Zero to Production

A GIS Professional's Process for Building Real Apps with AI

Adam Cabrera, Geo Engineering Manager, King County GIS Center

WAGISA Conference | May 20, 2026 | Bellevue, WA

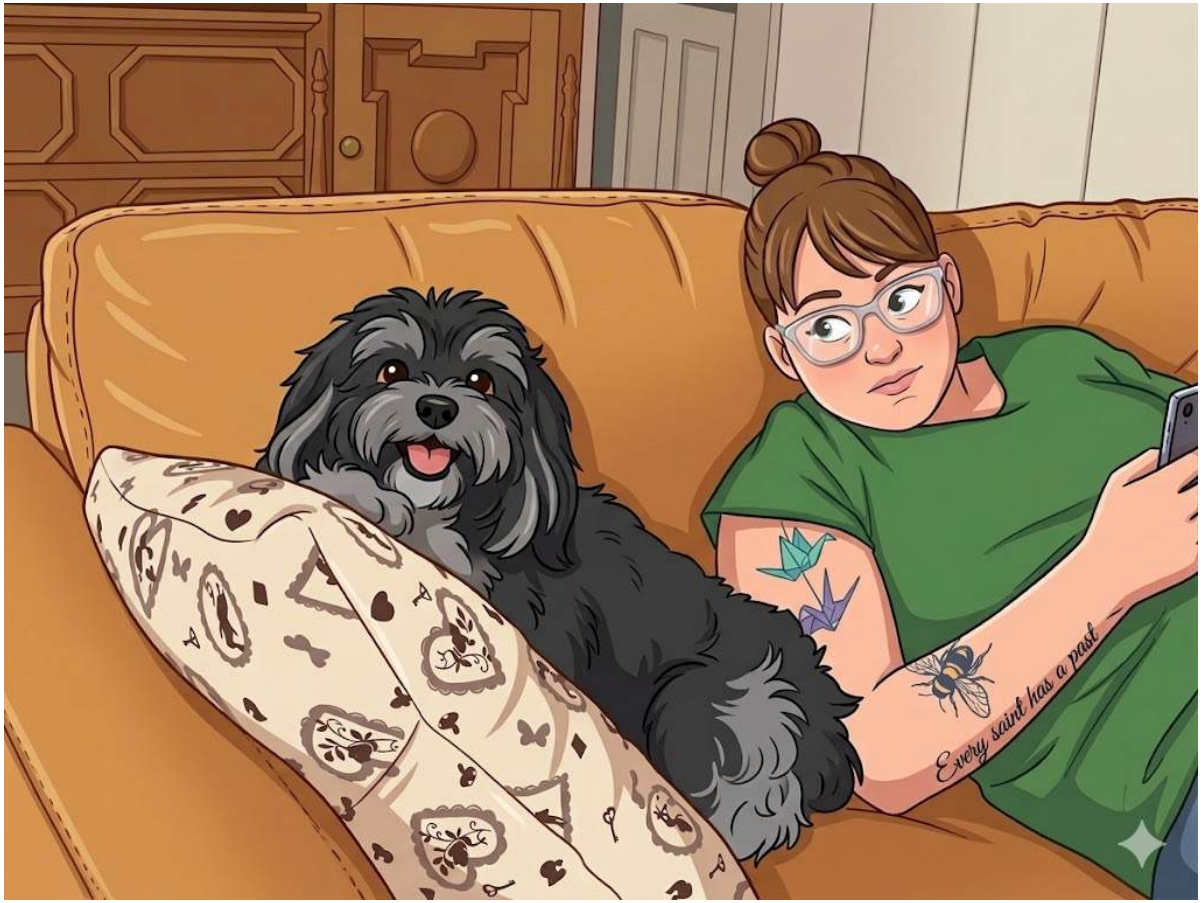
King County
Department of Information Technology

What You're Walking Out With

- 16 practices
  - Something for coders
  - Something for anyone with ideas and domain expertise
- A blog post covers the details for the ones we move through quickly
  - Scan the QR at the end
- Four steps you can take tonight
  - No coding background required

