# How Will AI Change Software?

Dean Wampler, Ph.D.
The AI Alliance
dwampler@thealliance.ai
January 14, 2026

aialliance.org

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)

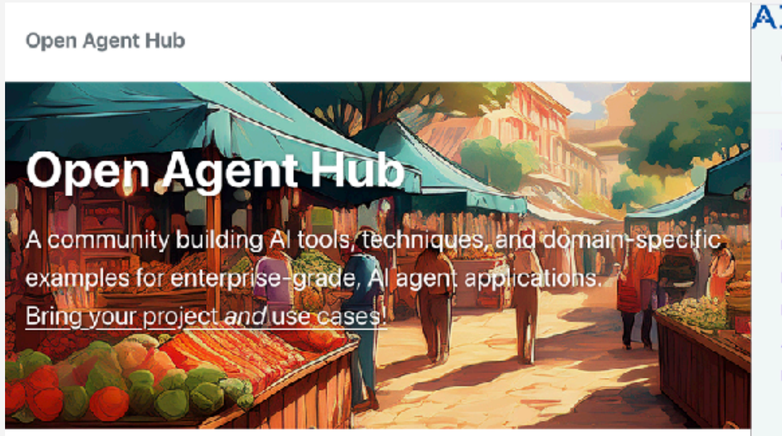
2. Accelerating your productivity.
   - Today, we speed up "old" ways of working.
   - How might AI fundamentally change SW Engineering?

