

# How Will AI Change Software?

Dean Wampler, Ph.D.  
The AI Alliance  
[dwampler@thealliance.ai](mailto:dwampler@thealliance.ai)  
January 14, 2026

[aialliance.org](http://aialliance.org)

[deanwampler.com/talks](http://deanwampler.com/talks)



# Outline (1/3)

- First, about the AI Alliance
- How do you use AI today? Probably two ways:
  1. Adding new capabilities to your apps that were previously not possible.
  2. Accelerating your productivity.

# Outline (2/3)

1. Adding new capabilities to your apps that were previously not possible.
  - Is this actually working?
  - Why are PoCs not transitioning to production?

# Outline (3/3)

## 2. Accelerating your productivity.

- Today, we speed up “old” ways of working.
- How might AI fundamentally change SW Engineering?

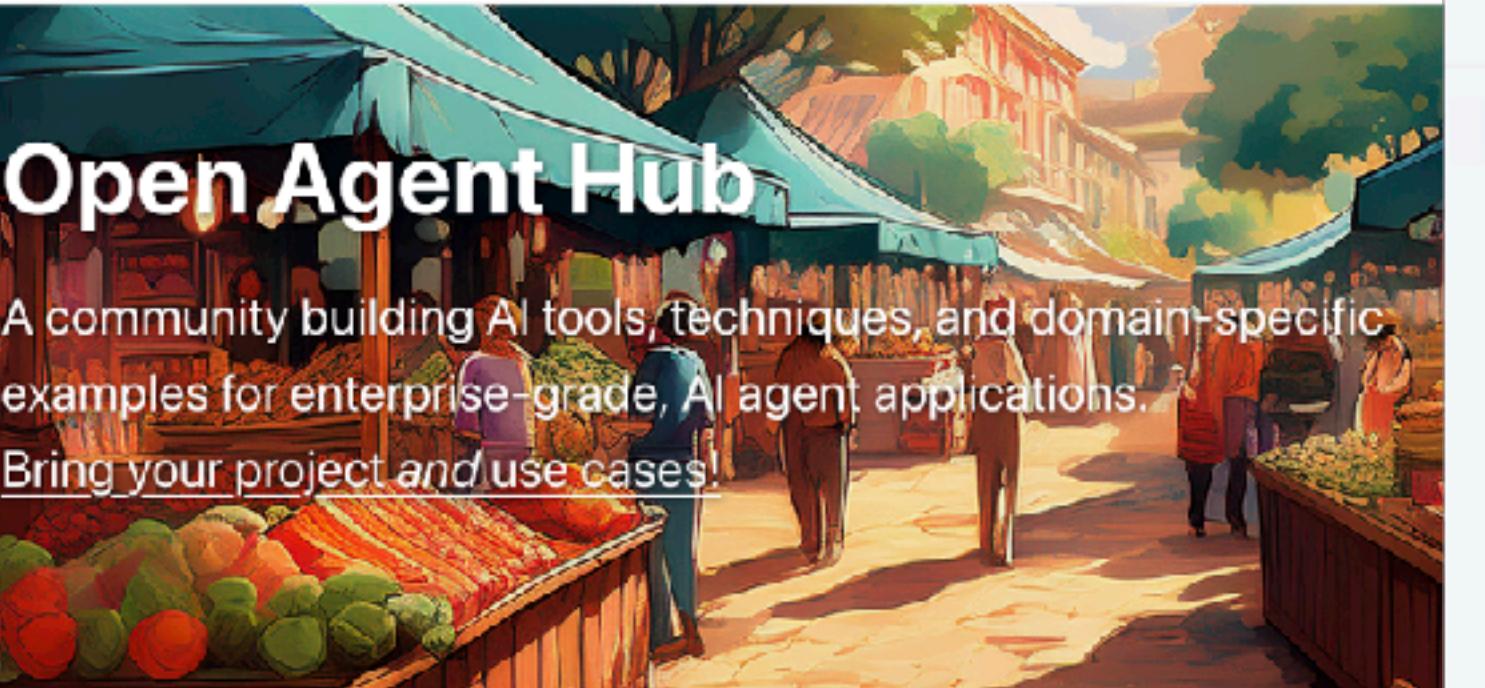
# AI ALLIANCE

195+ organizations in 25+ countries accelerating open innovation and adoption of AI

These project areas:  
[the-ai-alliance.github.io/](https://the-ai-alliance.github.io/)

## 1. Agents

### Open Agent Hub



A community building AI tools, techniques, and domain-specific examples for enterprise-grade, AI agent applications. Bring your project and use cases!

The AI Alliance Community is a non-profit (501(c)(3)) foundation that supports the success of essential projects in the AI community with particular focus on data, models, and agents, including their governance dedicated to the benefit of AI for all of society – not just industry and not just those who can pay to h

### AI ALLIANCE

#### Open Trusted Data Initiative

Start Here!

Trustworthiness

Dataset Catalog

Dataset Specification

How We Process Datasets

How to Contribute to OTDI

About Us

References

### AI ALLIANCE

Search Open Trusted Data Initiative

Browse the Datasets

Tell Us About Other Datasets

#### Open Trusted Data Initiative (OTDI)

We are building the world's largest, most diverse catalog of open and sourced datasets for AI. [Join us!](#)

#### Datasets for Languages

Datasets with different human languages.

##### Subcategories

African Languages

Languages in the Americas

Asian Languages

European Languages

Languages in the Middle East

### AI ALLIANCE

#### AI Alliance GitHub Organization

Home

Open Agent Hub Projects

Open Data and Model Foundry Projects

Safety, Governance, and Education Projects

Contributing

About Us

Microsite Cheat Sheet

### AI ALLIANCE

Search AI Alliance GitHub Organization

AI Alliance

GitHub Organization

Repos

AI Alliance Events

#### AI Safety, Governance, and Education

Collaborate on the necessary enablers of successful AI applications.