Today's Journey

- **Where I Started**
- What Changed
- The Methodology
- The Proof
- Your Path



About Me

Adam Cabrera

Geo Engineering Manager
King County GIS Center

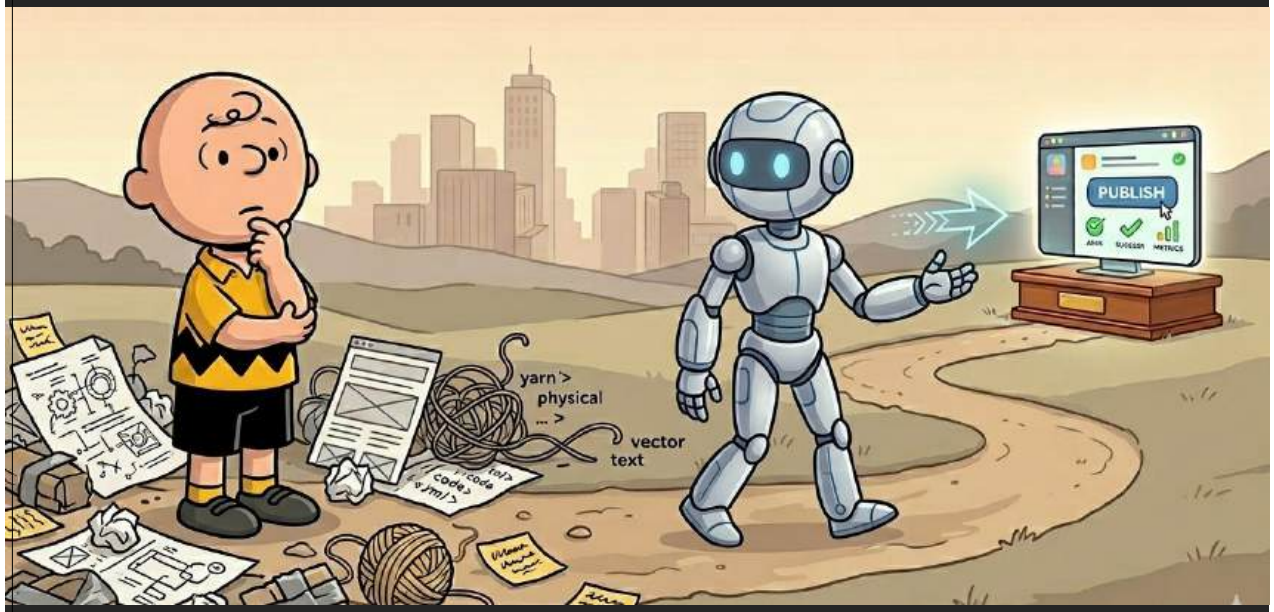
35 years in GIS

Command-line to cloud. Technician to manager.

Team of eight supporting King County's public-facing GIS infrastructure.

