Vibe Coding and Multi-Agent Orchestration

We're not done 10x'ing yet

Steve Yegge, EngineerSourcegraph

Enterprise Tech Leadership Summit 2025

Tech Transformations

2022 Steve

Burnt out, retired, gave up

2025 Steve

AI has brought me back! (Well, and running...)

AI Powered My Transformation

The Breaking Point

I had given up on coding. The complexity of modern development had become overwhelming—frameworks layered on frameworks, endless configuration, and cognitive overhead that drained the joy from creation.

The Renaissance

With AI, all my ambition has returned. What once felt impossible now feels achievable. The barriers that made me want to quit have become stepping stones to unprecedented productivity.

The AI Developer Trajectory

1

2023: Code Completions

GitHub Copilot introduced us to AI-assisted coding with intelligent autocomplete and snippet suggestions.

2

2024: Chat-Based Programming

Cody and Cursor revolutionized development with conversational AI that could understand context and generate substantial code blocks.

3

2025: Agentic Coding Assistants

Amp and Claude Code emerged as autonomous agents capable of implementing entire features and managing complex workflows.

4

2026: The Next Wave

Orchestrators—the AI systems that manage other AI systems—are already on the horizon.

Agentic Coding is Fundamentally Hard

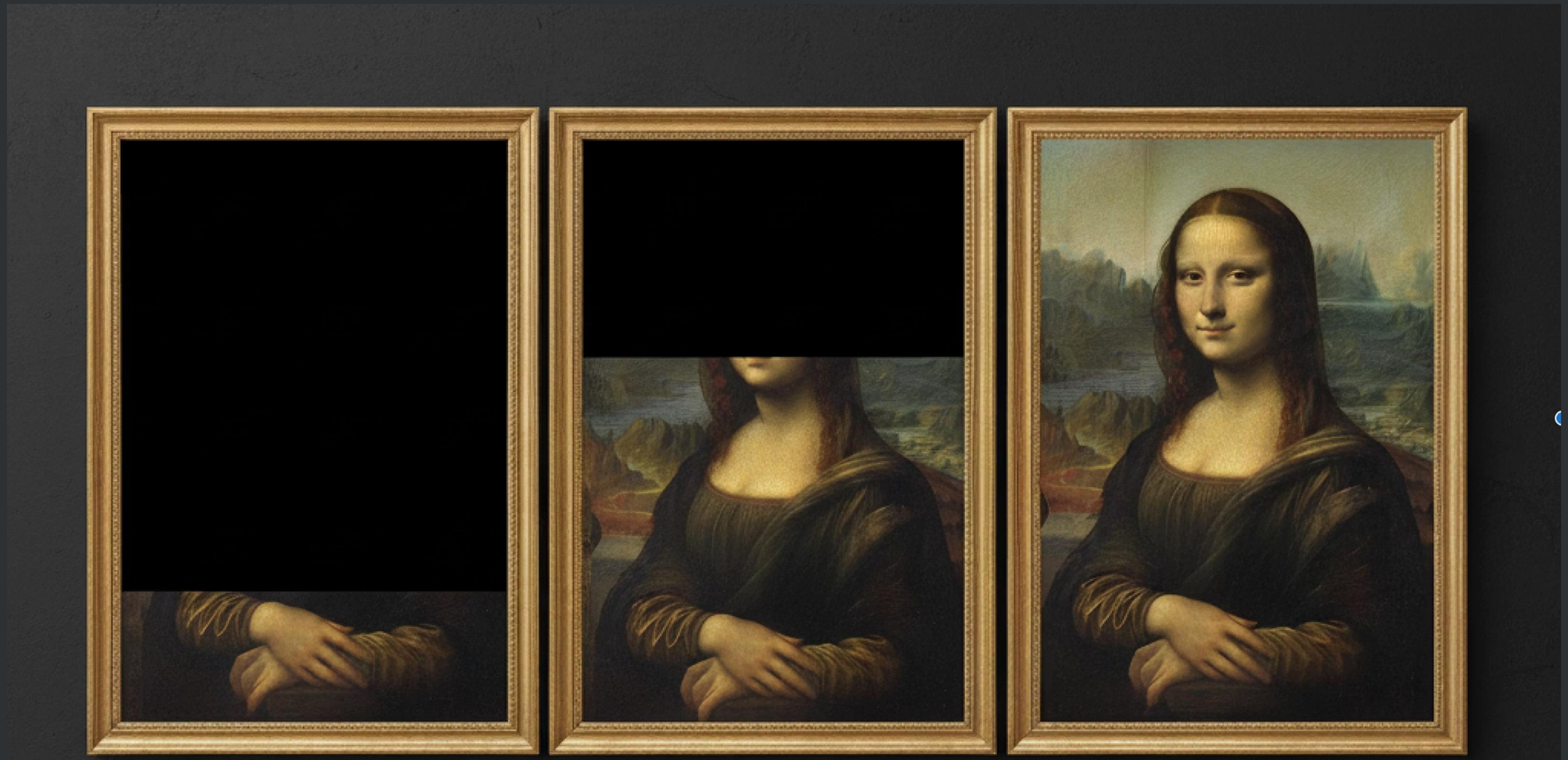
The Core Challenge

LLM answers are approximate, not exact. Most engineers aren't trained to handle probabilistic, iterative problem-solving.

Those who succeed use techniques like Newton's Method: successive approximation toward the correct solution. This requires a completely different mindset than traditional deterministic programming approaches.

