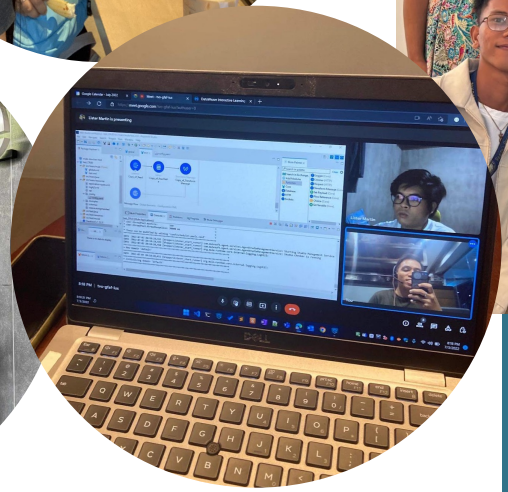
Human in the Lead: Staying in charge as AI reshapes Software Development

Aj De Guzman
Sr. Integration Developer, J4RVIS

August 2026

From Campus to Career



whoami

- Sr. Integration Developer (MuleSoft)
- Stack: Java, MuleSoft, Azure/AWS, DataWeave

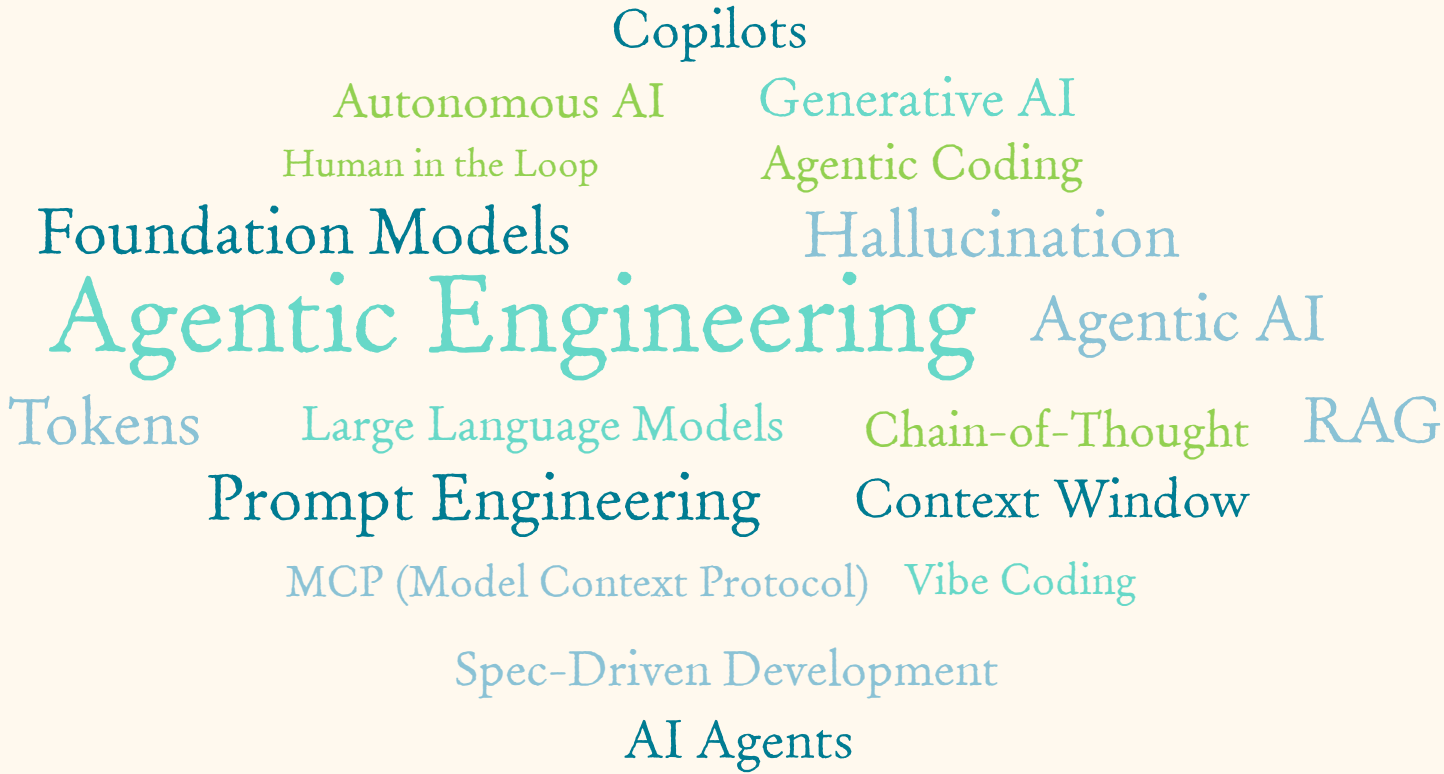
- **J4RVIS** - AI-native and systems integrator consultancy in Australia: financial services, media and entertainment, energy and utilities.