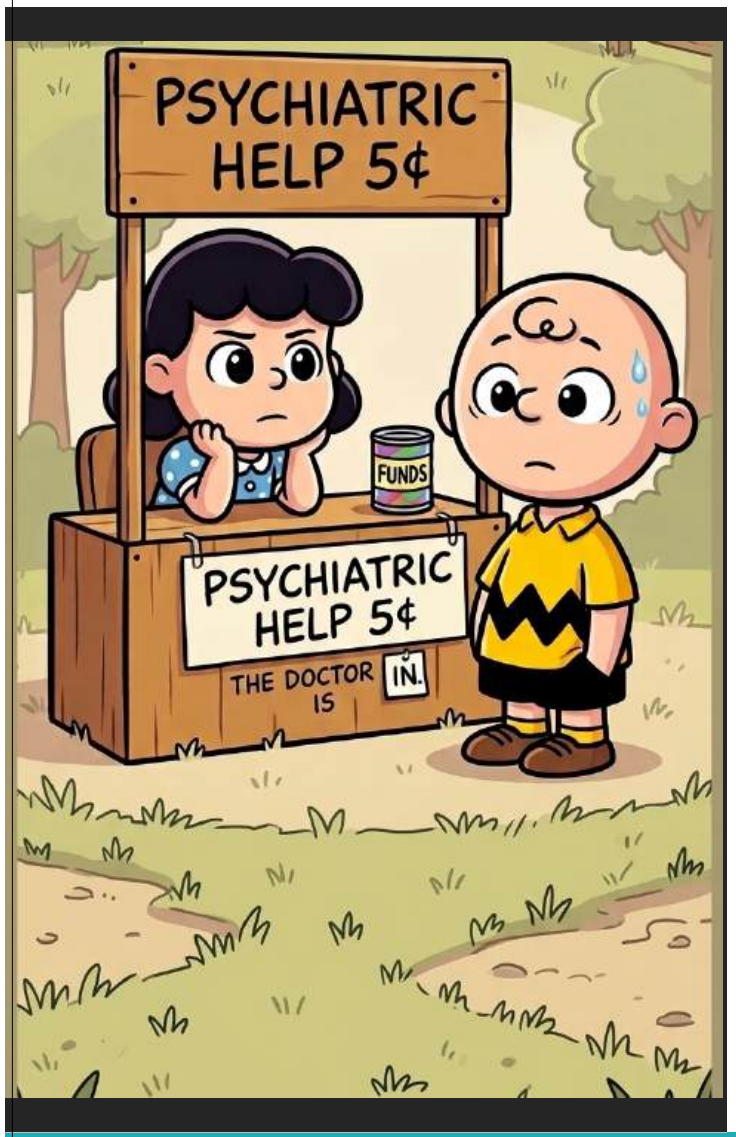
A Manager, Not a Coder

- Production code is not my day job anymore
- From the keyboard into the leadership role
- The craft stopped being what I was paid to do



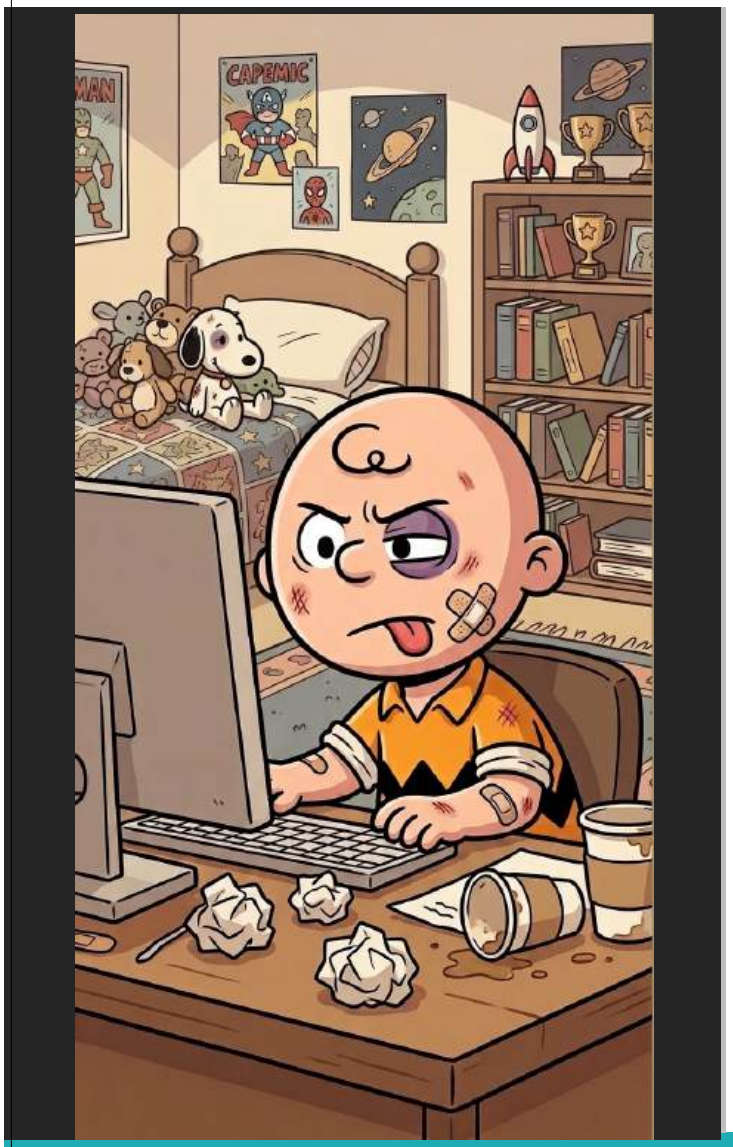
”Using AI, Could I Help My Team?"