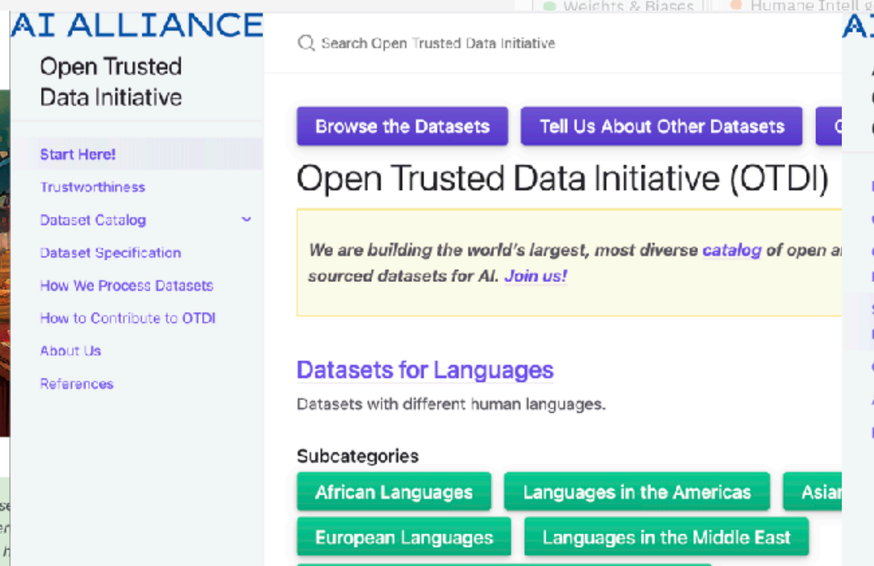
# AI ALLIANCE

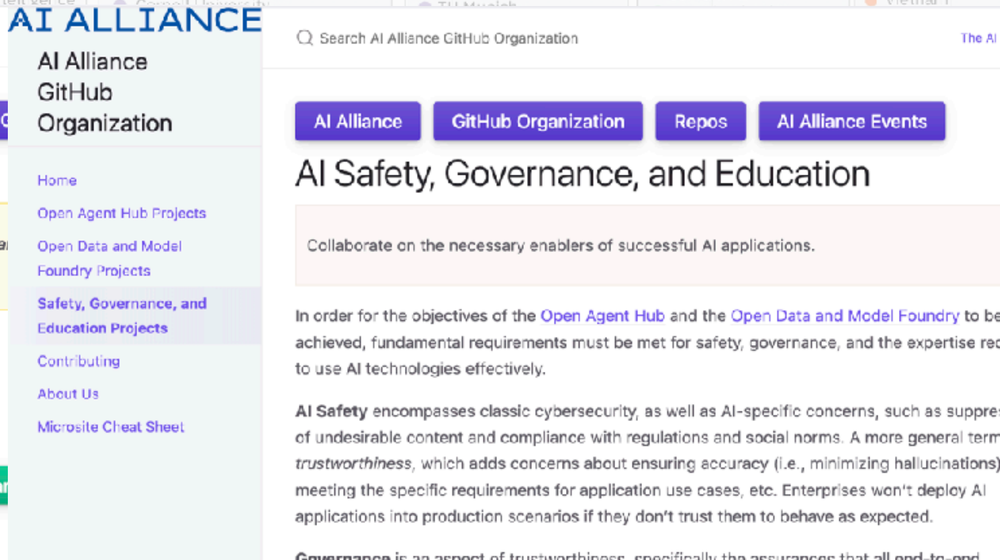
195+ organizations in 25+ countries accelerating open innovation and adoption of AI


These project areas:
the-ai-alliance.github.io/

1. Agents
2. Data & Models
3. Safety & Governance



# 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.

# Is this working?

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

Search

FORTUNE

### 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

Add us on

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 Decou