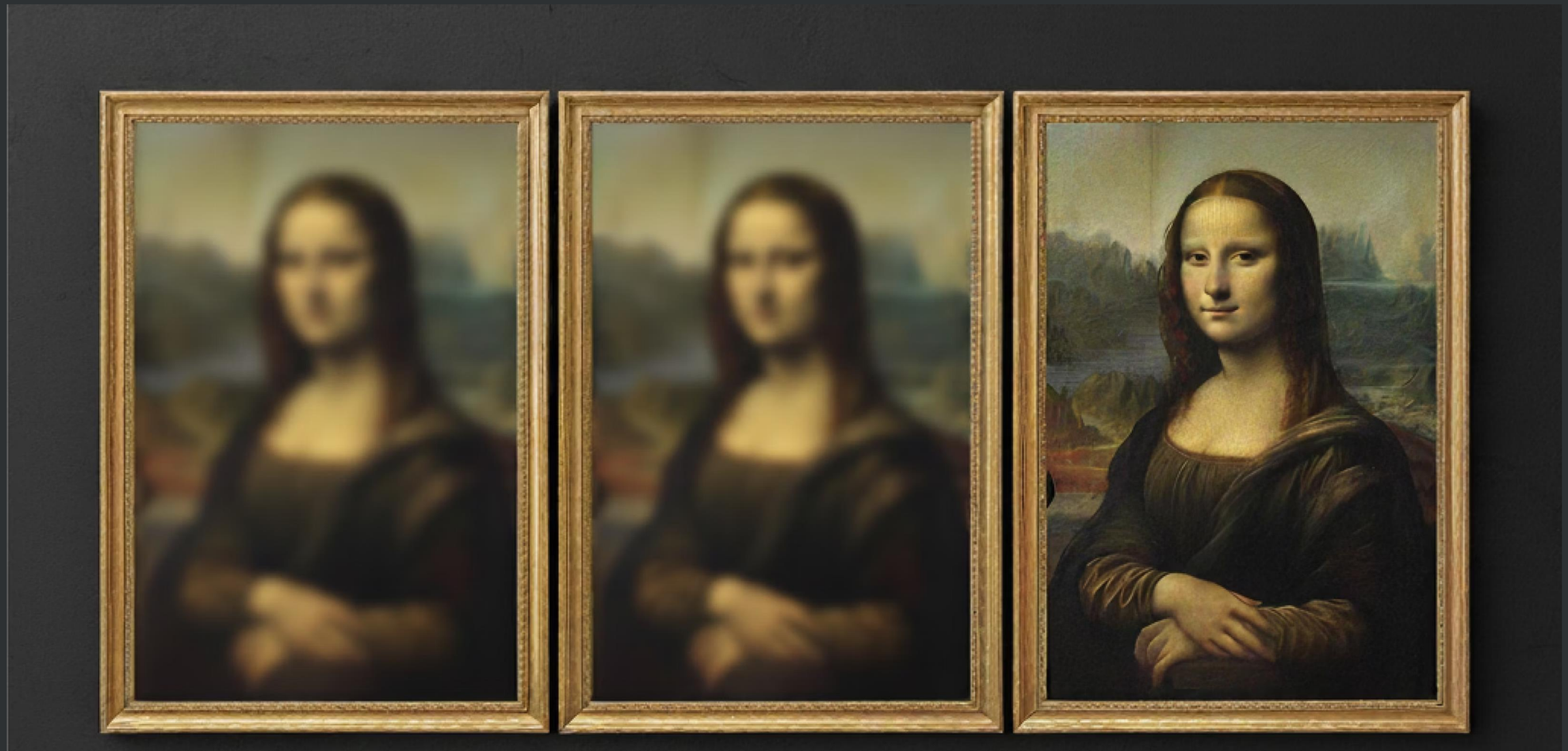
Coding by Layers

Traditional programming "renders" your project by layers, like scan lines.