- Not thinking about becoming a developer again
- Thinking about my team
- Had to try it myself to answer honestly



The Lucy Problem

- I was using free tools
- Cheap format, shallow result
- The AI tools weren’t qualified
  - Not every tool is up to the task
- They didn't have the complete picture
  - No context, no history, no follow-up
- **Results – Responses were incomplete and sub-par**
- Sound familiar?

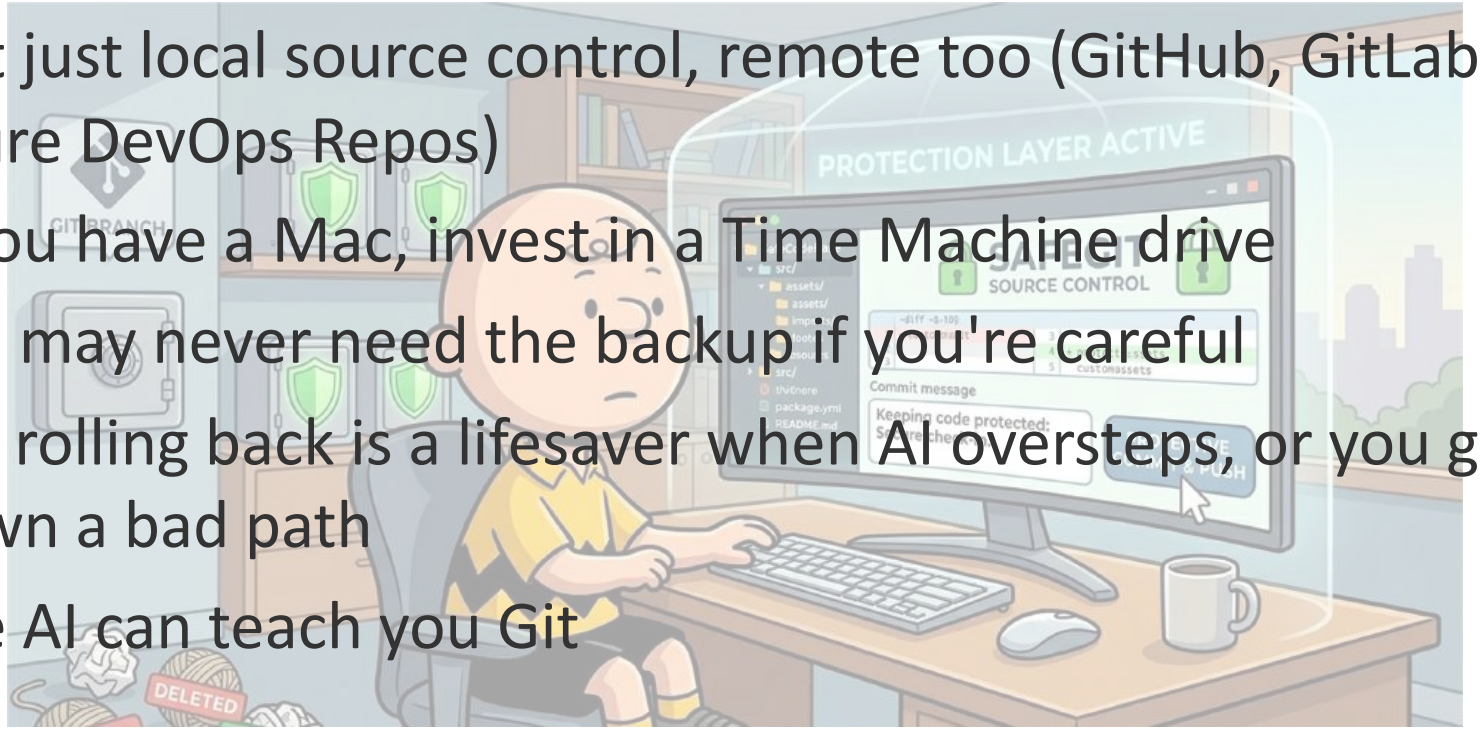


QuerySimple Is Where I Bled

- First real attempt: Custom Esri ExB widget with AI
- Thirty-five years into GIS, zero experience with AI coding
- No methodology and no model I could trust