In order for the objectives of the [Open Agent Hub](#) and the [Open Data and Model Foundry](#) to be achieved, fundamental requirements must be met for safety, governance, and the expertise required to use AI technologies effectively.

AI Safety encompasses classic cybersecurity, as well as AI-specific concerns, such as suppression of undesirable content and compliance with regulations and social norms. A more general term is *trustworthiness*, which adds concerns about ensuring accuracy (i.e., minimizing hallucinations) and meeting the specific requirements for application use cases, etc. Enterprises won't deploy AI applications into production scenarios if they don't trust them to behave as expected.

Governance is an aspect of trustworthiness, specifically the assurances that all end-to-end



# Outline (2/3)

1. Adding new capabilities to your apps that were previously not possible.
  - Is this actually working?
  - Why are PoCs not transitioning to production?

# 1. Adding new capabilities to your apps that were previously not possible.



# The GenAI Divide STATE OF AI IN BUSINESS 2025

## MIT NANDA

Aditya Challapally  
Chris Pease  
Ramesh Raskar  
Pradyumna Chari  
July 2025

[https://mlq.ai/media/quarterly\\_decks/v0.1\\_State\\_of\\_AI\\_in\\_Business\\_2025\\_Report.pdf](https://mlq.ai/media/quarterly_decks/v0.1_State_of_AI_in_Business_2025_Report.pdf)

# Is this working?

≡ Search

## MIT report: 95% of generative AI pilots at companies are failing

By Sheryl Estrada  
Senior Writer And Author Of CFO Daily

August 18, 2025, 6:54 AM ET

<https://fortune.com/2024/08/18/mit-report-95-percent-generative-ai-pilots-at-companies-failing-cfo/>

ePRnews

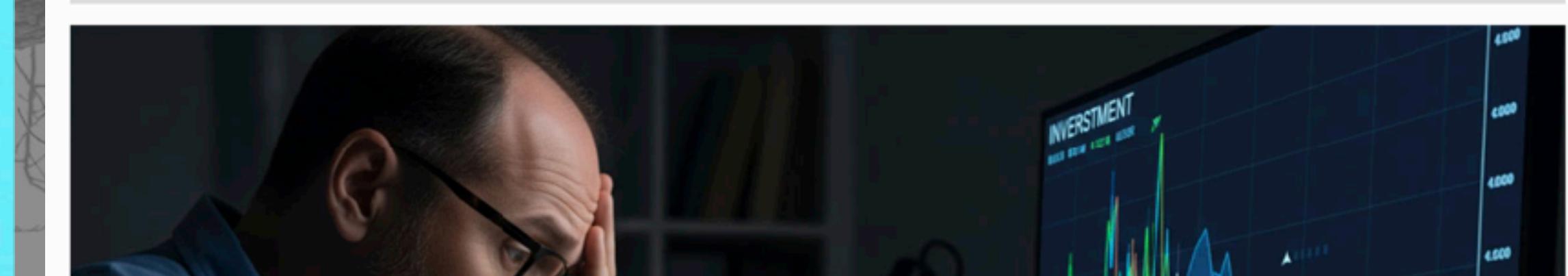
Home News Articles Pricing RSS

## MIT Report Reveals Shocking 95% Failure Rate for Corporate AI Projects Despite \$30-40 Billion Investment

Published On: Aug 25, 2025 (UTC) 4389

Copy Link

*Study Exposes 'GenAI Divide' as Most Enterprise AI Pilots Stall at Early Stages, Delivering No Measurable Business Returns*



<https://eprnews.com/mit-report-reveals-shocking-95-failure-rate-for-corporate-ai-projects-686740/>

**LATEST POSTS**



The Great Decoupling is Finally Growing

JAN 12, 2026



Biohacking: Three Solutions

DEC 05, 2025



Boosting Social Role of AI in Data

NOV 26, 2025



The World's First AI Browser

Browsing with A

DEC 17, 2025

“...the core issue isn't the quality of AI models themselves, but rather the ‘learning gap’ for both tools and organizations...”

“Generic tools like ChatGPT excel for individuals because of their **flexibility**, but they stall in enterprise use since they don't learn from or adapt to workflows,”

— Aditya Challapally

ng?

UNE

iling-cfo/

#### LATEST POSTS



The Great Decoupling is Finally Growing  
JAN 12, 2026



Biohacking: Three Solutions  
DEC 05, 2025



Boosting Social: The Role of AI in Data  
NOV 26, 2025



Fellow AI Browser  
Browsing with A



## Why Do Enterprise Agents Fail? Insights from IT-Bench using MAST

Mert Cemri, Melissa Pan, Ion Stoica and the MAST Team (UC Berkeley)

Saurabh Jha, Rohan Arora, Daby Sow, Nicholas Fuller (IBM Research)

Posted: December 19, 2025

💡 **Agentic LLMs are increasingly adopted in real world IT tasks**, for tasks like triaging incidents, querying logs/metrics and generating Kubernetes actions. However, evaluating these agentic systems is hard. Existing benchmarks, such as IT-Bench, typically provide just a single number (e.g., success rate) which is insufficient to understand where these systems fail and how to fix them. In this post, we aim to alleviate this challenge by using **MAST** (Multi-Agent System Failure Taxonomy) to turn **ITBench execution traces from SRE scenarios** into structured failure signatures that not only show whether a run fails, but can also explain *how and why* the run failed, thus providing insights into how to fix it.

- **Beyond Accuracy:** Success rates on ITBench (SRE, Security, FinOps tasks) only tell you *if* an agent failed. **MAST** reveals *how it failed*.
- **The "Isolated" vs. "Cascading" Divide:** Our analysis identifies a **Failure Complexity Hierarchy**. Frontier models like Gemini-3-Flash exhibit "Isolated Failures" (2.6 failure modes/trace), typically failing at a single, discrete bottleneck. In contrast, GPT-OSS-120B suffers from "Cascading Collapse" (5.3 failure modes/trace), where one minor reasoning mismatch triggers a compounding, systemic breakdown.
- **Fatal vs. Non-Fatal (Benign) Failure Modes:** We separate fatal failure modes such as (i) agents not knowing when to stop or reasoning-action misalignment of agents from (ii) benign and non-fatal failure modes such as messy behavior that can still

# Is this working?

Recent academic work  
from UC Berkeley and  
IBM Research on ways  
agents fail.

