

SCALE 23x | Pasadena, March 5-8, 2026

# Beyond Vibe Coding

How to Scale AI-Assisted Development Without Architectural Chaos

Mushegh Gevorgyan

Creator of SpecMind



Got questions? Scan and submit anytime. We'll review them at the end.

# The Rise of Vibe Coding

10x

faster feature development with AI assistants

Claude Code

Codex

Cursor

Windsurf

Individual engineers are shipping faster than ever before

# The Hidden Problem

What happens when a **whole team** vibe codes?

Engineer A + AI

->

Creates new DB table

Engineer B + AI

->

Creates another DB table **(same data!)**

Engineer C + AI

->

Spins up new service **(existing one works)**

Each engineer + AI works in isolation with limited context

# The Review Problem

We have code reviews because engineers make wrong decisions. But now...

**AI generates** -> 500 - 2000 lines per PR (**impossible to review**)

**Already merged** -> Too late to catch issues (**damage done**)

**Review feedback** -> "Just rewrite everything" (**exhausting & slow**)

The mechanisms we used to prevent bad code just don't work anymore

# The Result

5 engineers × AI × several sprints

x Shipping fast, but codebase becoming unmaintainable

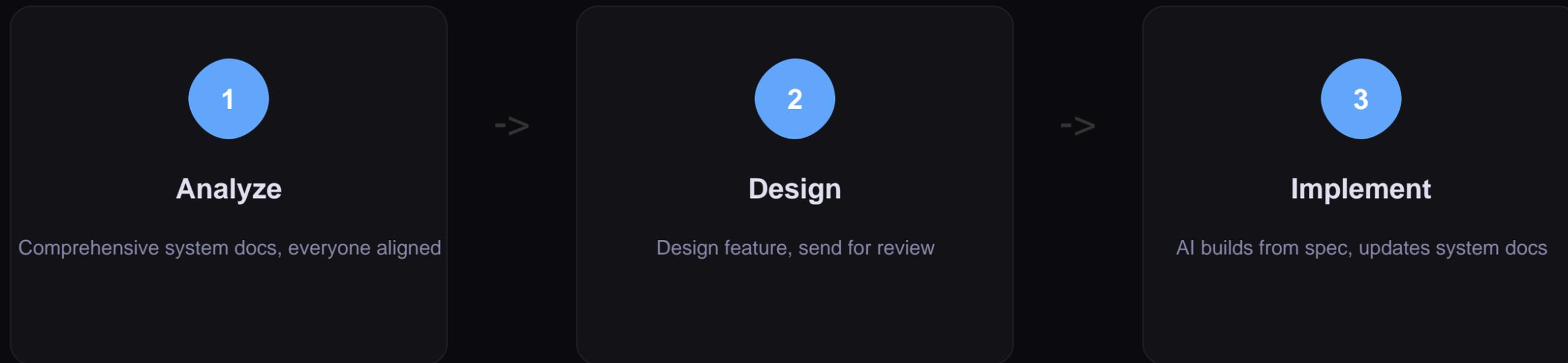
x Duplicate tables, redundant services

x Inconsistent patterns across the codebase

We don't need better code review. We need to review the *design* before AI writes the code.

# Introducing SpecMind

The idea is pretty simple



Document -> Design -> Review -> Implement -> Update docs

# Phase 1: Analyze

/analyze

## Tree-sitter Parsing

Understands structure, not just text

## Detects Everything

Services, layers, databases, APIs, dependencies, frameworks, ORMs

## Full Architecture Docs

Mermaid diagrams for the entire system

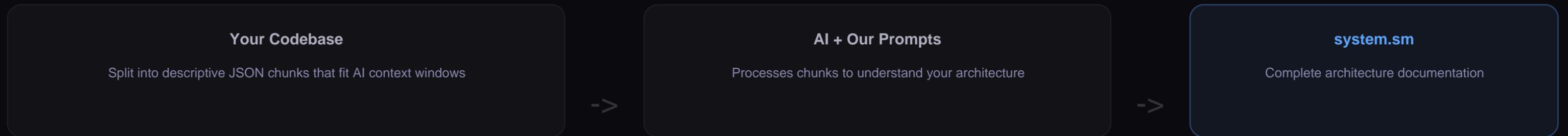
LIVE DEMO

# Analyzing a Real Codebase

`/analyze`

*Press F to toggle fullscreen*

# What Analyze Produces



**System Architecture**  
Global view of all services, databases, and integrations

**Per-Service Breakdowns**  
Each service broken down by layer

**Cross-Service Flows**  
Sequence diagrams showing how services talk to each other

**ER Diagrams**  
Entity relationship diagrams for your database models

LIVE DEMO

# Analyze Results

*Press F to toggle fullscreen*

# Phase 2: Design

`/design Real-time Notifications`

AI reads your `system.sm` -- knows existing services, tables, patterns

Creates a spec showing how the feature fits into your architecture

Color-coded diagrams: `new``changed``removed`

Send the spec to your team for review

LIVE DEMO

# Designing a Feature

/design Feature Name

*Press F to toggle fullscreen*

# Why Review Designs, Not Code

Reviewing AI Code

**2,000+**

lines of generated code

"Is this code correct?"

vs

Reviewing a Design Spec

**1**

.sm file with diagrams

"Does this design make sense?"

Catch problems **before** code is written -- saves days or weeks of refactoring

LIVE DEMO

# Design Results

*Press F to toggle fullscreen*

# Phase 3: Implement

`/implement Real-time Notifications`

1

Takes your feature spec and implements it as instruction

2

Automatically updates system.sm documentation

3

Adds entry to system changelog

LIVE DEMO

# Implementing with Context

`/implement Feature Name`

*Press F to toggle fullscreen*

# The Feedback Loop

1

Analyze codebase (*once, for project setup*)

v

2

Design feature specification

v

3

Team reviews design spec

v

4

Implement

v

5

Architecture docs auto-update

^ Next developer sees current state

Architecture evolves with your code, not after it

LIVE DEMO

# Implementation Results

*Press F to toggle fullscreen*

# AI Assistant Integration

## Claude Code

Supported

`.claude/commands/ slash commands`

```
/analyze /design /implement
```

## Windsurf

Supported

`.windsurf/workflows/ Cascade`

```
/analyze /design /implement
```

## Cursor

Supported

`.cursor/prompts/ custom prompts`

```
@analyze @design @implement
```

## Codex

Coming Soon

Custom prompts + bash

Same prompts, same SpecMind CLI under the hood

# Multi-Language Support

Because we use tree-sitter, we get multi-language support

<b>TypeScript</b> Supported	<b>JavaScript</b> Supported	<b>Python</b> Supported	<b>C#</b> Supported
<b>Go</b> Planned	<b>Rust</b> Planned	<b>Java</b> Planned	<b>C++</b> Planned

Tree-sitter supports 50+ languages -- great area for contribution

# Before & After

## Without SpecMind

- x Each dev's AI makes isolated decisions
- x Architecture drifts
- x Catch problems too late in code review

VS

## With SpecMind

- / AI has full architectural context
- / Designs reviewed before coding
- / Architecture stays consistent
- / Docs evolve with code automatically

Keep the speed, lose the chaos

# Get Involved

[github.com/specmind/specmind](https://github.com/specmind/specmind)

Star the repo

Try it on your project

Contribute language support

Create issues on GitHub



```
npx specmind setup
```

Works with Claude Code, Windsurf, and Cursor today