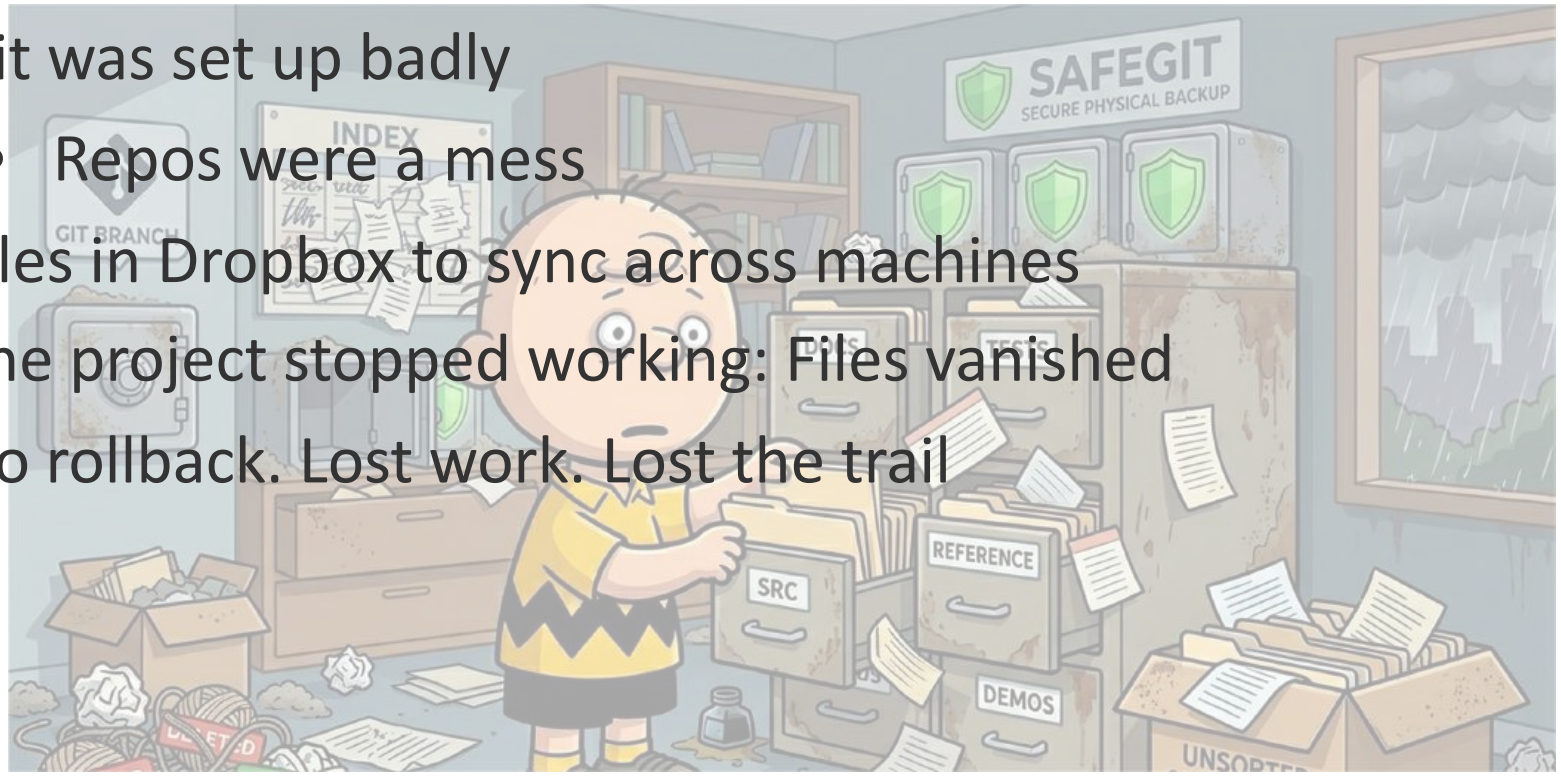
Practice 1: Source Control Is Step Zero

- Not just local source control, remote too (GitHub, GitLab, Azure DevOps Repos)
- If you have a Mac, invest in a Time Machine drive
- You may never need the backup if you're careful
- But rolling back is a lifesaver when AI oversteps, or you go down a bad path
- The AI can teach you Git



My First Hard Lesson Was About Git, Not Prompts

- Git was set up badly
  - Repos were a mess
- Files in Dropbox to sync across machines
- The project stopped working: Files vanished
- No rollback. Lost work. Lost the trail