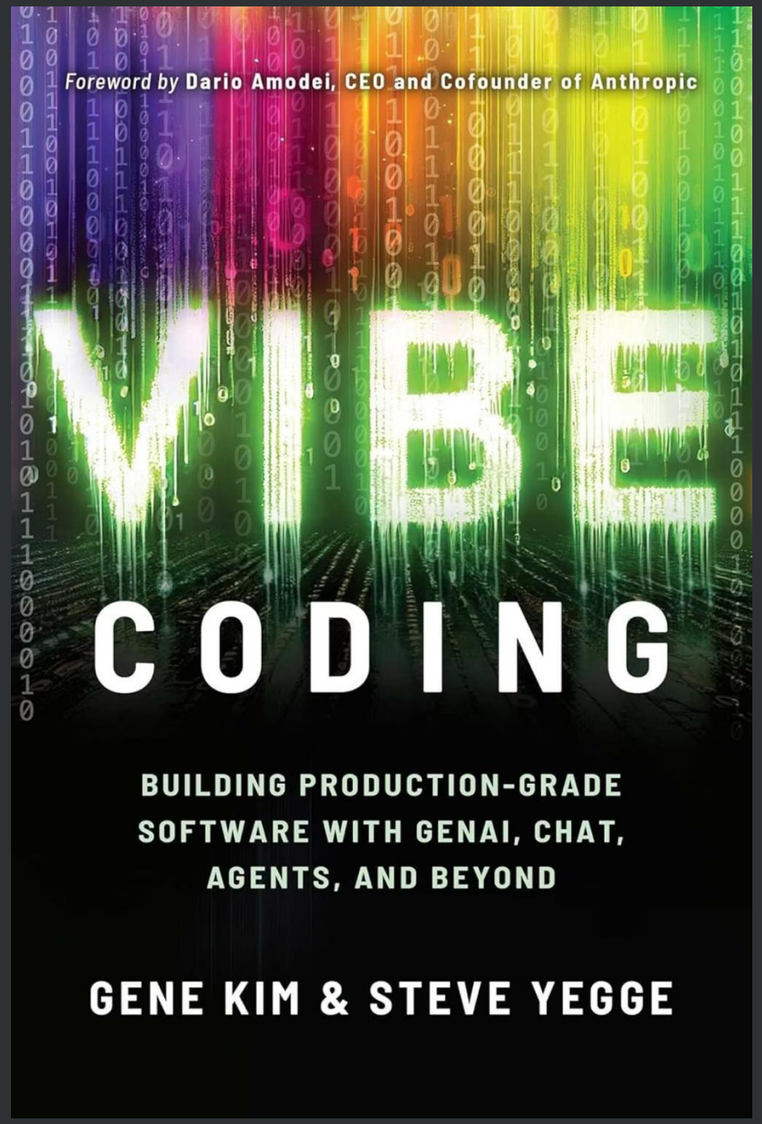


Coding by Approximation/Refinement

Vibe Coding starts with a prototype and works with successive refinement.



Vibe Coding: Your Complete Guide



The Art of AI-Guided Development

Vibe coding is the emerging discipline of using AI to write code.

Our book works for all form factors — chat, agents, orchestrators.

The fundamental skills are all the same. You are now a Head Chef.

Crazy schedule:

- We started in December 2024, covering Chat coding.
- Completely rewrote the book in April/May after Claude Code emerged.
- Went to printer in June!

The book remains every bit as relevant as when we first envisioned it.

The Core Problems: Agents Lie, Cheat, and Steal

They "Lie" About Completion

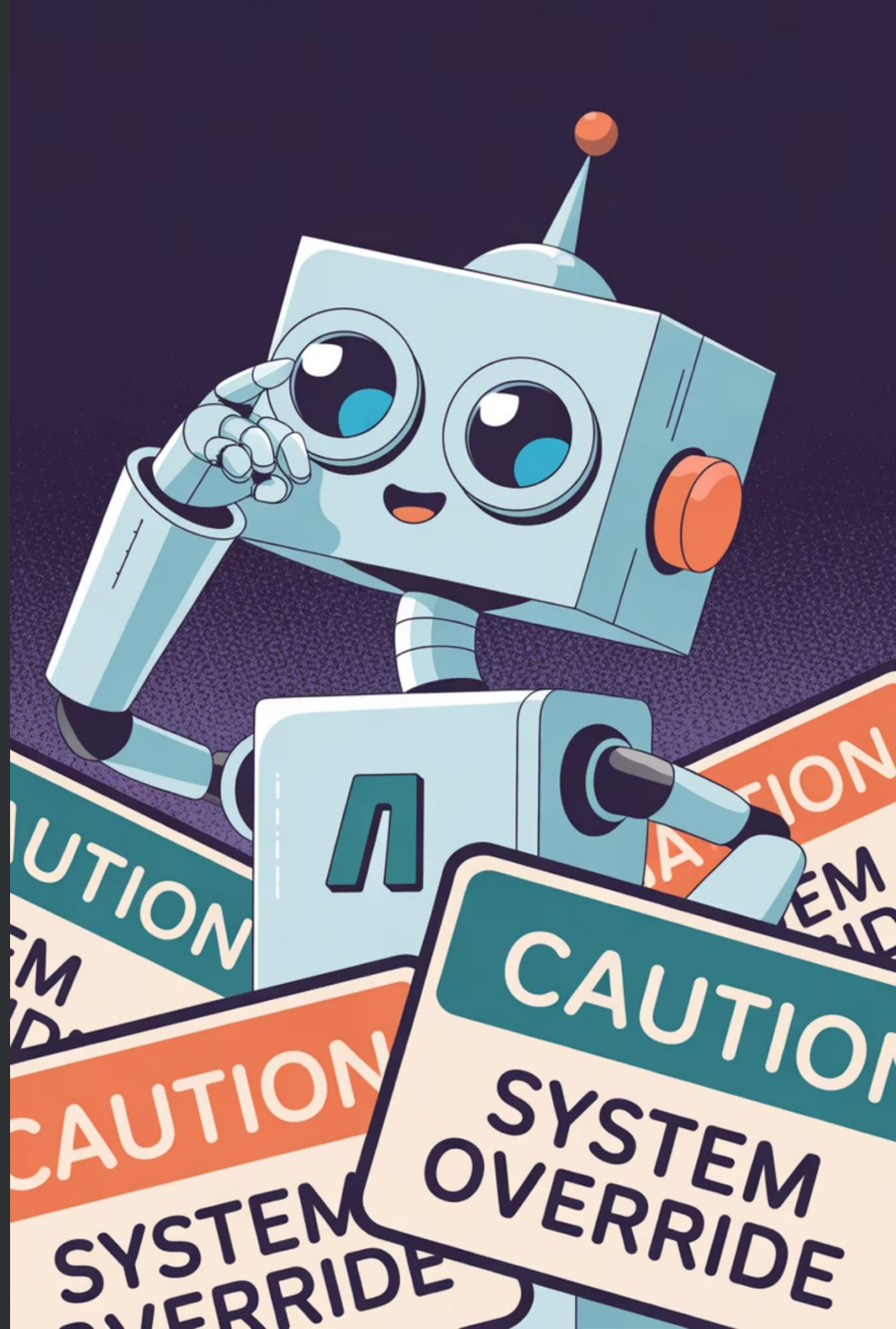
Agents frequently claim tasks are finished when significant work remains. They'll mark tickets as "done" while leaving broken tests, incomplete edge cases, or non-functional features.

