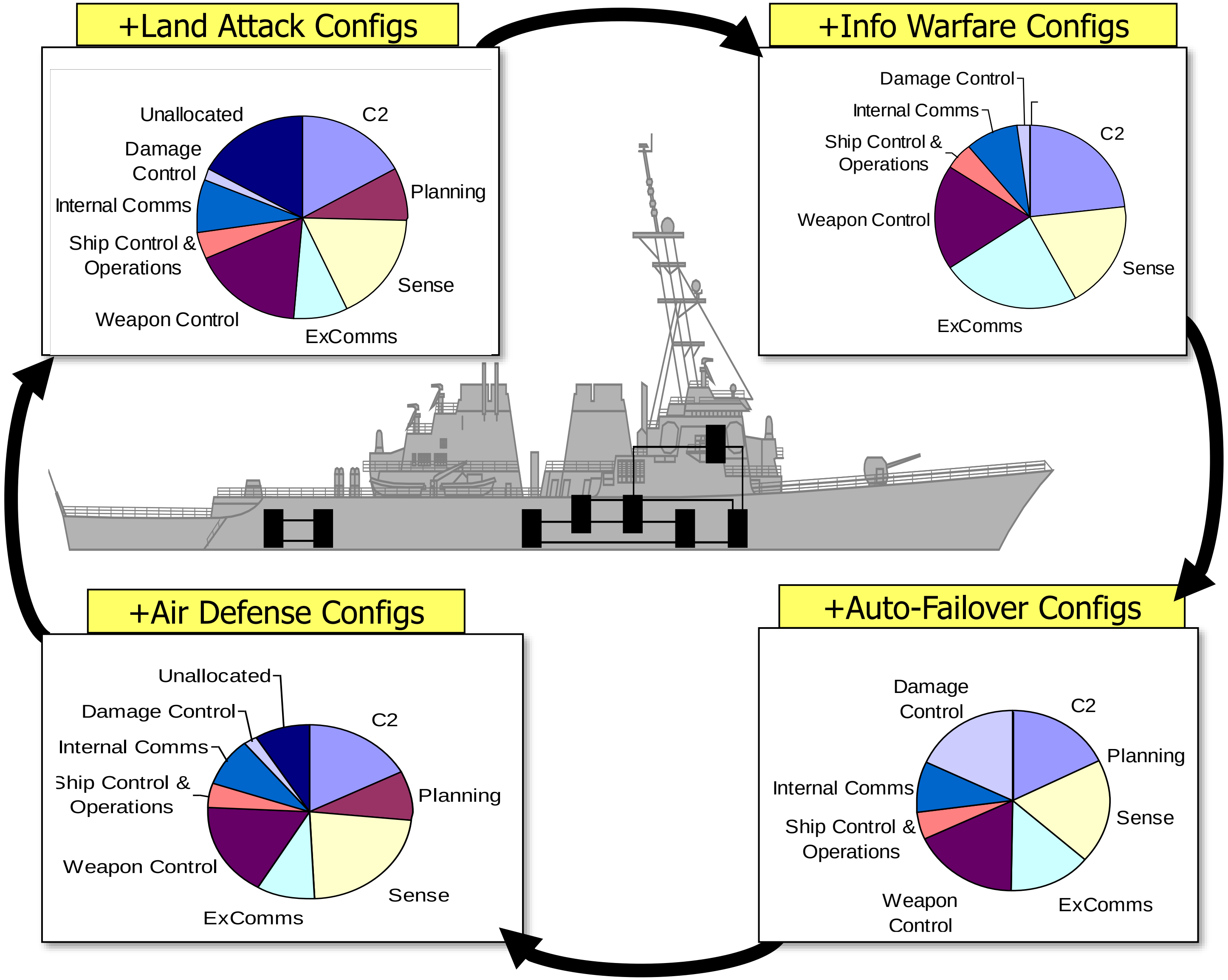
- Addressing these R&D challenges is particularly important for the DoD
  - Must continue to focus on "shift left" testing



See resources.sei.cmu.edu/library/asset-view.cfm?assetid=447143

Impact on Engineering AI-enabled Software Systems

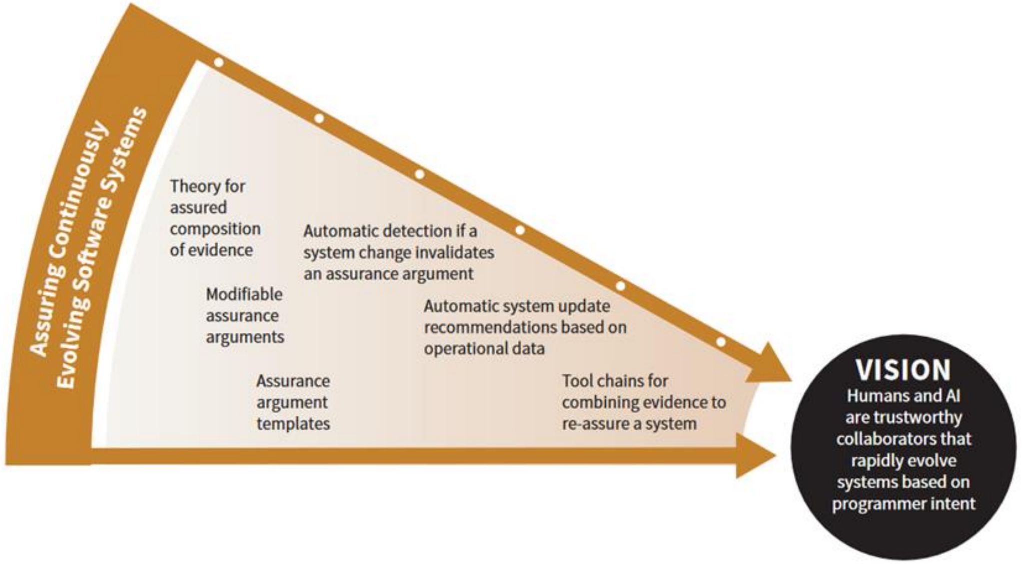
- Addressing these R&D challenges is particularly important for the DoD
  - Must continue to focus on “shift left” testing
  - Must also perform continuous quality assurance at runtime



See www.dre.vanderbilt.edu/~schmidt/PDF/JSS-DynamicResourceManagement.pdf

Impact on Engineering AI-enabled Software Systems

- Addressing these R&D challenges is particularly important for the DoD
  - Must continue to focus on "shift left" testing
  - Must also perform continuous quality assurance at runtime



Assuring Continuously Evolving Software Systems

Theory for assured composition of evidence

Modifiable assurance arguments

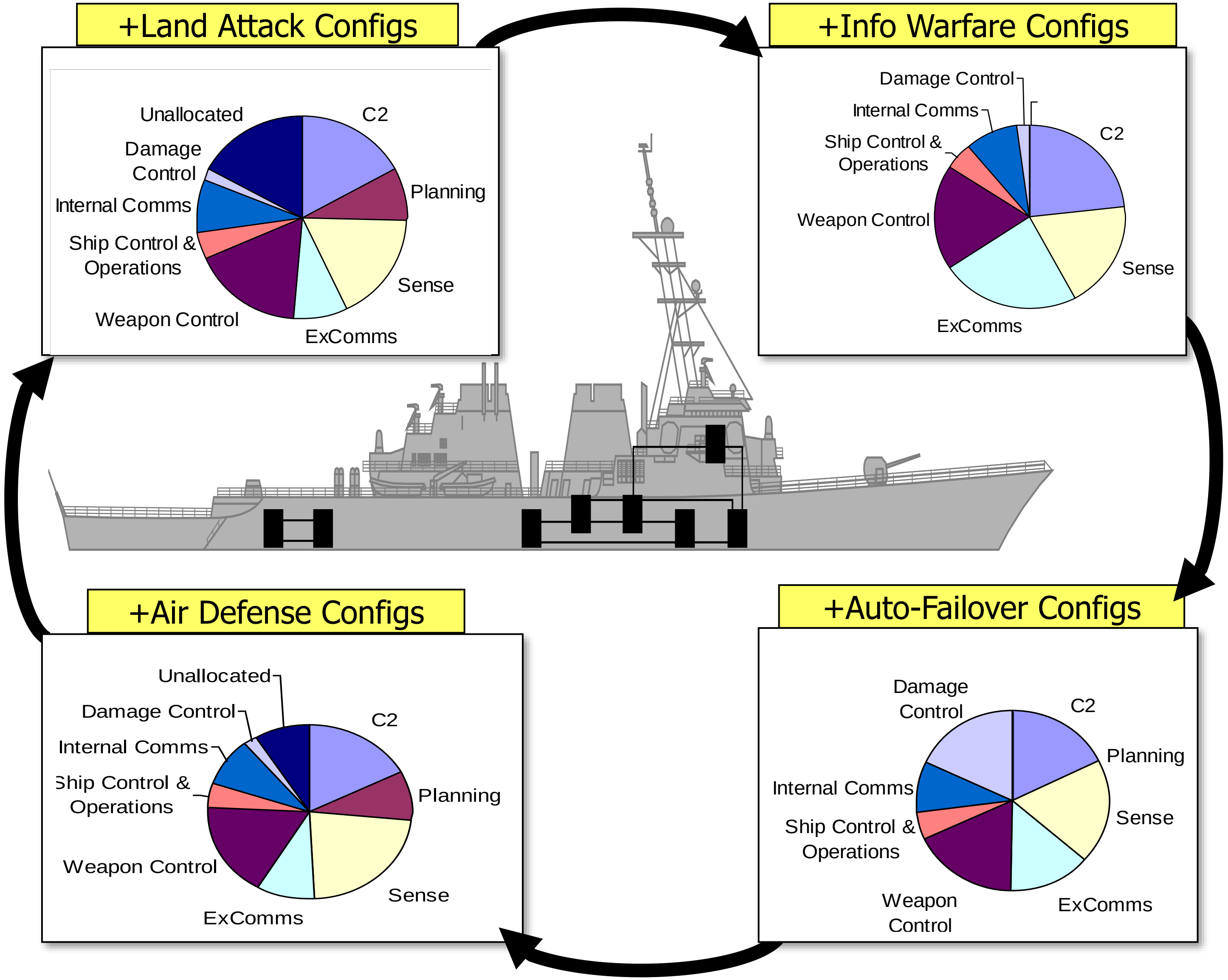
Assurance argument templates

Automatic detection if a system change invalidates an assurance argument

Automatic system update recommendations based on operational data

Tool chains for combining evidence to re-assure a system

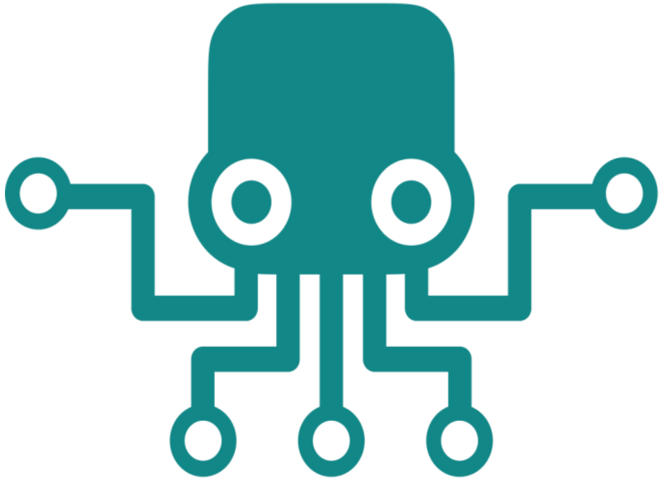
VISION
Humans and AI are trustworthy collaborators that rapidly evolve systems based on programmer intent

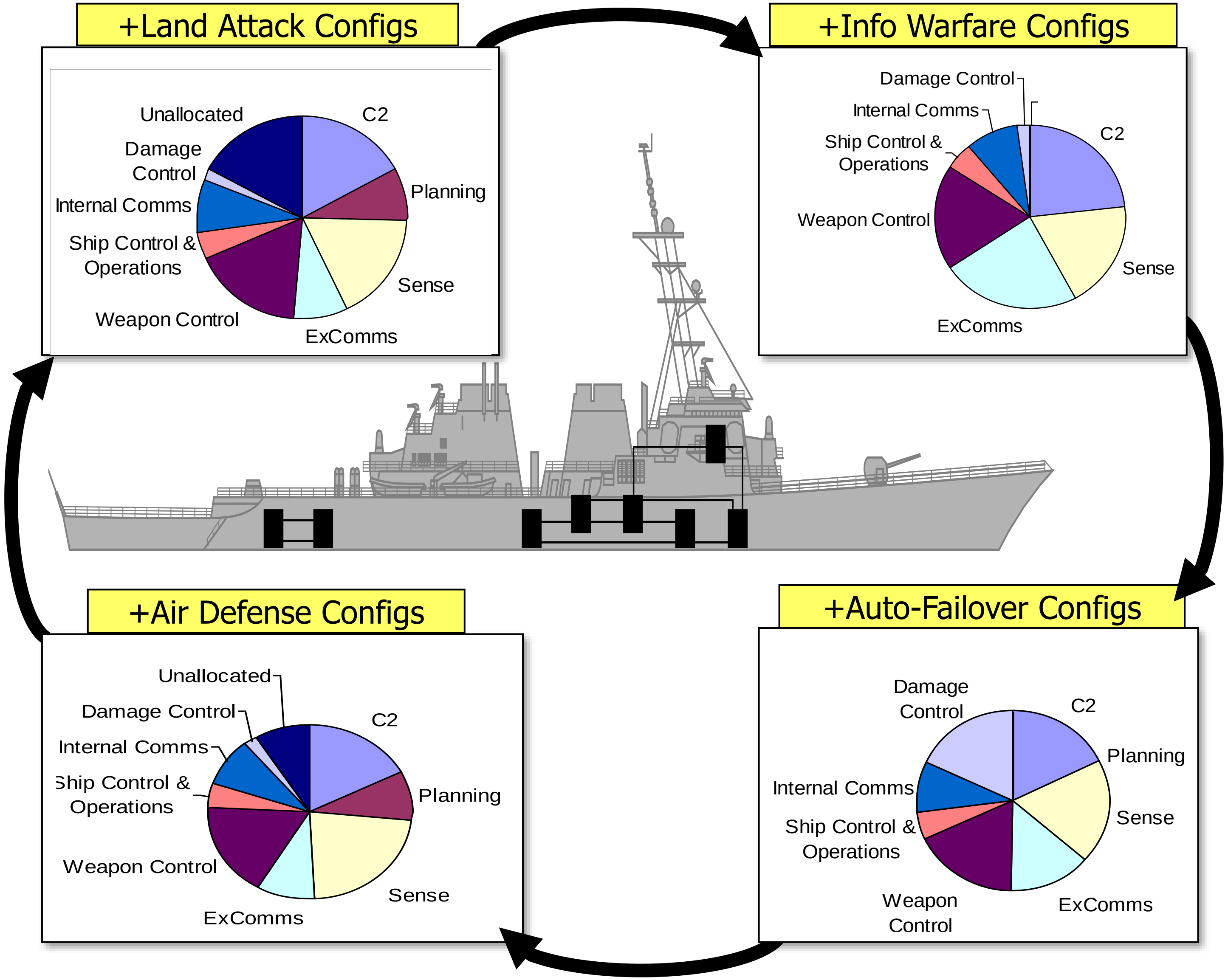


These types of systems are covered in other parts of the SEI's National Agenda Study

Impact on Engineering AI-enabled Software Systems

- Addressing these R&D challenges is particularly important for the DoD
  - Must continue to focus on “shift left” testing
  - Must also perform continuous quality assurance at runtime





See hacksheets.in/harnessing-auto-gpt-for-penetration-testing-with-osint-a-comprehensive-guide

Impact on Engineering AI-enabled Software Systems

- Addressing these R&D challenges is particularly important for the DoD
  - Must continue to focus on "shift left" testing
  - Must also perform continuous quality assurance at runtime
    - There are many implications for safety-critical real-time systems..

See www.darpa.mil/program/assured-autonomy

Applying Generative AI to Computer Science Courses at Vanderbilt

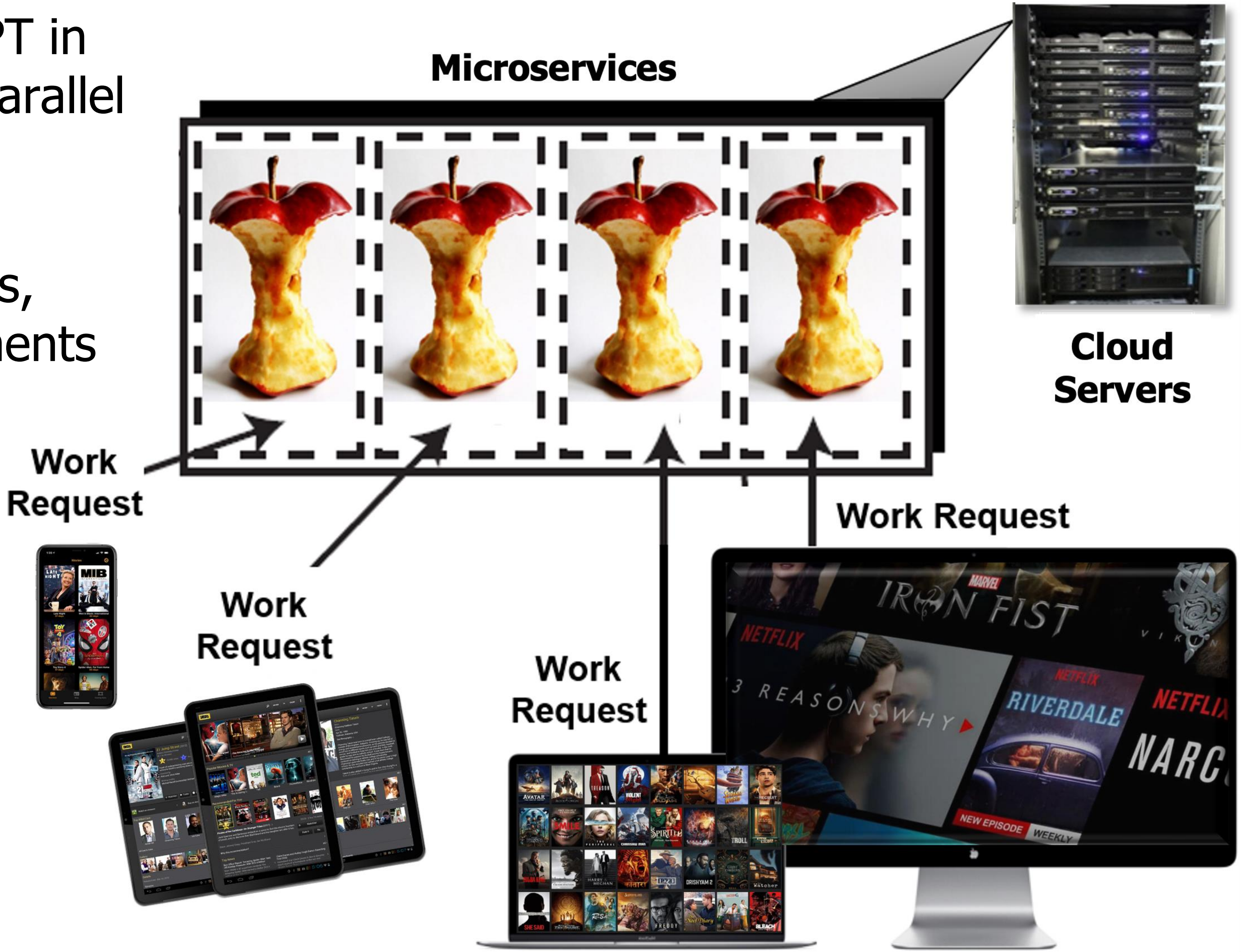
Applying Generative AI to Computer Science Courses at Vanderbilt