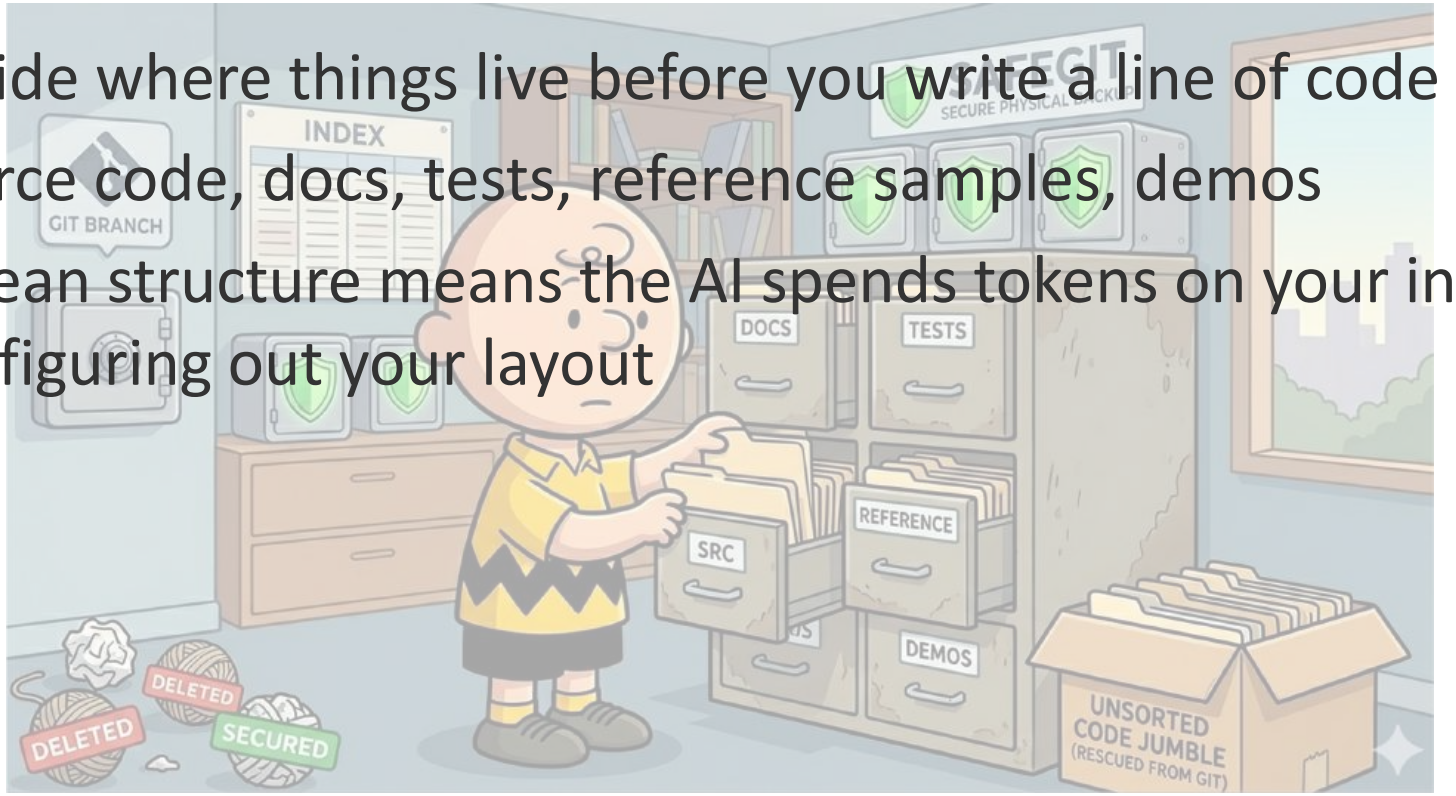


Don't Put Your Code in a Folder That Syncs

- Dropbox, OneDrive, iCloud, Google Drive
- They watch for changes and push updates. So does Git.
- Two systems fighting over the same files. You lose.