is Finally Growin
JAN 12, 2026


Biohacking: Thre
Solutions
DEC 05, 2025

Boosting Social
Role of AI in Dat
NOV 26, 2025

Fellou AI Browse
Browsing with A

"…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

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


# 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.

https://ucb-mast.notion.site/

# Is this working?

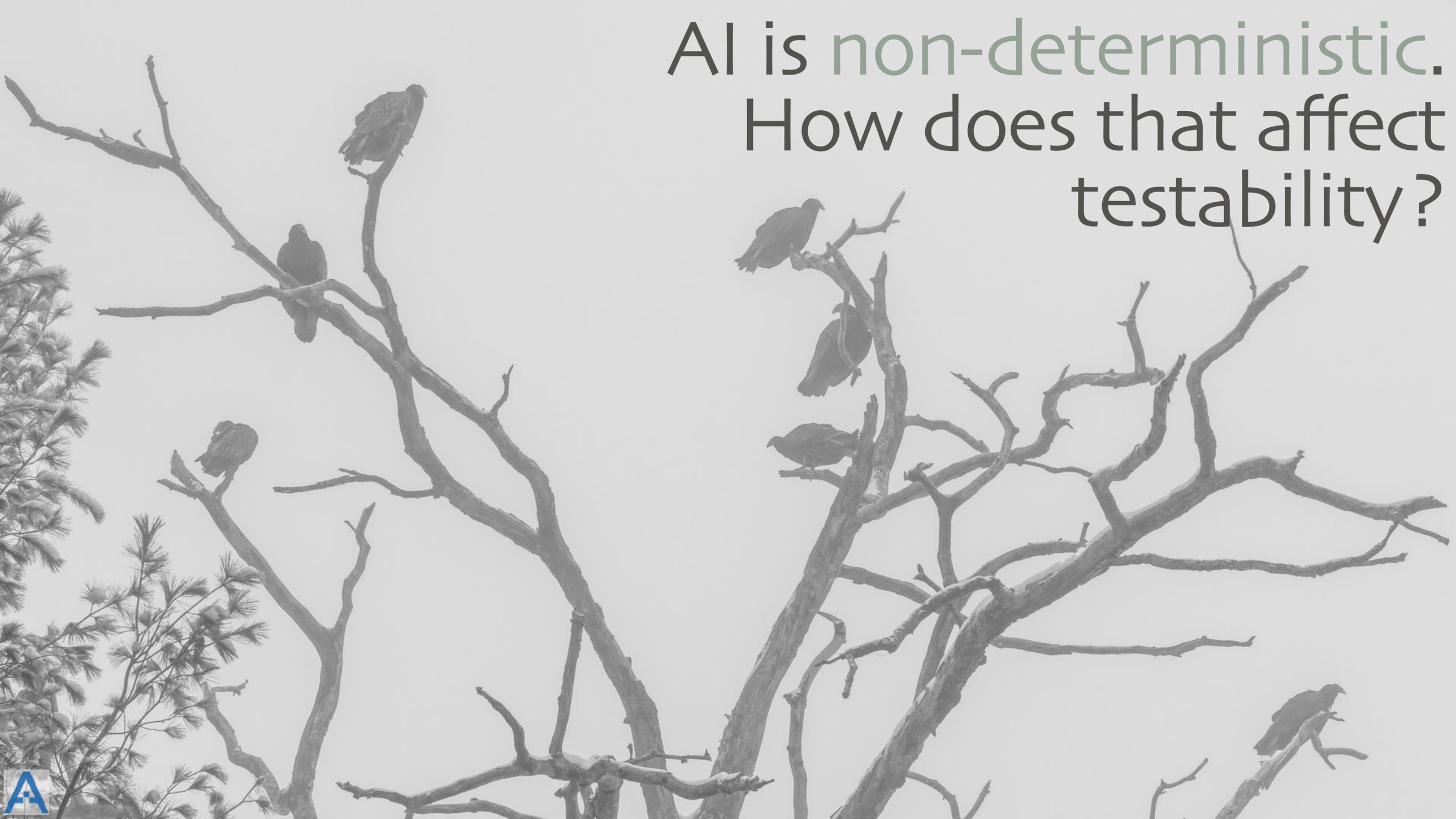
- Yes for …
  - Personal productivity acceleration.
    - … with careful supervision by the user.
- No for …
  - Grandiose, autonomy projects.