- The barrier to entry for learning & applying generative AI is *very* low

See chat.openai.com

Applying Generative AI to Computer Science Courses at Vanderbilt

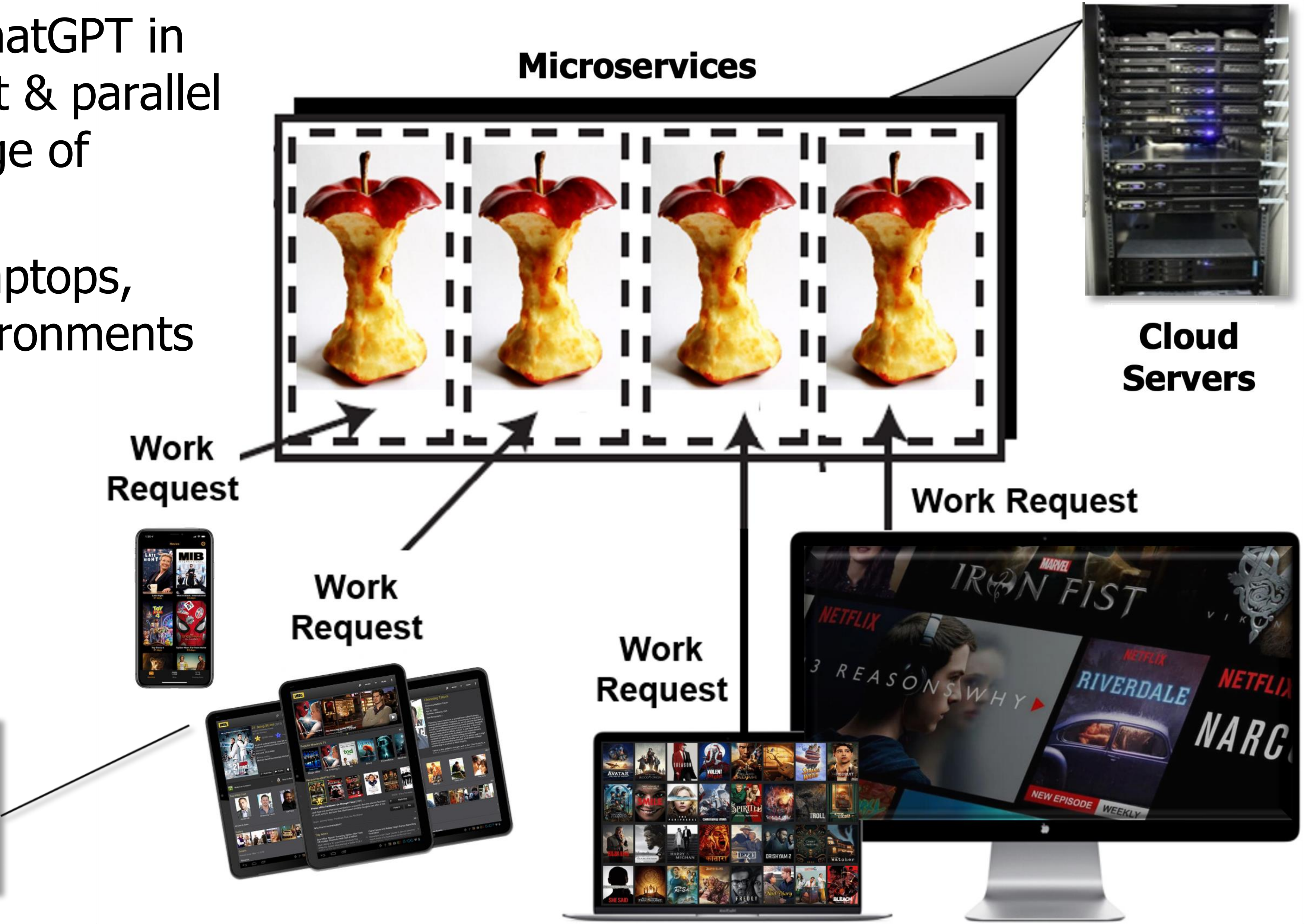
- I've therefore applied ChatGPT in my course on concurrent & parallel micro-services for a range of computing platforms
  - e.g., mobile devices, laptops, desktops, & cloud environments



See www.dre.vanderbilt.edu/~schmidt/cs891

Applying Generative AI to Computer Science Courses at Vanderbilt

- I've therefore applied ChatGPT in my course on concurrent & parallel micro-services for a range of computing platforms
    - e.g., mobile devices, laptops, desktops, & cloud environments

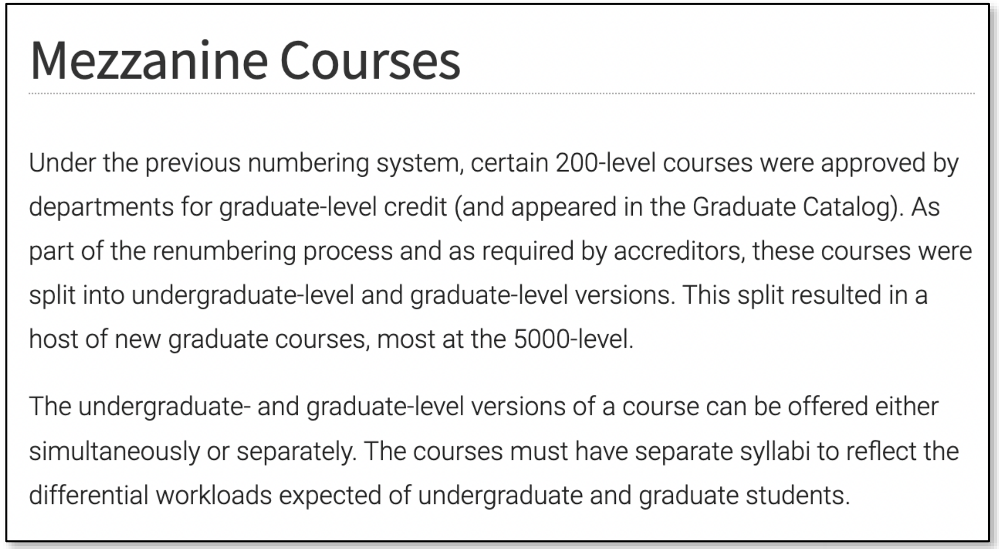


The course build a movie recommender web app with AI & non-AI parts

See www.dre.vanderbilt.edu/~schmidt/cs891

Applying Generative AI to Computer Science Courses at Vanderbilt

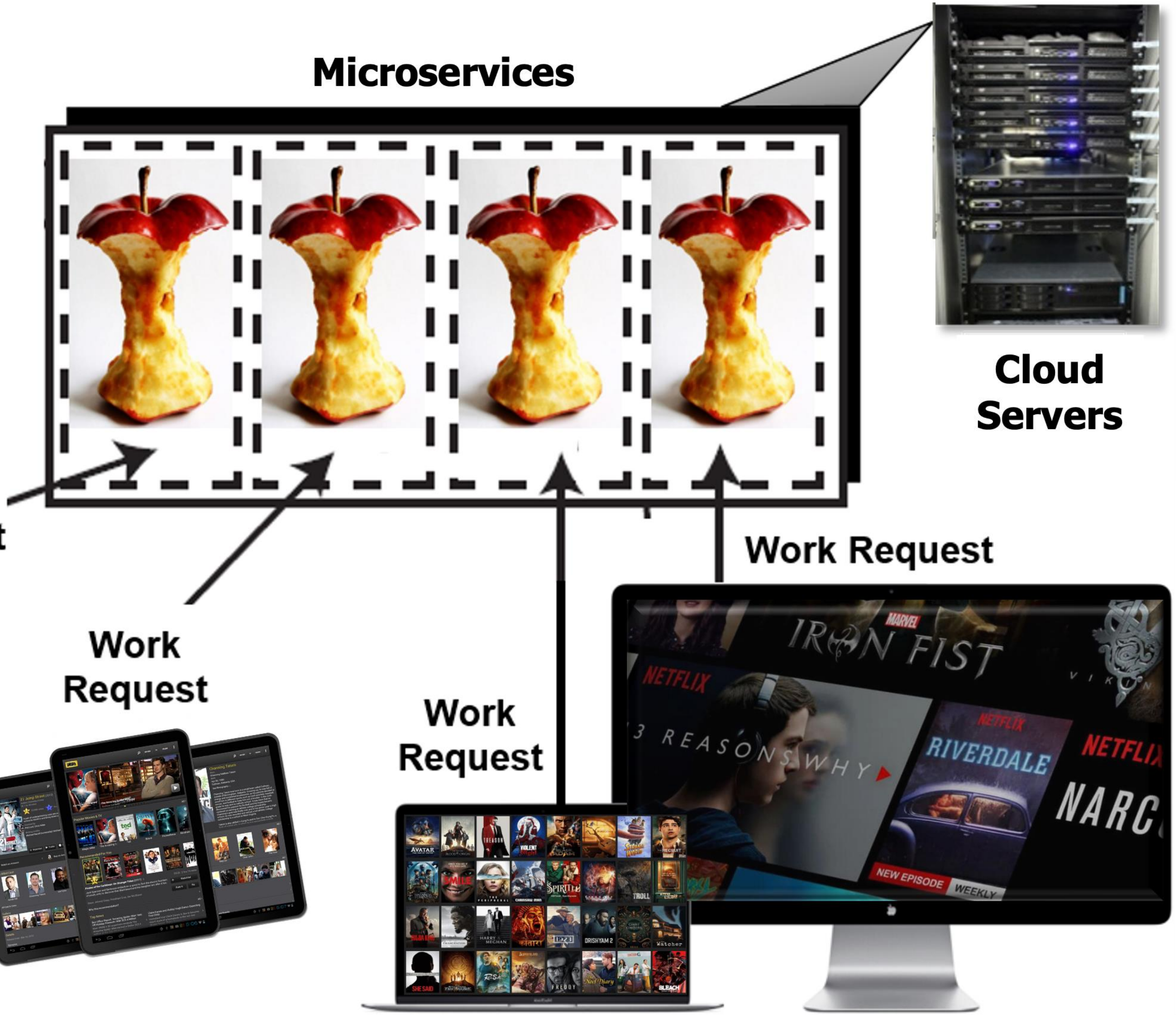
- Upper-level “mezzanine” course



Mezzanine Courses

Under the previous numbering system, certain 200-level courses were approved by departments for graduate-level credit (and appeared in the Graduate Catalog). As part of the renumbering process and as required by accreditors, these courses were split into undergraduate-level and graduate-level versions. This split resulted in a host of new graduate courses, most at the 5000-level.

The undergraduate- and graduate-level versions of a course can be offered either simultaneously or separately. The courses must have separate syllabi to reflect the differential workloads expected of undergraduate and graduate students.

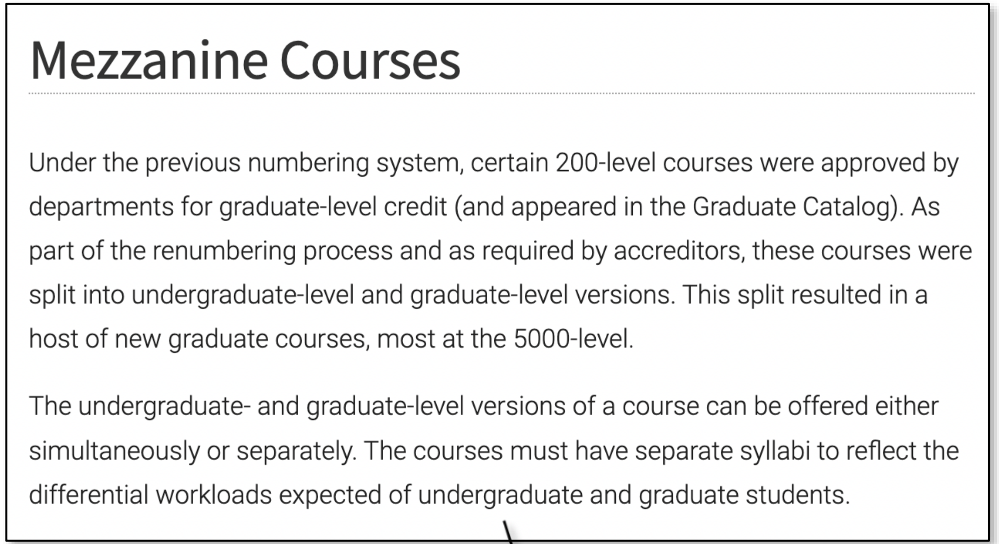


Request

See registrar.vanderbilt.edu/faculty-staff/course-renumbering/mezzanine-courses.php

Applying Generative AI to Computer Science Courses at Vanderbilt

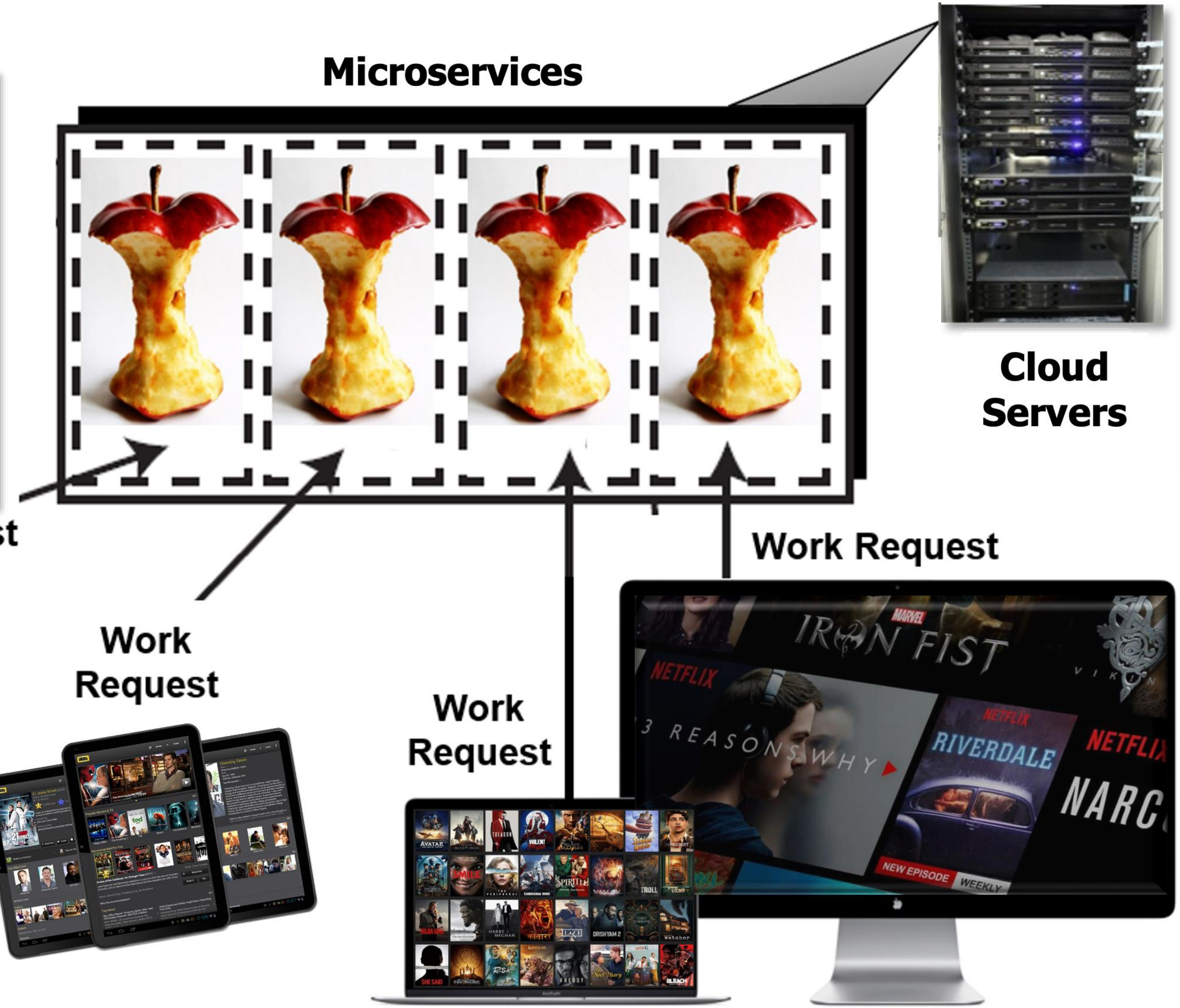
- Upper-level "mezzanine" course



Mezzanine Courses

Under the previous numbering system, certain 200-level courses were approved by departments for graduate-level credit (and appeared in the Graduate Catalog). As part of the renumbering process and as required by accreditors, these courses were split into undergraduate-level and graduate-level versions. This split resulted in a host of new graduate courses, most at the 5000-level.

The undergraduate- and graduate-level versions of a course can be offered either simultaneously or separately. The courses must have separate syllabi to reflect the differential workloads expected of undergraduate and graduate students.



Avoids some of the thornier issues associated with using ChatGPT in introductory classes

See registrar.vanderbilt.edu/faculty-staff/course-renumbering/mezzanine-courses.php

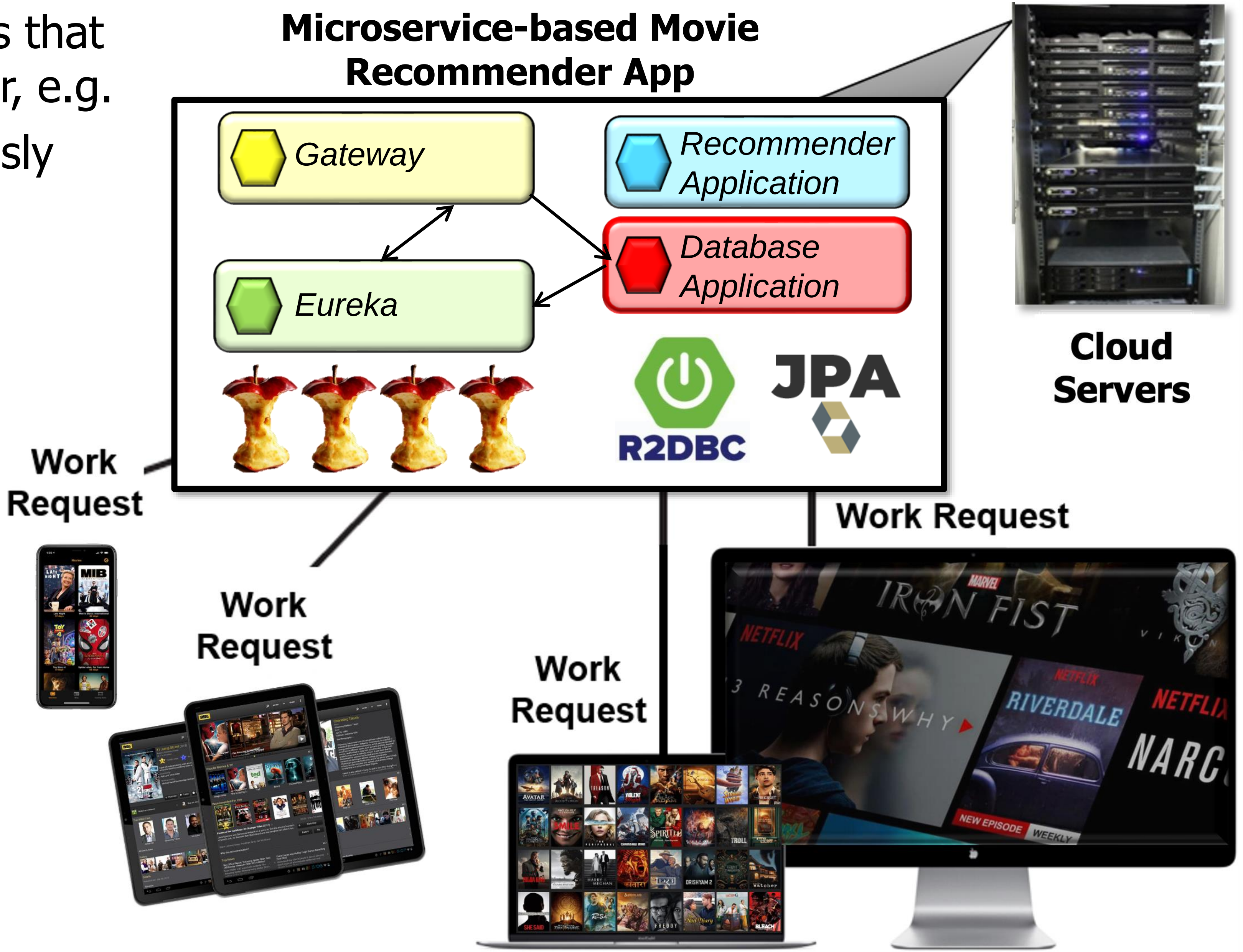
Examples of Using Generative AI in CS *891