Apps/Systems Integration

Food Delivery App - ordering from a local restaurant on a smartphone.

- Customer app
- Restaurant POS
- Payment gateway
- Courier dispatch

Human in the Lead: Staying in charge as AI reshapes Software Development


Copilots
Autonomous AI
Generative AI
Human in the Loop
Agentic Coding
Foundation Models
Hallucination
Agentic Engineering
Agentic AI
Tokens
Large Language Models
Chain-of-Thought
RAG
Prompt Engineering
Context Window
MCP (Model Context Protocol)
Vibe Coding
Spec-Driven Development
AI Agents

The Landscape

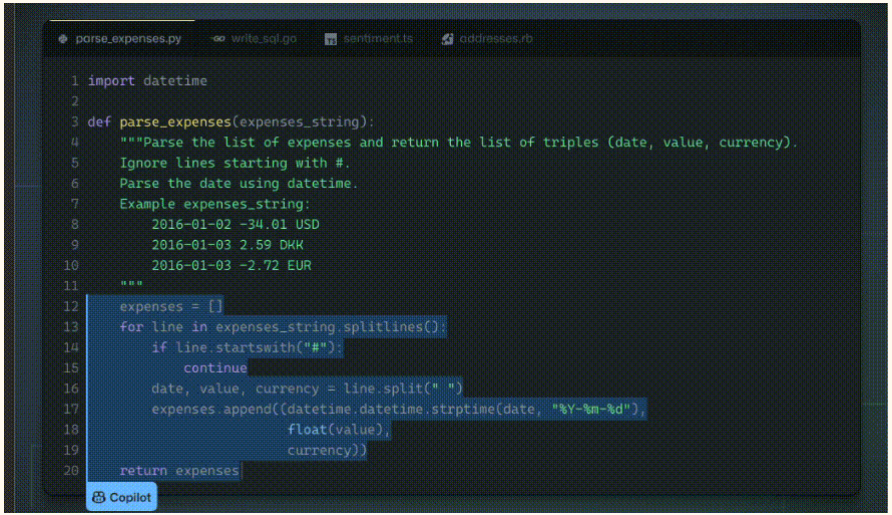
The three AI personas

1. **The Skeptic** - has seen hype cycles before and is waiting for evidence.
2. **The Overwhelmed** - believes it matters, does not know where to start.
3. **The Enthusiast** - already using it daily, sometimes uncritically.

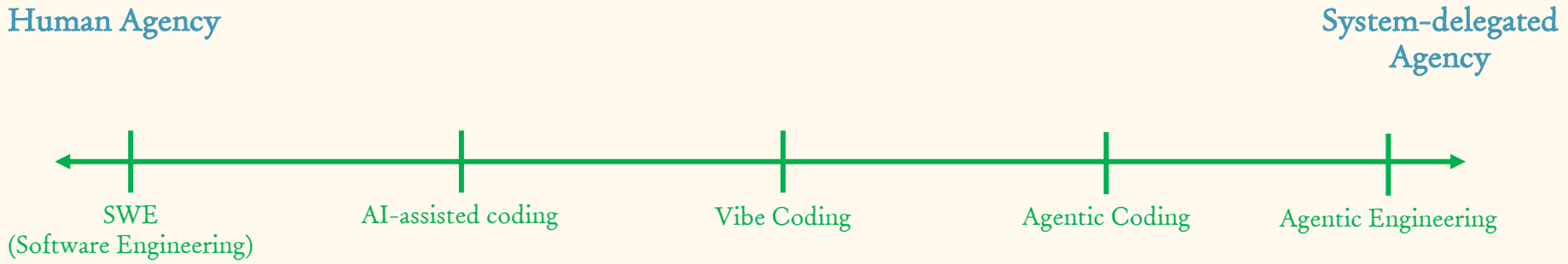
AI Capability Spectrum

Human does the work,
AI fills the gap

AI does the work,
human guides and verifies



Agency Spectrum



Source: daily.dev - IBM Technology

Thesis of this talk:

AI is not a replacement. It is a multiplier.