# What's Preventing the Bigger Projects from Working?

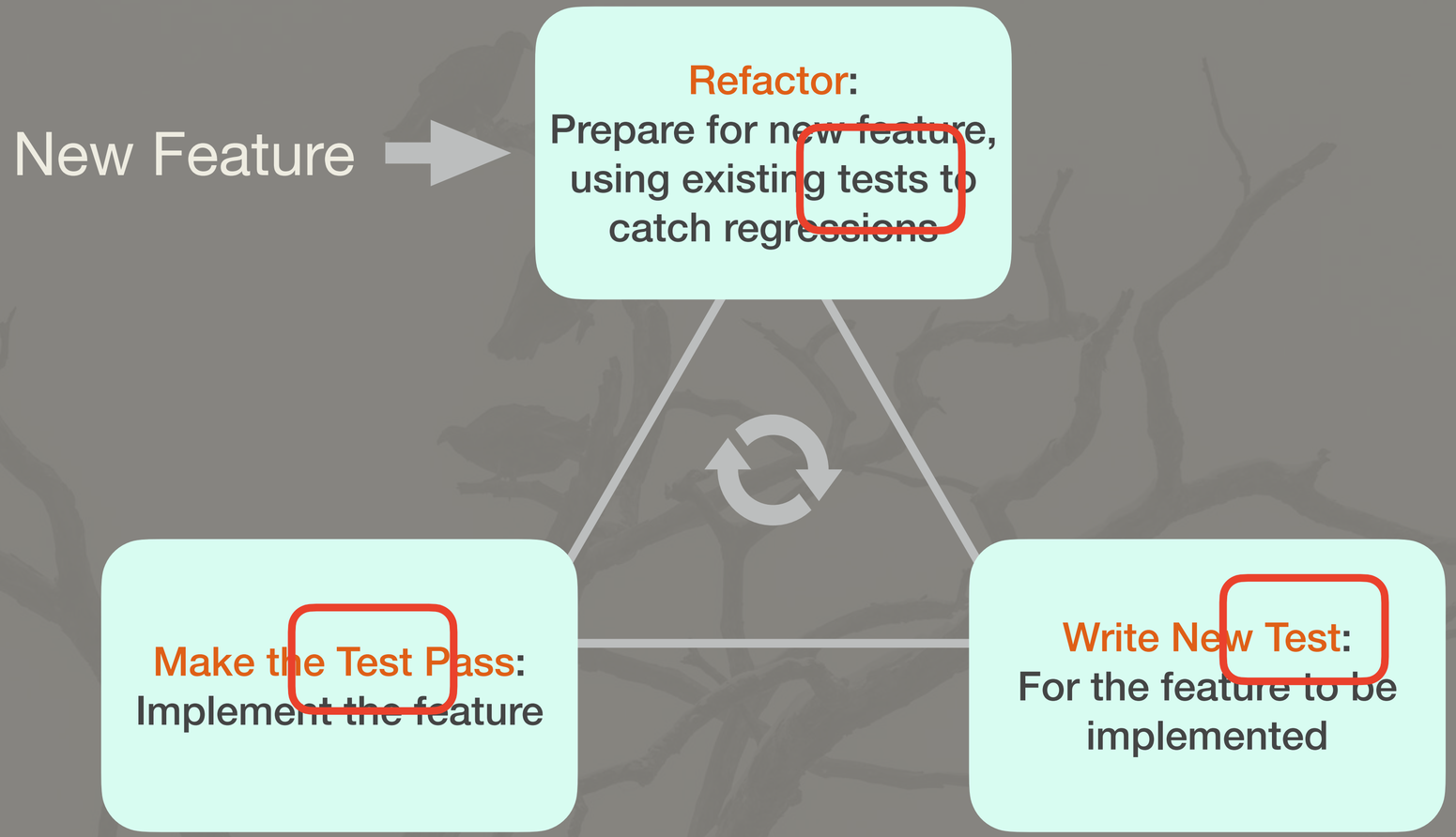
- Lots of things, but let's focus on one under-appreciated challenge:

> 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‡ loop?




Testing is the foundation of this process!

‡ Test-Driven Development

# What Do Developers Expect?

Developers expect software to be deterministic‡.
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!

‡ Distributed systems break this clean picture.

# What Do Developers Expect?

Develope inistic‡.
This help ucibility
enables a ons:

- The s
  - e.g.,
- The o !

Put another way, the determinism makes it easier to specify the system invariants. What should remain true before and after each step?

‡ 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)
```

https://github.com/deanwampler/tdd-hypothesis-example

# What do we get with generative AI?

Generative models are stochastic‡:

- 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

‡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[‡]:

- 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?

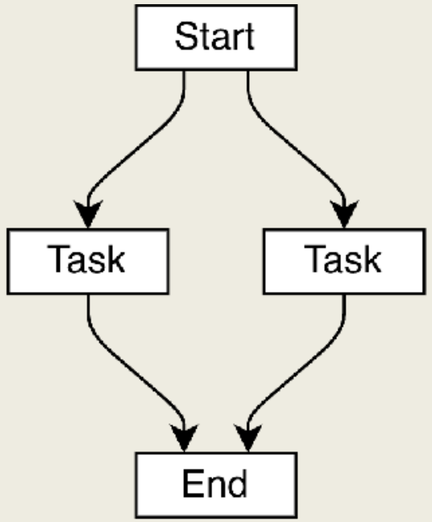
Generat

- The
  - cha
- With ite repe
  - Do or worse tha
  - Dic ?

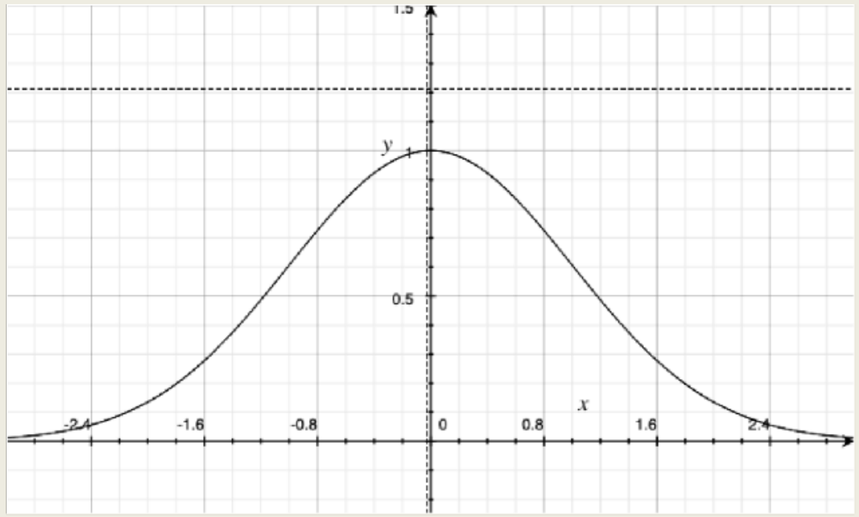
Put another way, the invariants are much less clear and therefore harder to define programmatically and enforce.

# But Data Scientists and AI Experts Are Accustomed to Stochasticity

Software Developer




**Deterministic Behavior**




**Stochastic Behavior**

AI Expert

**The Spectrum of Experience**

# 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

AI ALLIANCE