Examples of Using Generative AI in CS *891

- Develop lectures & assignments that expand my abilities as a teacher

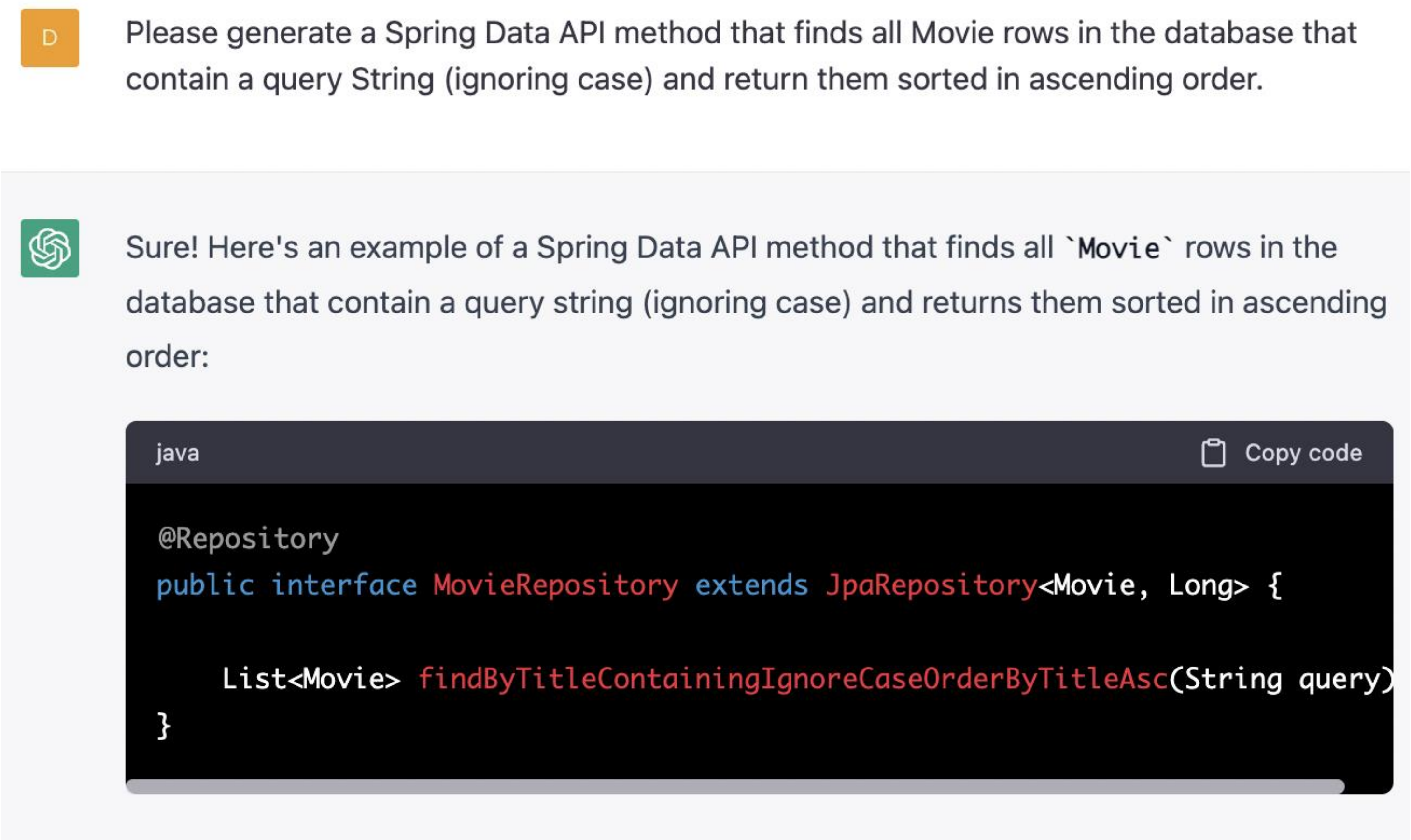
Examples of Using Generative AI in CS *891

- Develop lectures & assignments that expand my abilities as a teacher, e.g.
  - Add new topics that I previously lacked time to learn properly



Examples of Using Generative AI in CS *891

- Develop lectures & assignments that expand my abilities as a teacher, e.g.
  - Add new topics that I previously lacked time to learn properly, e.g.
    - Use natural language prompts to generate database query methods



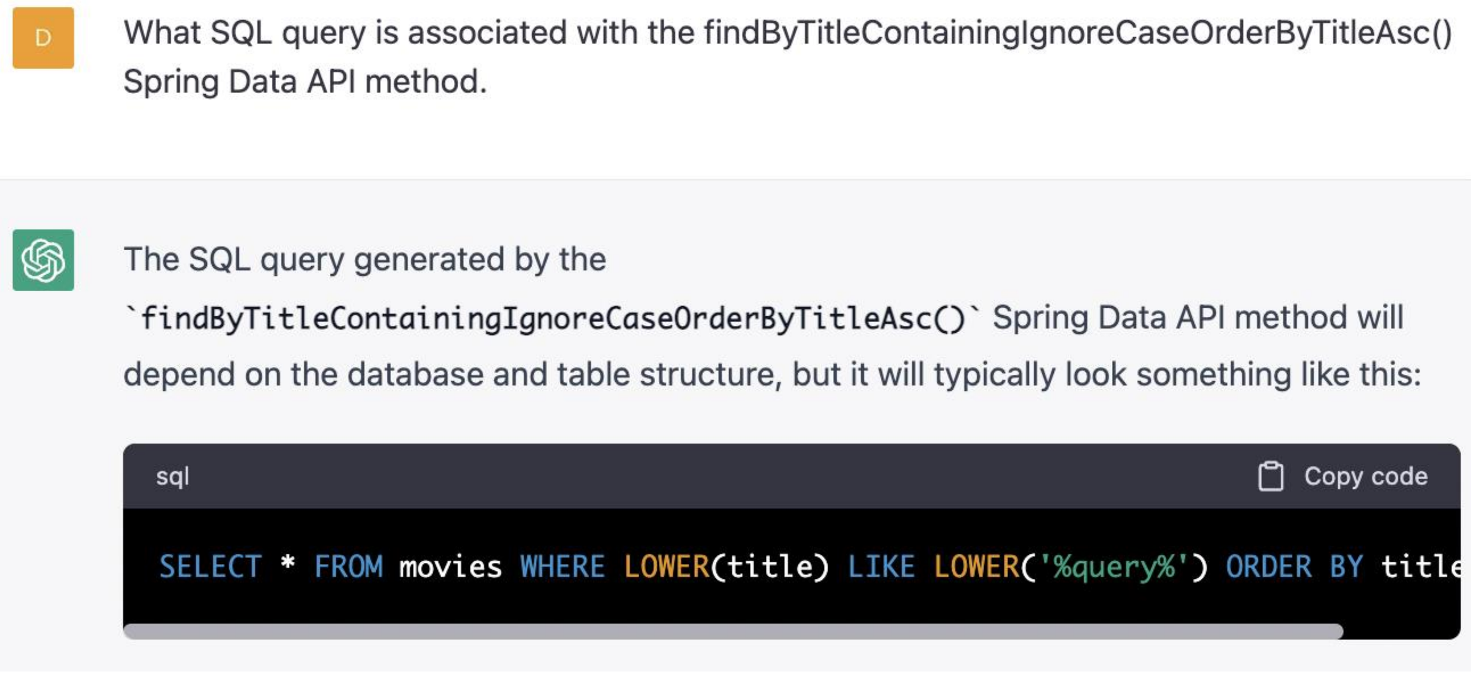
Please generate a Spring Data API method that finds all Movie rows in the database that contain a query String (ignoring case) and return them sorted in ascending order.

Sure! Here's an example of a Spring Data API method that finds all `Movie` rows in the database that contain a query string (ignoring case) and returns them sorted in ascending order:

```java
@Repository
public interface MovieRepository extends JpaRepository<Movie, Long> {

    List<Movie> findByTitleContainingIgnoreCaseOrderByTitleAsc(String query)
}
```

Examples of Using Generative AI in CS *891

- Develop lectures & assignments that expand my abilities as a teacher, e.g.
  - Add new topics that I previously lacked time to learn properly, e.g.
    - Use natural language prompts to generate database query methods
    - Explain what SQL queries are associated with these methods



What SQL query is associated with the findByTitleContainingIgnoreCaseOrderByTitleAsc() Spring Data API method.

The SQL query generated by the `findByTitleContainingIgnoreCaseOrderByTitleAsc()` Spring Data API method will depend on the database and table structure, but it will typically look something like this:

```sql
SELECT * FROM movies WHERE LOWER(title) LIKE LOWER('%query%') ORDER BY title
```

Examples of Using Generative AI in CS *891

- Develop lectures & assignments that expand my abilities as a teacher, e.g.
  - Add new topics that I previously lacked time to learn properly
  - Generate sample code using advanced techniques I was not facile with (yet)

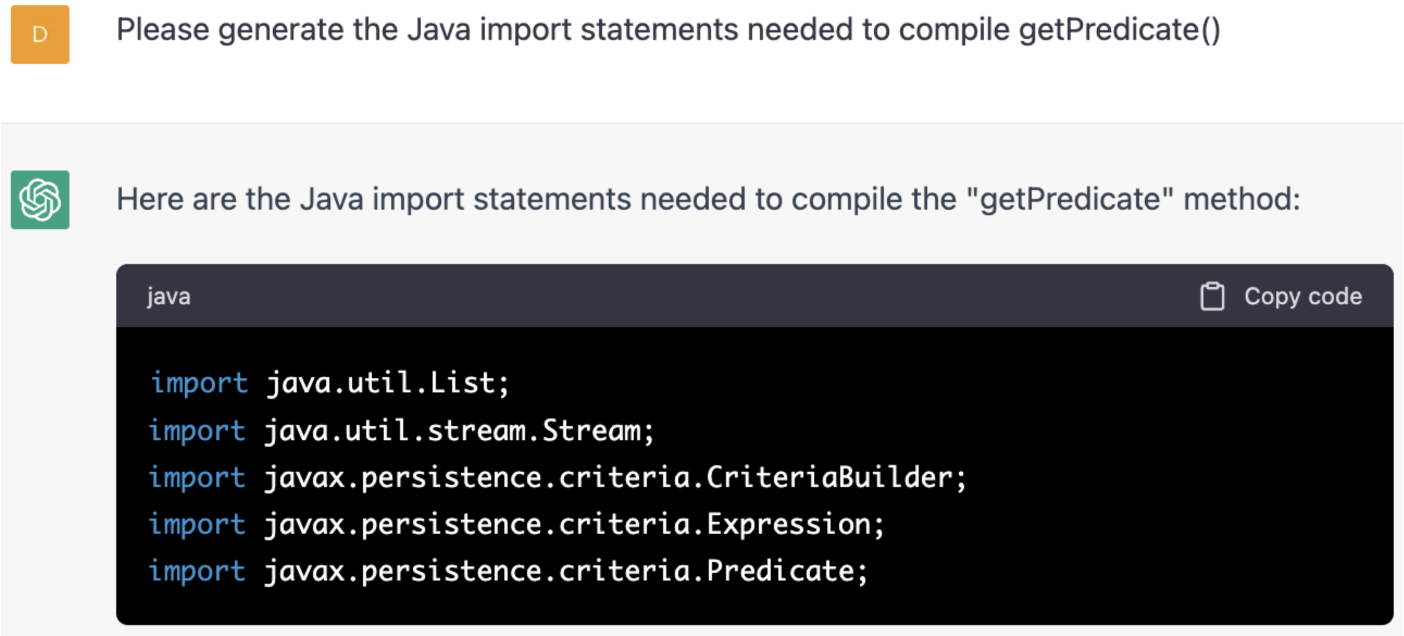
Examples of Using Generative AI in CS *891

- Develop lectures & assignments that expand my abilities as a teacher, e.g.
  - Add new topics that I previously lacked time to learn properly
  - Generate sample code using advanced techniques I was not facile with (yet), e.g.
    - Custom SQL queries

```
public static Predicate getPredicate
    (List<String> queries,
     CriteriaBuilder criteriaBuilder,
     Expression<String> idExpression) {
    return queries
        .stream()
        .map(String::toLowerCase)
        .map(query -> criteriaBuilder
             .like(idExpression,
                   "%" + query + "%"))
        .reduce(criteriaBuilder.conjunction(),
                criteriaBuilder::and);
}
```

Examples of Using Generative AI in CS *891

- Develop lectures & assignments that expand my abilities as a teacher, e.g.
  - Add new topics that I previously lacked time to learn properly
  - Generate sample code using advanced techniques I was not facile with (yet), e.g.
    - Custom SQL queries
    - Imports for custom SQL queries



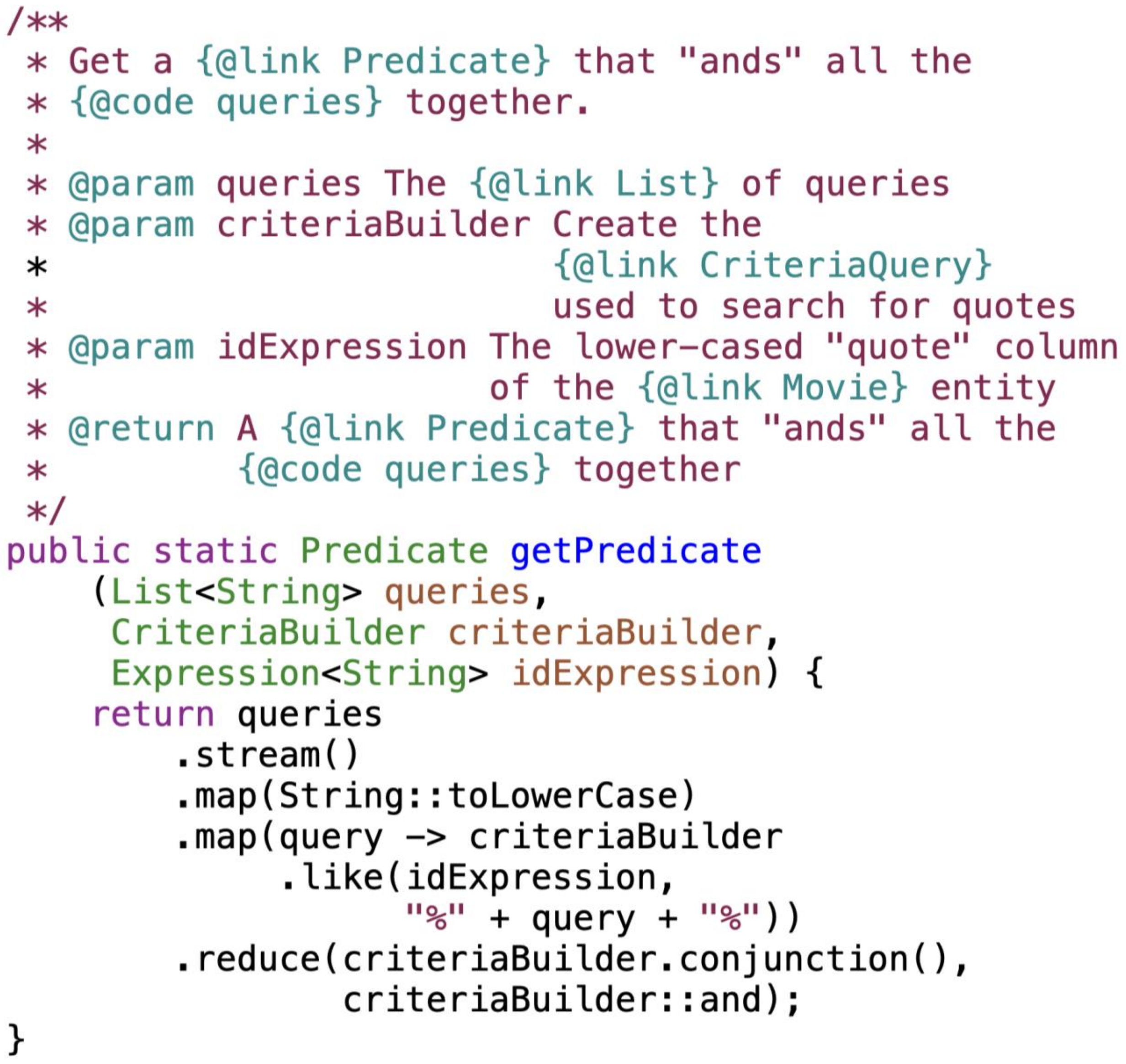
Please generate the Java import statements needed to compile getPredicate()

Here are the Java import statements needed to compile the "getPredicate" method:

```java
import java.util.List;
import java.util.stream.Stream;
import javax.persistence.criteria.CriteriaBuilder;
import javax.persistence.criteria.Expression;
import javax.persistence.criteria.Predicate;
```

Examples of Using Generative AI in CS *891

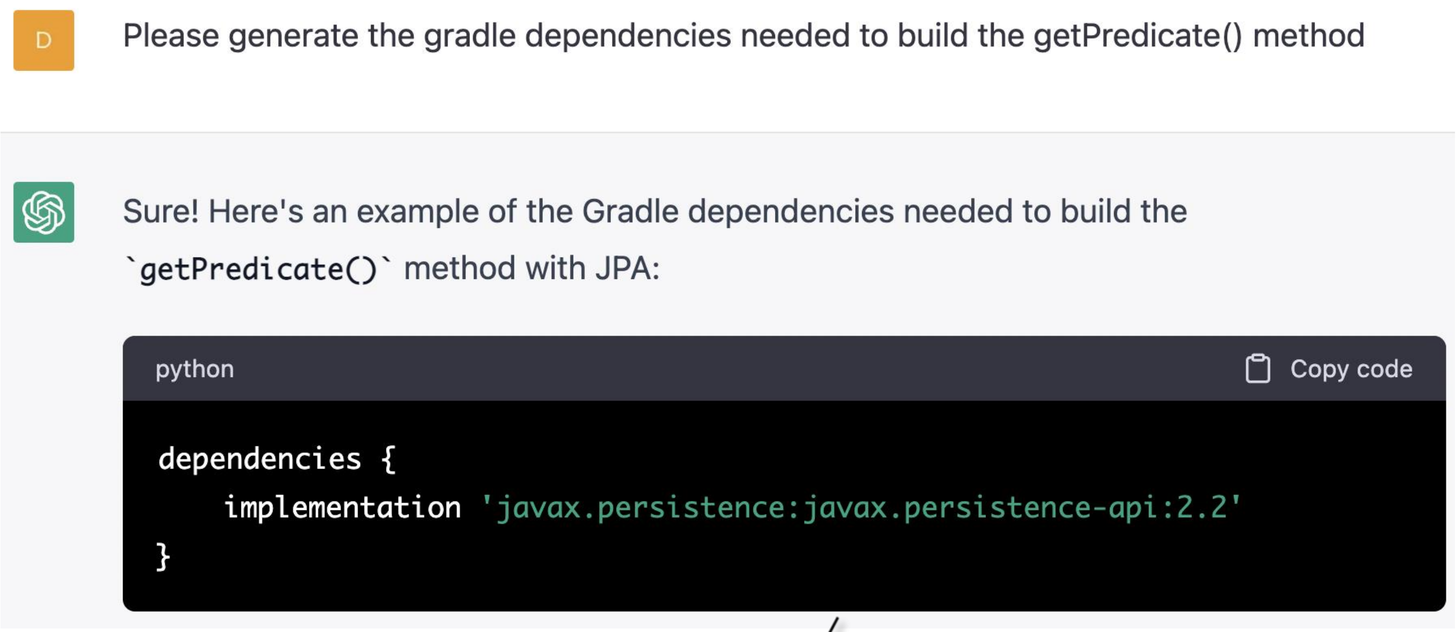
- Develop lectures & assignments that expand my abilities as a teacher, e.g.
  - Add new topics that I previously lacked time to learn properly
  - Generate sample code using advanced techniques I was not facile with (yet), e.g.
    - Custom SQL queries
    - Imports for custom SQL queries
    - Commenting the custom SQL queries



```
/**
 * Get a {@link Predicate} that "ands" all the
 * {@code queries} together.
 *
 * @param queries The {@link List} of queries
 * @param criteriaBuilder Create the
 *                        {@link CriteriaQuery}
 *                        used to search for quotes
 * @param idExpression The lower-cased "quote" column
 *                     of the {@link Movie} entity
 * @return A {@link Predicate} that "ands" all the
 *         {@code queries} together
 */
public static Predicate getPredicate
    (List<String> queries,
     CriteriaBuilder criteriaBuilder,
     Expression<String> idExpression) {
    return queries
        .stream()
        .map(String::toLowerCase)
        .map(query -> criteriaBuilder
             .like(idExpression,
                   "%" + query + "%"))
        .reduce(criteriaBuilder.conjunction(),
                criteriaBuilder::and);
}
```

Examples of Using Generative AI in CS *891

- Develop lectures & assignments that expand my abilities as a teacher, e.g.
  - Add new topics that I previously lacked time to learn properly
  - Generate sample code using advanced techniques I was not facile with (yet), e.g.
    - Custom SQL queries
    - Imports for custom SQL queries
    - Commenting the custom SQL queries
    - Build dependencies for custom SQL queries



Please generate the gradle dependencies needed to build the getPredicate() method

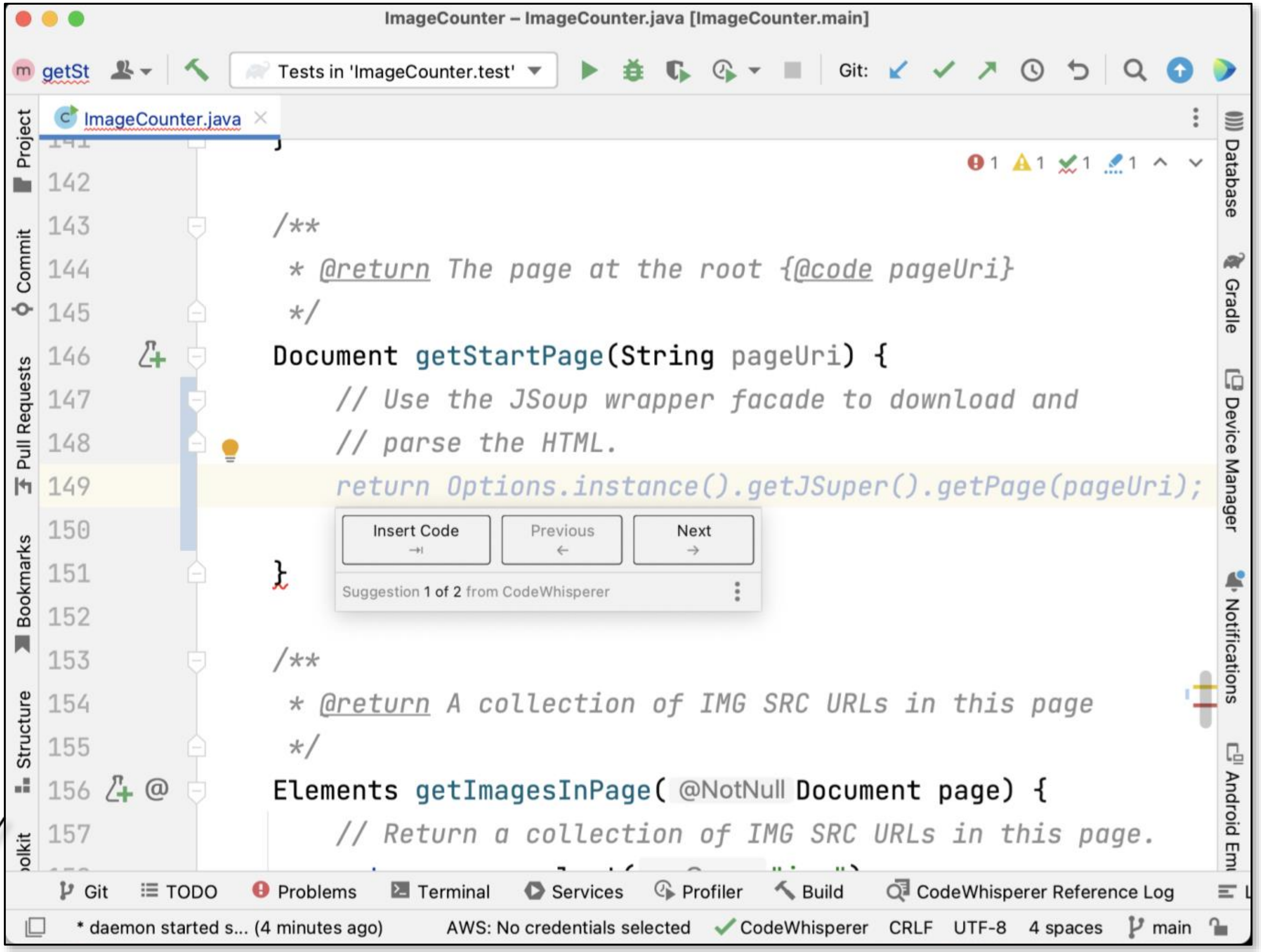
Sure! Here's an example of the Gradle dependencies needed to build the `getPredicate()` method with JPA:

```
dependencies {
    implementation 'javax.persistence:javax.persistence-api:2.2'
}
```

ChatGPT isn't aware of the replacement of the javax APIs with the Jakarta APIs..