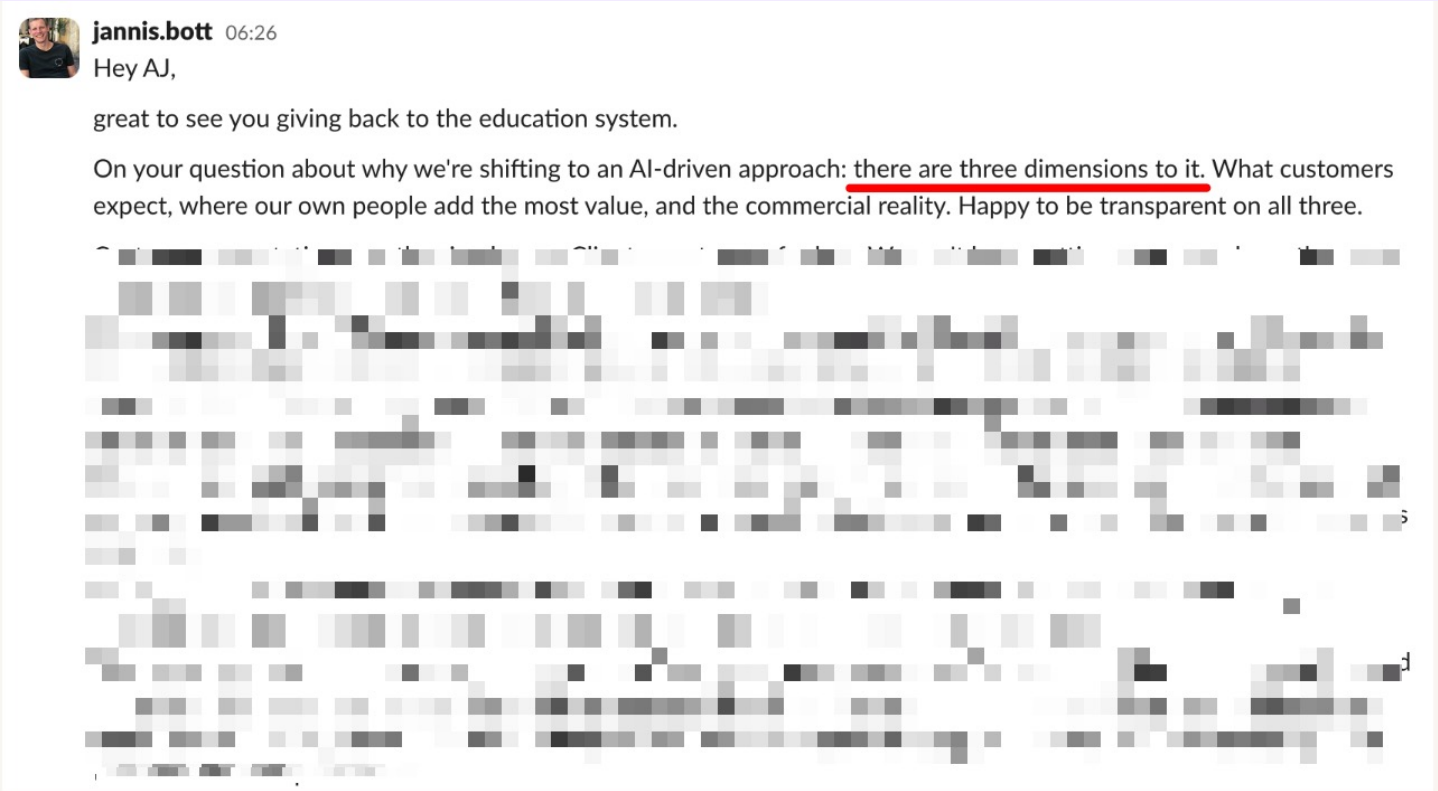
A multiplier works on what is already there. Multiply zero and you still get zero.

