IMEN574/IDSC574 Programing for Data Science

3. **Coding with AI**

Spring 2026

Instructor: **Sangwoo Mo**

Vibe Coding



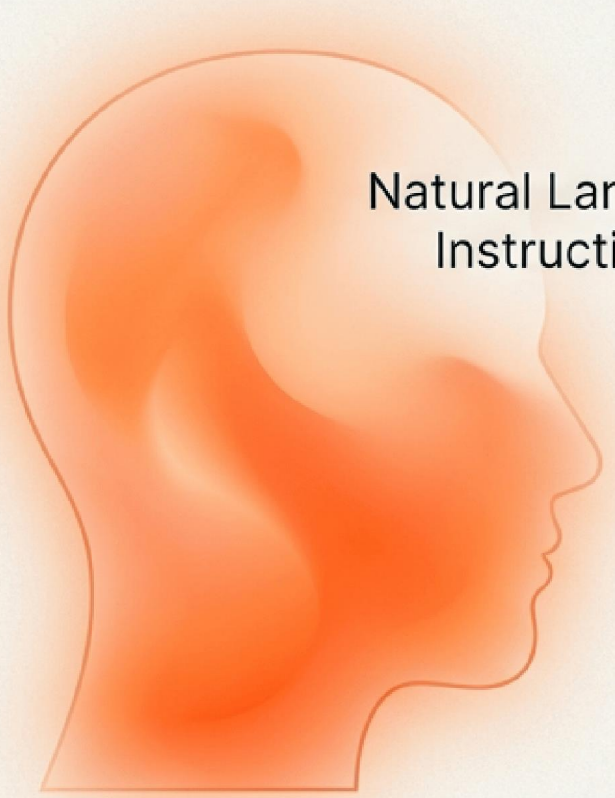
Andrej Karpathy 
@karpathy

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.

12:17 AM · Feb 3, 2025 · **4.1M** Views

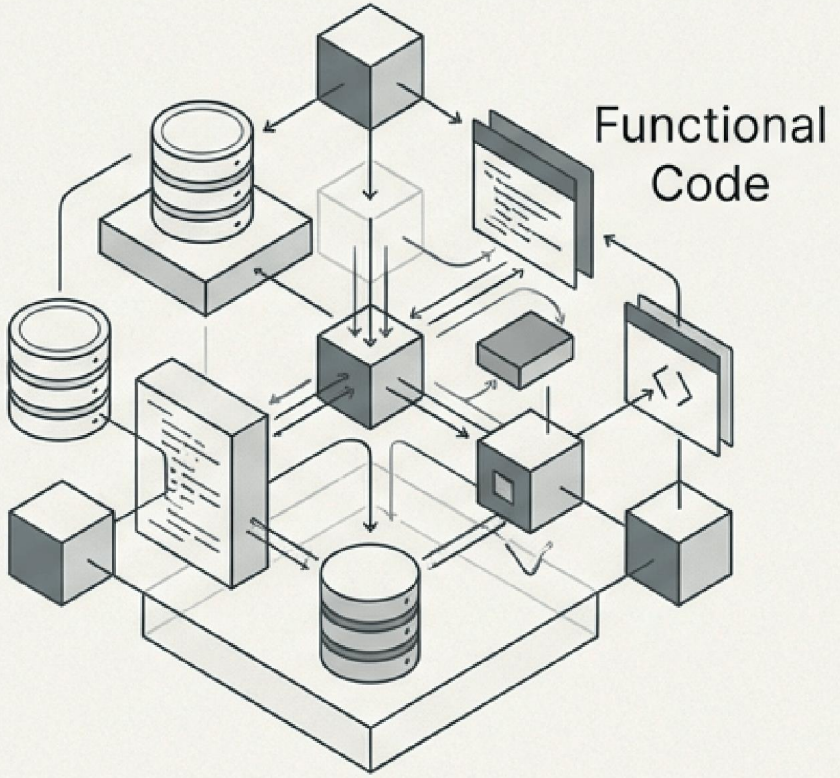
What We Mean by Vibe Coding

Input



Natural Language Instructions

Output



Functional Code

You focus on the WHAT. The AI handles the HOW.

NotebookLM



The Market Has Already Decided

of the Y Combinator 2025 startup cohort run on codebases that are almost entirely AI-generated.

IBM Google Microsoft WIX

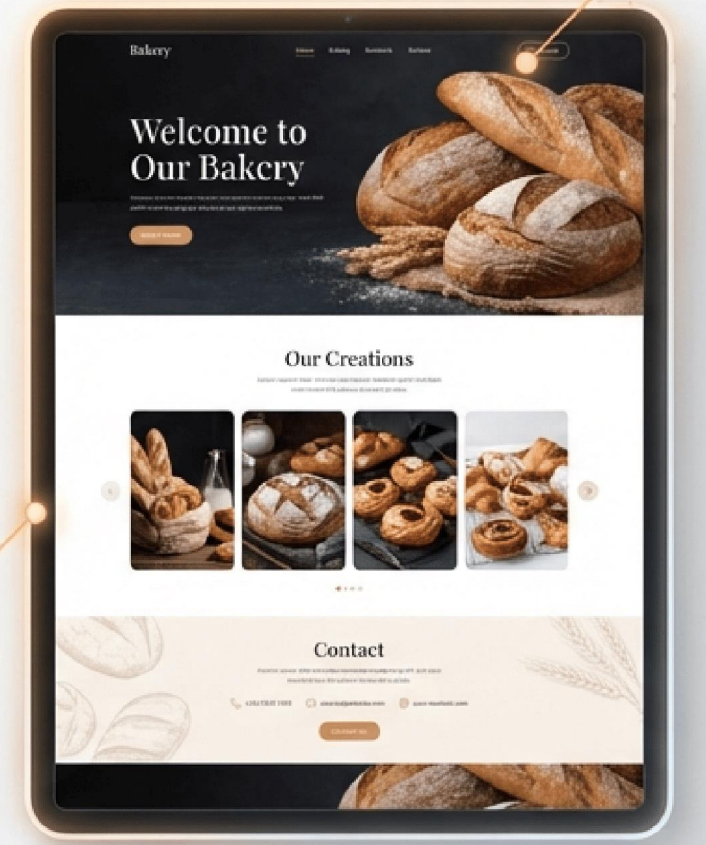
NotebookLM

The New Creative Standard

Plain English

I need a landing page for my bakery.

Focus on UX



Instant Prototype

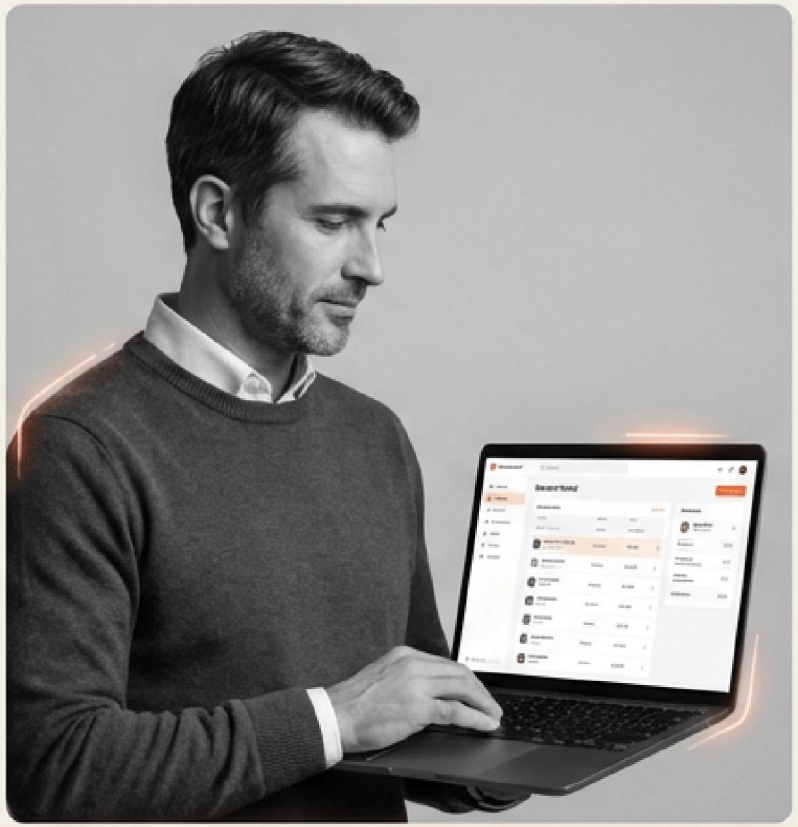
NotebookLM

Who is Vibe Coding For?