Examples of Using Generative AI in CS *891

- Develop lectures & assignments that expand my abilities as a teacher, e.g.
  - Add new topics that I previously lacked time to learn properly
  - Generate sample code using advanced techniques I was not facile with (yet), e.g.
    - Custom SQL queries
    - Imports for custom SQL queries
    - Commenting the custom SQL queries
    - Build dependencies for custom SQL queries



```
    }

    /**
     * @return The page at the root {@code pageUri}
     */
    Document getStartPage(String pageUri) {
        // Use the JSoup wrapper facade to download and
        // parse the HTML.
        return Options.instance().getJSuper().getPage(pageUri);
    }

    /**
     * @return A collection of IMG SRC URLs in this page
     */
    Elements getImagesInPage( @NotNull Document page) {
        // Return a collection of IMG SRC URLs in this page.
```

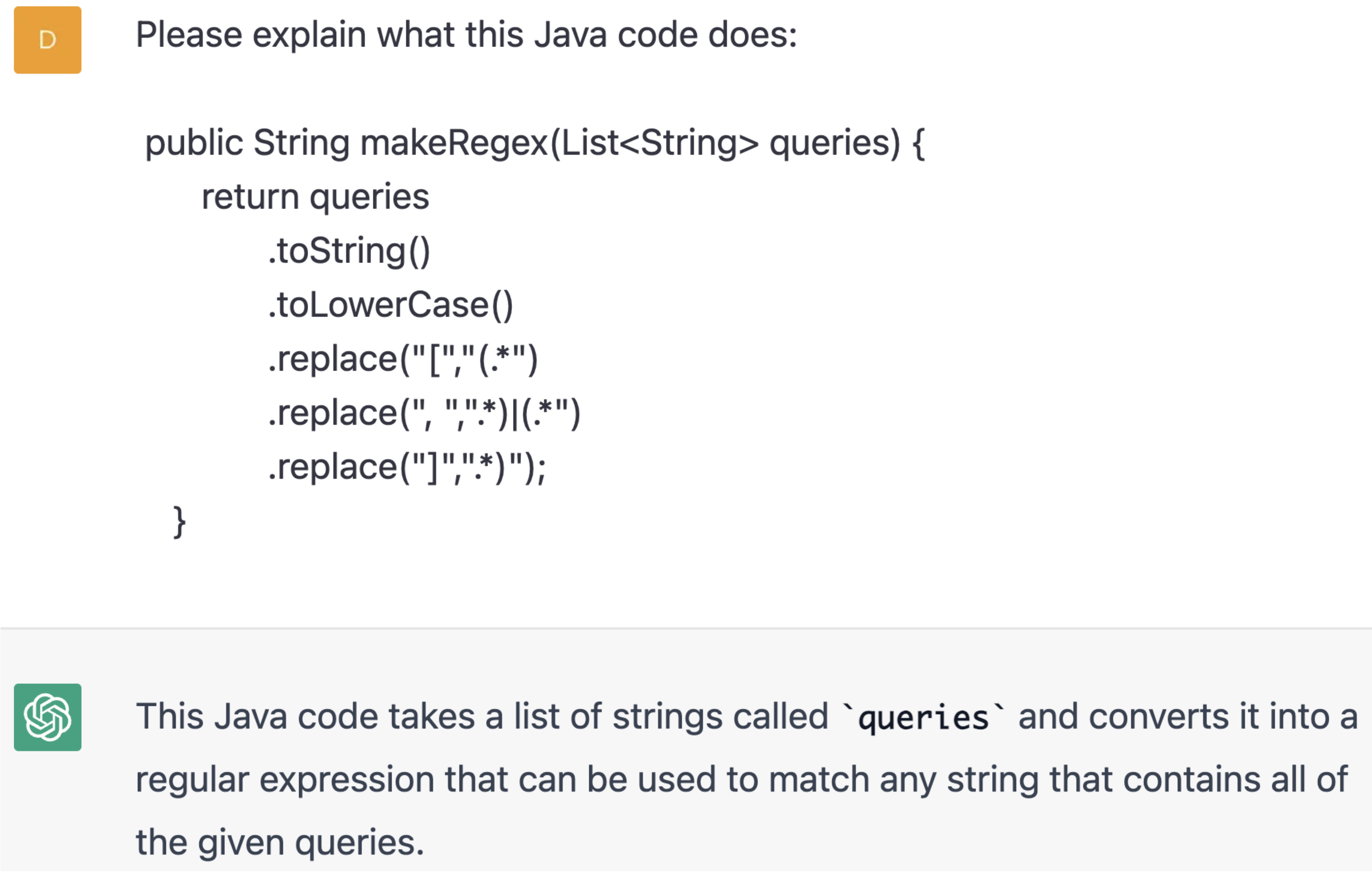
Suggestion 1 of 2 from CodeWhisperer

This type of code generation can now be done in real-time via plugins to interactive development environments (IDEs)!

See garystafford.medium.com/accelerating-development-with-generative-ai-powered-coding-tools-f1480664b6ab

Examples of Using Generative AI in CS *891

- Develop lectures & assignments that expand my abilities as a teacher, e.g.
  - Add new topics that I previously lacked time to learn properly
  - Generate sample code using advanced techniques I was not facile with (yet)
  - Understand & explain very clever solutions by my students



Please explain what this Java code does:

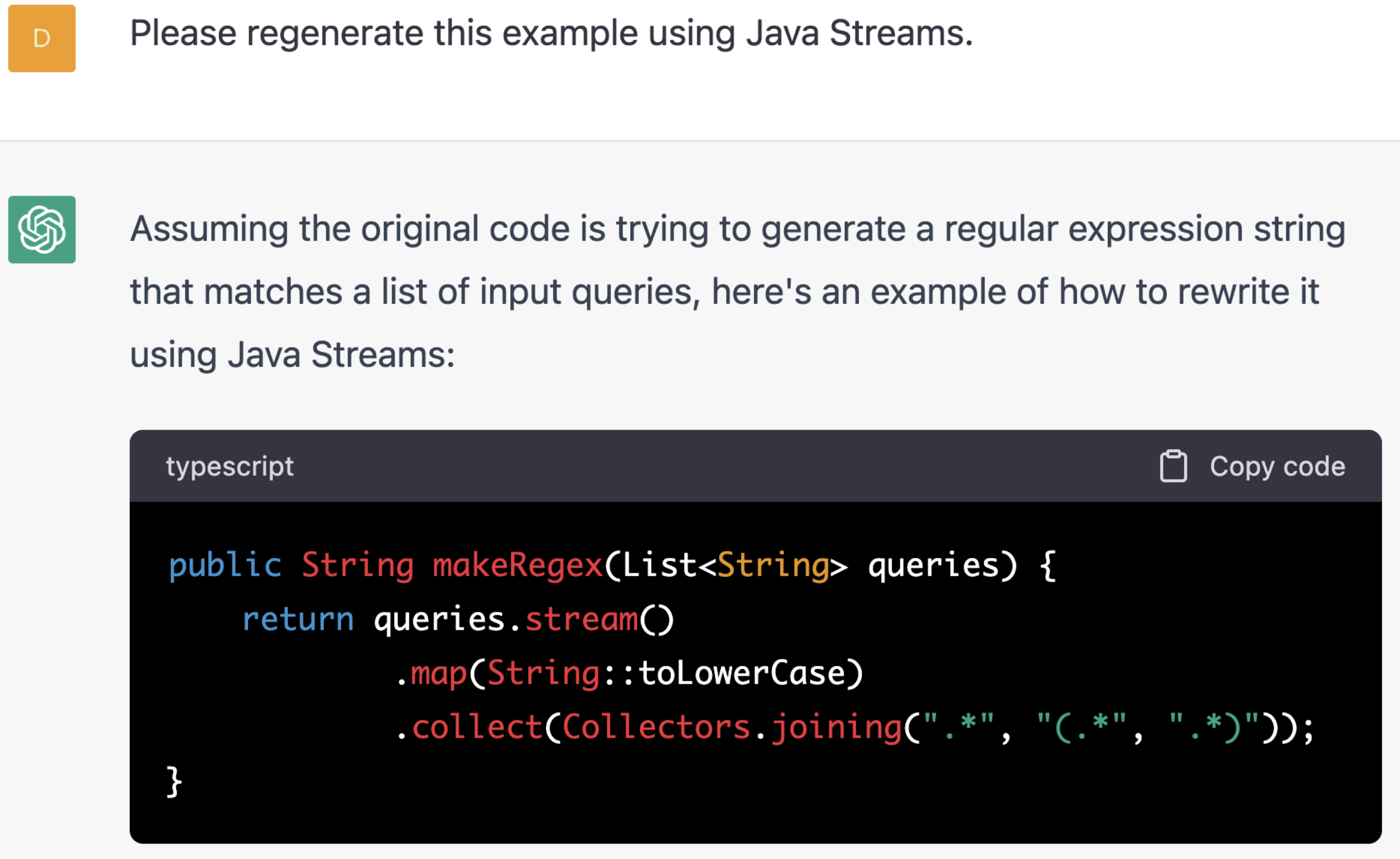
```
public String makeRegex(List<String> queries) {
    return queries
        .toString()
        .toLowerCase()
        .replace("[","(.*")
        .replace(", ",".*)|(.*")
        .replace("]",".*)");
  }
```

This Java code takes a list of strings called `queries` and converts it into a regular expression that can be used to match any string that contains all of the given queries.

See arxiv.org/pdf/2305.12865.pdf

Examples of Using Generative AI in CS *891

- Develop lectures & assignments that expand my abilities as a teacher, e.g.
  - Add new topics that I previously lacked time to learn properly
  - Generate sample code using advanced techniques I was not facile with (yet)
  - Understand & explain very clever solutions by my students
    - Automatically regenerate a more modern solution



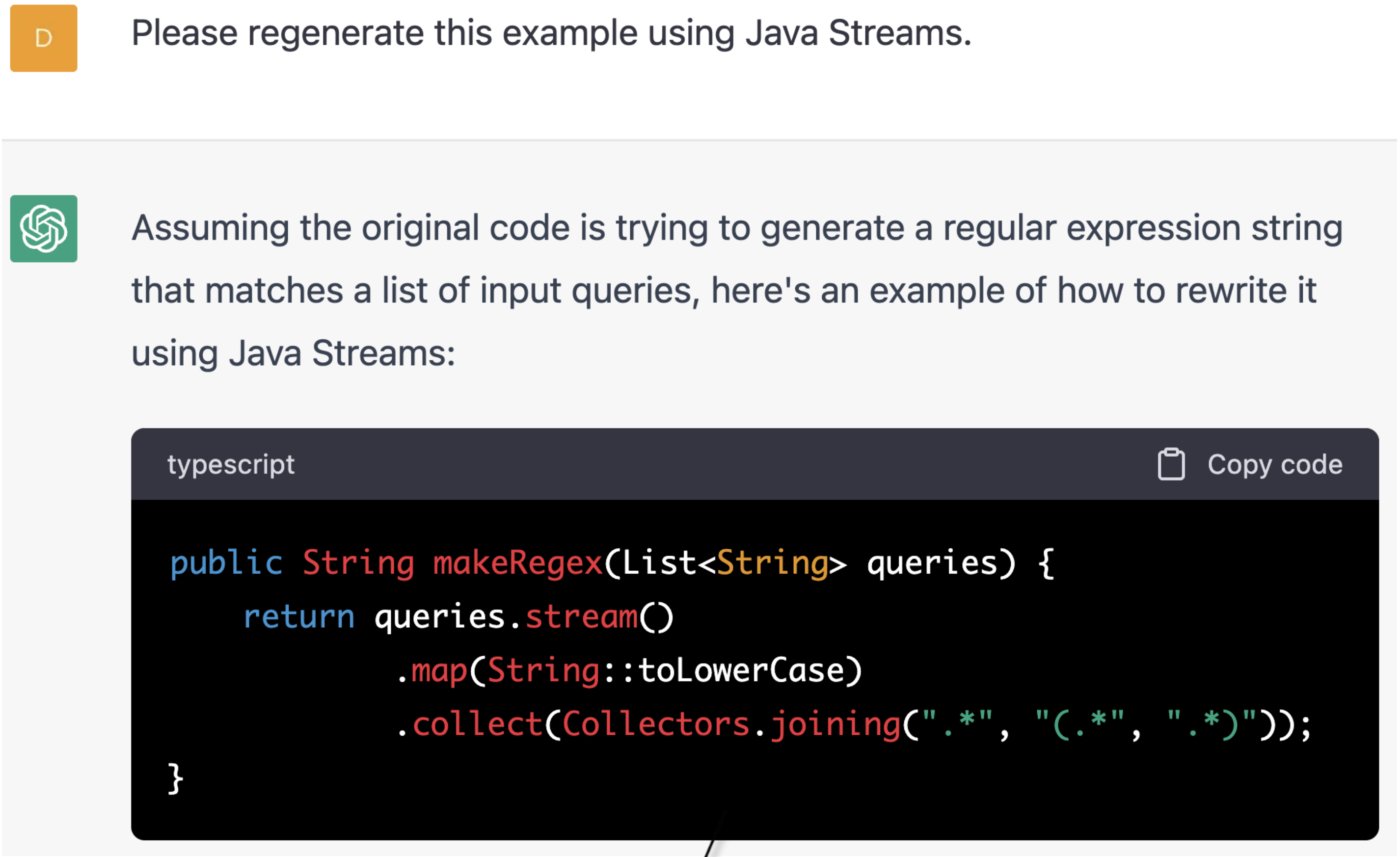
Please regenerate this example using Java Streams.

Assuming the original code is trying to generate a regular expression string that matches a list of input queries, here's an example of how to rewrite it using Java Streams:

```
public String makeRegex(List<String> queries) {
    return queries.stream()
            .map(String::toLowerCase)
            .collect(Collectors.joining(".*", "(.*", ".*)"));
}
```

Examples of Using Generative AI in CS *891

- Develop lectures & assignments that expand my abilities as a teacher, e.g.
  - Add new topics that I previously lacked time to learn properly
  - Generate sample code using advanced techniques I was not facile with (yet)
  - Understand & explain very clever solutions by my students
    - Automatically regenerate a more modern solution



Please regenerate this example using Java Streams.

Assuming the original code is trying to generate a regular expression string that matches a list of input queries, here's an example of how to rewrite it using Java Streams:

```
public String makeRegex(List<String> queries) {
    return queries.stream()
            .map(String::toLowerCase)
            .collect(Collectors.joining(".*", "(.*", ".*)"));
}
```

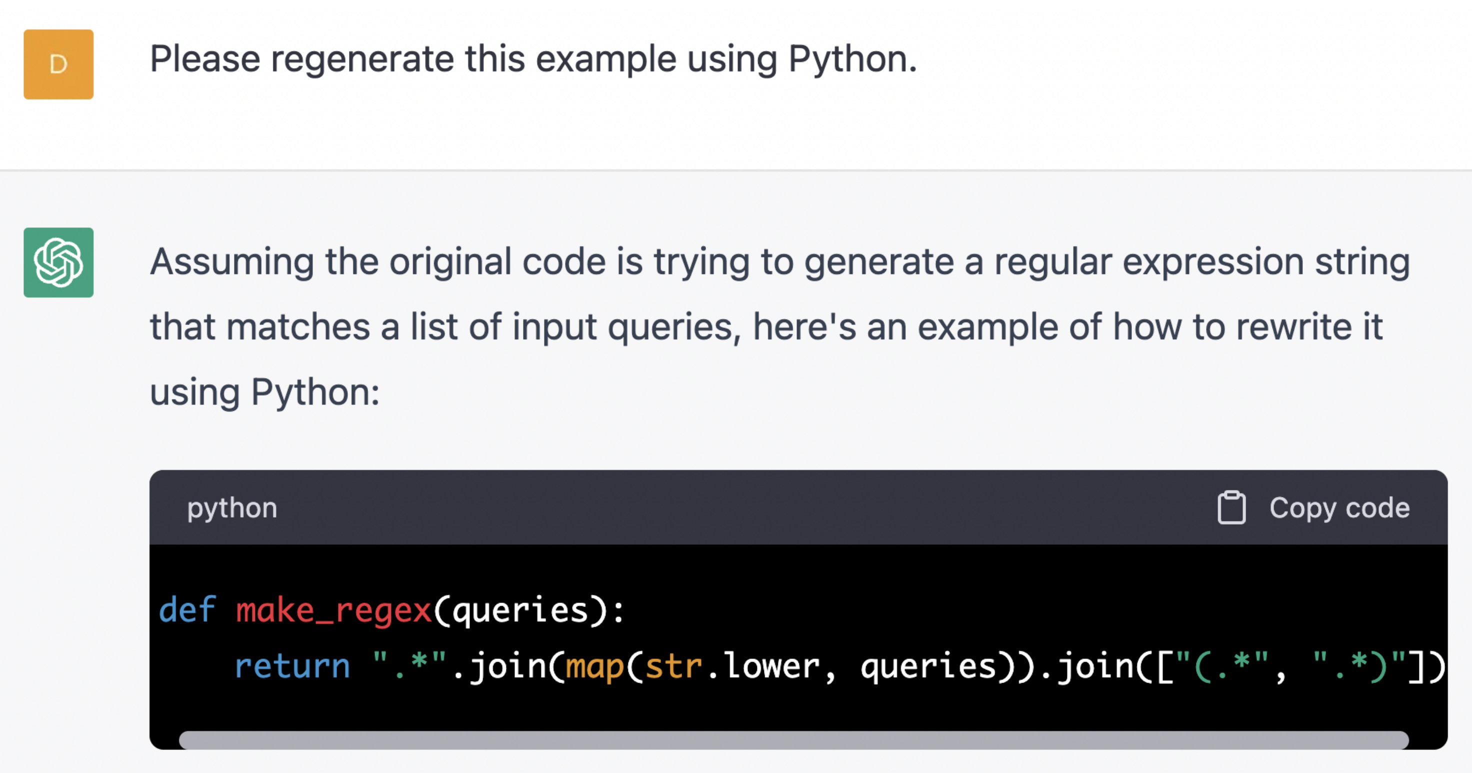
*ChatGPT "reads" & "learns" **all** the documentation, unlike me!*

Examples of Using Generative AI in CS *891

- Develop lectures & assignments that expand my abilities as a teacher, e.g.
  - Add new topics that I previously lacked time to learn properly
  - Generate sample code using advanced techniques I was not facile with (yet)
  - Understand & explain very clever solutions by my students
    - Automatically regenerate a more modern solution
      - & do this in multiple languages

Examples of Using Generative AI in CS *891

- Develop lectures & assignments that expand my abilities as a teacher, e.g.
  - Add new topics that I previously lacked time to learn properly
  - Generate sample code using advanced techniques I was not facile with (yet)
  - Understand & explain very clever solutions by my students
    - Automatically regenerate a more modern solution
      - & do this in multiple languages



Please regenerate this example using Python.