Our Company's Shift

Where the company started

- Requirements gathered through meetings and documents
- Developers write every line of code manually (and some are copied from Stack Overflow, different forums and threads.)
- Code reviews are fully human, line-by-line
- Testing - Unit tests, integration tests, are written by hand
- Debugging means reading stack traces

Why the shift to AI-driven approach?



jannis.bott 06:26

Hey AJ,

great to see you giving back to the education system.

On your question about why we're shifting to an AI-driven approach: there are three dimensions to it. What customers expect, where our own people add the most value, and the commercial reality. Happy to be transparent on all three.

Why the shift to AI-driven approach?

- **Customer Expectations** - Clients demand more value for less cost.
- **Internal Benefits** - AI improves output consistency and makes repetitive work cheaper to execute
- **Commercial Reality** - Budget pressure from clients creates competitive pressure.

Phase 1: AI-Assisted Development

What we adopted

- Established the AI Acceptable Use Policy (AUP)
- AI code assistants (Copilot, Claude) for code generation and completion
  - Primary AI tool: Claude
  - Company have required taking the Anthropic training before providing the license.
- AI-powered code review suggestions
- Automated test generation from existing code

AI Stack

General Purpose

Claude (Anthropic) - Chat, Design, Cowork, Tag, Platform, Agent SDK

Coding Assistant

Claude Code GitHub Copilot

Communication | Chat

slack Slackbot – Personal Assistant

Design Work

Figma

Diagramming

Lucidspark Lucidchart

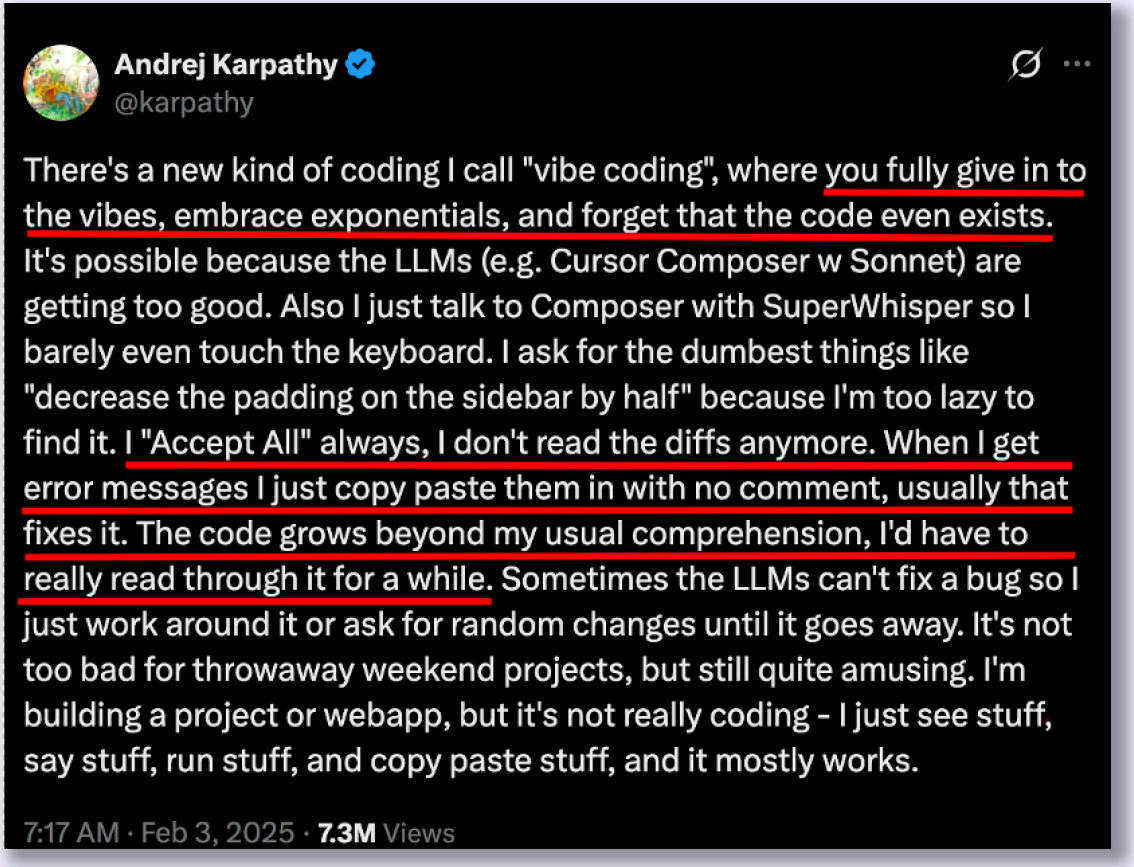
Voice AI

IIElevenLabs

Vibe Coding

Vibe coding is when you just tell an AI what you want, take whatever code it spits out, and don't really read it. Something breaks? Paste the error back, say "fix it," run it again. Repeat until it works.