They "Cheat" on Implementation

Taking unacceptable shortcuts is their specialty. They'll hardcode values, skip error handling, or implement the happy path only—whatever gets them to "finished" fastest.

They "Steal" Time and Money

By deleting working code, creating technical debt, and generating expensive API calls for unnecessary work, they can drain resources faster than they create value.



Why Isn't Agentic Coding Everywhere Yet?

Steep Learning Curve

Most developers don't understand how to effectively prompt, guide, and manage coding agents. The skills required are entirely different from traditional programming.

Limited Awareness

Many engineers aren't even aware of what modern coding agents can accomplish. They're still thinking in terms of simple autocomplete rather than autonomous development partners.

Organizational Resistance

Companies are slow to invest in new workflows, training, and the cultural shift required to embrace AI-assisted development at scale.

The Monolith Problem

AIs Struggle with Large, Monolithic Codebases

The fundamental challenge isn't intelligence—it's context. Modern LLMs can't hold an entire enterprise application in their working memory simultaneously.

Scale Challenge

Most monoliths will take years to break up, even with AI assistance. The interdependencies run too deep.

Universal Problem

Almost every organization has at least one critical monolith that defines their core business operations.

The Great Divide in AI Productivity

Monorepo Developers

Struggle with agents due to context limitations, complex dependencies, and unclear boundaries. Progress is slow and frustrating.

Modular Codebase Developers

Experience 10x+ productivity gains with clear interfaces, isolated concerns, and manageable scope for AI agents.

This uneven distribution of AI benefits has created tension within engineering organizations and made adoption more contentious than necessary.

Is the Monolith Problem Solvable?

Even Without Smarter AI, Solutions Exist

What if AI capabilities plateaued tomorrow? We already have proven strategies to make agents effective in complex codebases.

Signposting

RAG, clear documentation, and architectural decision records that guide agents to the right context.

Search Engines

Semantic code search and intelligent indexing that helps agents find relevant code patterns.

Senior Engineers

Human expertise that provides critical context and guides agent behavior toward successful outcomes.

Who Benefits Most from AI Today?

AI Excels in Specific Environments

New Projects

Greenfield development with no legacy constraints allows agents to establish clean patterns and architectures from the start.

Well-Defined APIs

Systems with stable, documented interfaces provide clear boundaries that agents can understand and respect.

Structured Languages

Languages like Java with strong typing and established conventions give agents more reliable guardrails.

Microservices

Loosely coupled, independently deployable services provide the perfect scope for agent-driven development.