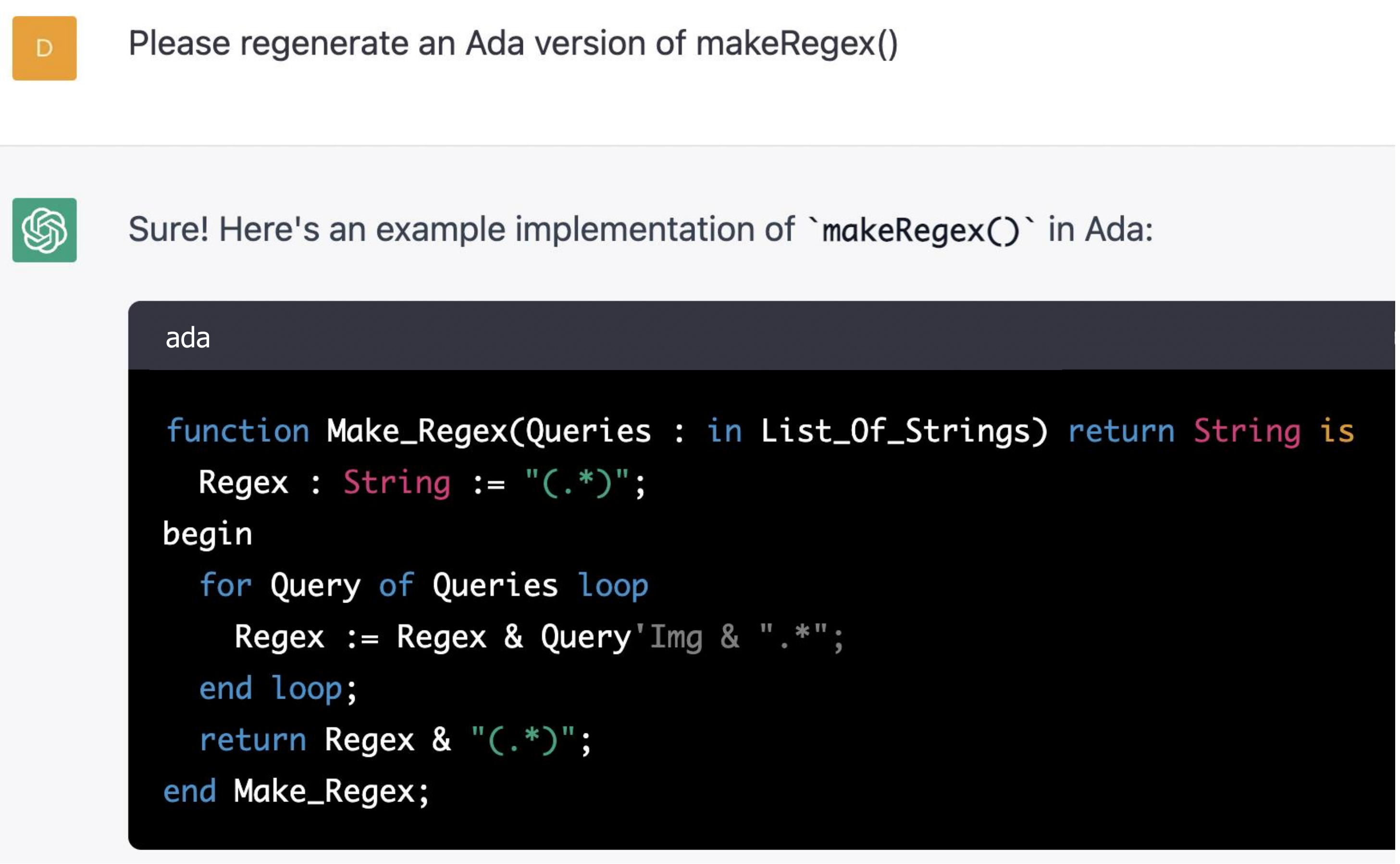
Assuming the original code is trying to generate a regular expression string that matches a list of input queries, here's an example of how to rewrite it using Python:

```python
def make_regex(queries):
    return ".*".join(map(str.lower, queries)).join(["(.*", ".*)"])
```

e.g., popular languages like Python

Examples of Using Generative AI in CS *891

- Develop lectures & assignments that expand my abilities as a teacher, e.g.
  - Add new topics that I previously lacked time to learn properly
  - Generate sample code using advanced techniques I was not facile with (yet)
  - Understand & explain very clever solutions by my students
    - Automatically regenerate a more modern solution
      - & do this in multiple languages



Please regenerate an Ada version of makeRegex()

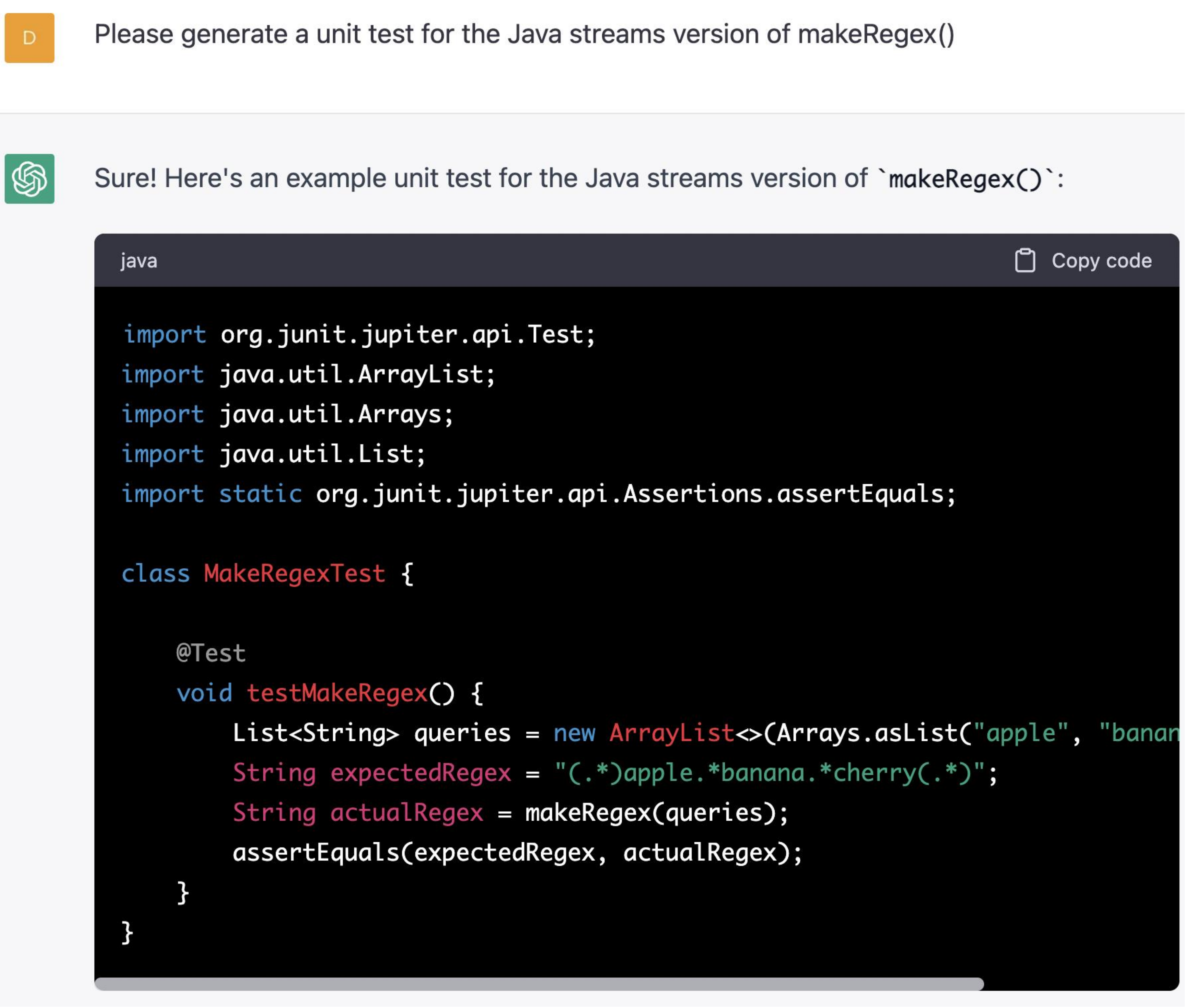
Sure! Here's an example implementation of `makeRegex()` in Ada:

```
function Make_Regex(Queries : in List_Of_Strings) return String is
  Regex : String := "(.*)";
begin
  for Query of Queries loop
    Regex := Regex & Query'Img & ".*";
  end loop;
  return Regex & "(.*)";
end Make_Regex;
```

e.g., more esoteric languages like Ada

Examples of Using Generative AI in CS *891

- Develop lectures & assignments that expand my abilities as a teacher, e.g.
  - Add new topics that I previously lacked time to learn properly
  - Generate sample code using advanced techniques I was not facile with (yet)
  - Understand & explain very clever solutions by my students
  - Generate unit tests for my examples



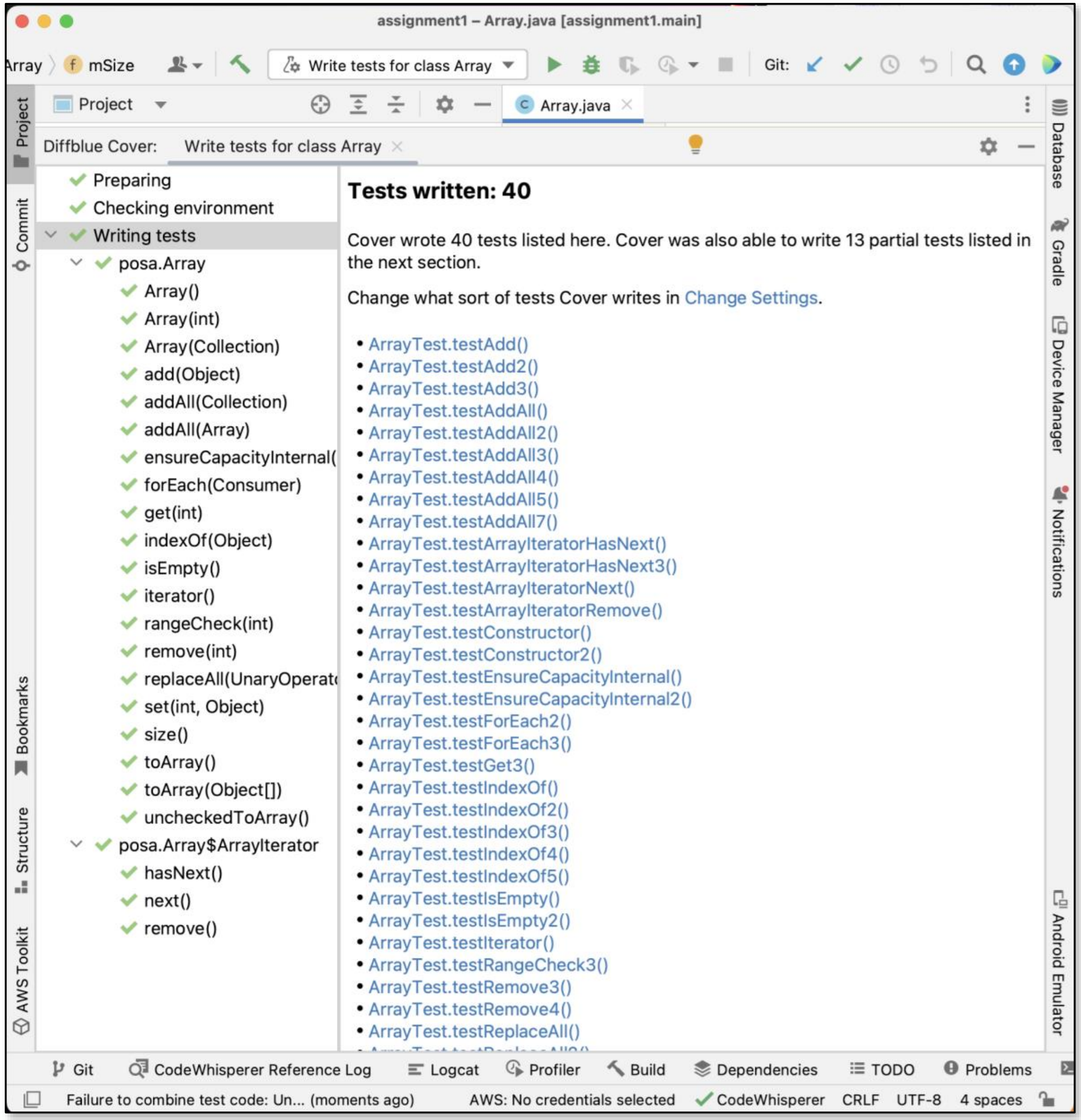
Please generate a unit test for the Java streams version of makeRegex()

Sure! Here's an example unit test for the Java streams version of `makeRegex()`:

```java
import org.junit.jupiter.api.Test;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import static org.junit.jupiter.api.Assertions.assertEquals;

class MakeRegexTest {

    @Test
    void testMakeRegex() {
        List<String> queries = new ArrayList<>(Arrays.asList("apple", "banan
        String expectedRegex = "(.*)apple.*banana.*cherry(.*)";
        String actualRegex = makeRegex(queries);
        assertEquals(expectedRegex, actualRegex);
    }
}
```

Examples of Using Generative AI in CS *891

- Develop lectures & assignments that expand my abilities as a teacher, e.g.
  - Add new topics that I previously lacked time to learn properly
  - Generate sample code using advanced techniques I was not facile with (yet)
  - Understand & explain very clever solutions by my students
  - Generate unit tests for my examples
    - Tools now exist that automate unit test generation in popular programming languages & IDEs



- ✓ Preparing
- ✓ Checking environment
- ✓ Writing tests
  - ✓ posa.Array
    - ✓ Array()
    - ✓ Array(int)
    - ✓ Array(Collection)
    - ✓ add(Object)
    - ✓ addAll(Collection)
    - ✓ addAll(Array)
    - ✓ ensureCapacityInternal(
    - ✓ forEach(Consumer)
    - ✓ get(int)
    - ✓ indexOf(Object)
    - ✓ isEmpty()
    - ✓ iterator()
    - ✓ rangeCheck(int)
    - ✓ remove(int)
    - ✓ replaceAll(UnaryOperat
    - ✓ set(int, Object)
    - ✓ size()
    - ✓ toArray()
    - ✓ toArray(Object[])
    - ✓ uncheckedToArray()
  - ✓ posa.Array$ArrayIterator
    - ✓ hasNext()
    - ✓ next()
    - ✓ remove()

Tests written: 40

Cover wrote 40 tests listed here. Cover was also able to write 13 partial tests listed in the next section.

Change what sort of tests Cover writes in Change Settings.

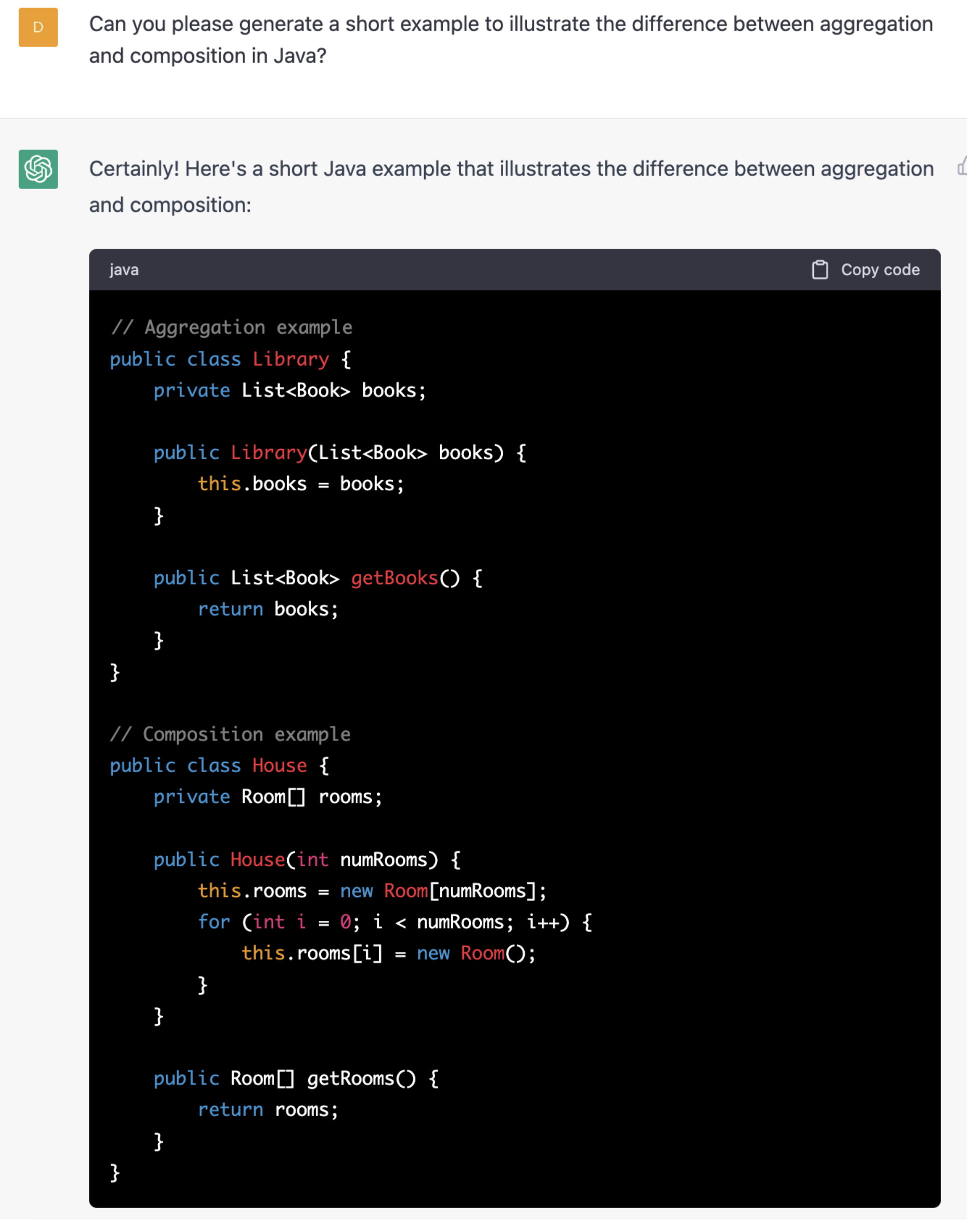
- ArrayTest.testAdd()
- ArrayTest.testAdd2()
- ArrayTest.testAdd3()
- ArrayTest.testAddAll()
- ArrayTest.testAddAll2()
- ArrayTest.testAddAll3()
- ArrayTest.testAddAll4()
- ArrayTest.testAddAll5()
- ArrayTest.testAddAll7()
- ArrayTest.testArrayIteratorHasNext()
- ArrayTest.testArrayIteratorHasNext3()
- ArrayTest.testArrayIteratorNext()
- ArrayTest.testArrayIteratorRemove()
- ArrayTest.testConstructor()
- ArrayTest.testConstructor2()
- ArrayTest.testEnsureCapacityInternal()
- ArrayTest.testEnsureCapacityInternal2()
- ArrayTest.testForEach2()
- ArrayTest.testForEach3()
- ArrayTest.testGet3()
- ArrayTest.testIndexOf()
- ArrayTest.testIndexOf2()
- ArrayTest.testIndexOf3()
- ArrayTest.testIndexOf4()
- ArrayTest.testIndexOf5()
- ArrayTest.testIsEmpty()
- ArrayTest.testIsEmpty2()
- ArrayTest.testIterator()
- ArrayTest.testRangeCheck3()
- ArrayTest.testRemove3()
- ArrayTest.testRemove4()
- ArrayTest.testReplaceAll()

Failure to combine test code: Un... (moments ago)

See www.diffblue.com/products

Examples of Using Generative AI in CS *891

- Develop lectures & assignments that expand my abilities as a teacher, e.g.
  - Add new topics that I previously lacked time to learn properly
  - Generate sample code using advanced techniques I was not facile with (yet)
  - Understand & explain very clever solutions by my students
  - Generate unit tests for my examples
  - Enables me to explain subtle design & programming concepts more effectively



Can you please generate a short example to illustrate the difference between aggregation and composition in Java?