Vibe Coding

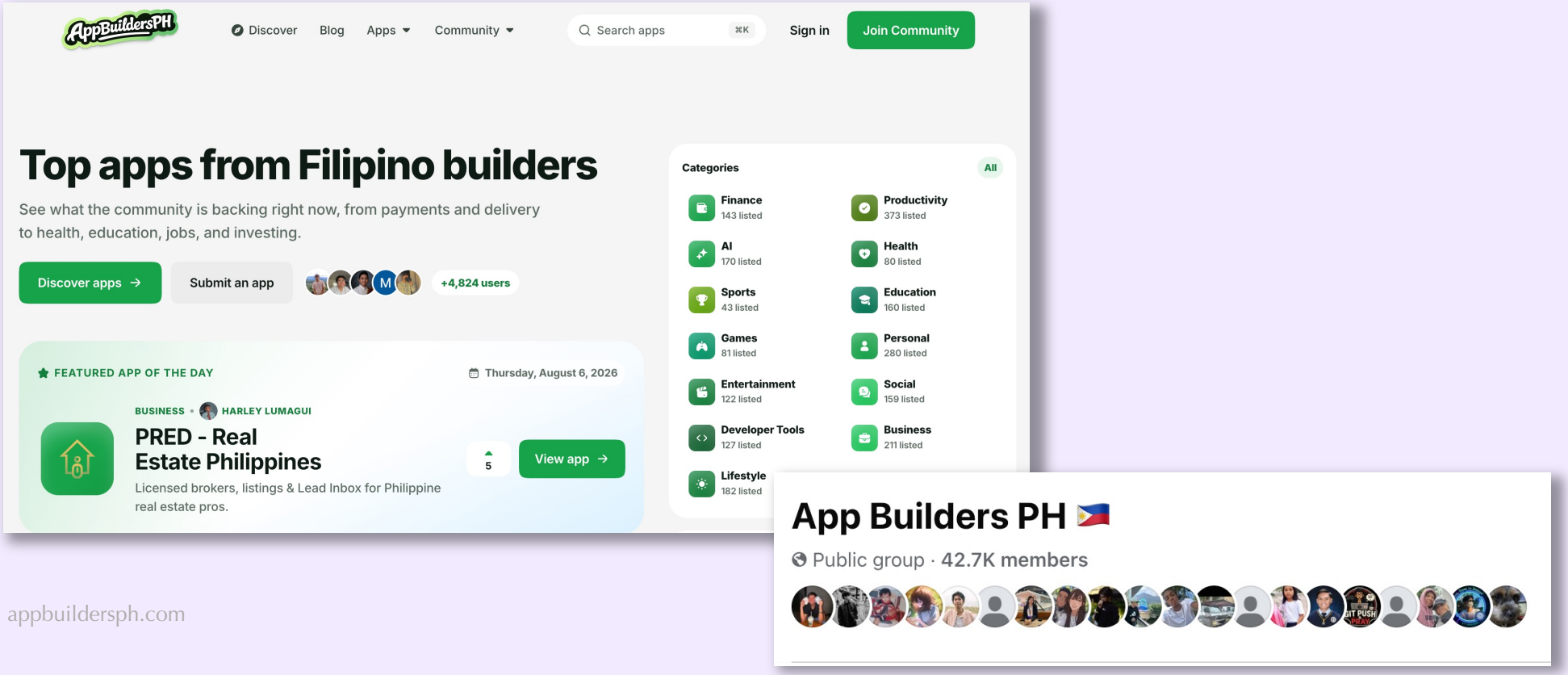


Andrej Karpathy
@karpathy

There's a new kind of coding I call "vibe coding", where you fully give in to the vibes, embrace exponentials, and forget that the code even exists. It's possible because the LLMs (e.g. Cursor Composer w Sonnet) are getting too good. Also I just talk to Composer with SuperWhisper so I barely even touch the keyboard. I ask for the dumbest things like "decrease the padding on the sidebar by half" because I'm too lazy to find it. I "Accept All" always, I don't read the diffs anymore. When I get error messages I just copy paste them in with no comment, usually that fixes it. The code grows beyond my usual comprehension, I'd have to really read through it for a while. Sometimes the LLMs can't fix a bug so I just work around it or ask for random changes until it goes away. It's not too bad for throwaway weekend projects, but still quite amusing. I'm building a project or webapp, but it's not really coding - I just see stuff, say stuff, run stuff, and copy paste stuff, and it mostly works.

7:17 AM · Feb 3, 2025 · **7.3M** Views

Vibe Coding



appbuildersph.com

Vibe Coding


User Write Prompts
AI generates codes
Edit Prompt
AI Code
Desired State

Vibe Coding is skipping SDLC

Phase 2: Agentic Coding

What changed

- Claude Code running in the terminal, against the real repo
  - CLAUDE.md in every project - our conventions, naming, error-handling patterns