Testing Generative AI Agent Applications

Search Testing Generative AI Agent Applications

The AI Alliance

https://the-ai-alliance.github.io/ai-application-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 advar
**Testing of Generative AI Agent Applications**. V
you need to achieve the same testing *confidence*
your traditional applications.

**Note:**

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

## The Challenge We Face

We enterprise software developers know how to write Repeatable and Automatable

An AI Alliance project I lead to:

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

# Outline (3/3)

2. Accelerating your productivity.
   - Today, we speed up "old" ways of working.
   - How might AI fundamentally change SW Engineering?


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.

- Usi
  - Re
  - Re
  - W
  - Im
  - Screen resumés
  - …

These tasks predate AI; we just use AI to accelerate them.

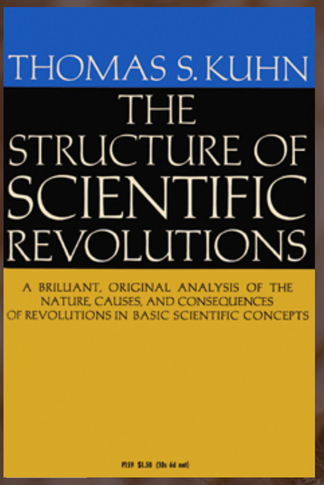
How will AI fundamentally change tasks themselves?

# 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.

https://simonwillison.net/2025/Oct/7/vibe-engineering/

# 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…
  - …

https://simonwillison.net/2025/Oct/7/vibe-engineering/

# 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."

https://simonwillison.net/2025/Oct/7/vibe-engineering/