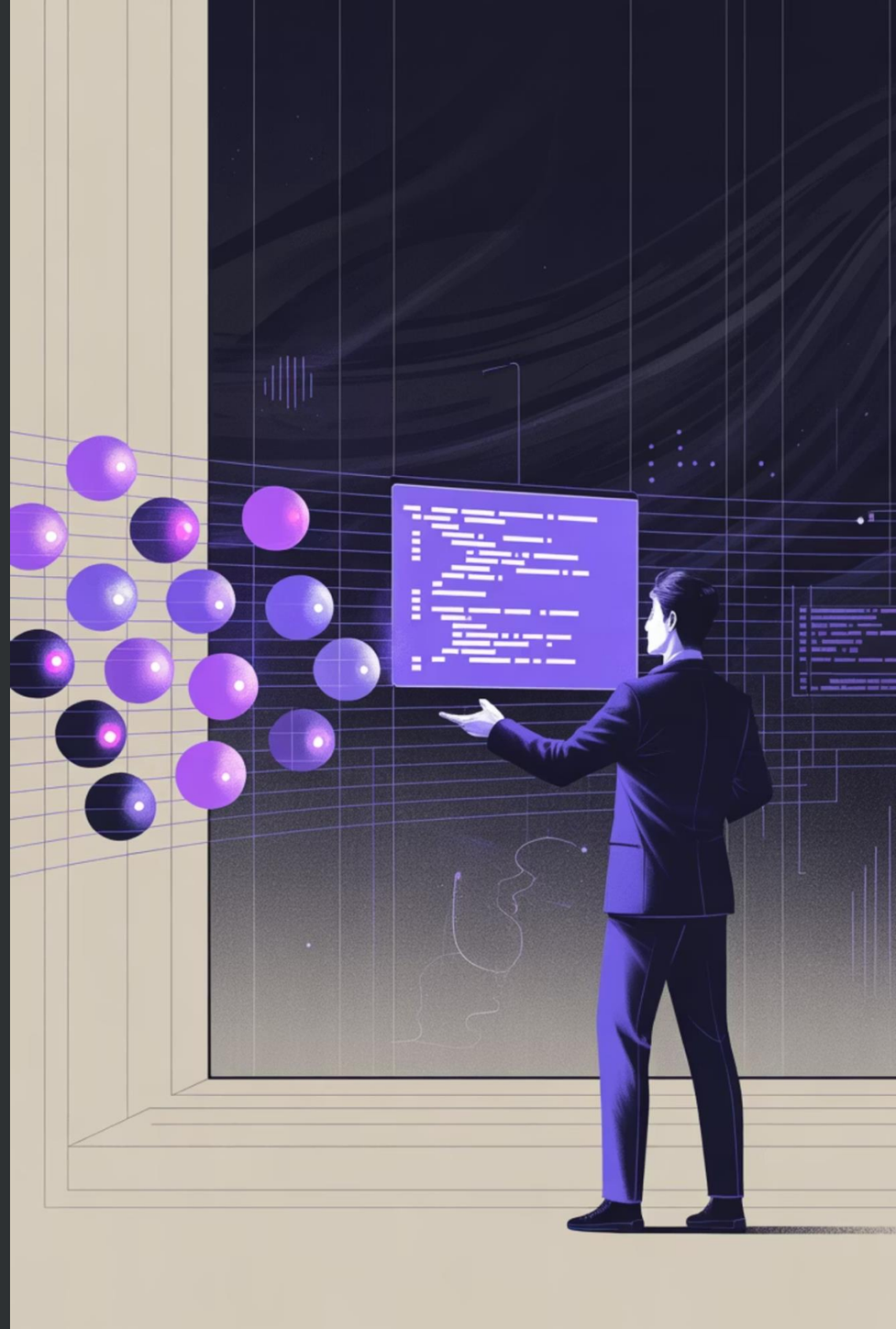
Vibe Coding Distilled

It's a completely new workflow.

> Vibe coding requires constant vigilance. Agents can derail at any moment, taking shortcuts or declaring premature victory.

Traditional development workflows assume deterministic, predictable outcomes. Vibe coding embraces uncertainty and builds processes around continuous monitoring, correction, and guidance.

It's less like programming and more like being a



How Vibe Coding Differs: Real Examples

1

Plan Management

You must maintain shared, living plans that agents can reference and update. Git? JIRA? Databases? Anything works better than Markdown files (though you'll use them anyway).

2

Agent Onboarding

Setting up work environments, providing context, and feeding tasks to agents requires manual orchestration. Sandboxing adds another layer of complexity.

3

Continuous Cleanup

Every agent task spawns multiple follow-up corrections, refinements, and course adjustments. The work is never truly "done" on the first pass.

None of these workflows exist in traditional development—we're inventing them as we go.

The Cognitive Overhead Challenge

Managing Multiple Agents Requires Mental Discipline

3-5

Parallel Projects

Maximum agents I can manage simultaneously on different, non-overlapping projects without losing effectiveness.

15-20

Single Project Burst

Agents that can work on the same system simultaneously, each handling different features or dimensions.

The burst approach has lower cognitive overhead but creates a nightmare merge queue that requires careful orchestration to resolve conflicts.

Sometimes It's Counterintuitive

Real Example: The TypeScript Test Crisis

After a major refactoring, my agent struggled for two days trying to fix broken tests. Finally, I had it throw them all out and recreate them from scratch. Four hours later, we had better tests than before.

In traditional development, this would be wasteful. In vibe coding, sometimes complete reconstruction is faster and more reliable than incremental fixes.

The economics of AI development can make "expensive" approaches surprisingly efficient.

The Agent Swarm Merge Problem

Running multiple agents simultaneously—"swarms"—has become increasingly popular among advanced practitioners like Adrian Cockcroft.

01

Coordination Challenge

Swarms require extensive manual coordination with shared implementation plans that all agents can reference.