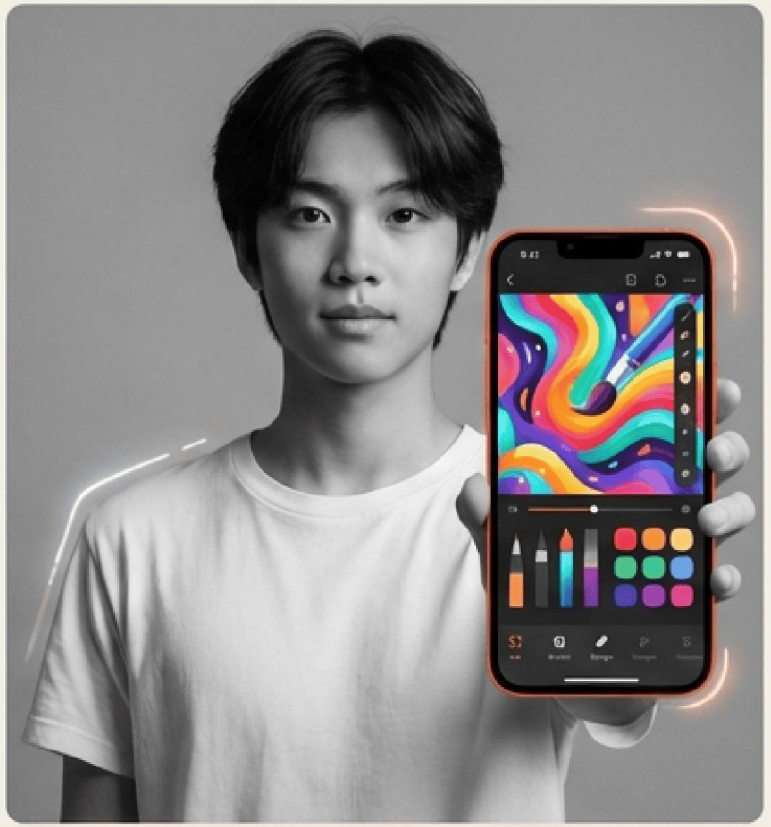
Removing the technical co-founder bottleneck.



The Entrepreneur



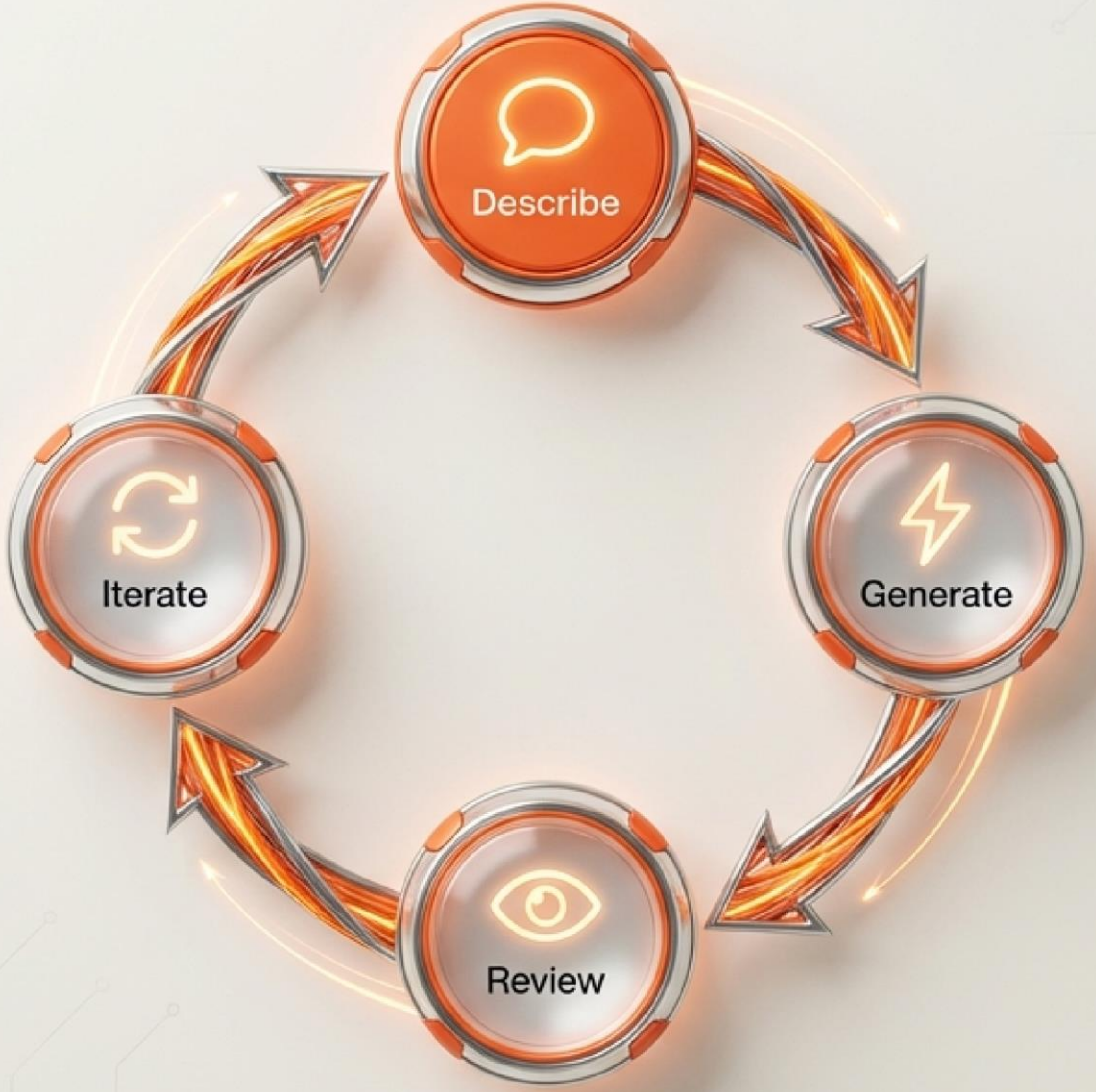
The Educator



The Designer

NotebookLM

The Cycle of Creation



NotebookLM

The Art of the Prompt

The Wrong Way

```
Make it better.
```



The Right Way

```
Change the button color to a darker
shade and make the chart interactive.
```



Vague inputs yield vague outputs. Be specific.

NotebookLM

Best Practices for Builders



Start Small: Core functionality first.



Provide Context: Explain user scenarios.



New Chats: Reset if you get stuck.



Checkpoints: Use version control.

NotebookLM

The Shift: From Chatbots to Reasoning Agents

The Old Way (Chatbot)



- Conversational & Polite
- Verbose Output
- Role-Playing Personas
- Markdown / JSON Syntax

The New Way (Agentic)



- Structural & Logical
- Concise Execution
- Domain-Specific Expertise
- XML Tag Syntax

Insight: Gemini 3 and Claude are planning engines. They Plan -> Execute -> Verify. Treating them like chat partners degrades performance.

NotebookLM

Vibe Coding & Visual POCs

Prompt Input

> Create an interactive map interface for a food discovery app. Show clusters of restaurants in different neighborhoods with color-coded markers based on cuisine type (e.g., Italian, Mexican, Asian). Allow zooming and clicking on clicking on a cluster to reveal individual locations with ratings and price range. Ensure a modern, dark mode aesthetic with clean fonts.