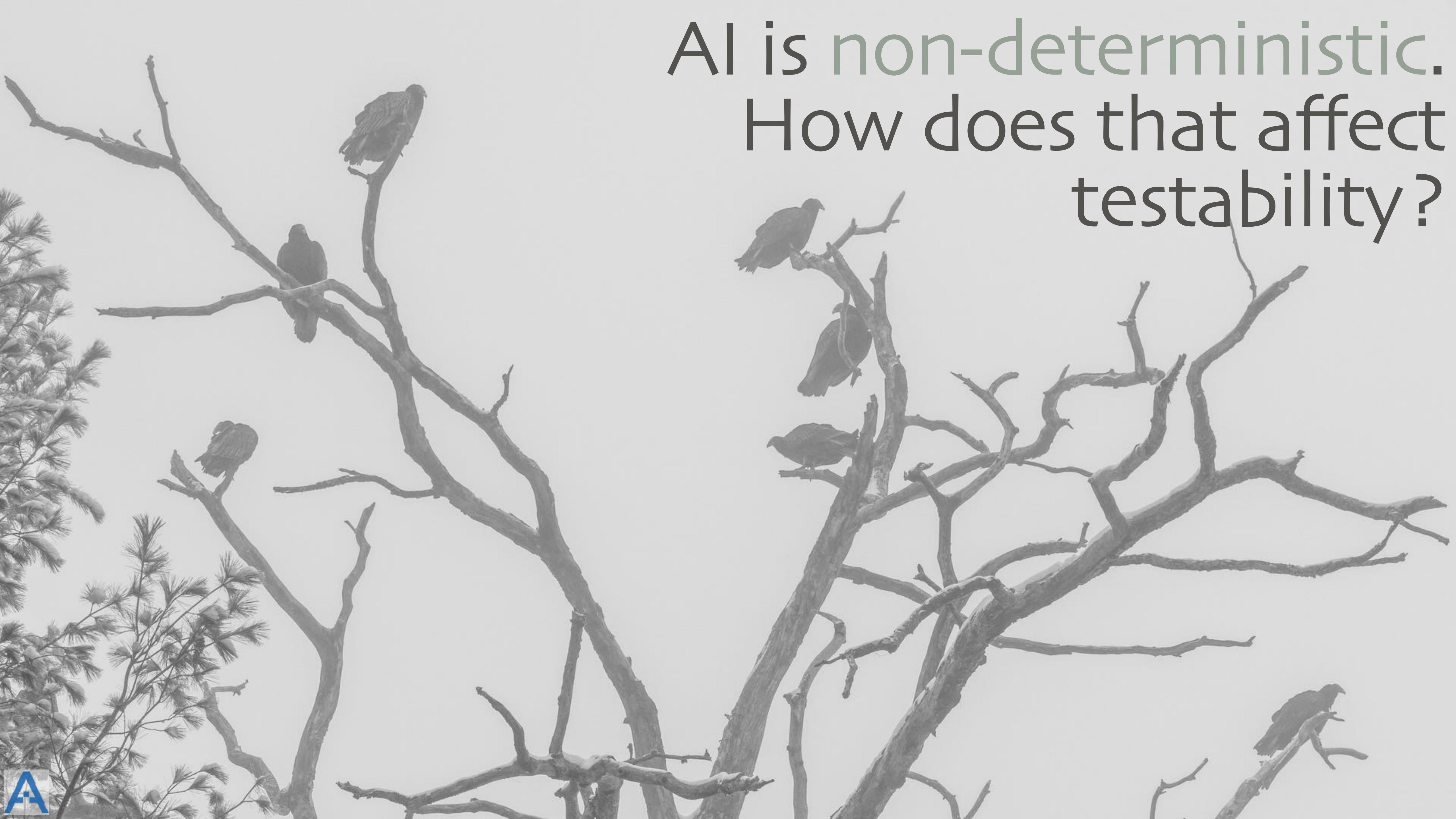
# Is this working?

- Yes for ...
  - Personal productivity acceleration.
    - ... with careful supervision by the user.
- No for ...
  - Grandiose, autonomy projects.

# What's Preventing the Bigger Projects from Working?

- Lots of things, but let's focus on one under-appreciated challenge:

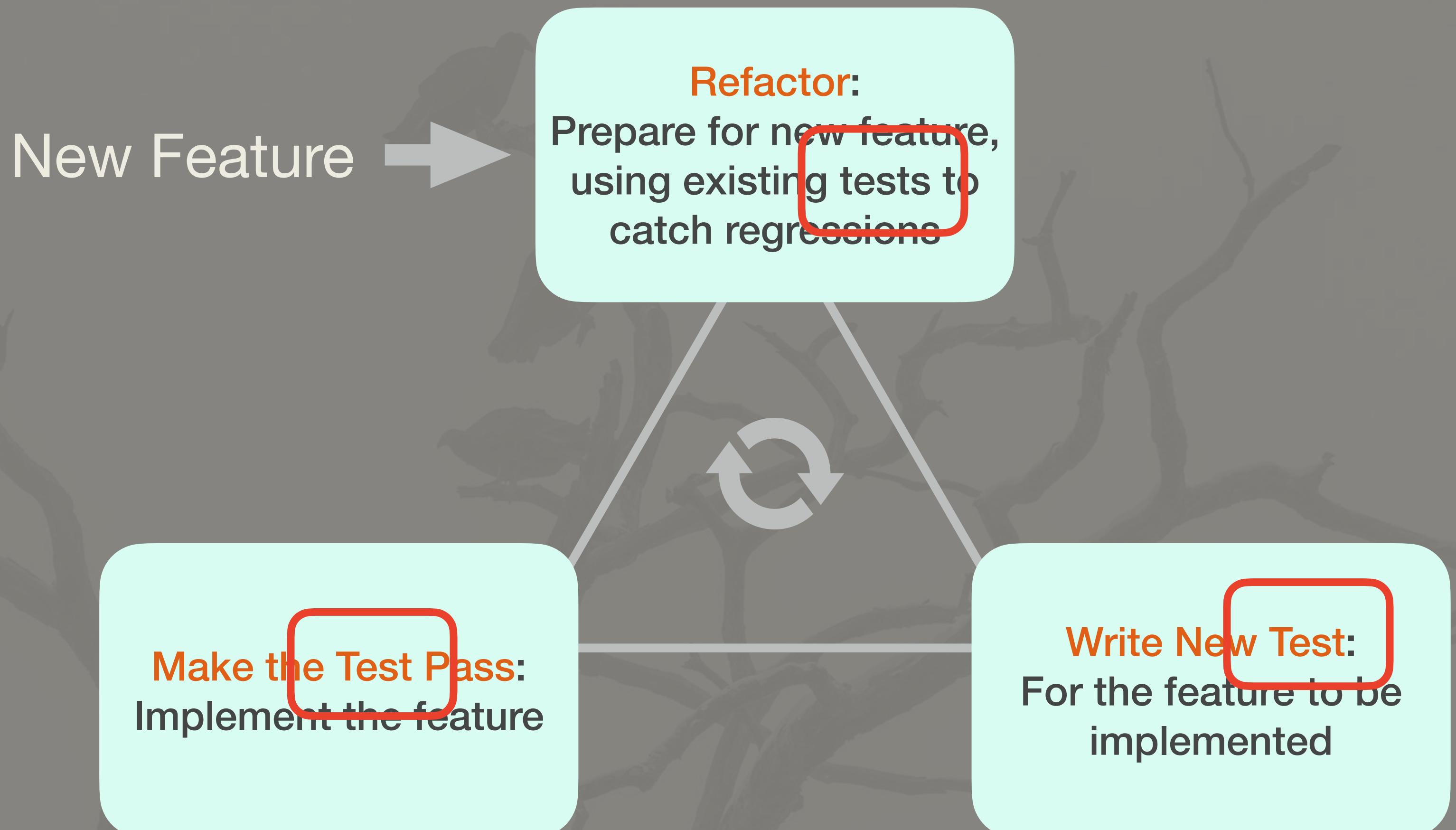
How do software developers test AI-enabled apps with the same **confidence** they have when testing traditional apps?



AI is non-deterministic.  
How does that affect  
testability?

# Remember the TDD<sup>‡</sup> loop?

Testing is the foundation of this process!



<sup>‡</sup> Test-Driven Development

# What Do Developers Expect?

Developers expect software to be deterministic<sup>‡</sup>. This helps ensure correctness, and reproducibility enables automation that catches regressions:

- The same input → the same output.
  - e.g.,  $\sin(\pi) = -1$
- The output changes? Something broke!

<sup>‡</sup> Distributed systems break this clean picture.

# What Do Developers Expect?

Developers expect determinism.  
This helps them reason about the system.

- The same result for the same inputs
- e.g., the same output for the same input
- The order of events is predictable

Put another way, the determinism makes it easier to specify the system invariants.

What should remain true before and after each step?

inistic<sup>‡</sup>.  
ucibility  
ons:

<sup>‡</sup> Distributed systems break this clean picture.

# What Do Developers Expect?

Functional Programming gave us property-based testing:

- E.g., QuickCheck, Hypothesis, ScalaCheck, ...
- Hypothesis example:

```
@given(st.integers(), nonzero_integers, st.integers(), nonzero_integers)
def test_two_non_identical_rationals_are_not_equal_to_each_other(self, numer1, denom1, numer2, denom2):
    """
    Rule: a/b == c/d iff ad == bc
    This is a better test, because it randomly generates different instances.
    However, the test has to check for the case where the two values happen to be
    equivalent!
    """
    rat1 = Rational(numer1, denom1)
    rat2 = Rational(numer2, denom2)
    if numer1*denom2 == numer2*denom1:
        self.assertEqual(rat1, rat2)
    else:
        self.assertNotEqual(rat1, rat2)
```

# What do we get with generative AI?

Generative models are stochastic<sup>‡</sup>:

- The same prompt → **different** output.
- `chatgpt("Write a poem")` → **insanity**

“Insanity is doing the same thing over and over again and expecting different results.”  
— not Einstein

<sup>‡</sup>Stochastic : described by a random probability distribution, e.g., flipping a coin, rolling dice, measuring the temperature, ...

# What do we get with generative AI?

Generative models are stochastic<sup>‡</sup>:

- The same prompt → **different** output.
  - chatgpt("Write a poem") → **insanity**
- Without **determinism**, how do you write repeatable, reliable tests for AI apps?
  - Does that new model perform **better** or **worse** than the previous model?
  - Did any **regressions** in behavior occur?