# 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.

https://the-ai-alliance.github.io/ai-application-testing/advanced-techniques/sdd/

# GitHub's SpecKit

- Phases:
  - 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.

https://github.blog/ai-and-ml/generative-ai/spec-driven-development-with-ai-get-started-with-a-new-open-source-toolkit/

# GitHub's SpecKit

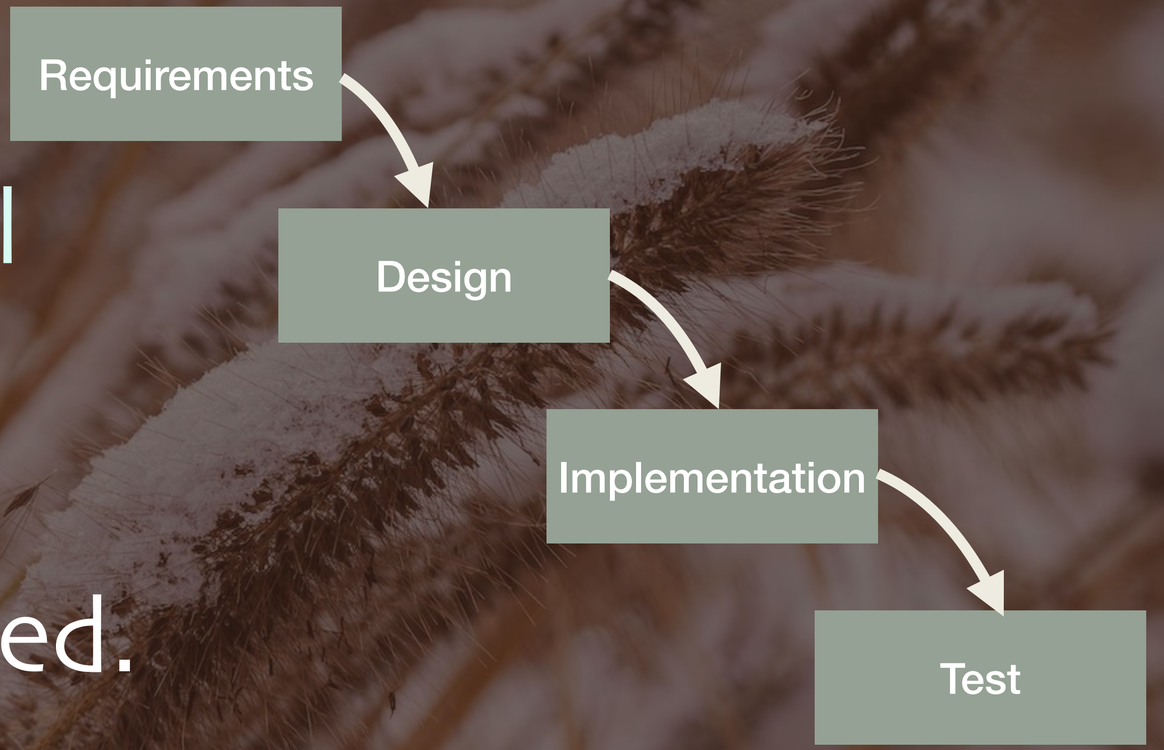
- Phases:
  - Sp
    re
  - Pl ate a
    hig
  - Ta ed
    ta
  - Im ..)
    using the plan.

"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."

— from the blog post (minor edits…)

https://github.blog/ai-and-ml/generative-ai/spec-driven-development-with-ai-get-started-with-a-new-open-source-toolkit/

# 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)

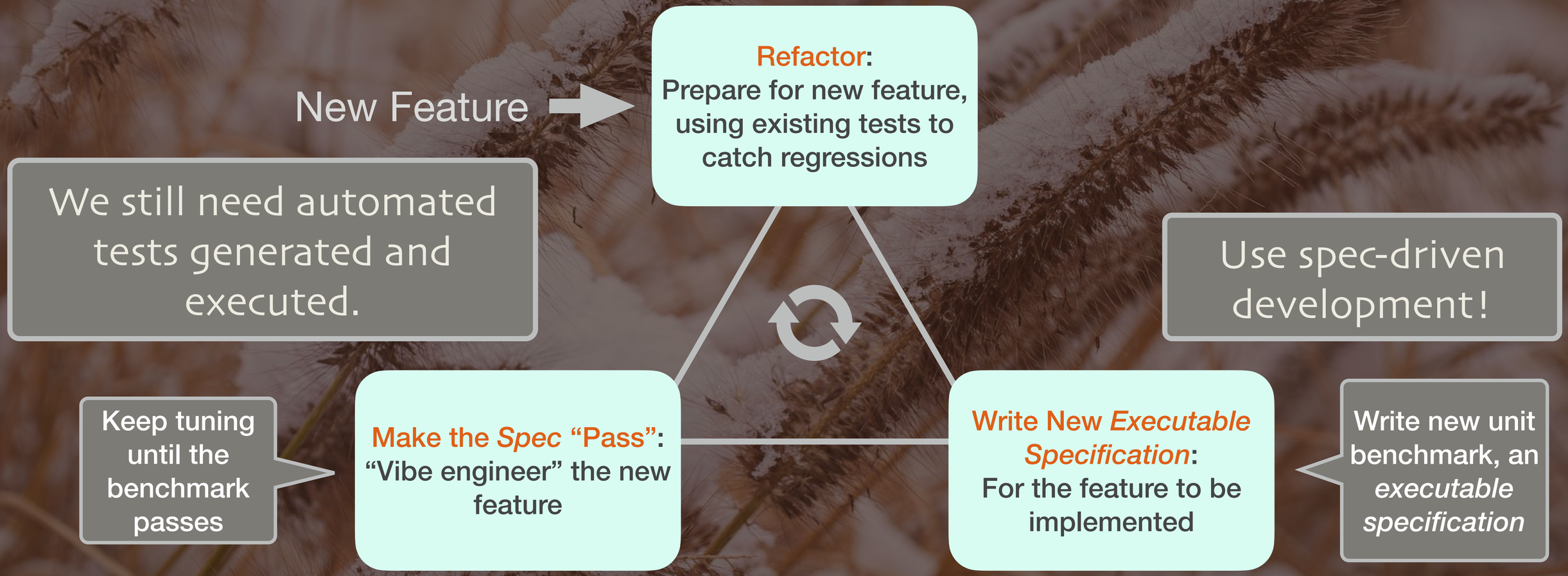
- Will learning prompt engineering be harder than just using the skills we already possess?
  - Maybe, but the productivity boost might make mastering prompt engineering worth it.
- Open-ended English is the worst possible API.
  - Carefully engineered system and user prompt templates will be essential.

# 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):



# Source code becomes obsolete?

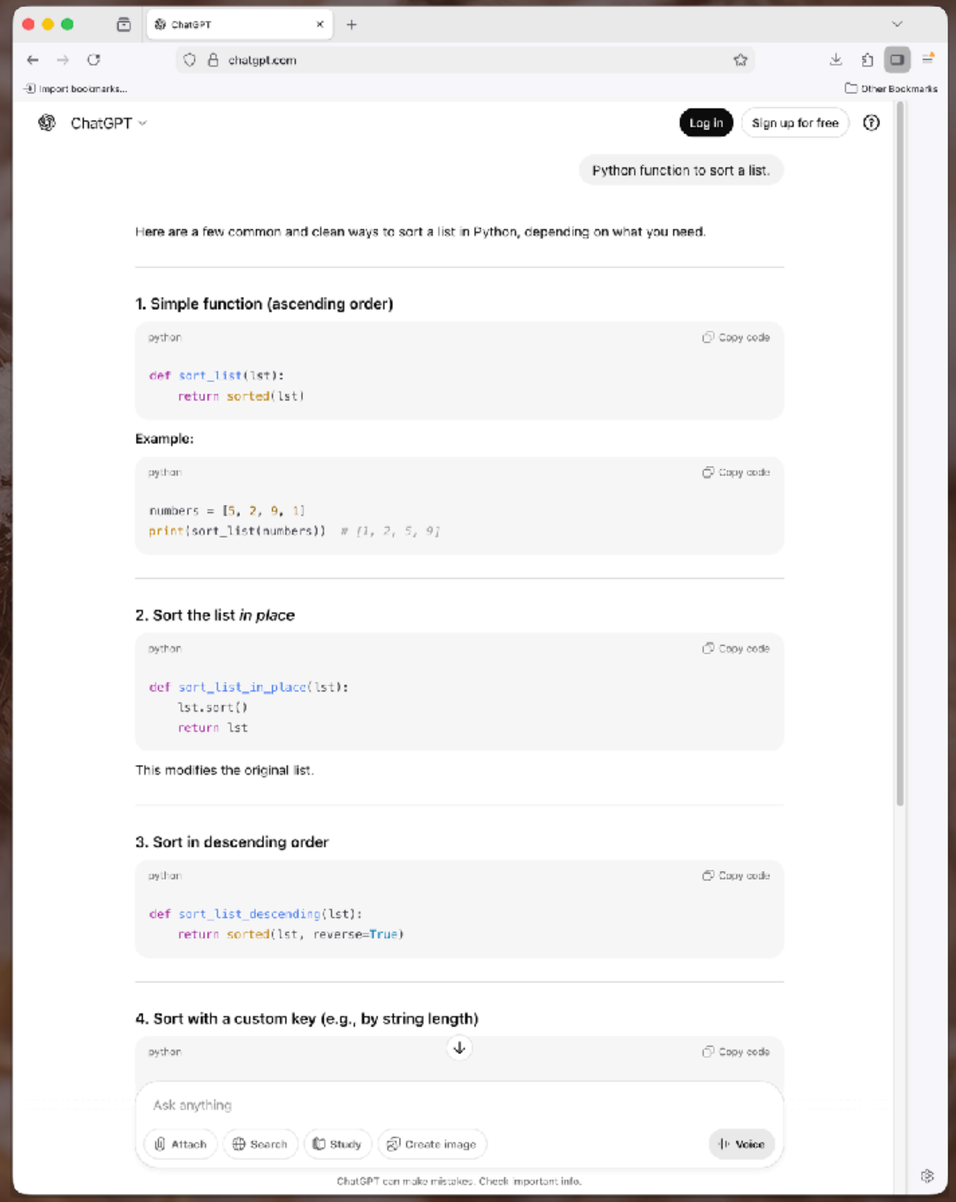
- ★ After ~70+ years, we still use source code!
  - Will AI make it obsolete?
    - We still need some sort of "representation" of execution constructs.
    - Code still needs to be human readable and debuggable.