Preview (Artifact)



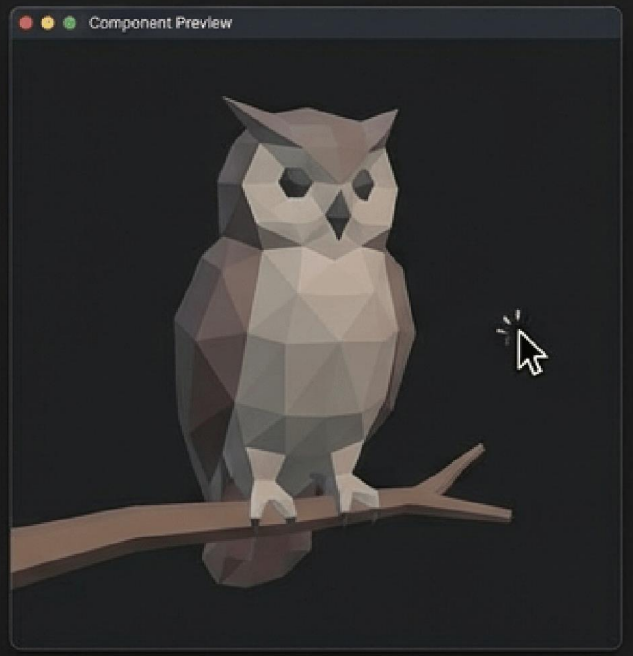
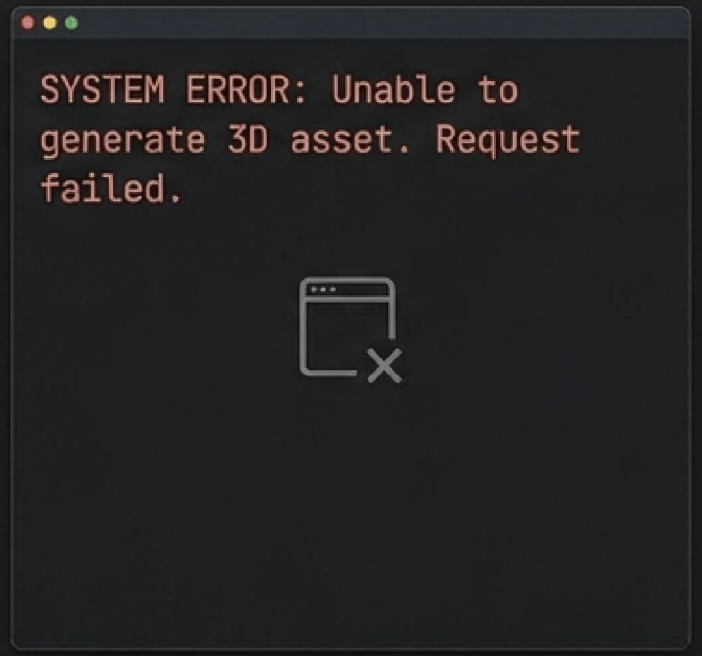
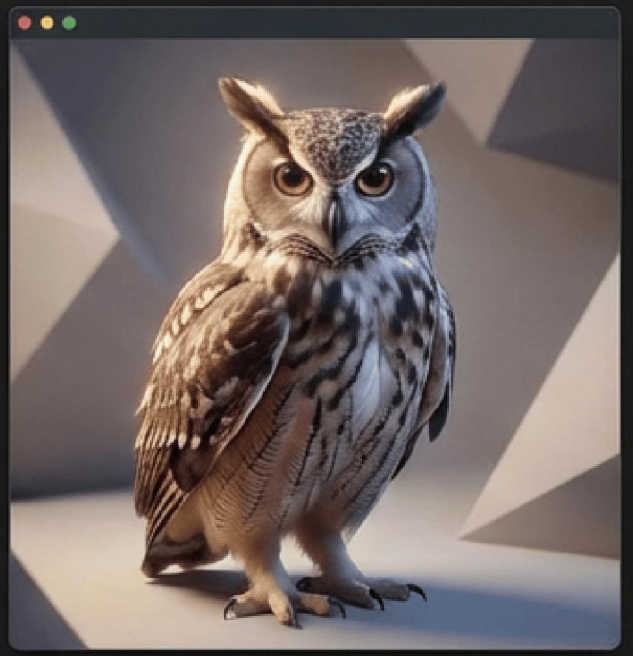
Definition: Vibe Coding

The use of LLMs to instantly generate functional UI/UX prototypes (Proof of Concepts) from natural language. It prioritizes the 'feel' and interactivity of the application over backend production readiness.

NotebookLM

Case Study: The 3D Owl

Prompt: “Create a 3D interactive Owl”

Gemini 3	GPT-4o	Claude Opus
		
Functional & Interactive. Reacts to mouse movement.	**Failed**. Could not generate the 3D asset.	**Aesthetic Superiority**. Best for complex CSS/Three.js rendering.

NotebookLM

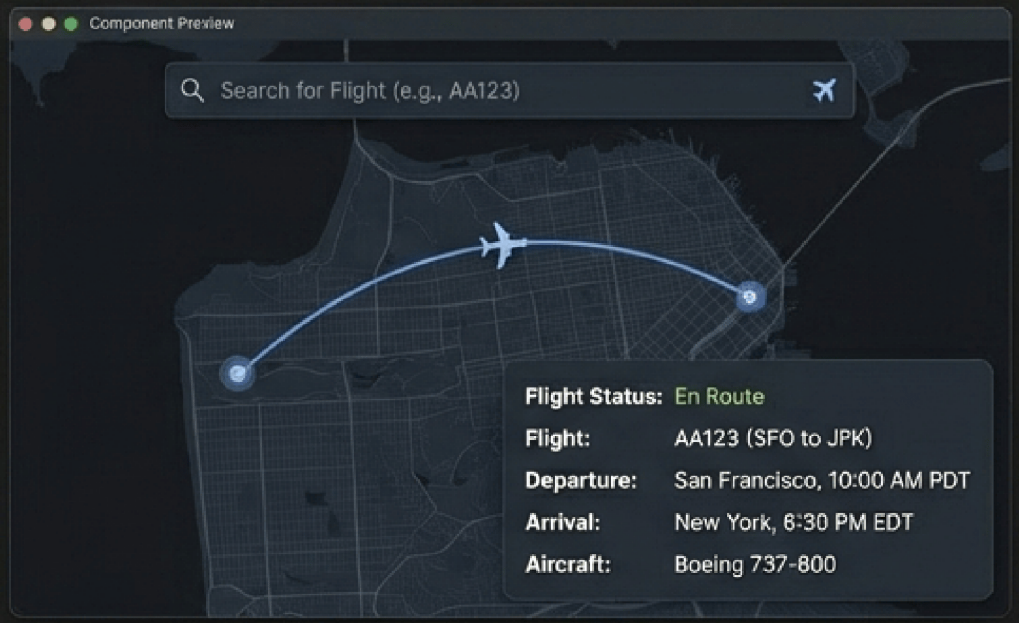
Case Study: Flight Tracker UI

Building an app with XML Prompts

The Prompt Structure

```
<main_task>Flight Tracker App POC
</main_task>
<step_1>Create search interface
(Destination, Flight #)</step_1>
<step_2>Visualize route on map</step_2>
<step_3>Mock data for demo
purposes</step_3>
```

The Result



The Anti-Gravity Advantage: Live Preview allows instant iteration.
It turns the LLM from a chatbot into a coding partner.

NotebookLM

Rule #1: Kill the Fluff

Gemini 3 performance degrades with verbosity.

Diff Check

```
001 002   [-] ~~Please could you be so kind as to look at this file and tell me the most important dates?~~
    002   [+] Analyze attached PDF. Extract key dates.
```

The Paradox: Unlike earlier models that thrived on long context, Gemini 3 treats polite conversational filler as **'noise'** in the token context.

Action: Remove articles (a, an, the) and polite phrases. Use **command-line style instructions**.

NotebookLM

Rule #2: The XML Revolution

```
<task>Analyze Data</task>
<context>2024 Sales Report.csv</context>

<constraints>
    - Output format: Table
    - Focus: Q4 Trends
</constraints>
```

Why XML?

It reduces ambiguity. Tags clearly delineate where an instruction starts and stops, preventing the model from “guessing” the boundaries of your prompt.

Shift from Markdown (# Headers) to explicit XML tagging for higher adherence.

NotebookLM

Rule #3: Domain over Role-Play

❌ The Persona Trap

```
You are a 10-year veteran data analyst with a friendly personality...
```

Analysis: Assigning personality traits causes the model to prioritize tone over factual accuracy.