# What do we get with generative AI?

## Generative

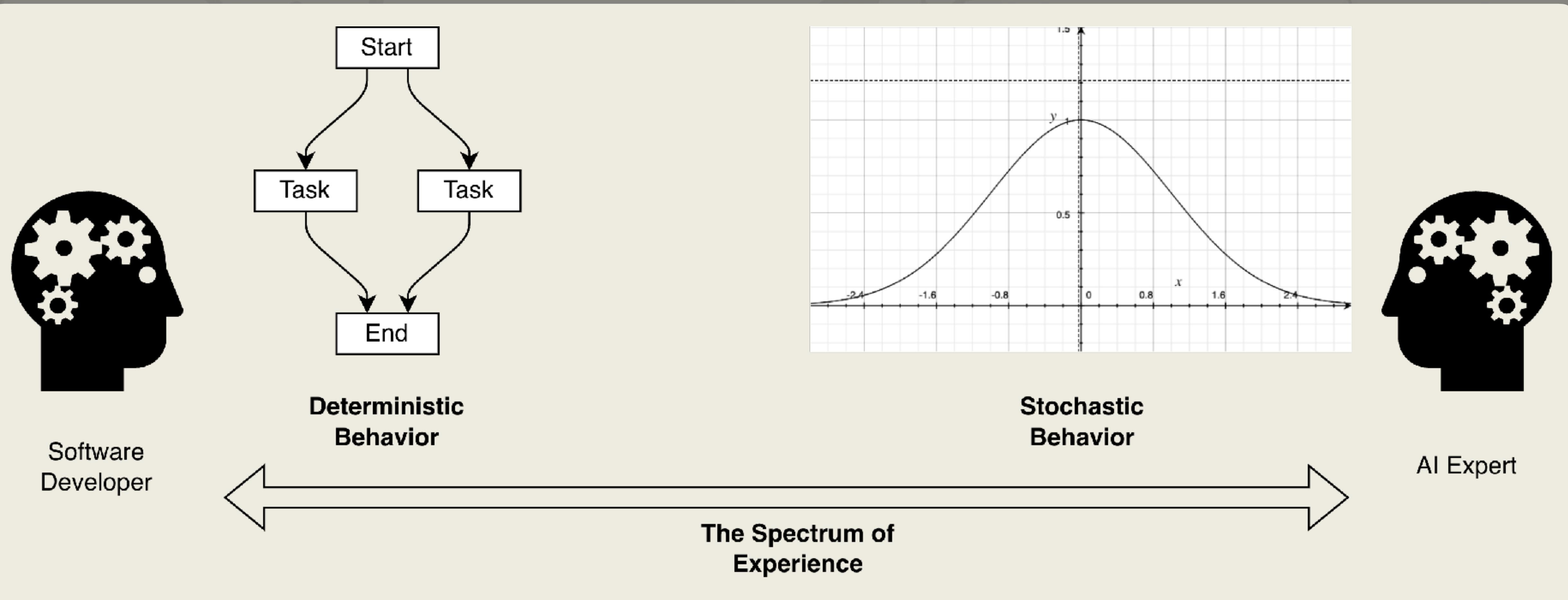
- The
- cha
- With
- repe
- Do
- tha
- Did

Put another way, the **invariants** are much less clear and therefore harder to define programmatically and enforce.

ite  
or worse

?

# But Data Scientists and AI Experts Are Accustomed to Stochasticity



# So, what should we developers do?



# So, what should we developers do?

- Learn the evaluation and benchmark tools and techniques used by data scientists, model builders, and AI safety experts.
- Adapt those tools and techniques for use in TDD and other testing methodologies.

# Specifically...

- Leverage what you already know about coupling and cohesion
- Use external tools for verification
- Scoped benchmarks - “unit benchmarks”
- Use an LLM as a judge
- Learn and use statistical techniques

# For More Details



<https://deanwampler.github.io/polyglotprogramming/papers/#Generative-AI-Should-We-Say-Goodbye-to-Deterministic-Testing>



[JOIN THIS PROJECT](#)[Github Repo](#)

# Testing Generative AI Agent Applications

(Previous Title: Achieving Confidence in Enterprise AI Agent Applications)

***I am an enterprise developer; how do I test my AI agent applications??***

*I know how to test my traditional software, which is **deterministic** (more or less...), but I don't know how to test my AI agent applications, which are uniquely **stochastic**, and therefore **nondeterministic**.*

Welcome to the **The AI Alliance** project to advance **Testing of Generative AI Agent Applications**. Whether you need to achieve the same testing *confidence* as you do with your traditional applications.

## Note:

This site isn't about using AI to generate convincing online resources about that topic. Instead, this is about testing of any kind when an application contains nondeterminism they introduce.

An AI Alliance project I lead to:

- Develop new developer testing tools and techniques adopted from data science.
- Teach developers how to use them.

## The Challenge We Face

We enterprise software developers know how to write **Repeatable** and **Automatable**.

# Outline (3/3)

## 2. Accelerating your productivity.

- Today, we speed up “old” ways of working.
- How might AI fundamentally change SW Engineering?

A close-up photograph of three red autumn leaves attached to a thin green branch. The leaves are ovate with serrated edges and prominent veins. The color is a vibrant red, with some yellow and orange hues at the base. The background is a soft, out-of-focus green.

2. Accelerating your productivity.

## 2. Accelerating your productivity.

- Using AI to speed up software tasks
- Generate unit tests
  - Or the TDD way: code from unit tests 😎
- Generate a PR to fix a small bug/change
- Gitflow processes
- ...

## 2. Accelerating your productivity.

- Using AI to speed up knowledge work tasks
  - Research a publicly-traded stock for investing
  - Research the law for a court case
  - Write a draft report
  - Improve the grammar and spelling in a doc
  - Screen resumés
  - ...

## 2. Accelerating your productivity.

- Using AI to do repetitive tasks
- Reusing AI models
- Reusing AI code
- Writing AI code
- Implementing AI
- Screen resumés
- ...