Practice 2: Organize Your Project Before You Build

- Decide where things live before you write a line of code
- Source code, docs, tests, reference samples, demos
- A clean structure means the AI spends tokens on your intent, not figuring out your layout



Today's Journey

- **Where I Started**
- What Changed
- The Methodology
- The Proof
- Your Path

Questions?

Today's Journey

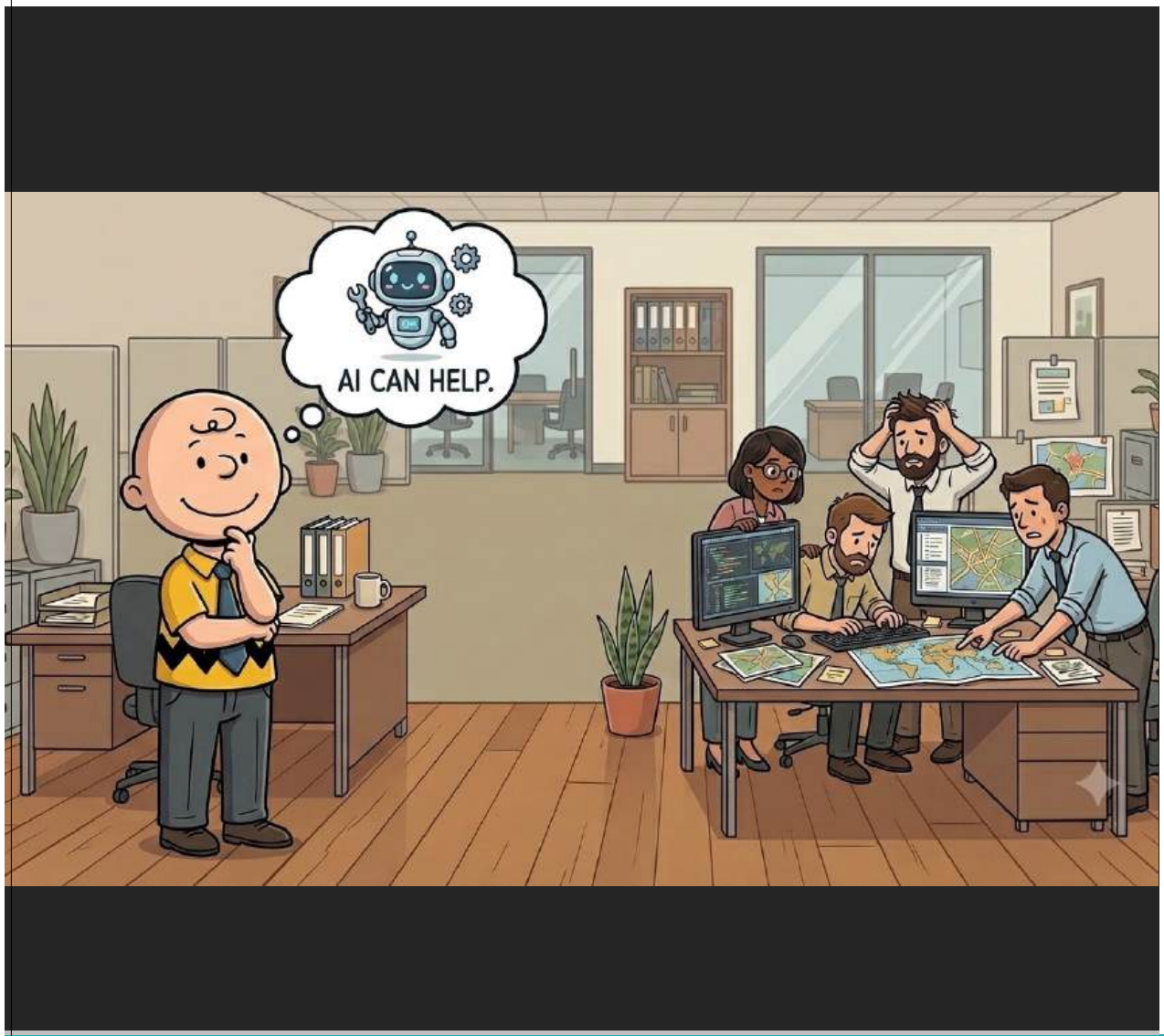
Where I Started

What Changed

The Methodology

The Proof

Your Path



"I Just Wanted to Help My Team"