✅ The Domain Definition

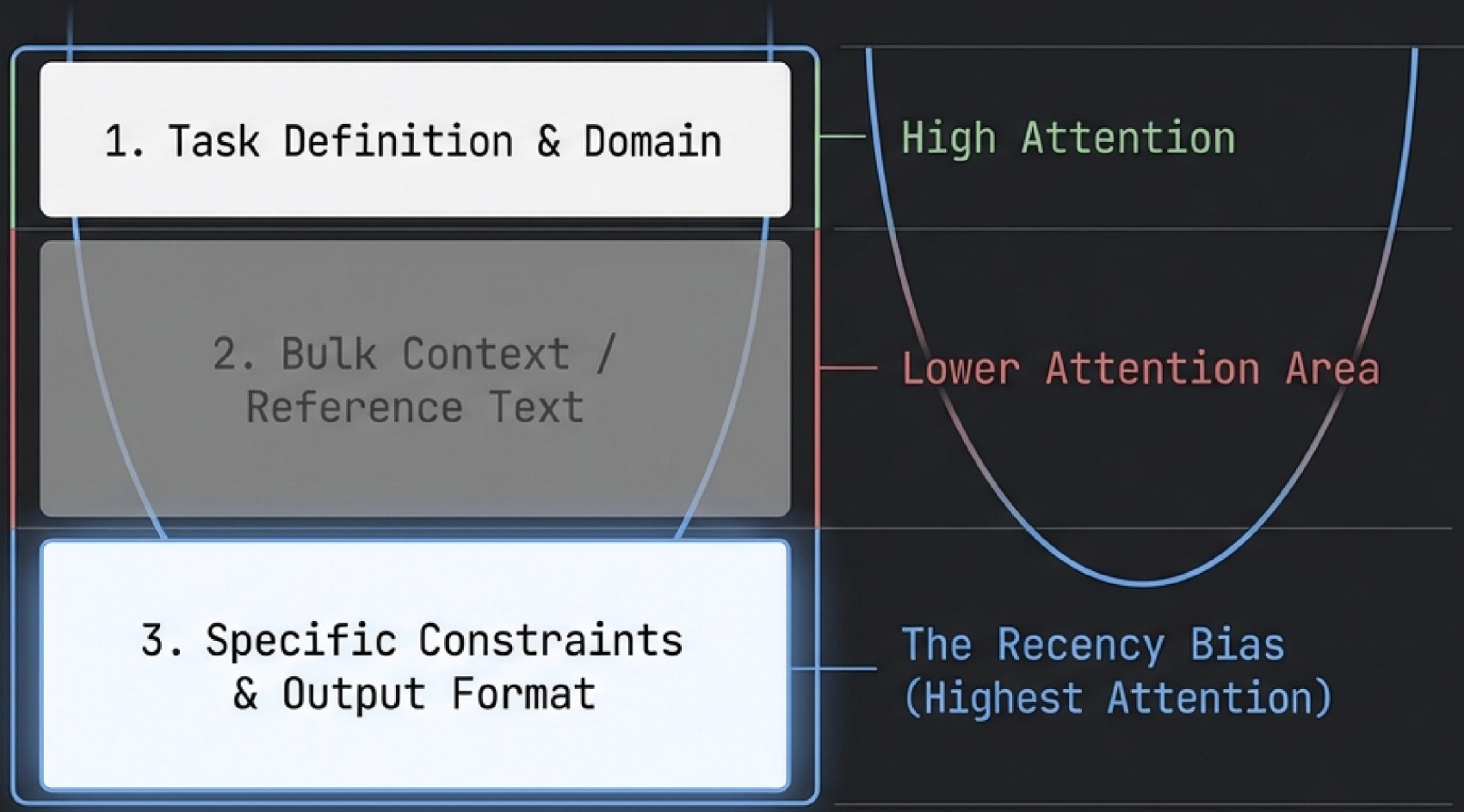
```
Domain: Data Science / Time-Series Analysis
```

Analysis: Defining the field of expertise grounds the model in the correct knowledge base without tonal distortion.

NotebookLM

Rule #4: The Golden Position

Combating “Lost in the Middle” Phenomenon



Protocol: Never paste source text *after* your constraints.

Always remind the model of the rules at the very end.

NotebookLM

Rule #5: The Verification Loop

Embedding quality control into the prompt.




```
<thinking_process>
      <step_1>Plan the analysis logic.</step_1>
      <step_2>Execute data extraction.</step_2>
      <step_3>VERIFY: Check if result meets <constraints>.</step_3>
      <step_4>Generate Final Answer with Line Citations.</step_4>
</thinking_process>
```

Impact: Forcing the model to explicitly 'check its work' reduces hallucinations significantly.

NotebookLM

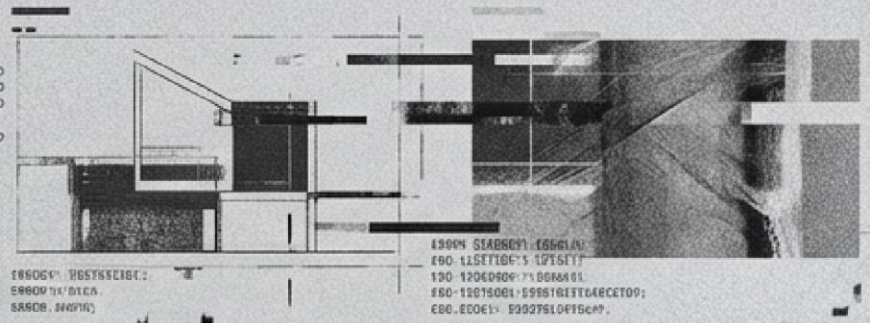
How We Vibe Code at a FAANG

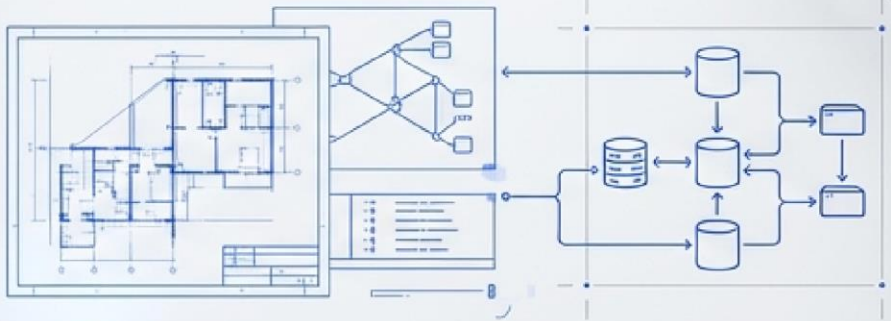
A 7-Step Protocol for AI-Augmented Production Engineering

JetBrains Mono

Source: r/vibecoding | Validated by Senior AI SWE

NotebookLM





THE MYTH

AI coding is junk. It cannot be used for production.



> I've seen a lot of flak coming from folks who don't believe AI-assisted coding can be used for production code. This is simply not true.