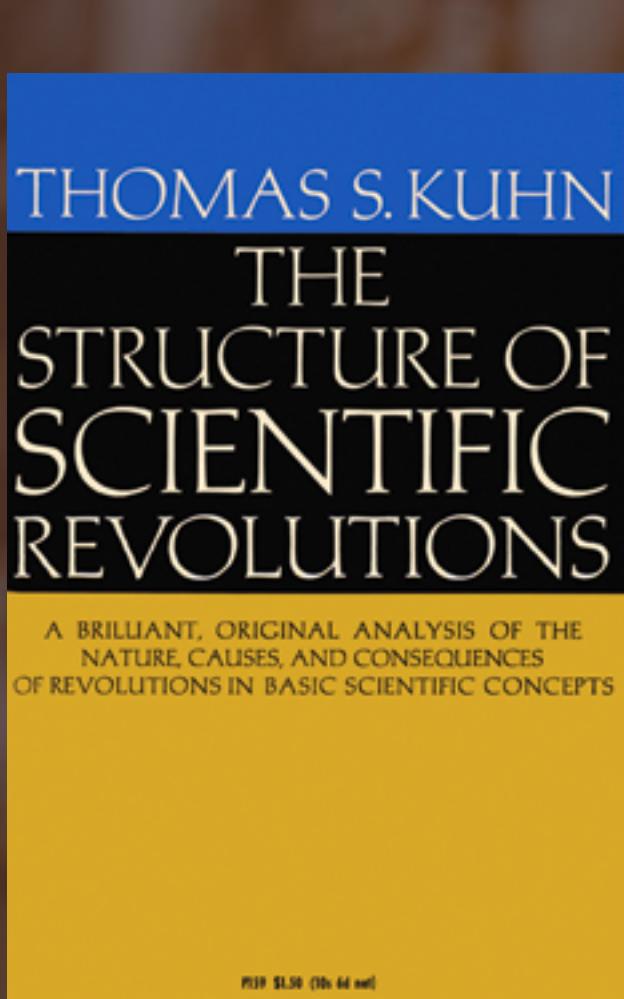
## 2. Accelerating your productivity.



How might AI fundamentally  
change SW Engineering?

# Thinking about a new perspective?

- The Structure of Scientific Revolutions
- We prefer to adapt our current theory to accommodate new data, rather than discard the theory and start over.
- But sometimes, we need to restart from first principles.



Will AI change software engineering in more fundamental ways?

# How might AI fundamentally change SW Engineering?

- Vibe Coding → Vibe Engineering
- Spec-Driven Development
- TDD → Continuous Tuning
- Source code becomes obsolete?

# Vibe Coding → Vibe Engineering

★ Vibe coding: Just prompt and if it looks good...

- The good: It allows non-coders to create code!



- The bad: It allows non-coders to create code!



Most results are  
unmaintainable messes!

# Vibe Coding → Vibe Engineering

- Can we really create quality, maintainable code just with prompts??
- Vibe engineering was coined half in jest by Simon Willison
  - ... but with the serious intent of considering what would be required for real software engineering to be doable with “vibing” only.

# Vibe Engineering

- Still requires
  - Expertise about algorithms, architecture
  - Careful review of work and fine-tuning the prompts to get precisely what we want
  - Working with AI tools is similar to working with more junior humans...
  - ...

# Vibe Engineering

- Still requires
- ...
- We have to get very good at prompt engineering to write specifications of what we want.

“AI tools amplify existing expertise.”

# Spec-Driven Development

- (A.k.a. Specification-Driven Development)
- An approach to principled, effective prompt construction and how to use them.
- Uses separate “phases”, each with its own prompt and corresponding tools.

- Phases:

# GitHub's SpeckKit

- **Specify**: Generate the specification (i.e., requirements)
- **Plan**: Add more technical details and generate a high-level plan for the project.
- **Tasks**: Decompose the plan into fine-grained tasks.
- **Implement**: Generate the app (with tests, ... ) using the plan.

- Phases:

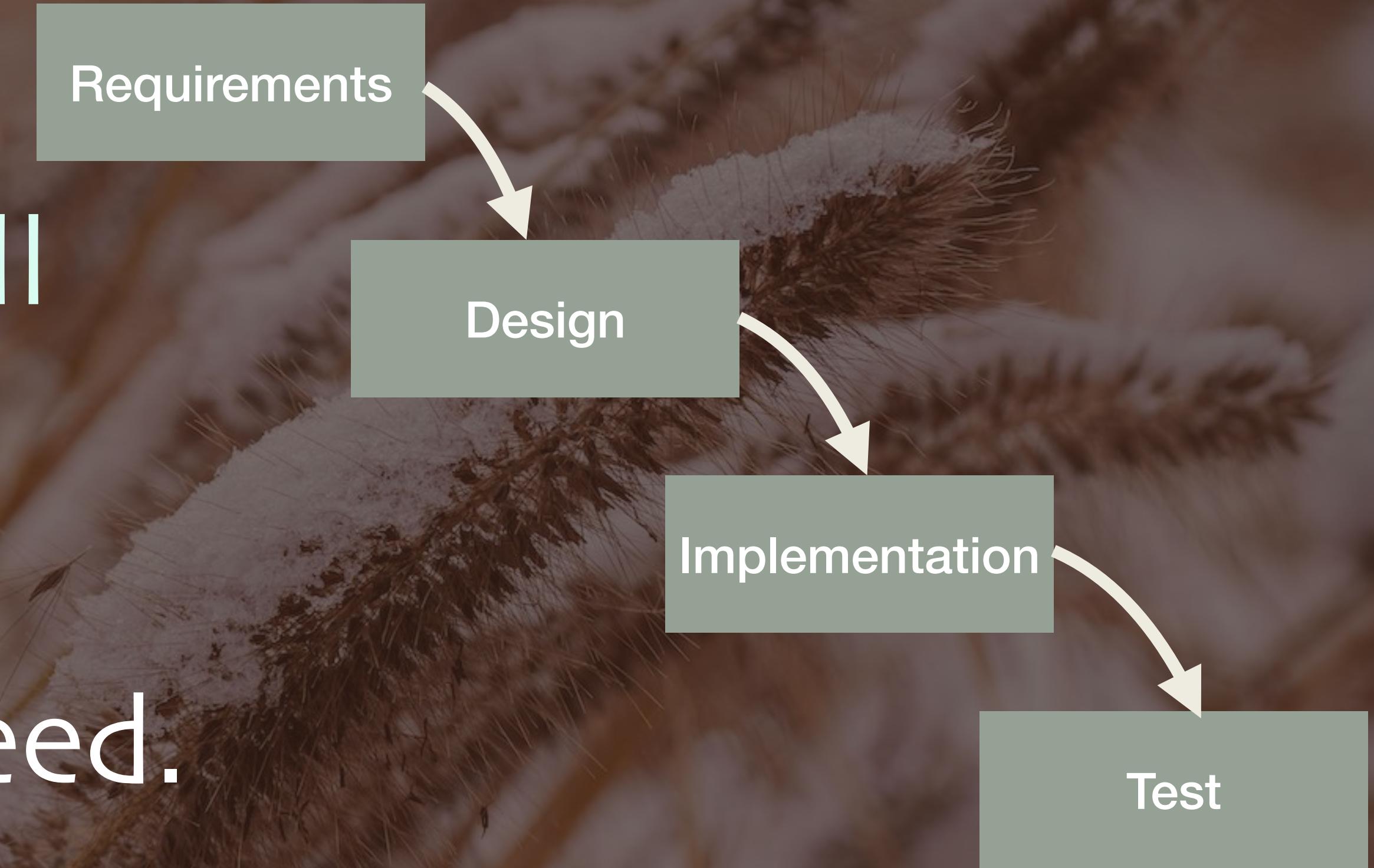
## GitHub's SpecKit

- Spec: “Instead of coding first and writing docs later, in SDD, you start with a spec. This is a contract for how your code should behave and becomes the source of truth your tools and AI agents use to generate, test, and validate code. The result is less guesswork, fewer surprises, and higher-quality code.”
- Plan: — from the blog post (minor edits...)

- Implement using the plan.

# Criticisms (1/2)

- Is this just the Waterfall Process reborn?
- It has to be done incrementally to succeed.
- TDD has a refactor step. That needs to be incorporated in the phases.



# Criticisms (2/2)

- Will learning prompt engineering be harder than just using the skills we already possess?
  - Maybe, but the productivity boost might make mastering prompt engineering worth it.
- Open-ended English is the worst possible API.
  - Carefully engineered system and user prompt templates will be essential.

# TDD → Continuous Tuning

- ★ What if we switch from **testing** for desired behavior to **tuning** for desired behavior?
  - We already tune models to improve domain-specific knowledge, chatbot behavior, etc.
  - Today: it's only done during model creation.
  - Tomorrow: continuously tune incrementally.

# TDD → Continuous Tuning

Changes to the TDD cycle (for model behaviors):

New Feature →

We still need automated tests generated and executed.

**Refactor:**  
Prepare for new feature, using existing tests to catch regressions

Use spec-driven development!

Keep tuning until the benchmark passes

**Make the Spec “Pass”:**  
“Vibe engineer” the new feature

**Write New Executable Specification:**  
For the feature to be implemented

Write new unit benchmark, an *executable specification*



# Source code becomes obsolete?

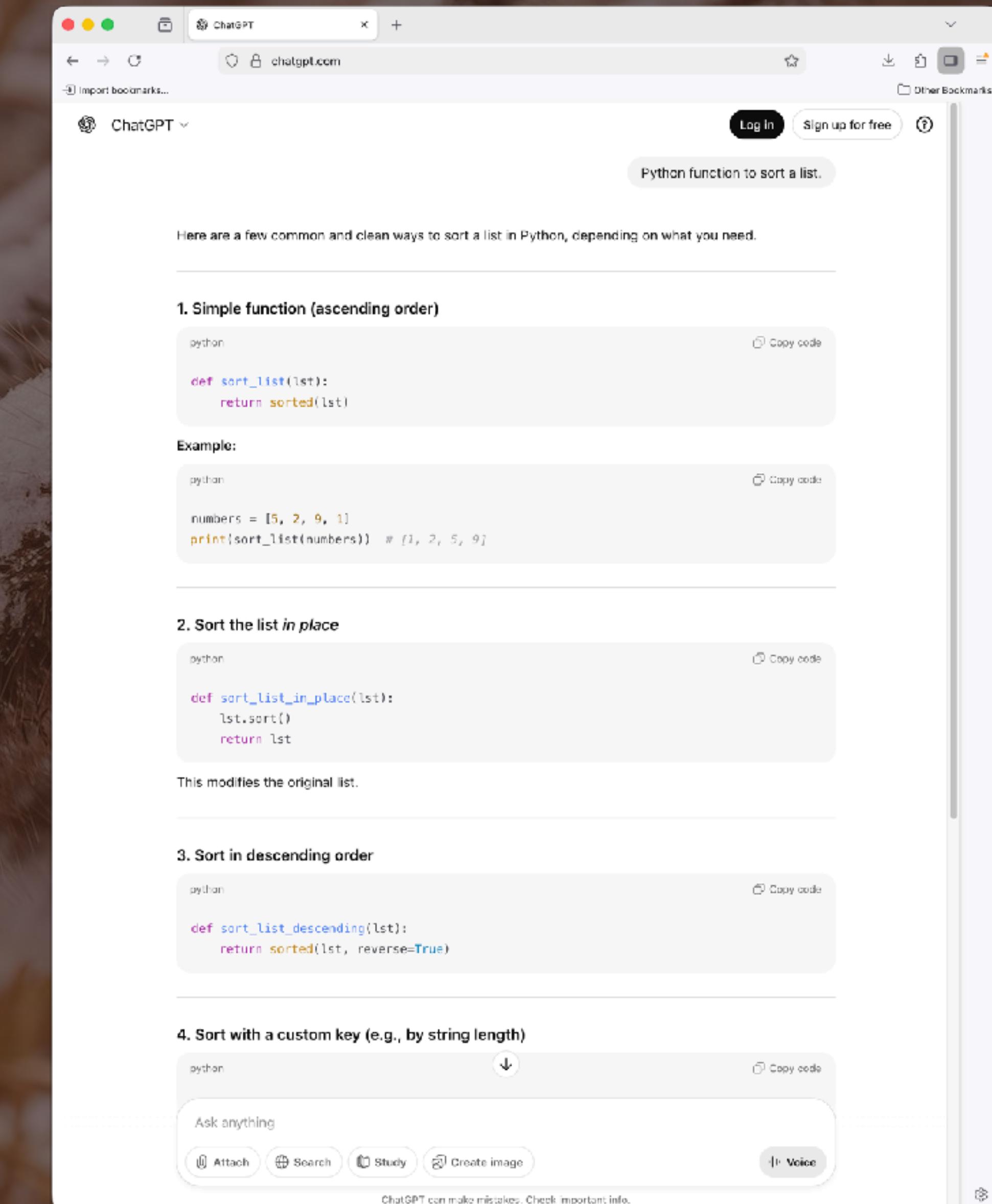
★ After ~70+ years, we still use source code!