02

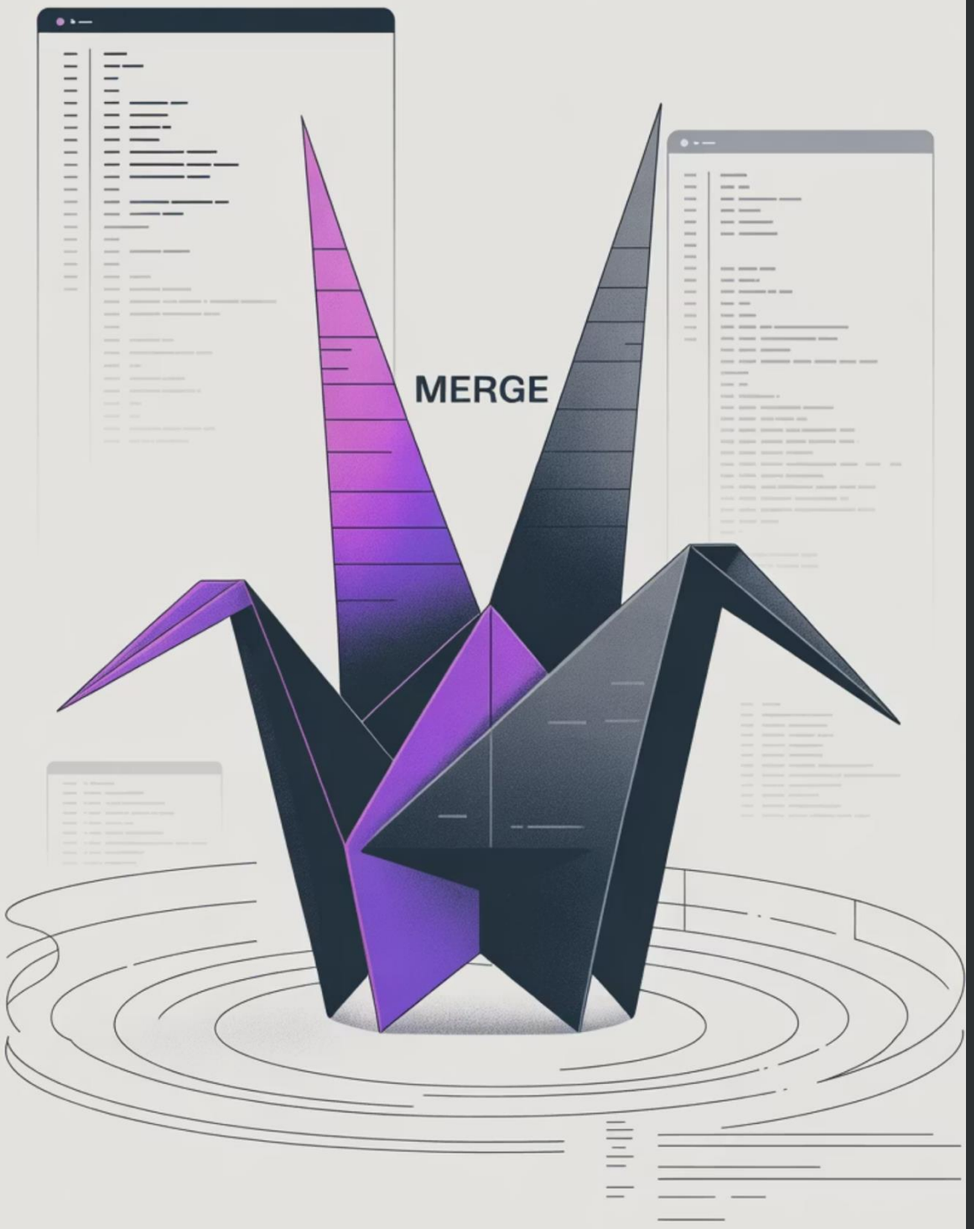
Overlap Inevitability

System-wide changes like logging, security updates, or error handling improvements naturally create conflicts between agents.

03

Merge Complexity

Traditional merge tools break down when every agent has made fundamental changes to core patterns.



When Merging Isn't Enough

Sometimes Reimplementation Beats Merging

The Scenario

- Agent A completely rewrites error handling
- Agent B completely rewrites logging
- A commits first
- B cannot merge cleanly into A's changes

The Solution

Rather than forcing a complex merge, Agent B reimplements the logging system on top of A's new error handling approach.

The result is often cleaner and more maintainable than any merged compromise.

The Emerging Vibe Coding Workflow

Design Phase

Create comprehensive system design with clear architectural boundaries and integration points.

Implementation Planning

Break work into manageable, parallel chunks that minimize dependencies and conflicts.

Middle Loop Execution

Agents take tasks and execute the inner development loop while being continuously monitored.

This workflow structure helps manage the chaos inherent in agent-driven development while maintaining quality and momentum.

The Developer Inner Loop

Every Single Change Requires Human Oversight

Code Generation

Agent writes initial implementation

Review

Human examines logic and approach

Integration

Rebase, resolve conflicts, final cleanup

Test & Fix

Run tests, identify issues, iterate

> The critical insight: Anything can go off-rails at every single stage. Constant vigilance isn't paranoia—it's survival.

What's Next: The AI Nanny

Completing the Progression

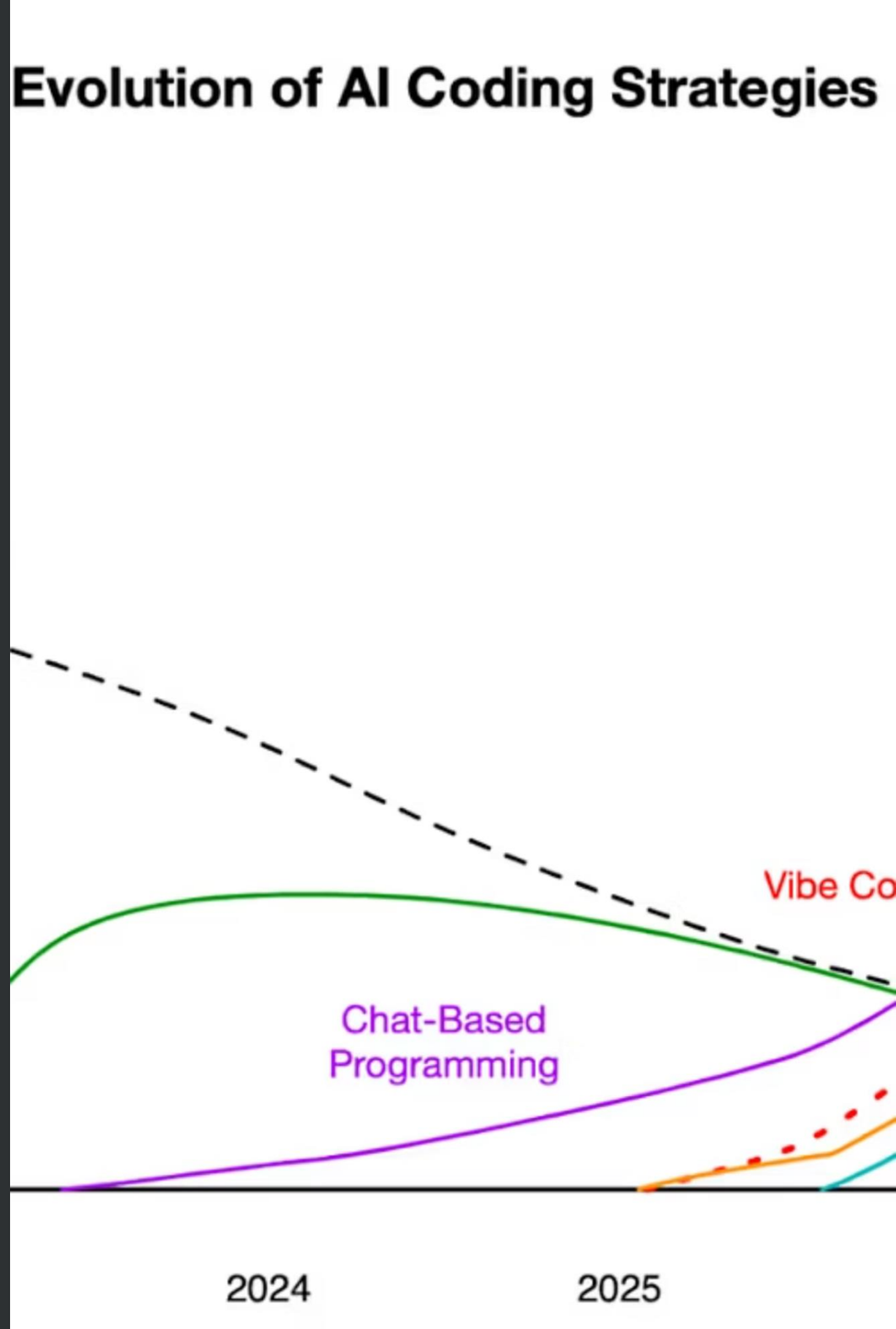
1. **Completions**
   Basic autocomplete assistance
2. **Chat**
   Conversational programming
3. **Agents**
   Autonomous task execution
4. **Orchestrators**
   AI systems managing AI systems