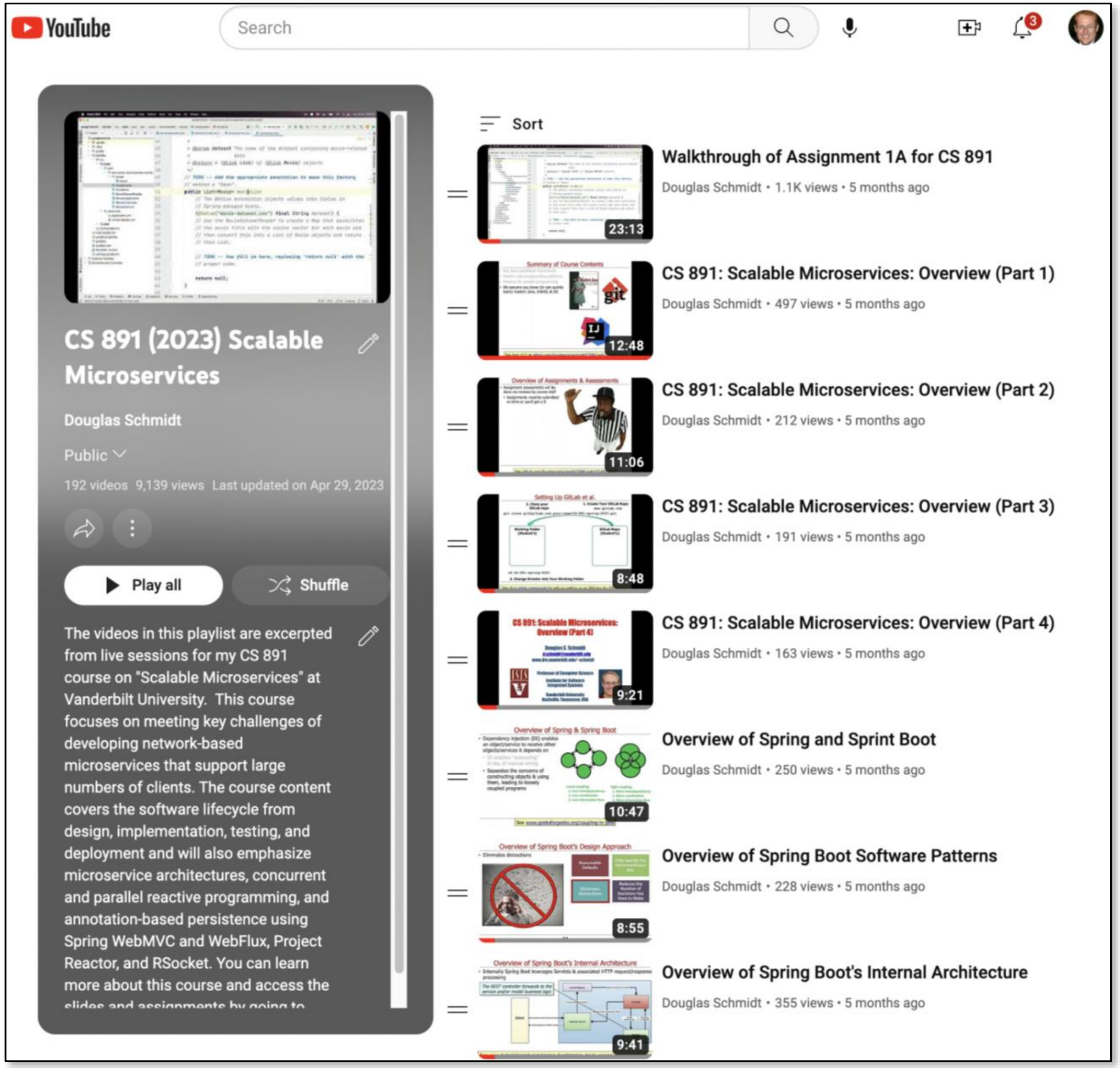
Certainly! Here's a short Java example that illustrates the difference between aggregation and composition:

```java
// Aggregation example
public class Library {
    private List<Book> books;

    public Library(List<Book> books) {
        this.books = books;
    }

    public List<Book> getBooks() {
        return books;
    }
}

// Composition example
public class House {
    private Room[] rooms;

    public House(int numRooms) {
        this.rooms = new Room[numRooms];
        for (int i = 0; i < numRooms; i++) {
            this.rooms[i] = new Room();
        }
    }

    public Room[] getRooms() {
        return rooms;
    }
}
```

See www.javatpoint.com/aggregation-vs-composition-in-java

Examples of Using Generative AI in CS *891

- Develop lectures & assignments that expand my abilities as a teacher, e.g.
  - Add new topics that I previously lacked time to learn properly
  - Generate sample code using advanced techniques I was not facile with (yet)
  - Understand & explain very clever solutions by my students
  - Generate unit tests for my examples
  - Enables me to explain subtle design & programming concepts more effectively
    - Particularly useful for responding to learners on my online forums



CS 891 (2023) Scalable Microservices

Douglas Schmidt

Public

192 videos 9,139 views Last updated on Apr 29, 2023

The videos in this playlist are excerpted from live sessions for my CS 891 course on "Scalable Microservices" at Vanderbilt University. This course focuses on meeting key challenges of developing network-based microservices that support large numbers of clients. The course content covers the software lifecycle from design, implementation, testing, and deployment and will also emphasize microservice architectures, concurrent and parallel reactive programming, and annotation-based persistence using Spring WebMVC and WebFlux, Project Reactor, and RSocket. You can learn more about this course and access the slides and assignments by going to

Walkthrough of Assignment 1A for CS 891
Douglas Schmidt • 1.1K views • 5 months ago

CS 891: Scalable Microservices: Overview (Part 1)
Douglas Schmidt • 497 views • 5 months ago

CS 891: Scalable Microservices: Overview (Part 2)
Douglas Schmidt • 212 views • 5 months ago

CS 891: Scalable Microservices: Overview (Part 3)
Douglas Schmidt • 191 views • 5 months ago

CS 891: Scalable Microservices: Overview (Part 4)
Douglas Schmidt • 163 views • 5 months ago

Overview of Spring and Sprint Boot
Douglas Schmidt • 250 views • 5 months ago

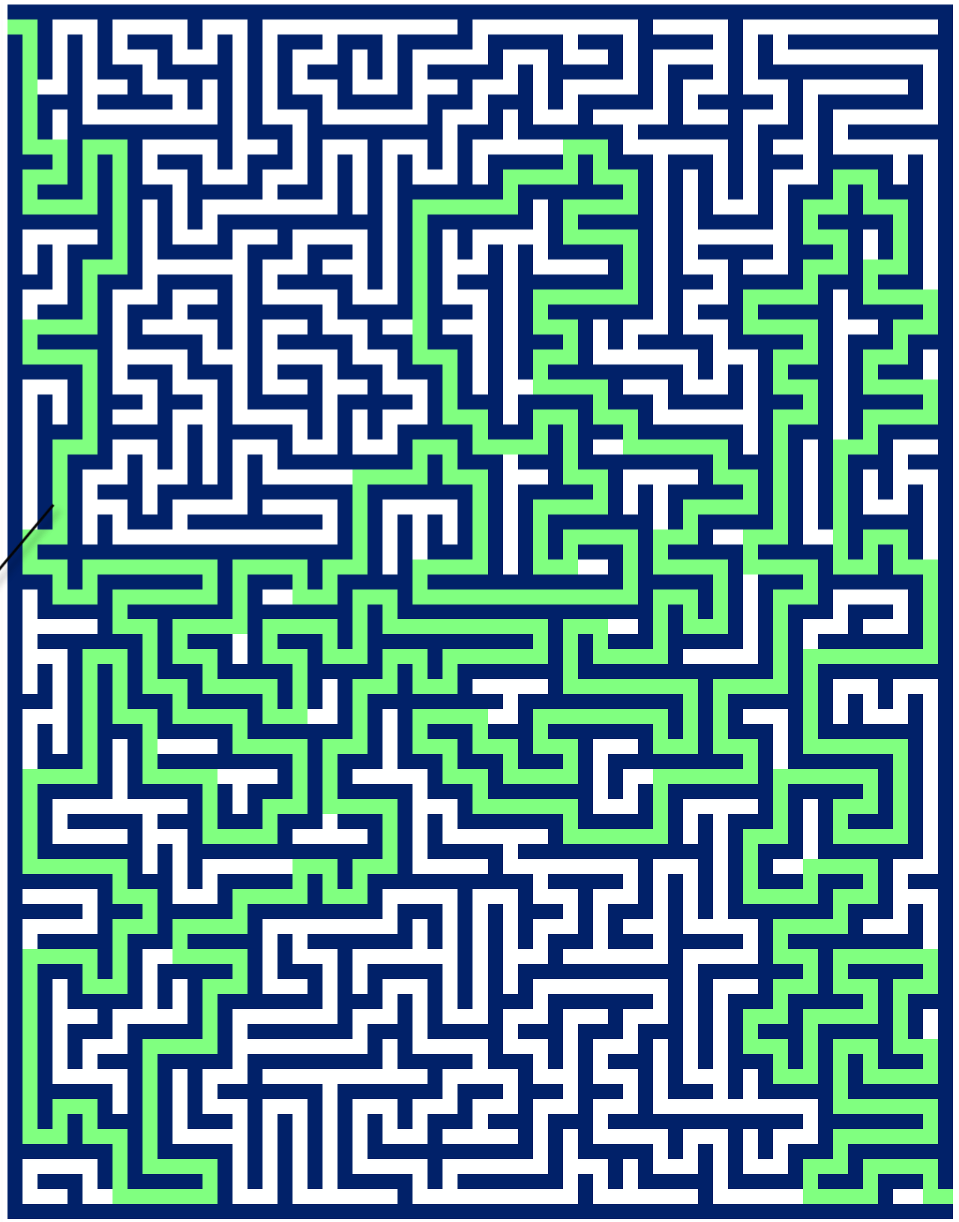
Overview of Spring Boot Software Patterns
Douglas Schmidt • 228 views • 5 months ago

Overview of Spring Boot's Internal Architecture
Douglas Schmidt • 355 views • 5 months ago

See www.youtube.com/c/DouglasSchmidt

Examples of Using Generative AI in CS *891

- Develop lectures & assignments that expand my abilities as a teacher
  - Add new topics that I previously lacked time to learn properly
  - Generate sample code using advanced techniques I was not facile with (yet)
  - Understand & explain very clever solutions by my students
  - Generate unit tests for my examples
  - Enables me to explain subtle design & programming concepts more effectively



Doing all this the "old fashioned" way would have taken me months (or forever) instead of minutes/hours/days

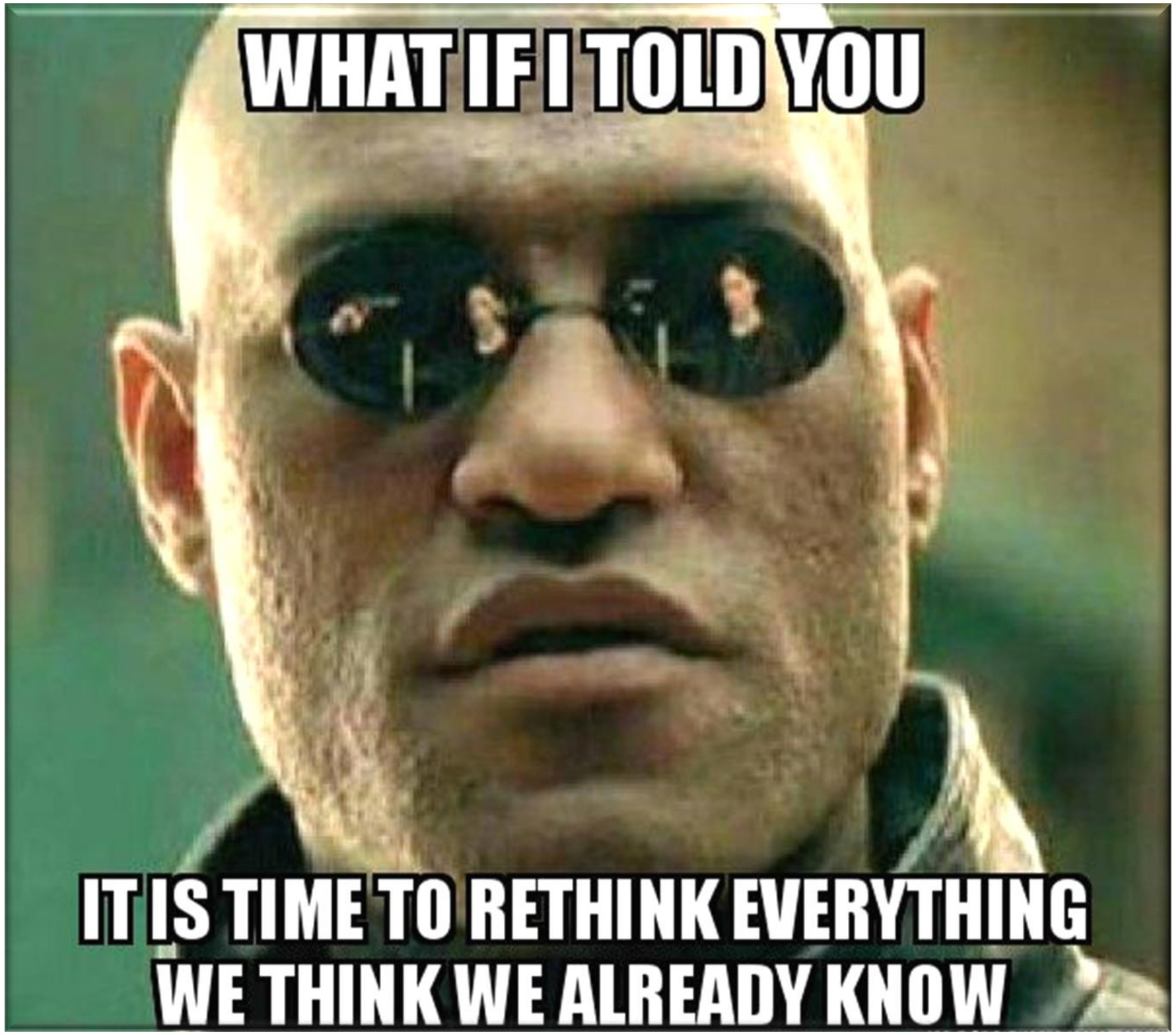
General Strategies for Using Generative AI Effectively

General Strategies for Using Generative AI Effectively

- Encourage—rather than discourage—the use of generative AI in various contexts

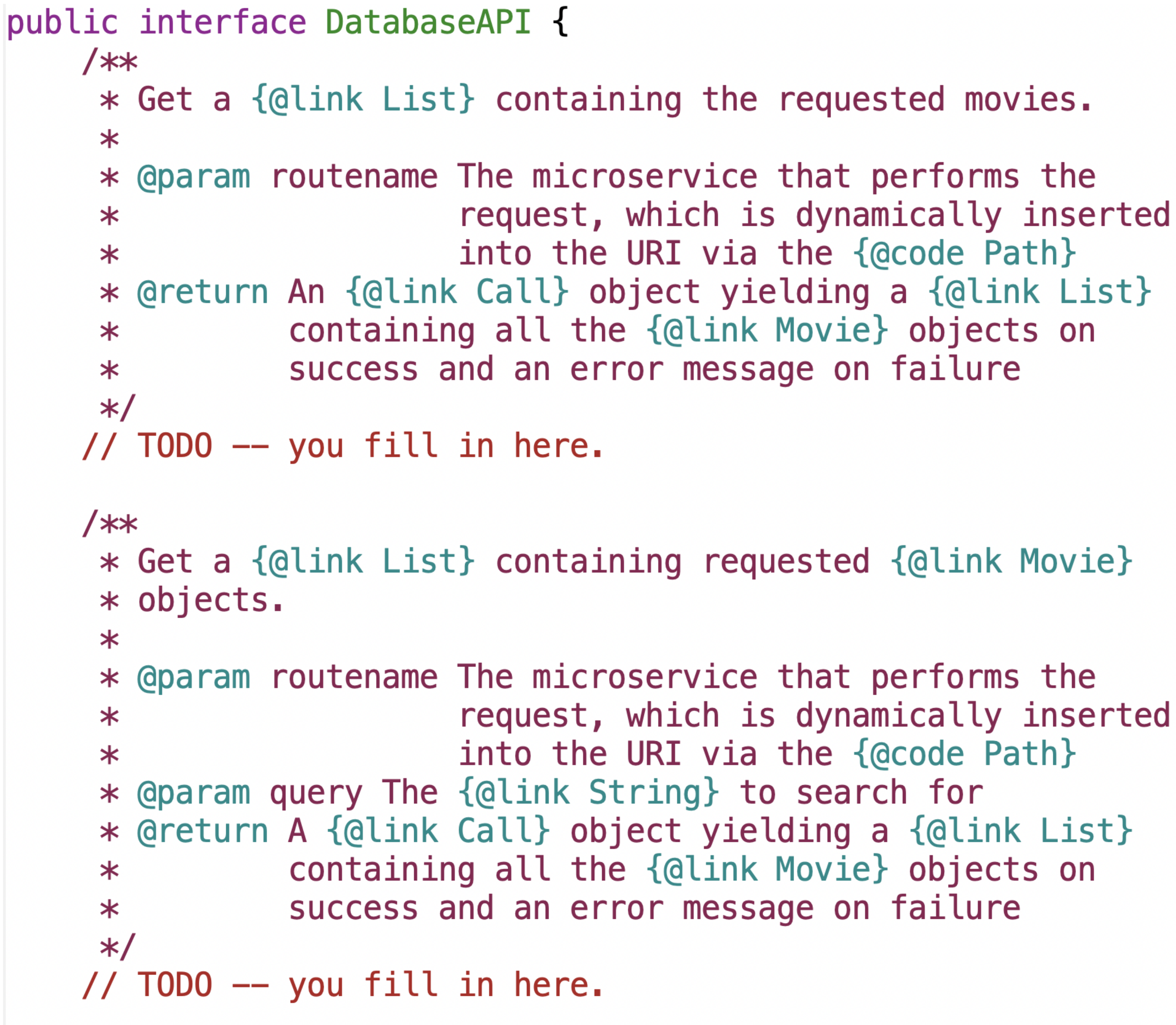
General Strategies for Using Generative AI Effectively

- Encourage—rather than discourage—the use of generative AI in various contexts
  - Requires rethinking assessments



General Strategies for Using Generative AI Effectively

- Encourage—rather than discourage—the use of generative AI in various contexts
  - Requires rethinking assessments, e.g.
    - Make assignments more "open ended" to demonstrate the power of ChatGPT

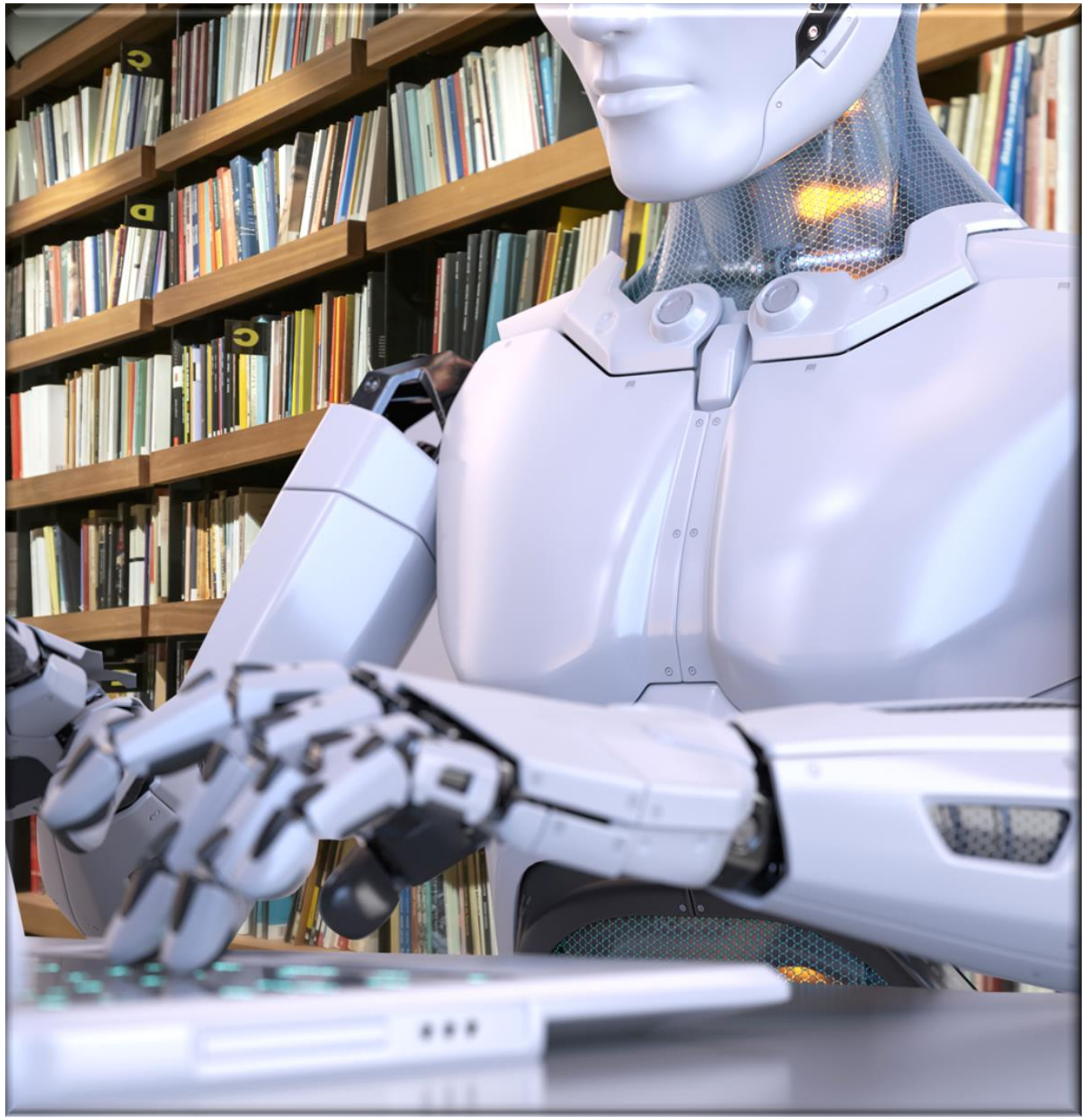


```
public interface DatabaseAPI {
    /**
     * Get a {@link List} containing the requested movies.
     *
     * @param routename The microservice that performs the
     *                  request, which is dynamically inserted
     *                  into the URI via the {@code Path}
     * @return An {@link Call} object yielding a {@link List}
     *         containing all the {@link Movie} objects on
     *         success and an error message on failure
     */
    // TODO -- you fill in here.

    /**
     * Get a {@link List} containing requested {@link Movie}
     * objects.
     *
     * @param routename The microservice that performs the
     *                  request, which is dynamically inserted
     *                  into the URI via the {@code Path}
     * @param query The {@link String} to search for
     * @return A {@link Call} object yielding a {@link List}
     *         containing all the {@link Movie} objects on
     *         success and an error message on failure
     */
    // TODO -- you fill in here.
```