- Will AI make it obsolete?
- We still need some sort of “representation” of execution constructs.
- Code still needs to be human readable and debuggable.

```
1 OPEN INPUT sales, OUTPUT report-out
2 INITIATE sales-report
3
4 PERFORM UNTIL 1 <> 1
5   READ sales
6   AT END
7     EXIT PERFORM
8   END-READ
9
10  VALIDATE sales-record
11  IF valid-record
12    GENERATE sales-on-day
13  ELSE
14    GENERATE invalid-sales
15  END-IF
16 END-PERFORM
17
18 TERMINATE sales-report
19 CLOSE sales, report-out
20 .
```

# How might this work?

- AI is good at knowing common (best?) practices...
- E.g., the most common way to sort lists seen in the training data is probably the best way (or at least good enough).



The screenshot shows a web browser window for ChatGPT. The URL is chatgpt.com. The page displays several examples of Python code for sorting lists. At the top, it says "Python function to sort a list." Below that, it says "Here are a few common and clean ways to sort a list in Python, depending on what you need." The examples are numbered 1 through 4:

- 1. Simple function (ascending order)**

```
python
def sort_list(lst):
    return sorted(lst)
```

Example:

```
python
numbers = [5, 2, 9, 1]
print(sort_list(numbers)) # [1, 2, 5, 9]
```
- 2. Sort the list *in place***

```
python
def sort_list_in_place(lst):
    lst.sort()
    return lst
```

This modifies the original list.
- 3. Sort in descending order**

```
python
def sort_list_descending(lst):
    return sorted(lst, reverse=True)
```
- 4. Sort with a custom key (e.g., by string length)**

```
python
```

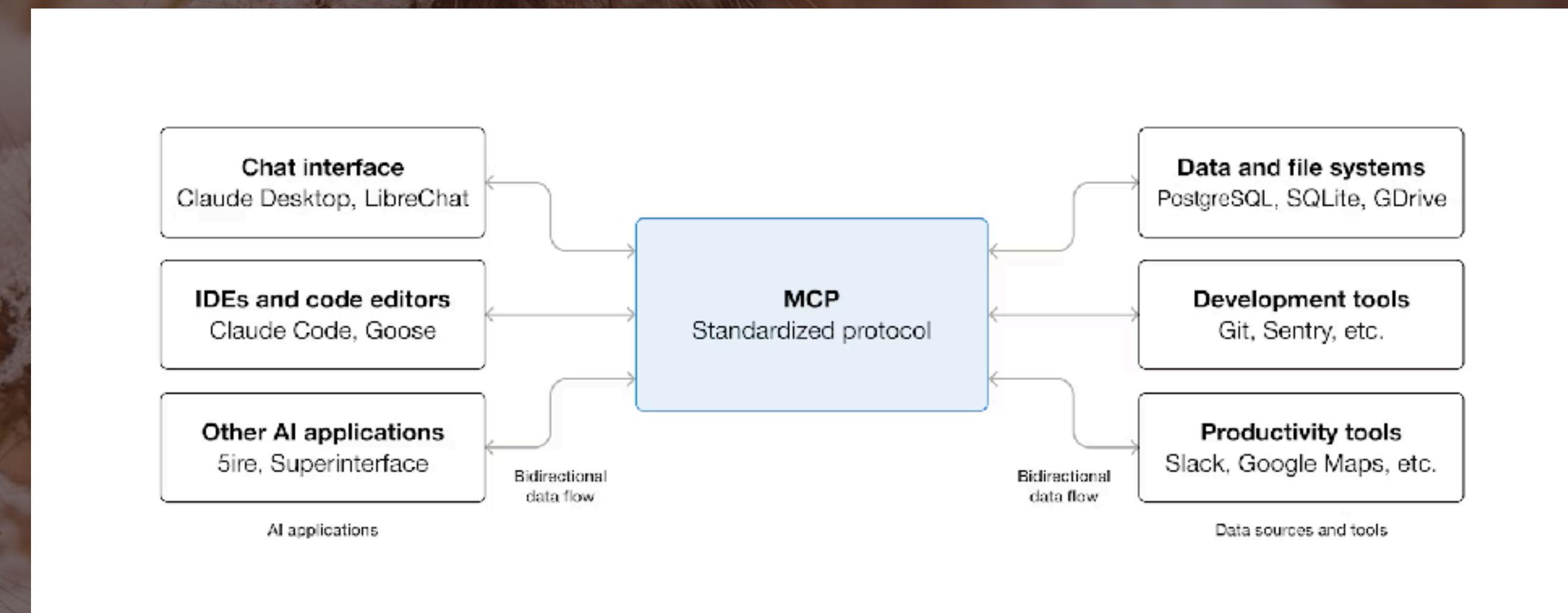
Ask anything

Attach Search Study Create image Voice

ChatGPT can make mistakes. Check important info.

# How might this work?

- AI is good at “gluing” things together.
- E.g., Model Context Protocol (MCP)
- Discover APIs and figure out to invoke them automatically.



# Source code becomes obsolete?

- So, Vibe Engineering becomes
  - Best practice component generation + MCP-based service invocation + glue that integrates them together?
  - Today's source code becomes assembly.
    - It's there, but few people need to understand or manipulate it.

# Questions?

[aialliance.org](http://aialliance.org)

[the-ai-alliance.github.io/](http://the-ai-alliance.github.io/)

[dwampler@thealliance.ai](mailto:dwampler@thealliance.ai)

Mastodon and Bluesky: @deanwampler

[aialliance.org](http://aialliance.org)

[deanwampler.com/talks](http://deanwampler.com/talks)