THE REALITY

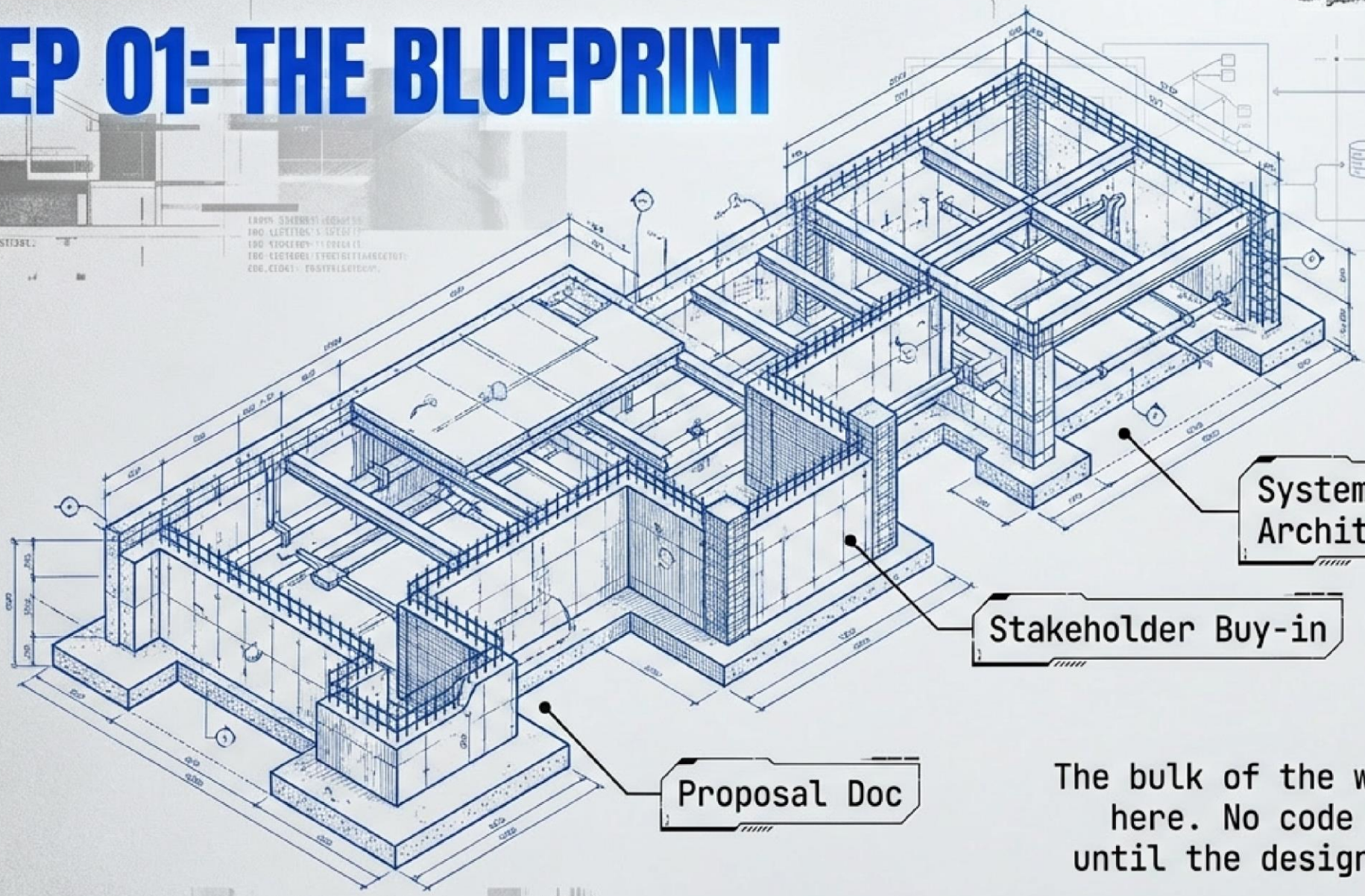
AI is a Force Multiplier. But only with Architecture.



DATE: 2024.18.27 // LATITUDE: 34.8522° N // LONGITUDE: 118.2437° W // SYSTEM STATUS: OPTIMIZED

NotebookLM

STEP 01: THE BLUEPRINT



The bulk of the work happens here. No code is written until the design is solid.

NotebookLM

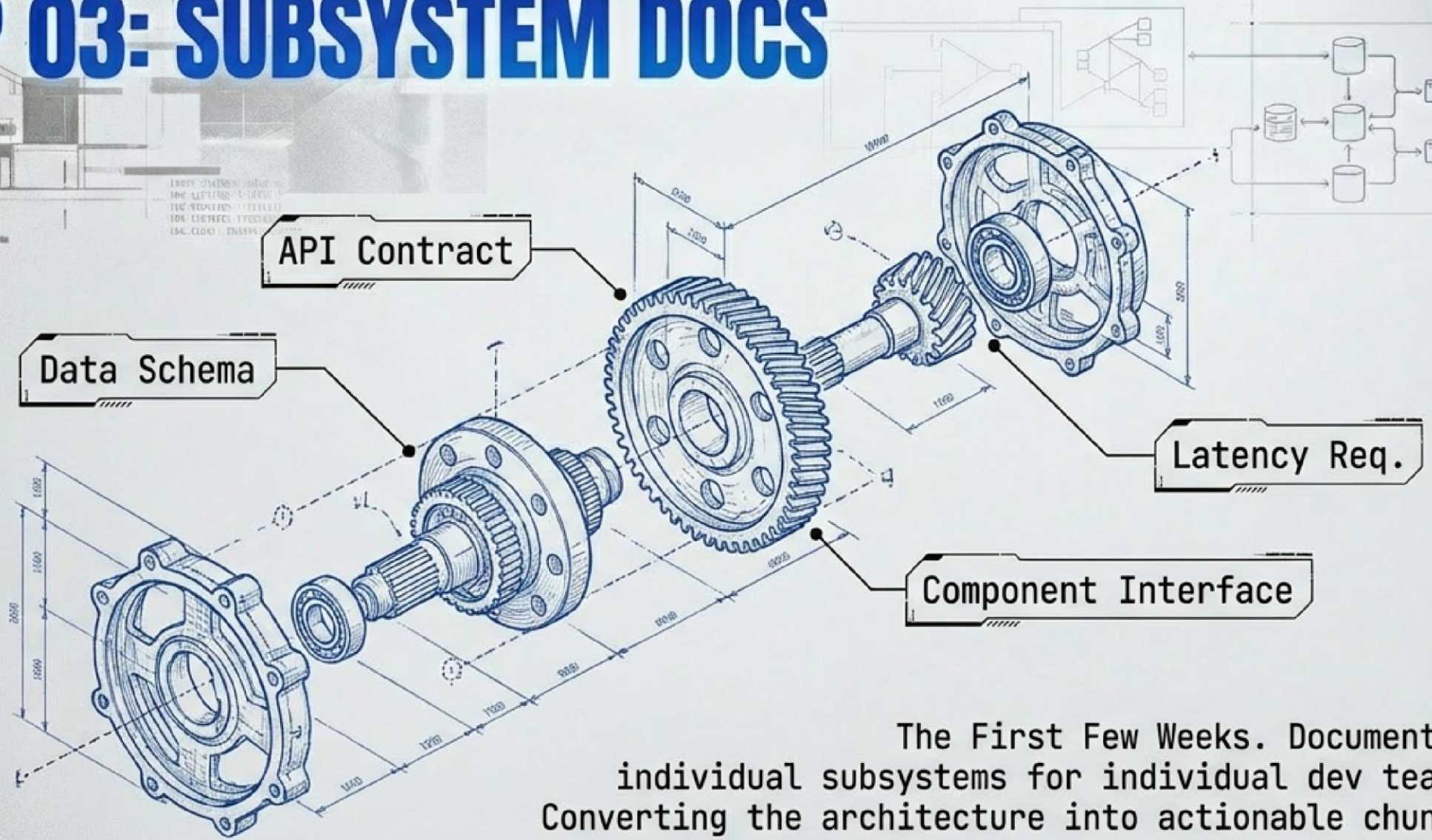
STEP 02: FRONT-LOADING THE PAIN



The Design Review. Your team's design doc gets absolutely shredded by Senior Engineers. This is good. This is the filter.

NotebookLM

STEP 03: SUBSYSTEM DOCS



The First Few Weeks. Documenting individual subsystems for individual dev teams. Converting the architecture into actionable chunks.

NotebookLM

STEP 04: THE BACKLOG



Sprint Planning. Hammering out discrete tasks. Precision ordering before execution.