See www.youtube.com/watch?v=WUpQ0V7D4bU

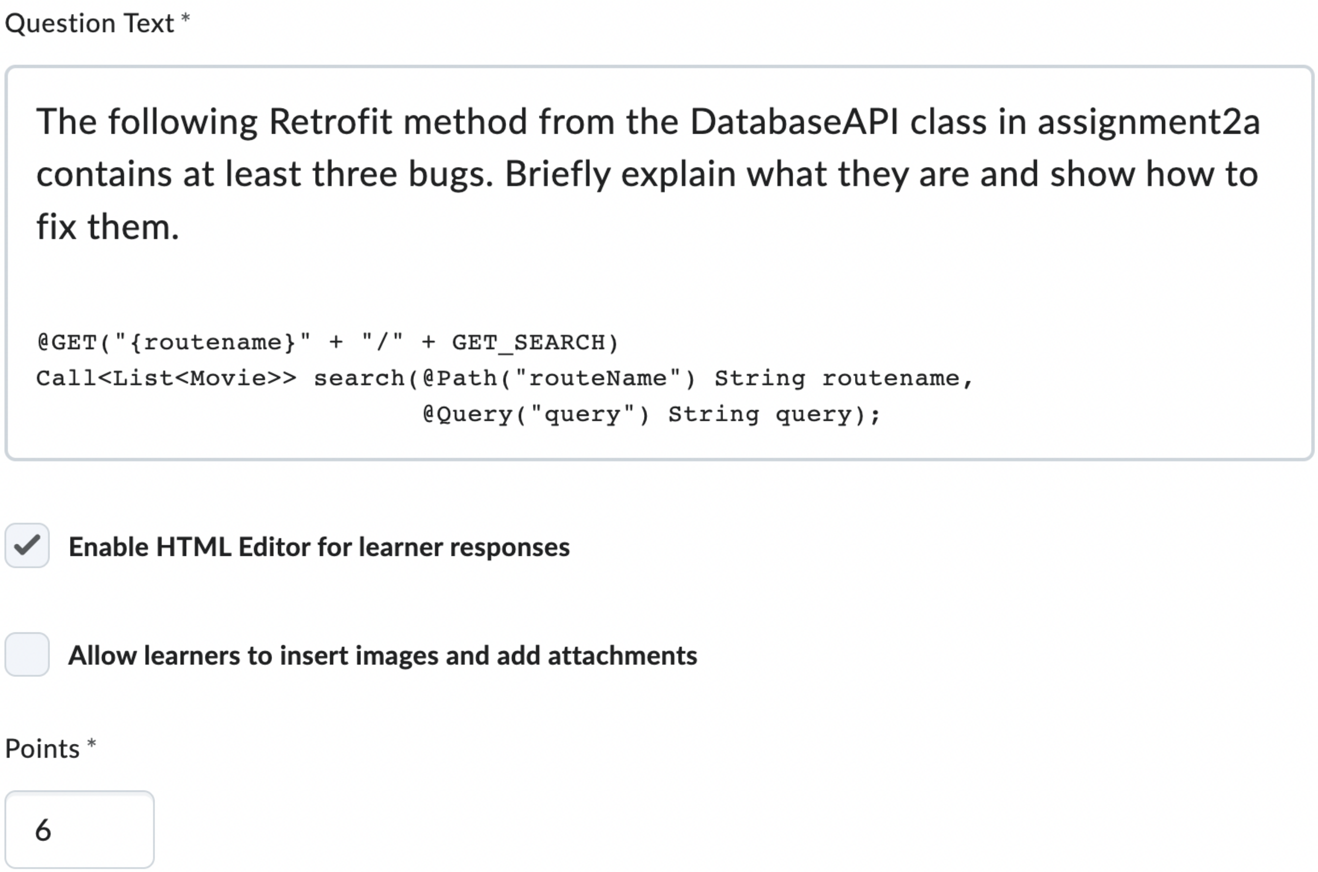
General Strategies for Using Generative AI Effectively

- Encourage—rather than discourage—the use of generative AI in various contexts
  - Requires rethinking assessments, e.g.
    - Make assignments more "open ended" to demonstrate the power of ChatGPT
    - Make quizzes/exams harder to do even if ChatGPT is used



General Strategies for Using Generative AI Effectively

- Encourage—rather than discourage—the use of generative AI in various contexts
  - Requires rethinking assessments, e.g.
    - Make assignments more "open ended" to demonstrate the power of ChatGPT
    - Make quizzes/exams harder to do even if ChatGPT is used, e.g.
      - More "contextual"



Question Text *

The following Retrofit method from the DatabaseAPI class in assignment2a contains at least three bugs. Briefly explain what they are and show how to fix them.

```
@GET("{routename}" + "/" + GET_SEARCH)
Call<List<Movie>> search(@Path("routeName") String routename,
                         @Query("query") String query);
```

☑ Enable HTML Editor for learner responses

☐ Allow learners to insert images and add attachments

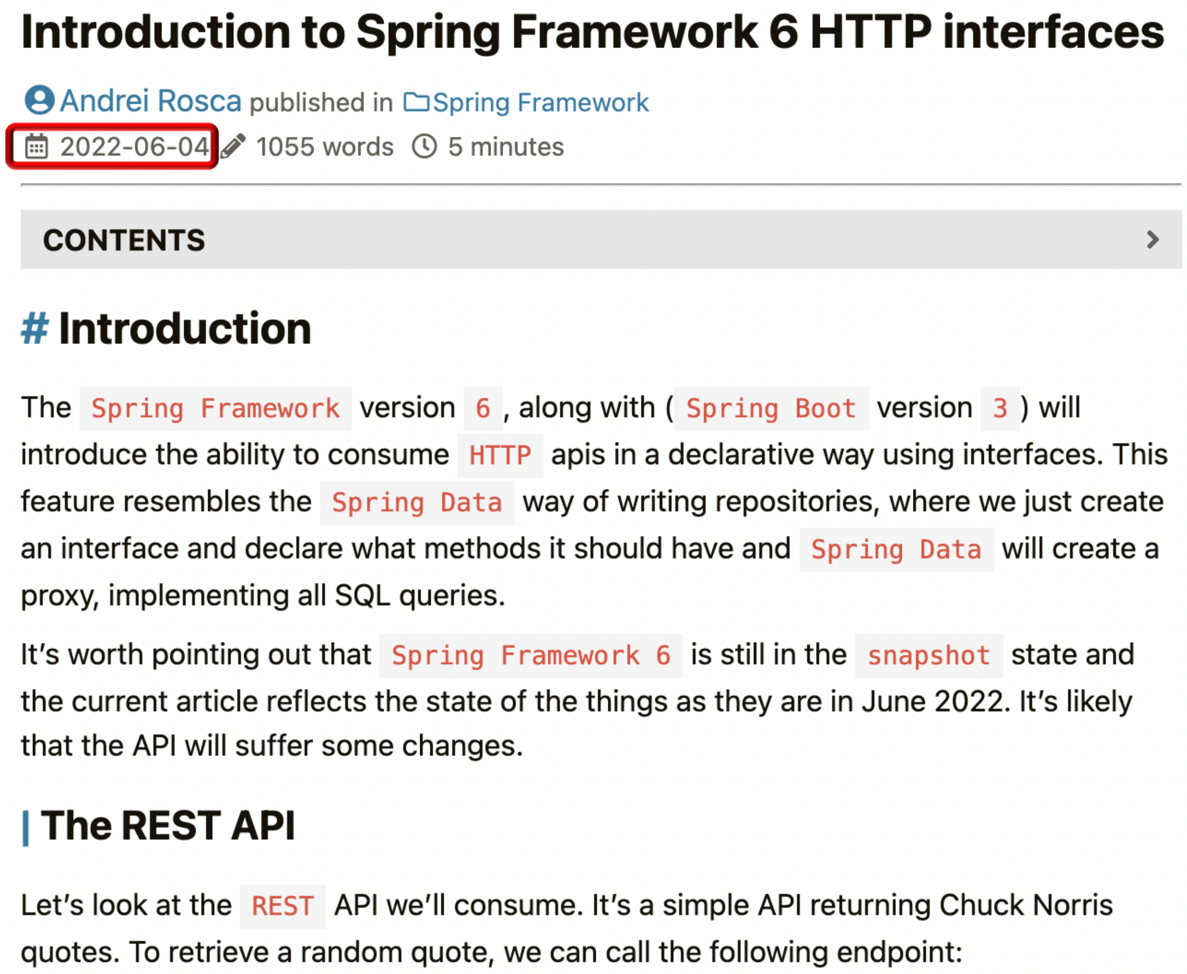
Points *

6

See www.nytimes.com/2022/05/13/opinion/college-university-remote-pandemic.html

General Strategies for Using Generative AI Effectively

- Encourage—rather than discourage—the use of generative AI in various contexts
  - Requires rethinking assessments, e.g.
    - Make assignments more "open ended" to demonstrate the power of ChatGPT
    - Make quizzes/exams harder to do even if ChatGPT is used, e.g.
      - More "contextual" & more "modern"



Introduction to Spring Framework 6 HTTP interfaces

Andrei Rosca published in Spring Framework
2022-06-04 1055 words 5 minutes

CONTENTS

Introduction

The `Spring Framework` version `6`, along with (`Spring Boot` version `3`) will introduce the ability to consume `HTTP` apis in a declarative way using interfaces. This feature resembles the `Spring Data` way of writing repositories, where we just create an interface and declare what methods it should have and `Spring Data` will create a proxy, implementing all SQL queries.

It's worth pointing out that `Spring Framework 6` is still in the `snapshot` state and the current article reflects the state of the things as they are in June 2022. It's likely that the API will suffer some changes.

| The REST API

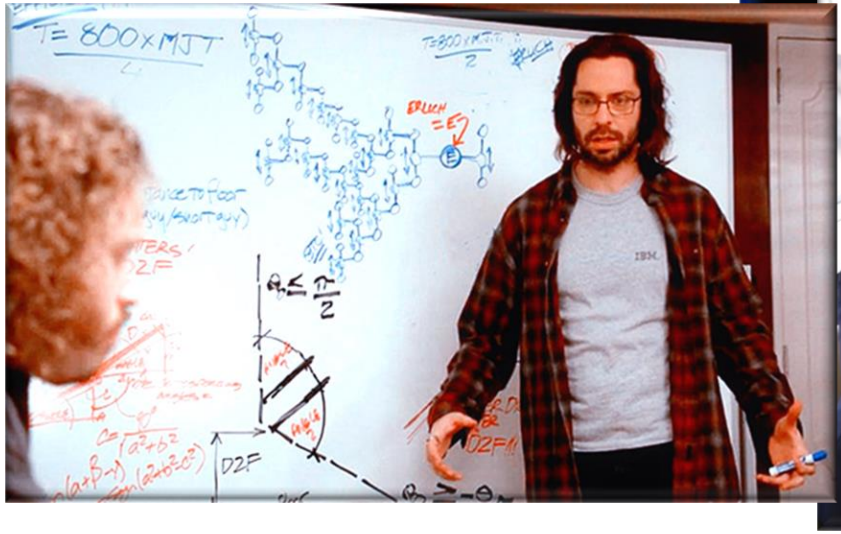
Let's look at the `REST` API we'll consume. It's a simple API returning Chuck Norris quotes. To retrieve a random quote, we can call the following endpoint:

See www.quora.com/Why-has-ChatGPT-only-limited-Kknowledge-until-2021

General Strategies for Using Generative AI Effectively

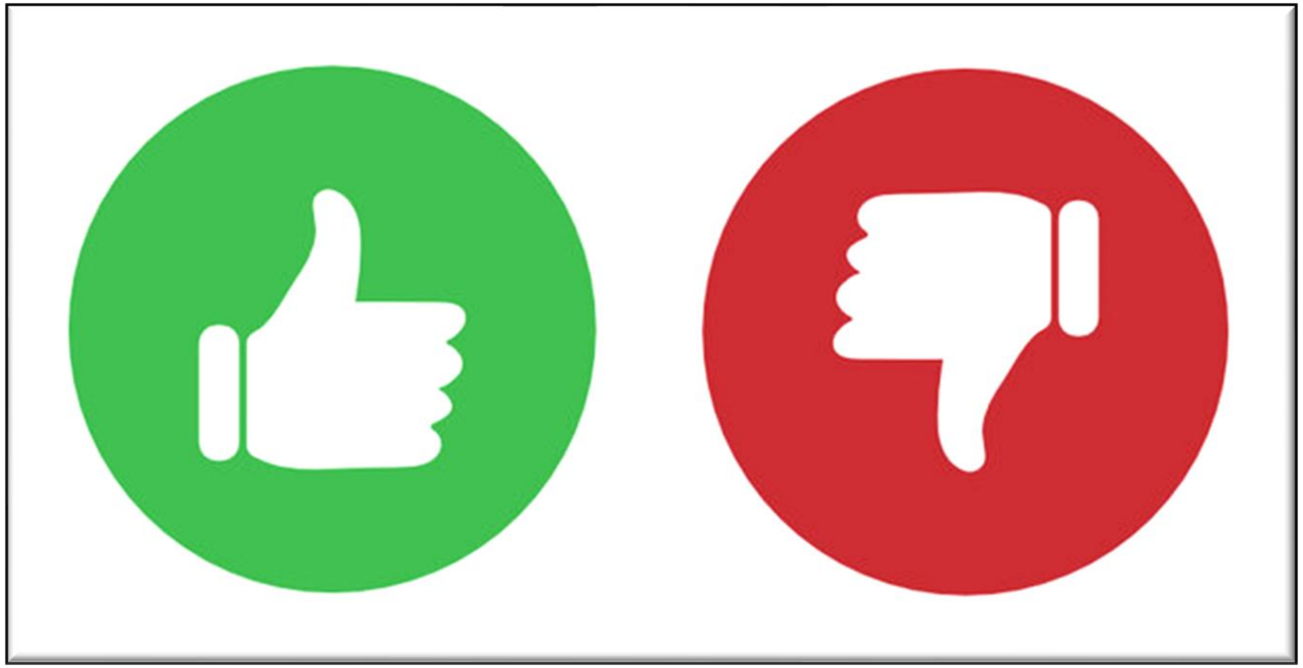
- Encourage—rather than discourage—the use of generative AI in various contexts
  - Requires rethinking assessments

Generative AI is causing us to rethink assessments models whenever individual performance must be evaluated accurately



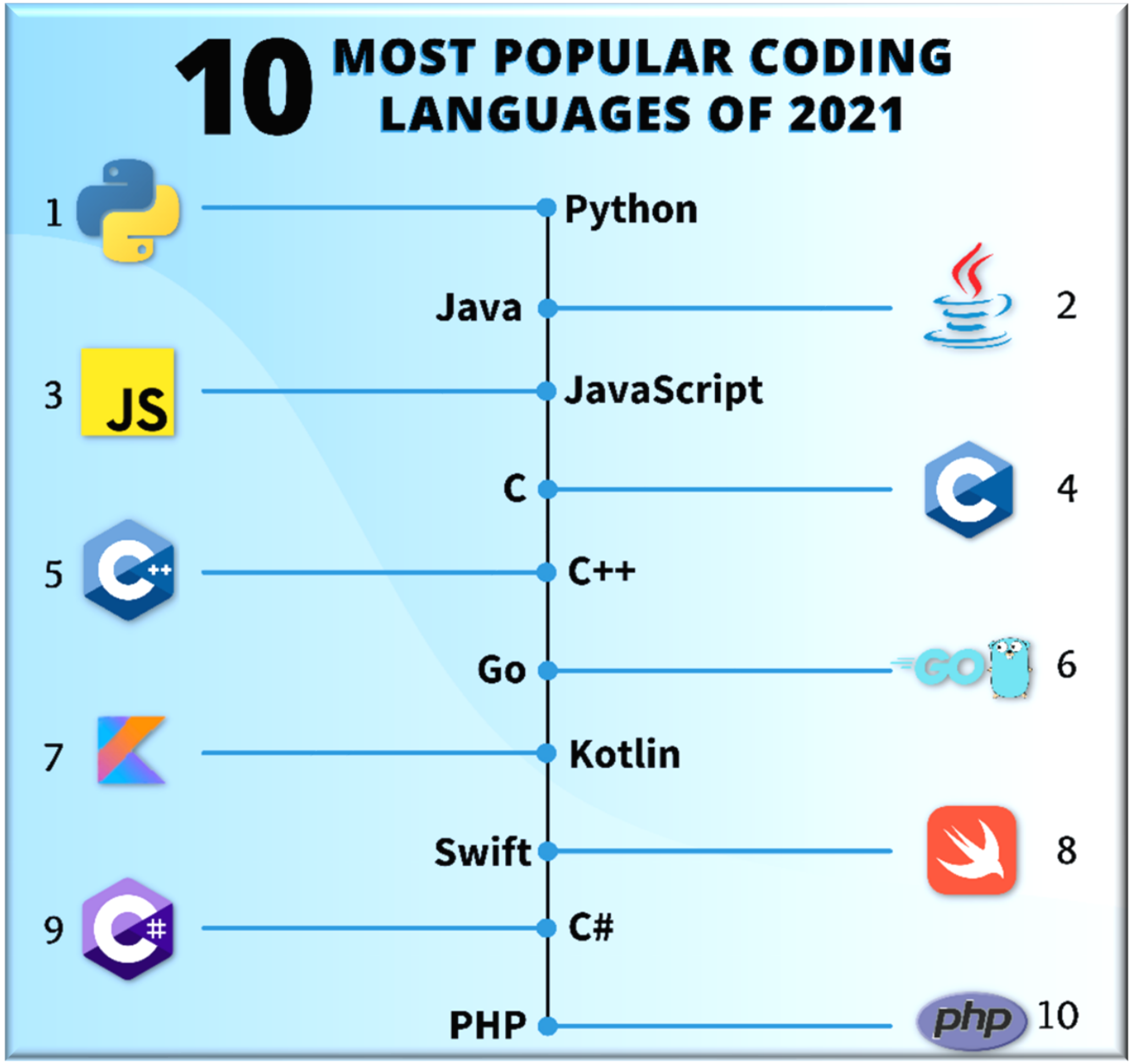
General Strategies for Using Generative AI Effectively

- Encourage—rather than discourage—the use of generative AI in various contexts
  - Requires rethinking assessments
  - Ensure users understand both the benefits & limitations of generative AI tools



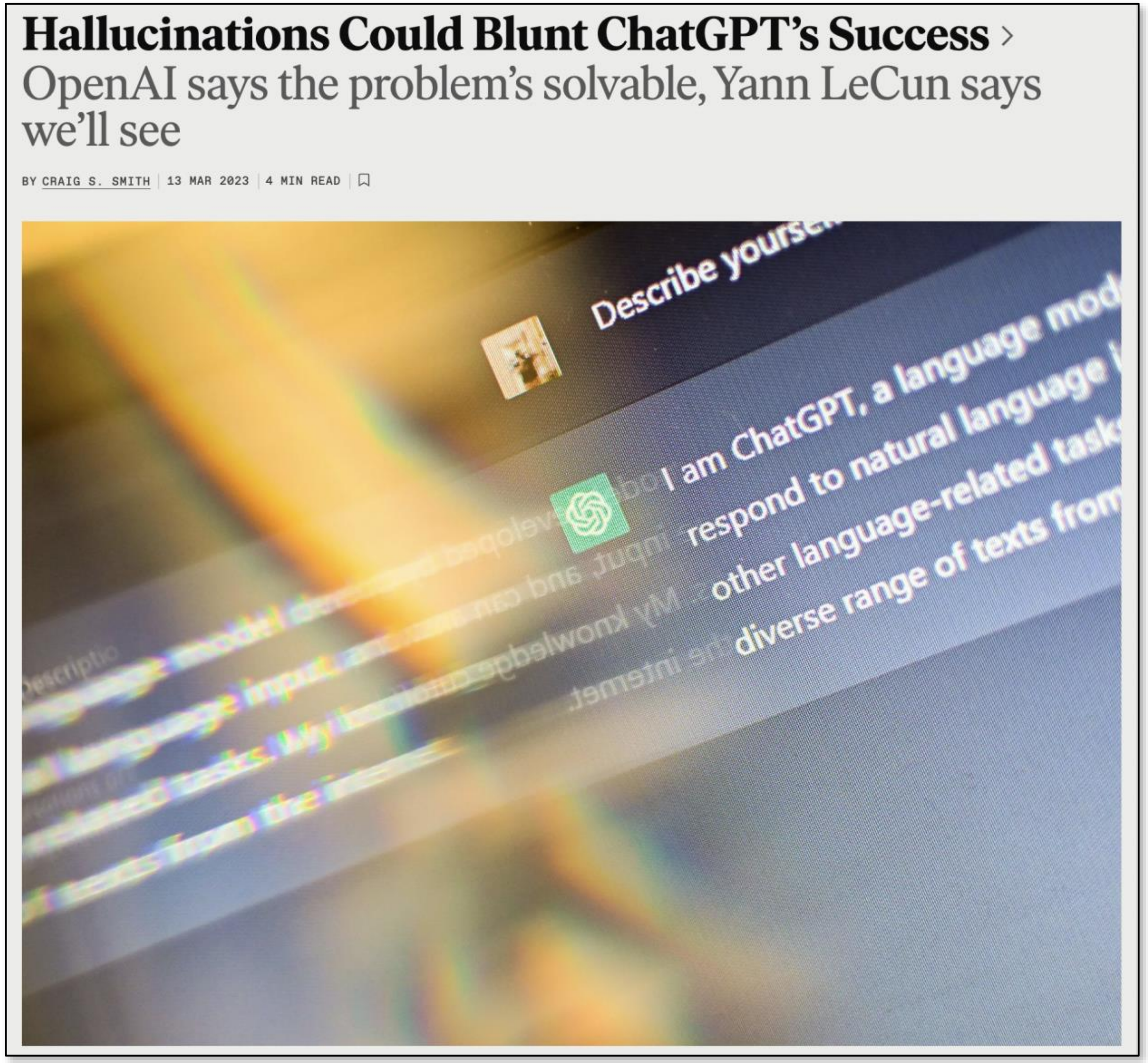
General Strategies for Using Generative AI Effectively