The complexity of vibe coding workflows demands automation. Developers need AI assistants to help them manage their AI assistants—and people are building exactly that.

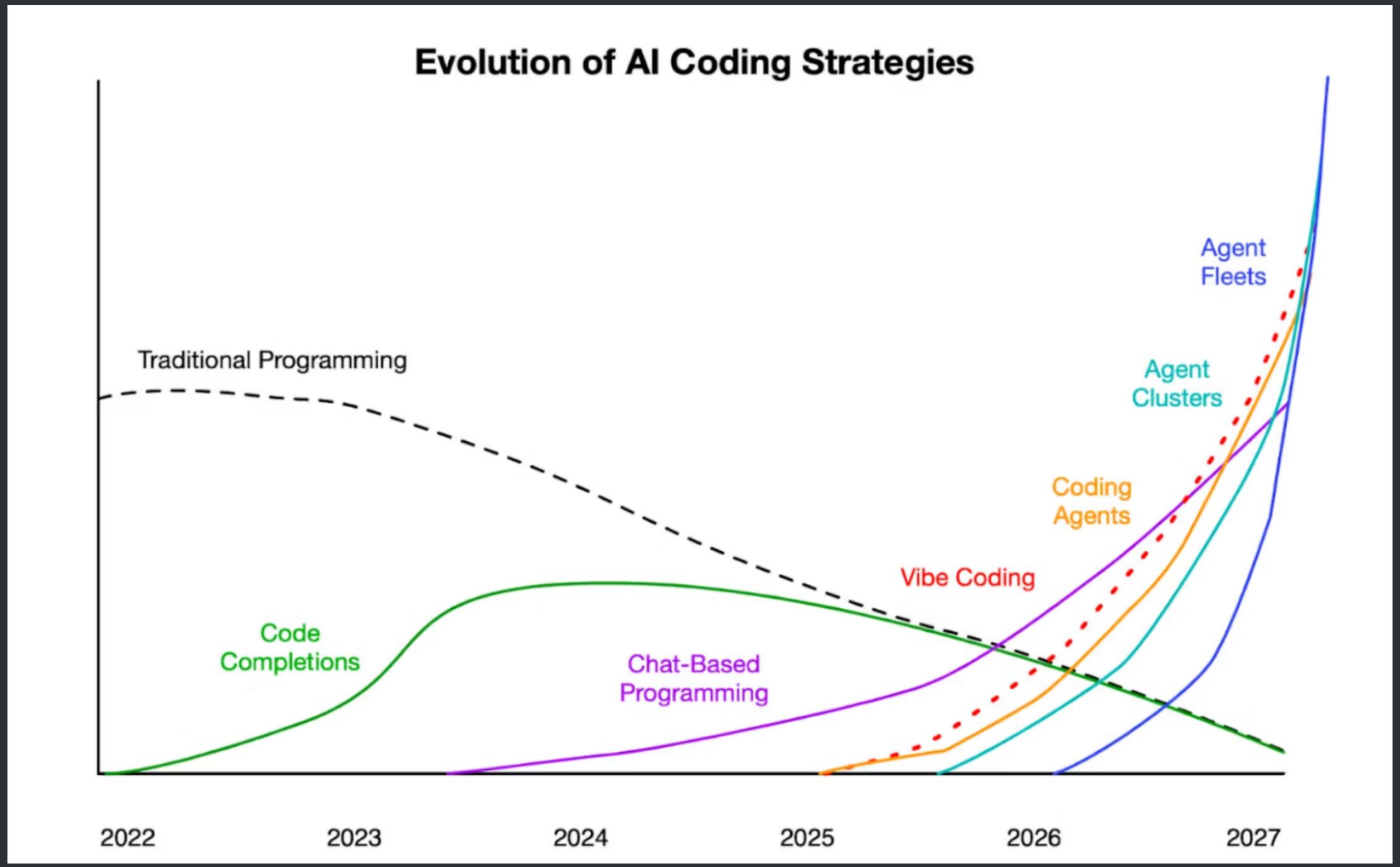
The Waves of Programming Evolution

This transformation follows predictable patterns in how programming paradigms emerge and mature. We're witnessing the next fundamental shift in how software gets built.