NotebookLM

STEP 05: HANDS ON KEYBOARD

THE AI FORCE MULTIPLIER

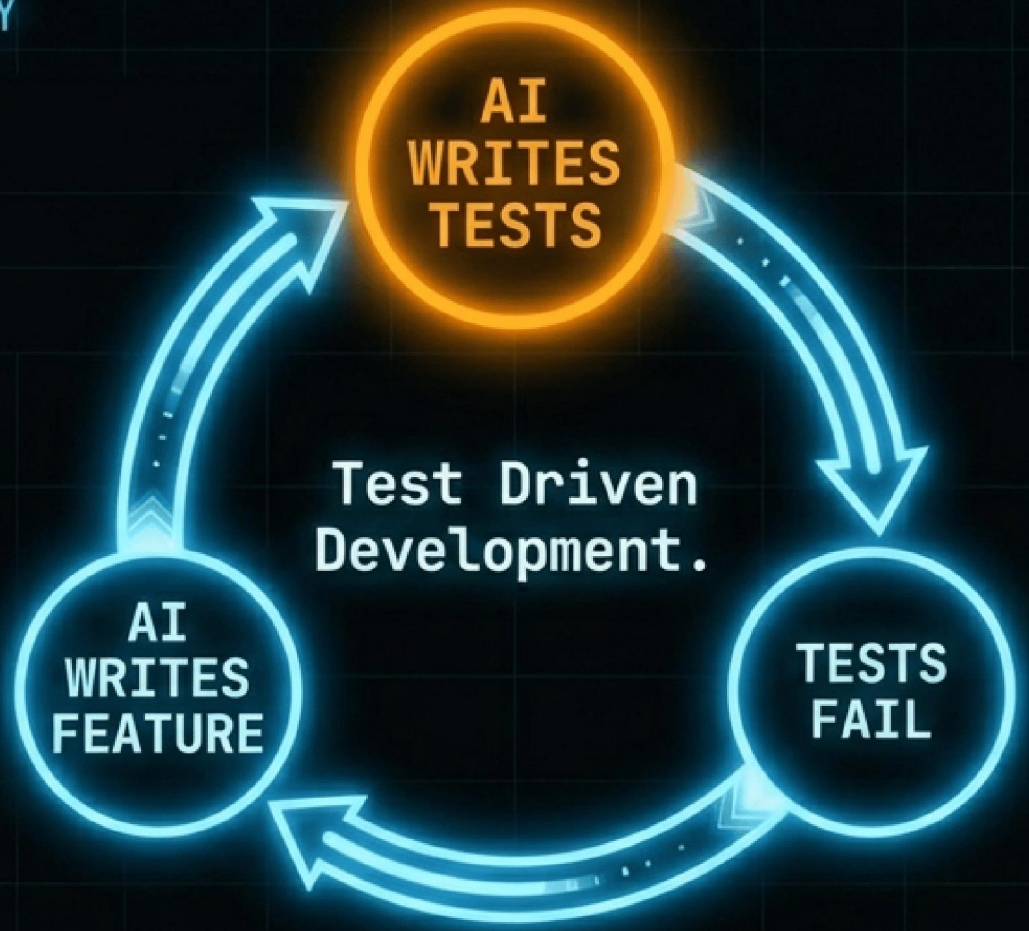


JetBrains Mono / Helvetica Now Display
Execution phase. Actively coding and implementing tasks. AI tools significantly increase speed and output.

NotebookLM

THE SECRET SAUCE: TDD

HELVETICA NOW DISPLAY



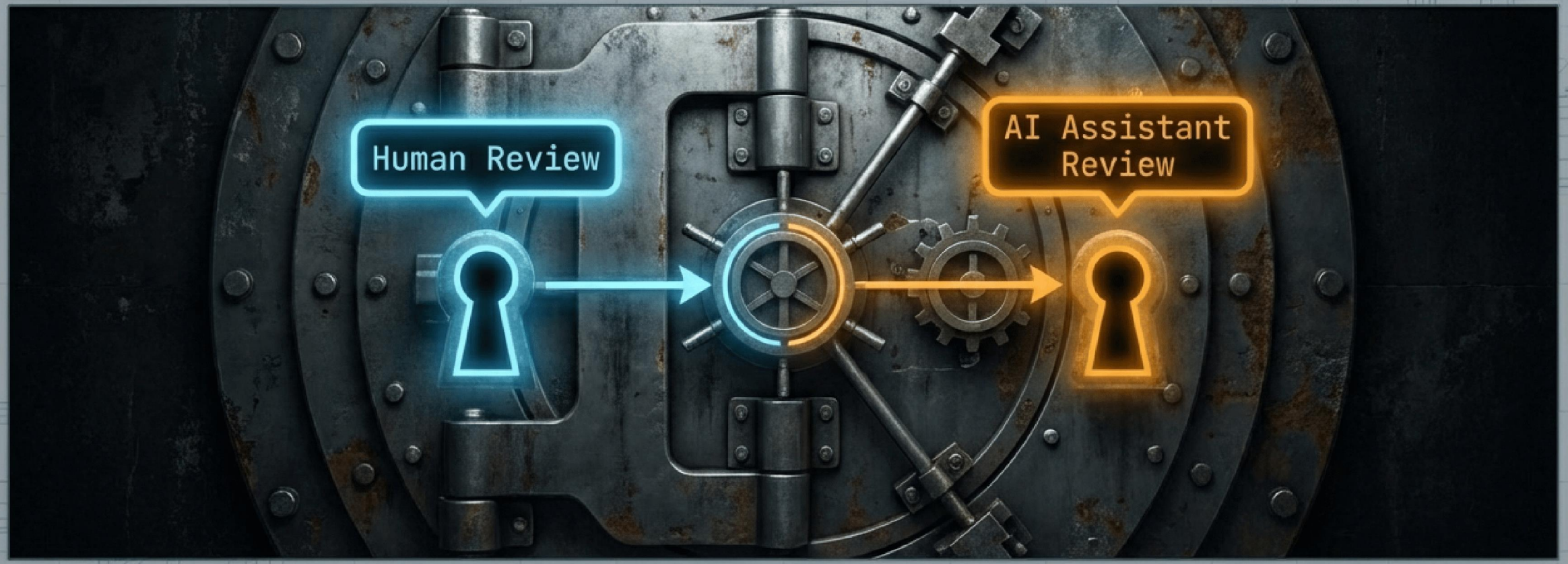
We do not build the feature until the AI has written the test.

TECHNICAL BRUTALISM MEETS SWISS PRECISION // DATE: 2024.10.27 //
DOSSIER STATUS: CLASSIFIED // SYSTEM STATUS: OPTIMIZED // TERMINAL MODE ENGAGED

NotebookLM

STEP 06: THE TWO-DEV APPROVAL

THE AI FORCE MULTIPLIER



Code Submission.
Two distinct approvals required before merging to main.
AI assists, but humans decide.