- Encourage—rather than discourage—the use of generative AI in various contexts
  - Requires rethinking assessments
  - Ensure users understand both the benefits & limitations of generative AI tools
    - They can be useful on topics trained on by using many good examples



General Strategies for Using Generative AI Effectively

- Encourage—rather than discourage—the use of generative AI in various contexts
  - Requires rethinking assessments
  - Ensure users understand both the benefits & limitations of generative AI tools
    - They can be useful on topics trained on by using many good examples
    - However, "hallucinations" are common for topics w/limited quality training sets



Hallucinations Could Blunt ChatGPT's Success ›

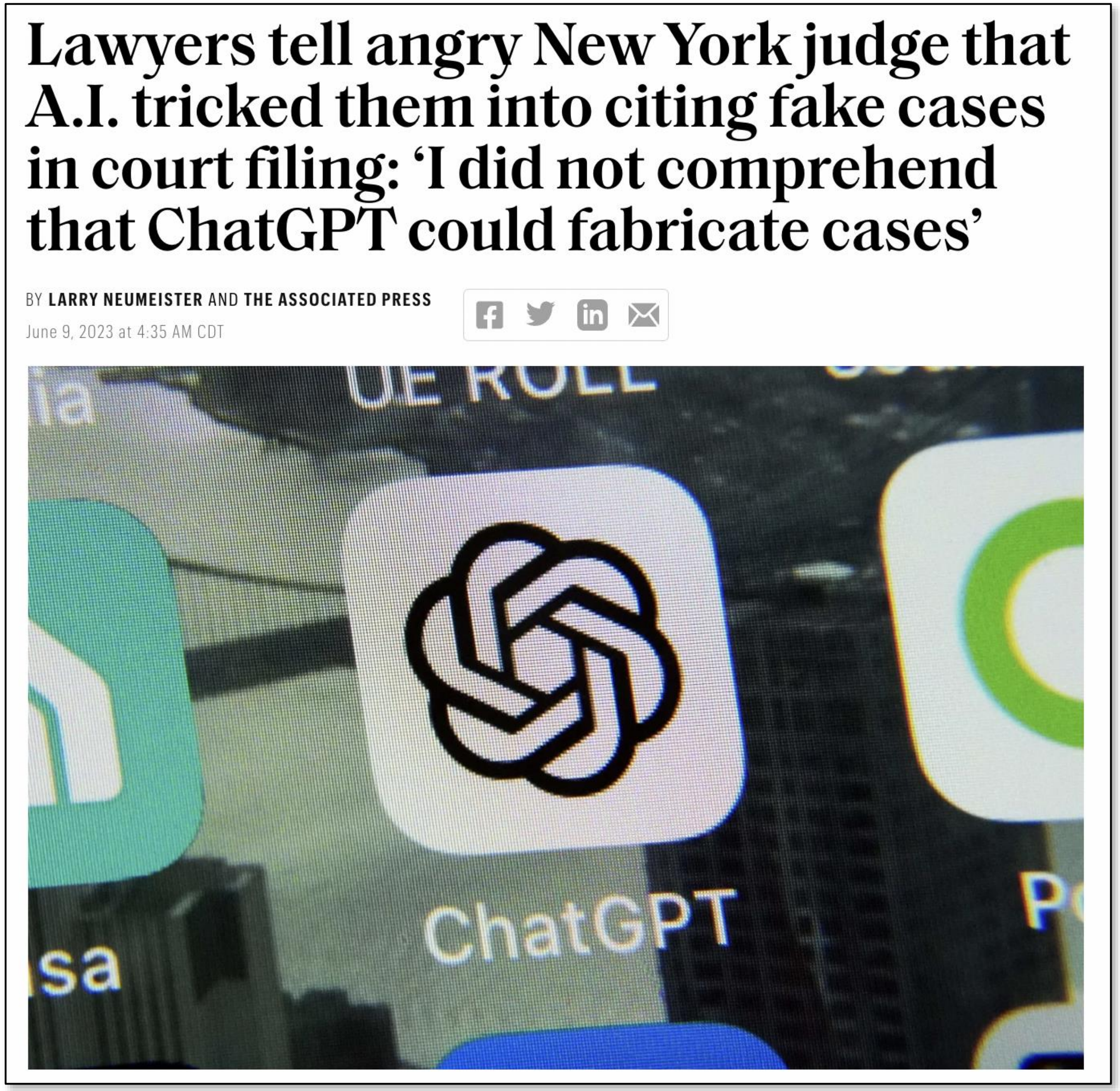
OpenAI says the problem's solvable, Yann LeCun says we'll see

BY CRAIG S. SMITH | 13 MAR 2023 | 4 MIN READ

Describe yourse
I am ChatGPT, a language mod
respond to natural language
other language-related task
diverse range of texts from

General Strategies for Using Generative AI Effectively

- Encourage—rather than discourage—the use of generative AI in various contexts
  - Requires rethinking assessments
  - Ensure users understand both the benefits & limitations of generative AI tools
    - They can be useful on topics trained on by using many good examples
    - However, "hallucinations" are common for topics w/limited quality training sets
      - This has led to highly embarrassing outcomes..



Lawyers tell angry New York judge that A.I. tricked them into citing fake cases in court filing: 'I did not comprehend that ChatGPT could fabricate cases'

BY **LARRY NEUMEISTER** AND **THE ASSOCIATED PRESS**

June 9, 2023 at 4:35 AM CDT

See fortune.com/2023/06/09/lawyers-angry-new-york-judge-ai-tricked-them-citing-fake-cases-court-filing-chatgpt

General Strategies for Using Generative AI Effectively

- Encourage—rather than discourage—the use of generative AI in various contexts
  - Requires rethinking assessments
  - Ensure users understand both the benefits & limitations of generative AI tools
    - They can be useful on topics trained on by using many good examples
    - However, "hallucinations" are common for topics w/limited quality training sets
    - Generative AI tools are most effective when they supplement—rather than supplant—the learning process

General Strategies for Using Generative AI Effectively

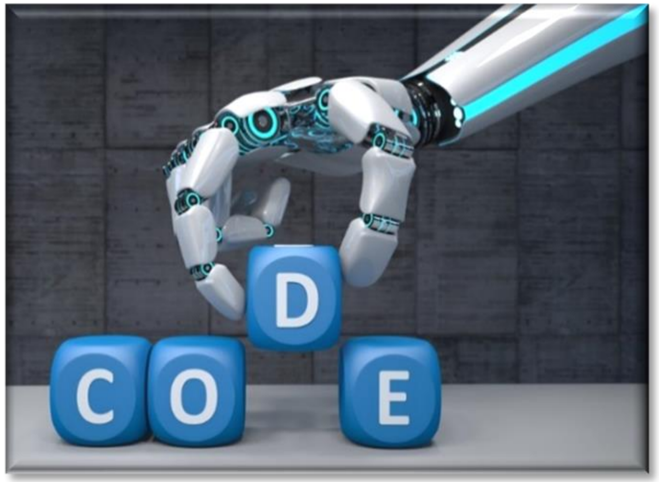
- Encourage—rather than discourage—the use of generative AI in various contexts
  - Requires rethinking assessments
  - Ensure users understand both the benefits & limitations of generative AI tools
    - They can be useful on topics trained on by using many good examples
    - However, "hallucinations" are common for topics w/limited quality training sets
    - Generative AI tools are most effective when they supplement—rather than supplant—the learning process

It's particularly important for novices to use these tools carefully

Wrapping Up

Wrapping Up

- Learning to use generative AI tools effectively is a step towards achieving our vision of software development in which humans & AI work together as ***trustworthy*** collaborators



ARCHITECTING THE FUTURE OF SOFTWARE ENGINEERING
A National Agenda for Software Engineering Research & Development
Carnegie Mellon University
Software Engineering Institute

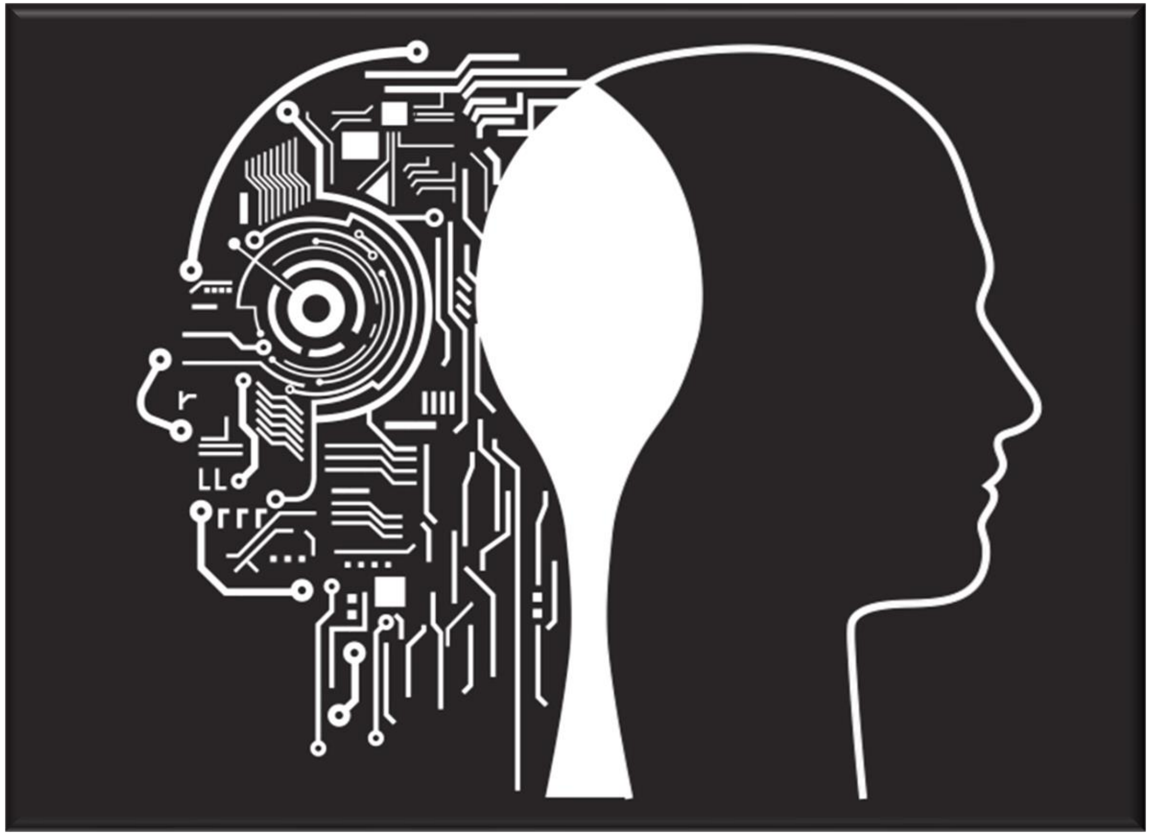
Wrapping Up

- Learning to use generative AI tools effectively is a step towards achieving our vision of software development in which humans & AI work together as ***trustworthy*** collaborators

Achieving "trustworthiness" may be the 10-15 year horizon of our original vision..

Wrapping Up

- Learning to use generative AI tools effectively is a step towards achieving our vision of software development in which humans & AI work together as ***trustworthy*** collaborators, e.g.
  - Rapidly evolve robust software-reliant systems based on programmer intent



Wrapping Up

- Learning to use generative AI tools effectively is a step towards achieving our vision of software development in which humans & AI work together as ***trustworthy*** collaborators, e.g.
  - Rapidly evolve robust software-reliant systems based on programmer intent
    - Soon everyone will be a programmer!
      - "A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT," Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, & Douglas C. Schmidt, https://arxiv.org/abs/2302.11382
      - "ChatGPT Prompt Patterns for Improving Code Quality, Refactoring, Requirements Elicitation, & Software Design," Jules White, Sam Hays, Quchen Fu, Jesse Spencer-Smith, & Douglas C. Schmidt, https://arxiv.org/abs/2303.07839

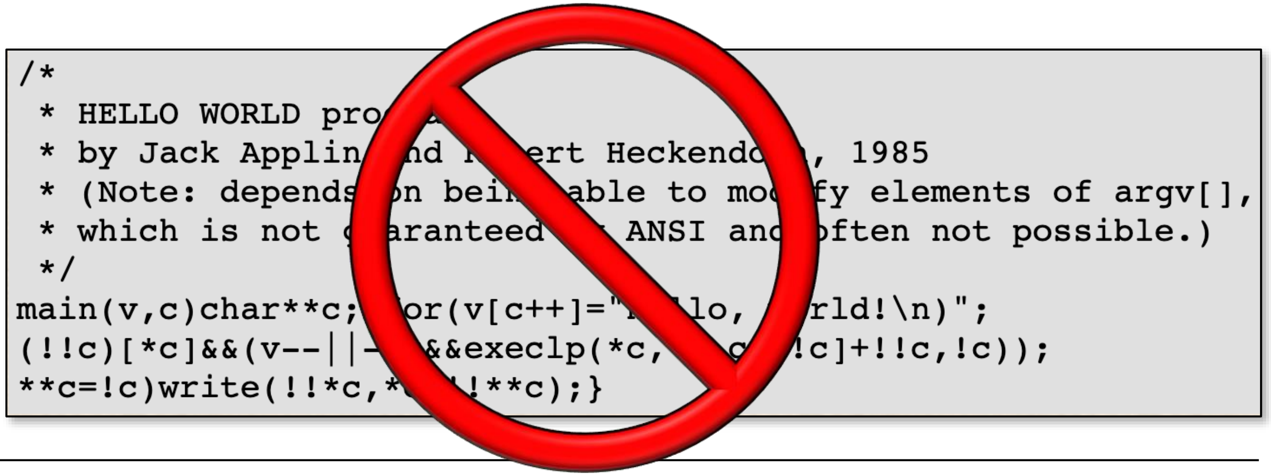
This image was AI-generated with Midjourney

Wrapping Up

- Learning to use generative AI tools effectively is a step towards achieving our vision of software development in which humans & AI work together as ***trustworthy*** collaborators, e.g.
  - Rapidly evolve robust software-reliant systems based on programmer intent
  - Refocus humans on more creative & architectural aspects of software

Wrapping Up

- Learning to use generative AI tools effectively is a step towards achieving our vision of software development in which humans & AI work together as ***trustworthy*** collaborators, e.g.
  - Rapidly evolve robust software-reliant systems based on programmer intent
  - Refocus humans on more creative & architectural aspects of software
    - Rather than mundane details



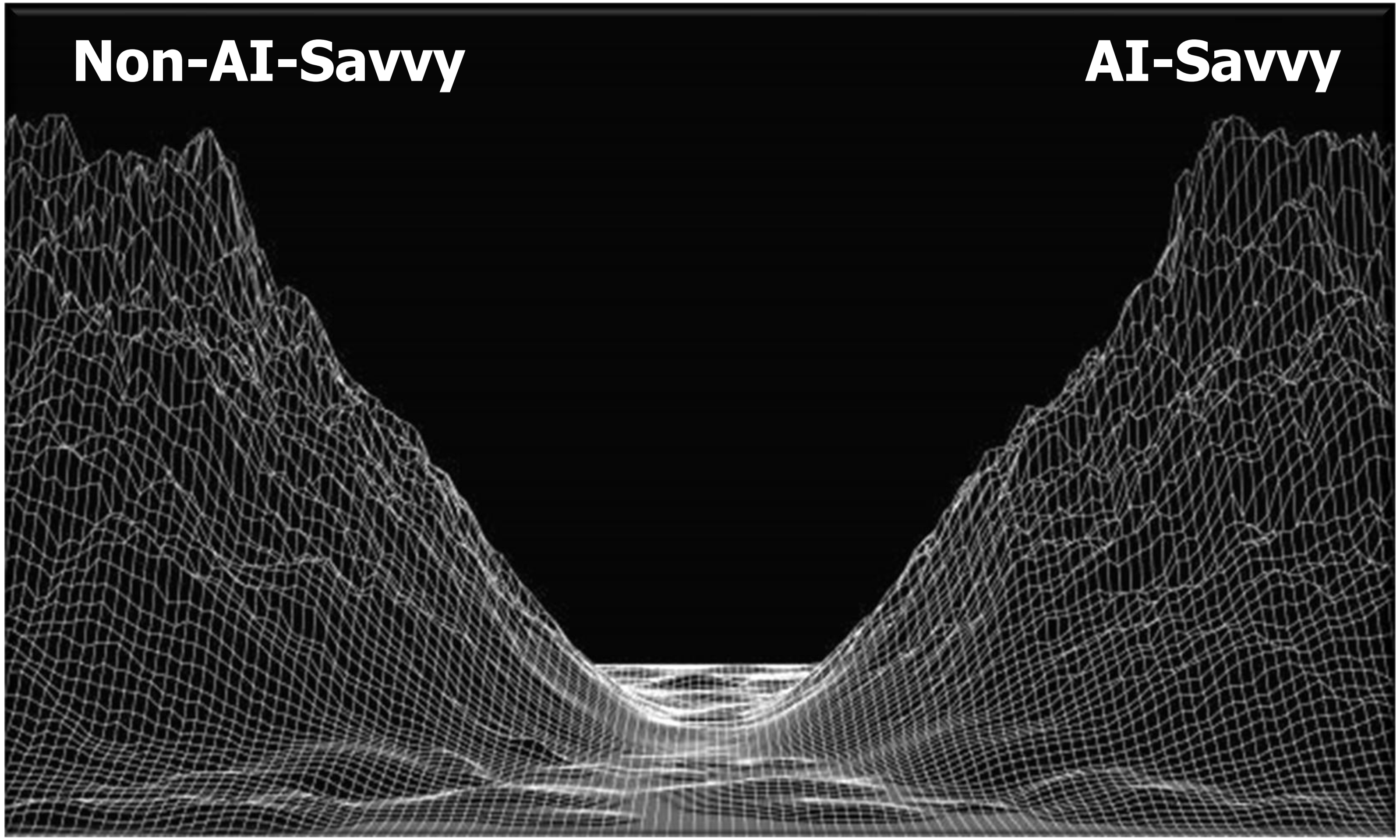
```
/*
 * HELLO WORLD pro
 * by Jack Applin  nd   ert Heckendo  , 1985
 * (Note: depends  n bein  able to mod  fy elements of argv[],
 * which is not    aranteed    ANSI and  ften not possible.)
 */
main(v,c)char**c;  or(v[c++]="    lo,   rld!\n)";
(!!c)[*c]&&(v--||-  &&execlp(*c,    c  !c]+!!c,!c));
**c=!c)write(!!*c,*   !**c);}
```

Wrapping Up

- Much R&D remains before we can achieve the vision of humans & AI working together as trusted collaborators, but there's no time like the present to jump in!

Wrapping Up

- Much R&D remains before we can achieve the vision of humans & AI working together as trusted collaborators, but there's no time like the present to jump in!
  - Must be(a)ware of the forthcoming "digital chasm"..



See fortune.com/2023/04/25/artificial-intelligence-ai-replace-humans-prompt-engineers-chatgpt

"Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Networking and Information Technology Research and Development Program."

The Networking and Information Technology Research and Development
(NITRD) Program

Mailing Address: NCO/NITRD, 2415 Eisenhower Avenue, Alexandria, VA 22314

Physical Address: 490 L'Enfant Plaza SW, Suite 8001, Washington, DC 20024, USA Tel: 202-459-9674,
Fax: 202-459-9673, Email: nco@nitrd.gov, Website: https://www.nitrd.gov