  - Plan mode first: the agent proposes, a developer approves, then it edits
  - MCP servers connecting it to our API specs and outside tools
- Sub-agents in parallel: one drafts the script, one writes unit tests

What did not change

- A human approves the plan, reads the diff, and owns the merge commit

The model will use its imagination if you don't specify.

Spec-Driven Development

Traditional Development

Test-Driven Development

Spec-Driven Development

Think, then Spec, then Plan, then Code

Spec-Driven Development

Vibe Coding

```
Write a functional component for a
login page...
```

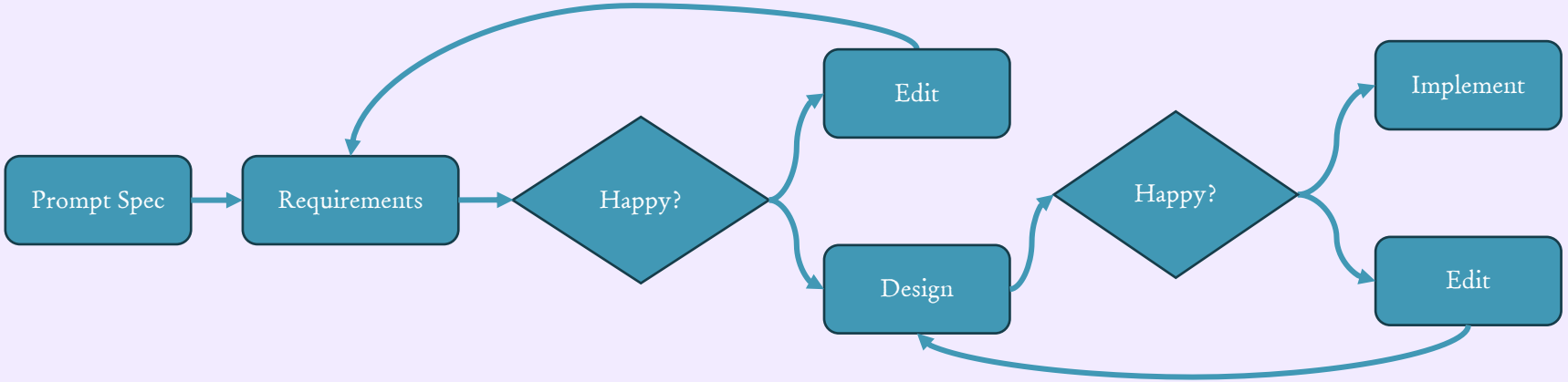
Spec Coding

```
Feature: User Authentication
Endpoint: /login POST
Accept: {user, pass}
Failure: 400 if missing user
Tests: valid creds, return 200
```

Toolkit: Github's Spec-Kit

https://github.com/github/spec-kit

Spec-Driven Development



Phase 3: Agentic Engineering (In Progress)

What we are building now

- Skills: our conventions packaged so the agent applies them unprompted
  - Naming, error handling, logging, design patterns
- Plugins: bundles of skills, commands, and connectors shared across the team
- MCP servers wired into our own systems, not just public ones
- Hooks and gates: the agent cannot merge, deploy, or touch production

The shift

- We stopped trying to prompt better. We started building the environment it works in.