```
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 .
```

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

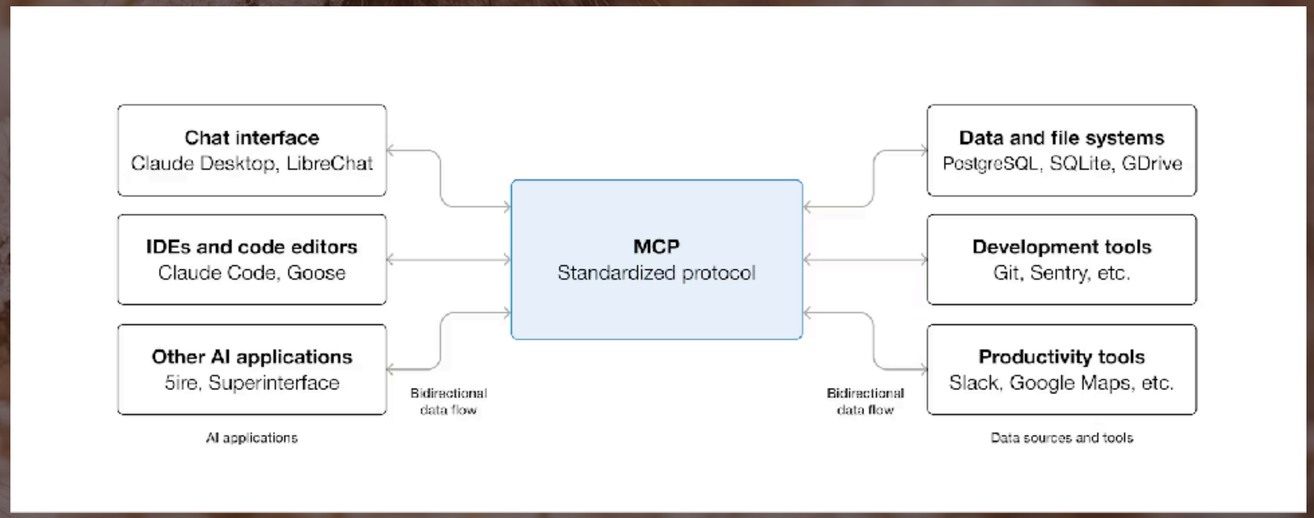
# How might this work?

- 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).



# 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.




https://modelcontextprotocol.io/docs/getting-started/intro

# 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
the-ai-alliance.github.io/

dwampler@thealliance.ai
Mastodon and Bluesky: @deanwampler

aialliance.org


deanwampler.com/talks