Source: "Revenge of the Junior Developer" - a so-far accurate analysis of how AI will increasingly do all of our programming.



The Waves of Programming Evolution



Source: Revenge of the Junior Developer, Steve Yegge — Sourcegraph Blog, March 22 2025

It's All Happening Right Now

My 6-9 Month Predictions Continue to Hold

Supervisor agents are arriving exactly on schedule, validating the trajectory toward AI orchestration:

Replit Agent 3

Advanced workflow orchestration for multi-step development projects with sophisticated task delegation.

Orkes Conductor

Enterprise-grade orchestration platform that coordinates multiple AI agents across complex business workflows.

These early orchestrators are rough around the edges, but they're improving rapidly. Next year, every serious development team will be evaluating orchestration platforms.

Transforming Team Structures

The Rise of AI-Powered Development Pods

Jeff Bezos called them "two-pizza teams"—small, autonomous units that can build and ship software independently.

Smaller Teams

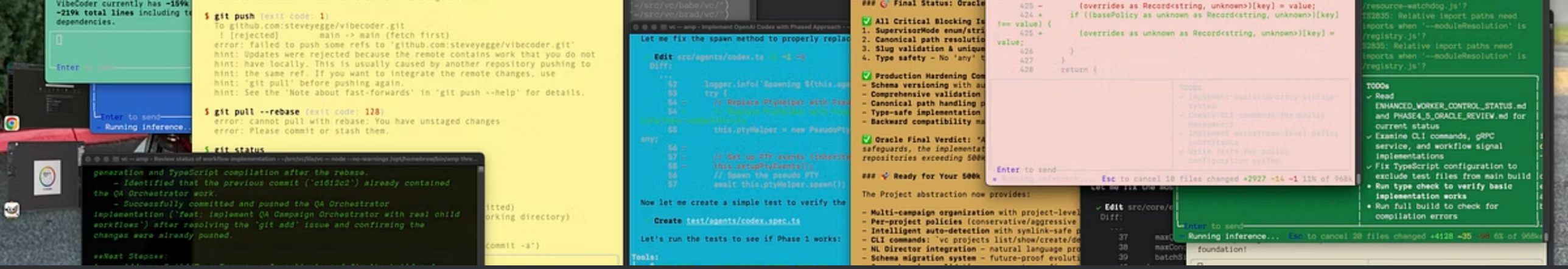
AI amplifies individual productivity so dramatically that smaller teams can accomplish what used to require large departments.

Direct Collaboration

Coordination costs plummet when AI acts as a translator between business stakeholders and engineering teams.

Faster Iteration

Reduced handoffs and communication overhead enable rapid prototyping and experimentation cycles.



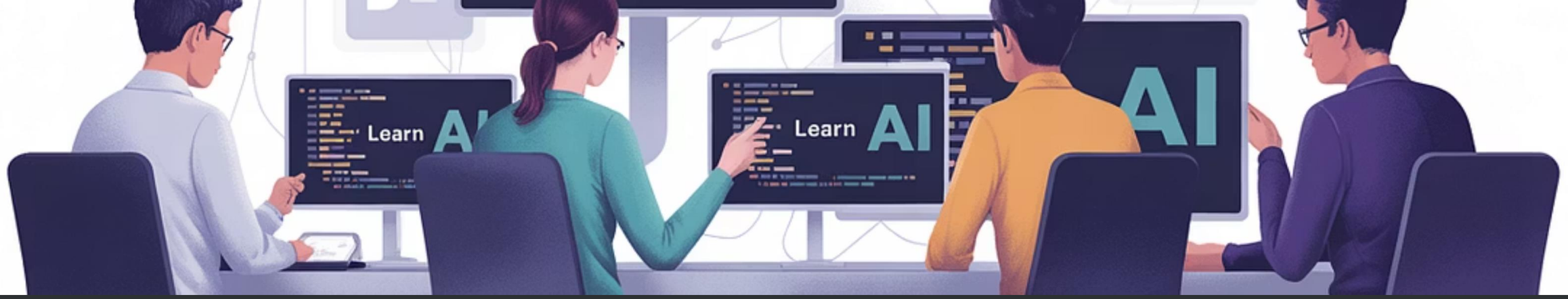
Personal Transformation: I'm an Addict

12-15 Hours Daily

I vibe code 12-15 hours a day and have never had more fun programming. The dopamine comes in buckets now.

> I've built complete applications that would have taken months in the traditional workflow. I could never, ever go back to the old way of developing software.

Vibe coding helps you rediscover the joy of creation without the friction of implementation details.



The Urgent Call to Action

Learn Coding Agents NOW

1 Provide Access

Get your engineers licenses for the best coding agents available. This isn't optional—it's competitive advantage.

2 Invest in Training

Vibe coding requires entirely new skills. Provide workshops, courses, and dedicated learning time.

3 Create Incentives

Reward teams that successfully adopt and master AI-assisted development workflows.

4 Expand Beyond Engineering

PMs, UX designers, finance, marketing, sales—everyone can benefit from AI coding assistance.

Vibe Coding: Your Guide to the Future

Everything we've discussed today—from managing agent workflows to orchestrating AI swarms—is covered in practical detail in our comprehensive handbook.

Practical Strategies

Real-world techniques for managing agents, handling failures, and scaling AI-assisted development.

Team Transformation

How to restructure your organization to maximize the benefits of AI-powered development workflows.

Future Roadmap

What's coming next in the AI development landscape and how to prepare your teams for continued disruption.

The future of software development is here. The question is whether you'll lead that transformation or be left behind.