What Agentic Engineering Actually Requires

- **Specification:** writing down what correct means before any code exists
- **Decomposition:** tasks small enough to finish and clear enough to verify
- **Constraint design:** deciding what the agent must never be allowed to do
- **Domain knowledge:** you cannot specify what you do not understand

Human-in-the-Lead

“Human in the Lead” not just “Human in the Loop”

~ Julie Sweet, CEO at Accenture

Four Pillars of Human-Led AI Development

- **Comprehension** - can you explain what it does, in standup, without notes?
- **Override** - can you tell it no, and say why?
- **Guardrails** - what is the agent explicitly not allowed to touch?
- **Accountability** - your name is on the merge commit, not the model's.

Own Your AI

- The model changes under you, and behavior changes with it
- Your code, your client's data, your students' work - all leave the building
- Local and open-weight models are slower and weaker, and entirely yours
- For teaching, a modest model you control can beat a strong one you do not

"Own your AI. AI in the cloud is not aligned with you; it's aligned with the company that owns it." ~Mitko Vasilev

What Stays Human

- The requirement nobody wrote down
- Deciding what not to build
- Telling a client no
- Judging whether a trade-off is acceptable to the business
- Accountability at 2am when it breaks in production

AI has no stake in the outcome. You do.

“You can delegate authority, but you cannot delegate responsibility.”

~ Former U.S. Senator Byron Dorgan

What this means for the Classroom?

From “how to think about AI”, to “how to think with AI”.

Your Students Are Already Using AI

- They're using ChatGPT to write assignments
- They're using Copilot to finish lab exercises
- They're submitting AI-generated code they don't fully understand
- And some of them are getting *really good at it*

Cognitive Offloading

Remove the friction and you remove the skill.

- Writing code is not only how code gets produced - it is how thinking gets trained
- Stop lifting and you lose strength. Stop struggling and you lose reasoning.
- A student who never gets stuck never builds the judgment to review

Which Friction to Remove, Which to Keep

Worth removing

- Environment and toolchain setup
- Looking up syntax you already understand
- Mechanical, repetitive refactors
- Formatting and configuration files

Worth keeping

- Debugging something you wrote yourself
- Designing before writing any code
- Reading unfamiliar code without help
- Explaining your solution out loud
- Being stuck for a while

Beginners and Advanced Need Opposite Things

Years 1-2: build judgment

- AI in explain-mode, not write-mode
- Code written by hand, defended aloud
- Read more code than they write
- Fundamentals get more time, not less
- Goal: know what good looks like

Years 3-4: build the workflow

- Spec-driven development as a graded skill
- Reviewing AI output as a core exercise
- Agent workflows, MCP, tool integration
- Ship something that has a real user
- Goal: know when to say no