NotebookLM

STEP 07: STAGING TO PROD

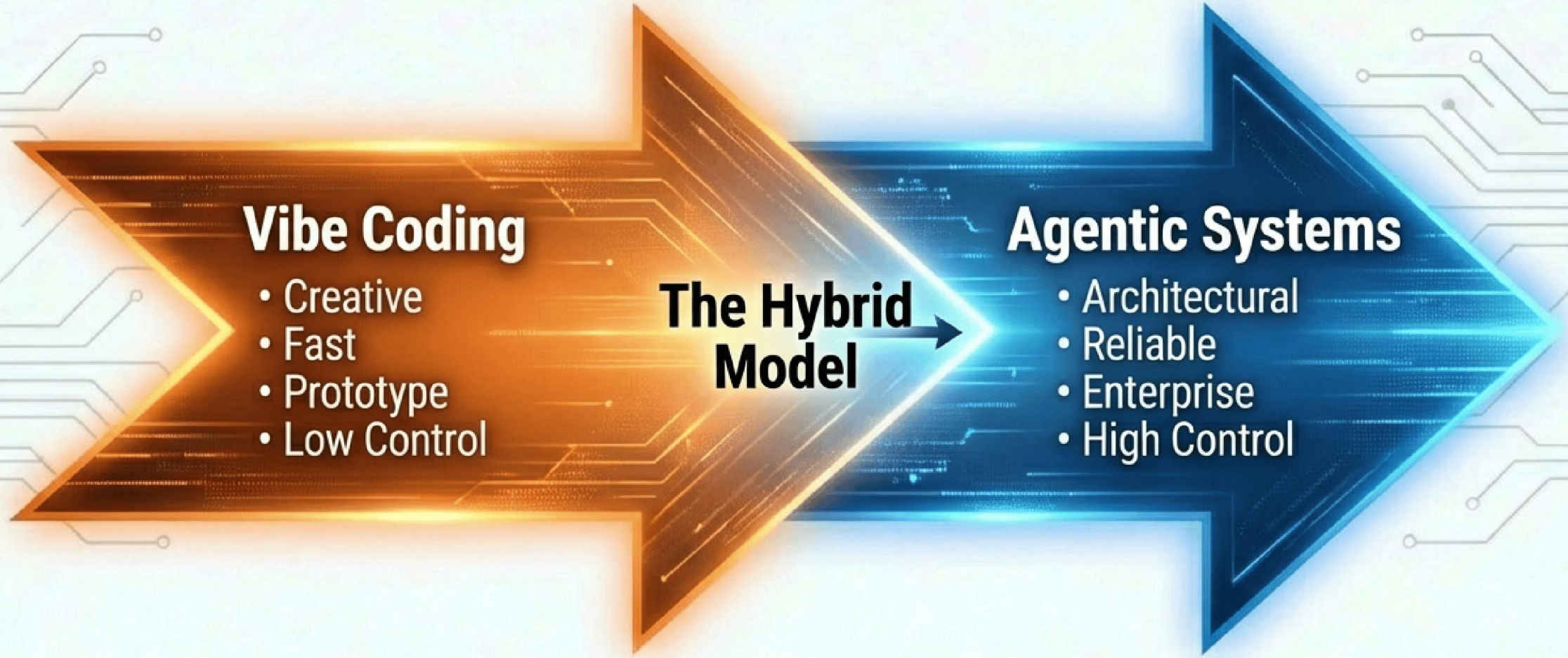
HELVETICA NOW DISPLAY



The Final Mile.
If staging is good, we ship.

NotebookLM

The Shift: Vibe to Agentic

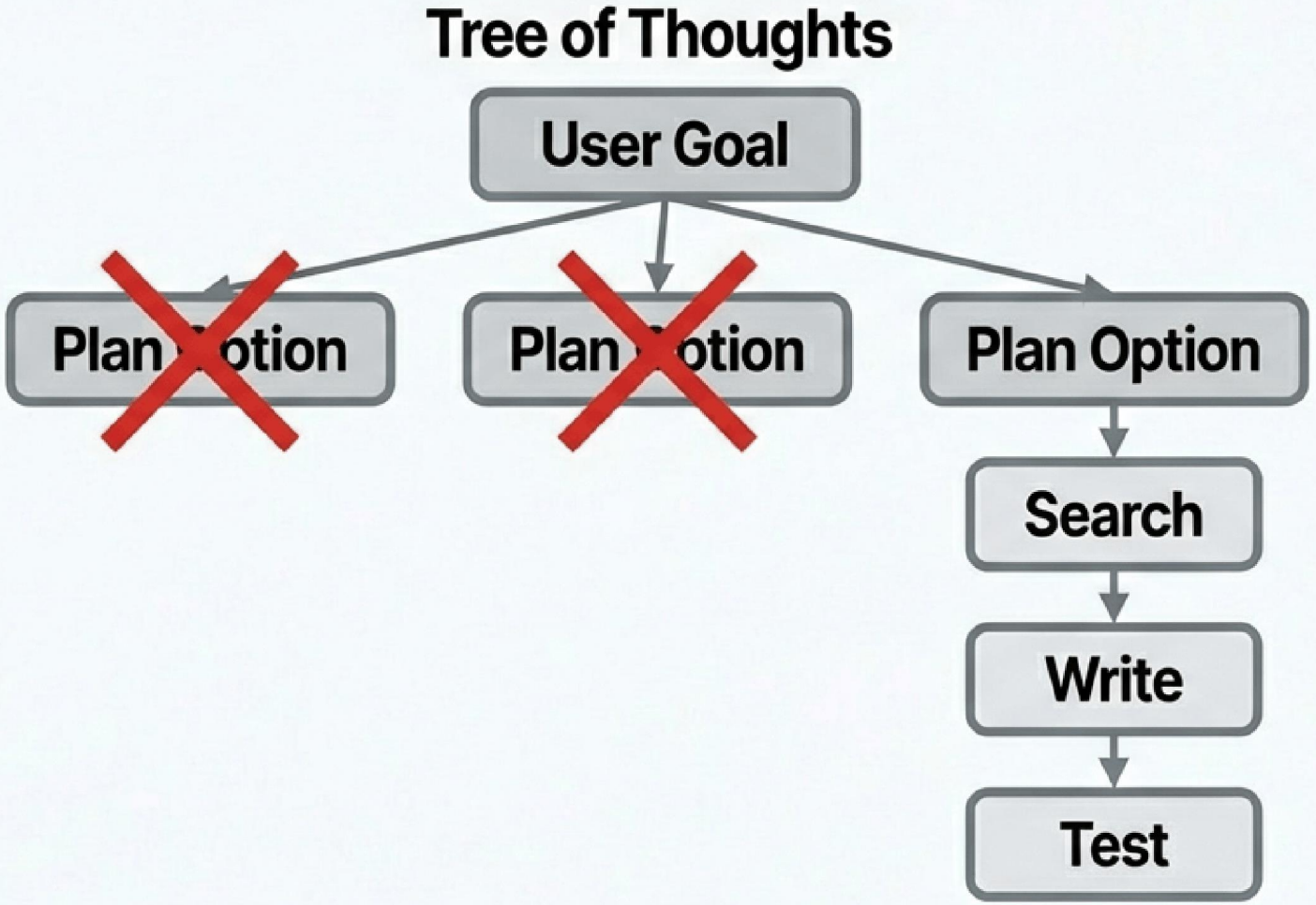


NotebookLM

Planning: Thinking Before Coding

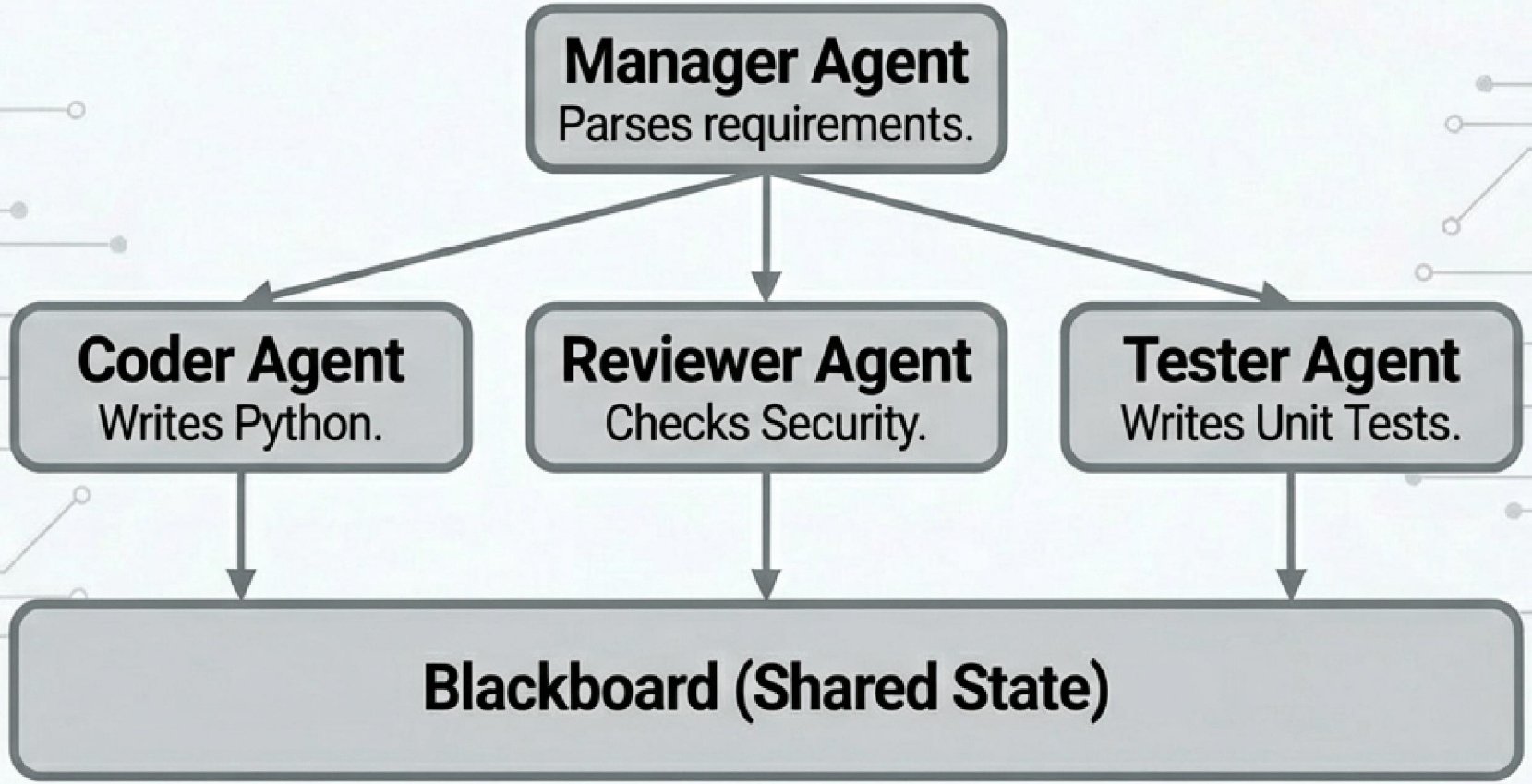


Solves the "Next Token" problem.
Agents simulate execution paths before writing code (MCTS / Self-Planning).



NotebookLM

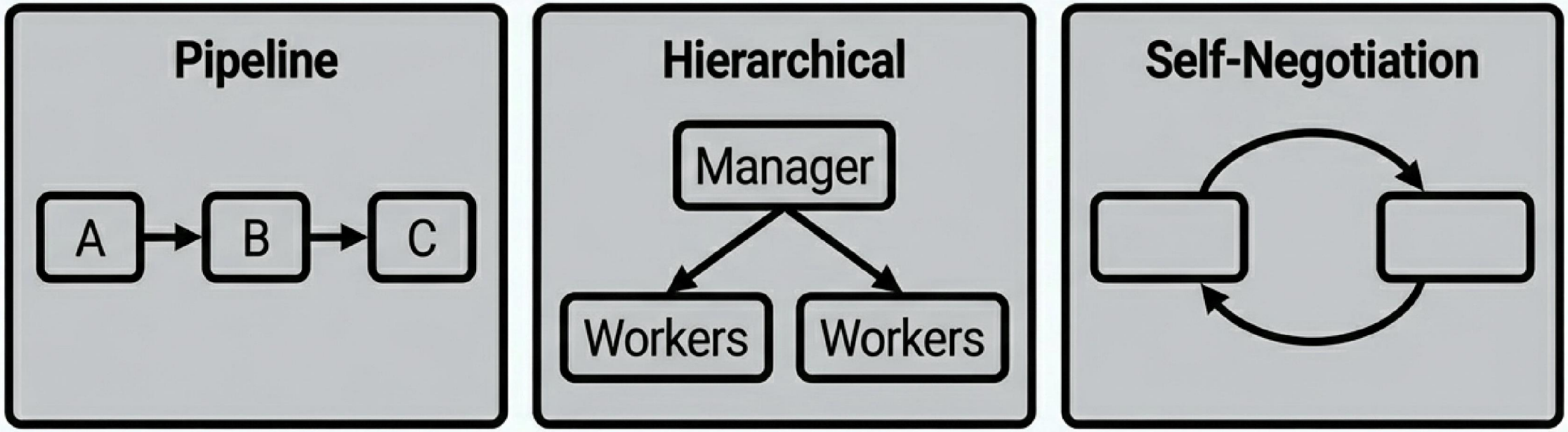
Scaling Up: Multi-Agent Systems



Role-Playing: Specialized agents collaborating in a shared memory space.

NotebookLM

Workflow Paradigms

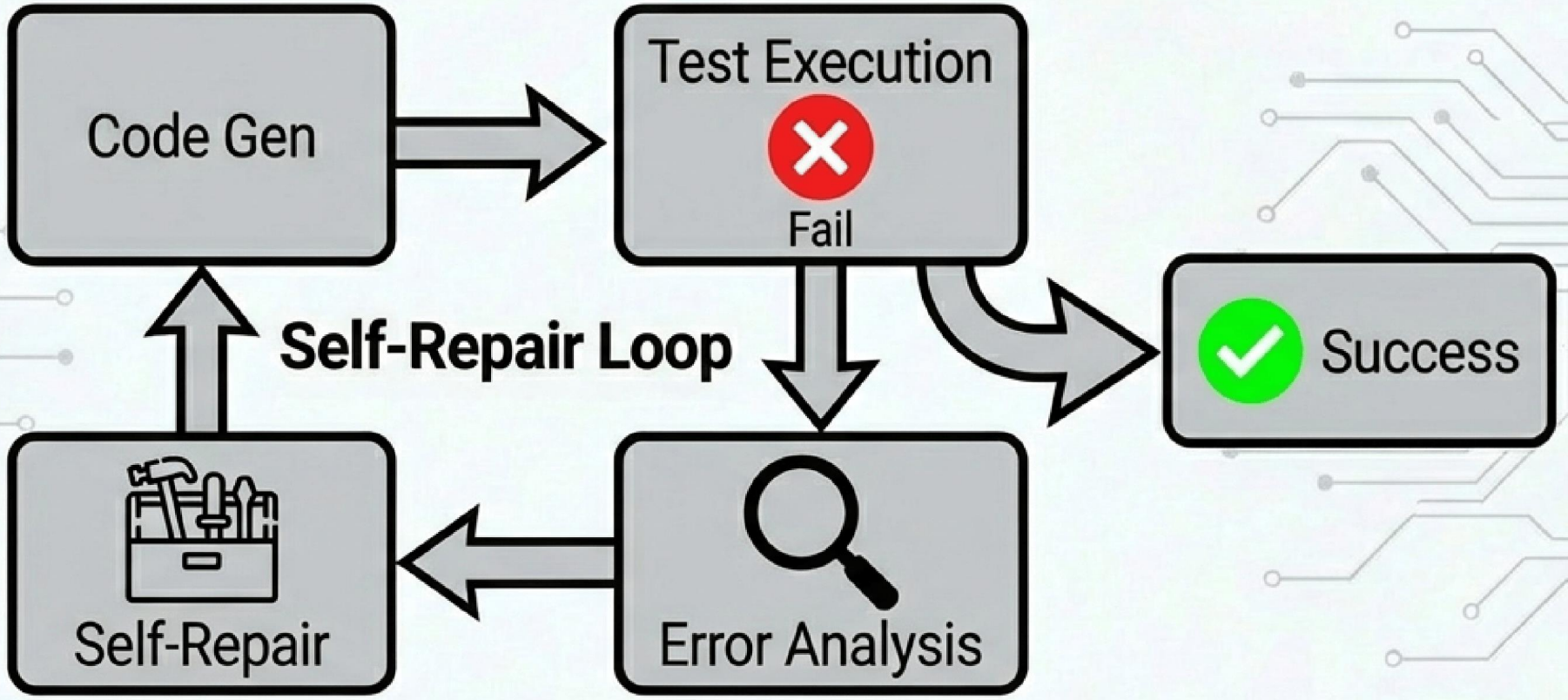


Waterfall | Self-Organized | Consensus/ChatDev

Moving from linear hand-offs to dynamic negotiation loops to reduce hallucinations.

NotebookLM

The Killer Feature: Self-Correction



Agents execute their own code, read the error message, and fix the bug without human intervention (AutoCodeRover).

NotebookLM

The Tool Landscape

Co-Pilot	Collaborator	Autonomous Team
		Emerging Frontier
• Examples: GitHub Copilot, Tongyi Lingma • Feature: IDE Extension	• Examples: Cursor • Feature: AI-Native IDE, Local Context	• Examples: Devin, Claude Code • Feature: End-to-End, Sandbox Execution

NotebookLM

The Tool Landscape

Co-Pilot	Collaborator	Autonomous Team
		Emerging Frontier
• Examples: GitHub Copilot, Tongyi Lingma • Feature: IDE Extension	• Examples: Cursor • Feature: AI-Native IDE, Local Context	• Examples: Devin, Claude Code • Feature: End-to-End, Sandbox Execution

4. LLM Prompting 5. Modern IDE 6. Coding Agents

NotebookLM