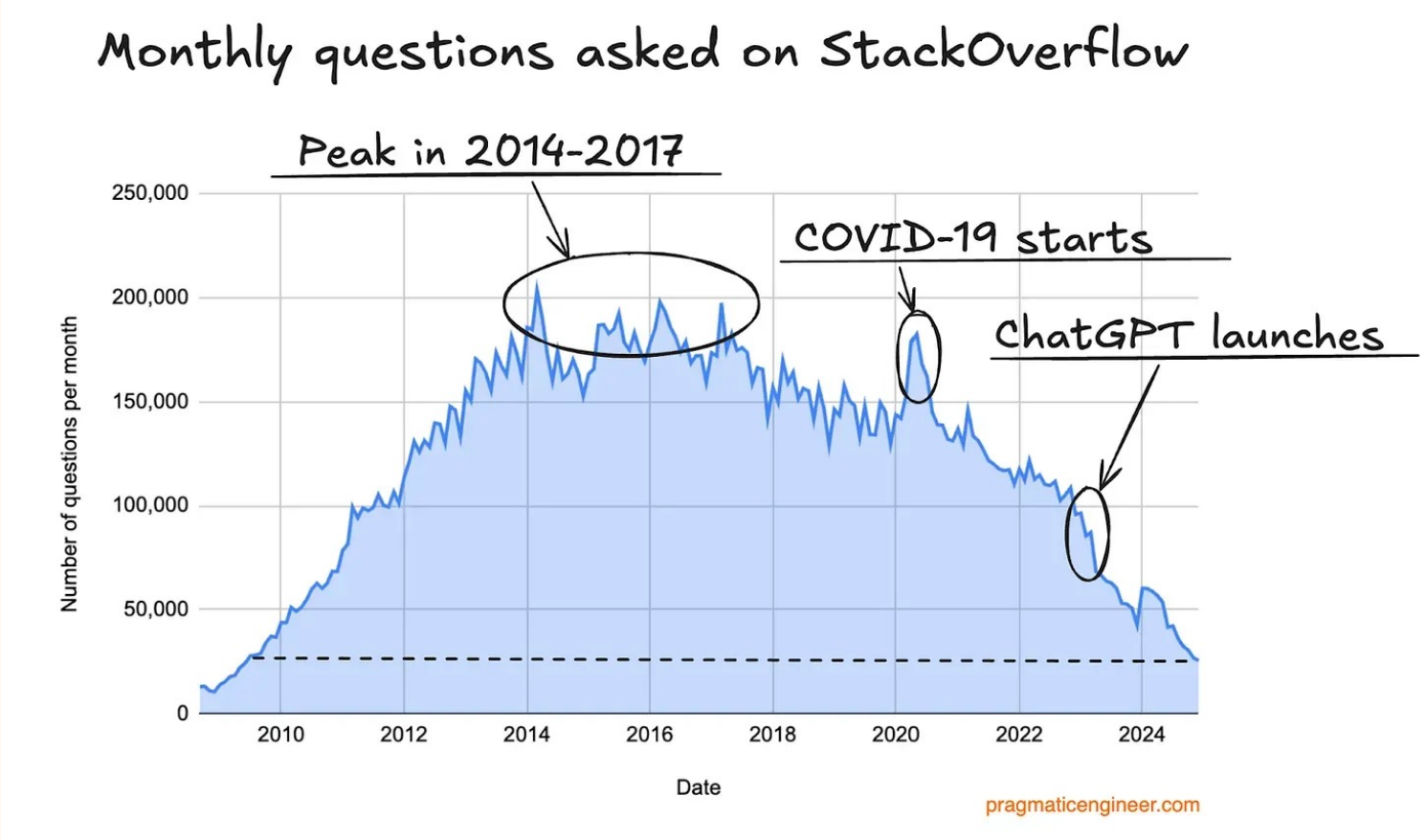
Assessment Is the Real Problem

- Oral defense: they explain the code they submitted, line by line
- Grade the code review, not the code
- Hand them AI-generated code with three bugs planted in it
- Assess the commit history and the prompts, not only the artifact
- Keep some in-person, no-tools work for the fundamentals

What Industry Needs From the Graduates

- How to read and debug code they didn't write
- How to write a clear, structured prompt
- How to evaluate whether code is correct, secure, and maintainable
- How to learn tools that didn't exist when they were in school
- That "it works" is not the same as "it's good"

Where Did the Learning Go?



Filling the gaps (New courses or modules to consider)

- Prompt Engineering and AI Communication
- AI-Assisted Software Development Lab
- Ethics of AI in Software Engineering
- Code Review and Quality Assurance in an AI Era
- Agentic Systems Design

What I Tell New Graduates

- Work on your foundations. Tools change; principles last.
- Learn how to learn - it is the only skill with no expiry date.
- Get comfortable with getting uncomfortable.
- Everyone will know how to code a little. The ones who master it will lead them.

Three Things to Try Next Semester

- **One** lab where AI is required - and the grade is on the review, not the output.
- **One** existing assignment with an oral defense added. Change nothing else.
- **One** AI policy per assignment, not per syllabus - students must know which mode they are in.

AI is not a replacement. It is a multiplier.

Thank you!

Q & A