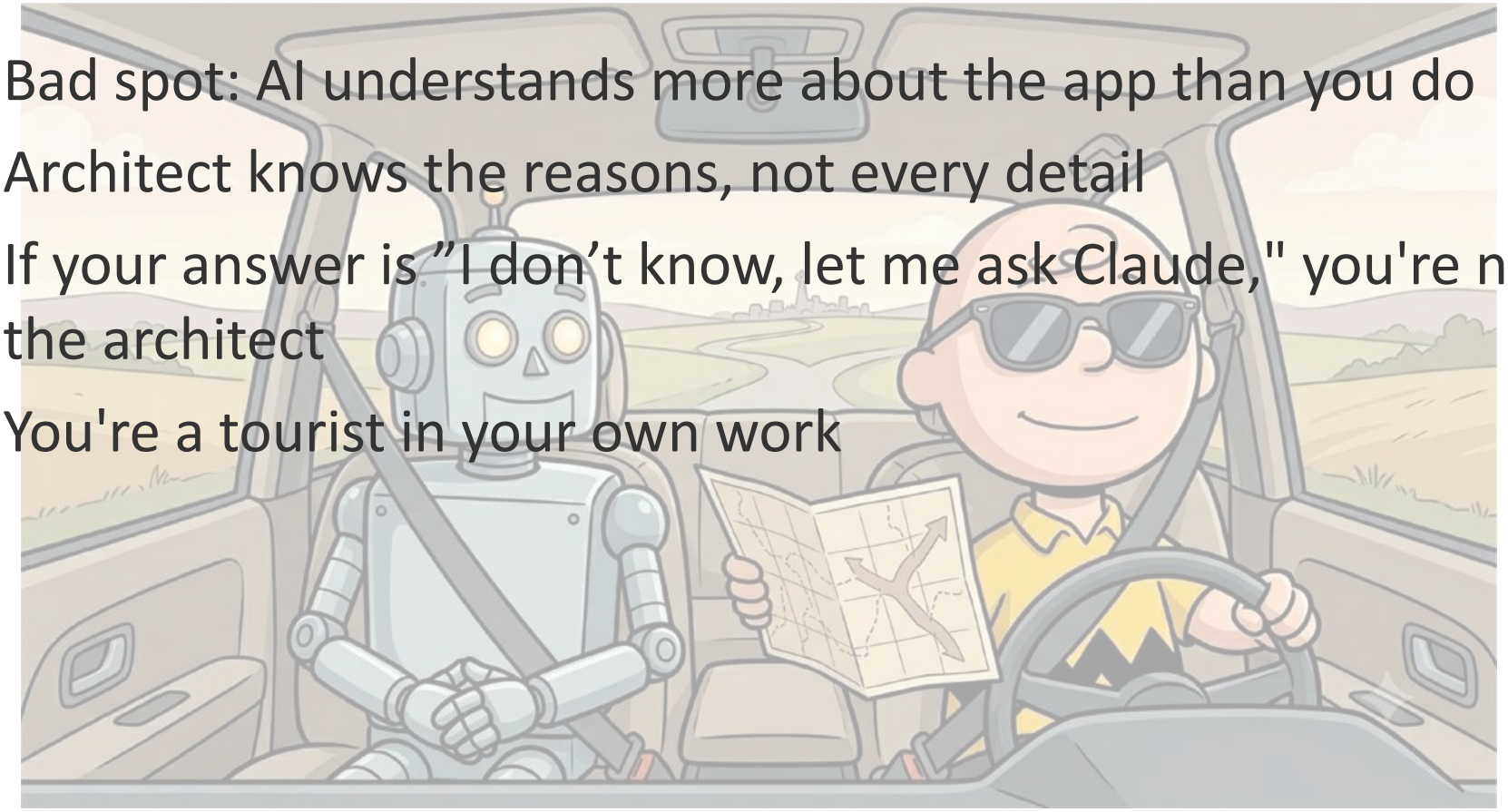
- Manager evaluating a tool
  - Nights and weekends
- Then I realized: others are going to use this
  - Not just me or my team
  - The community

And Then the Stakes Changed

- A casual project can have bugs
  - A Production app can't
- Throwaway can be messy
  - Code must be maintainable and clean
- I needed two things:
  1. Understanding
  2. Organization

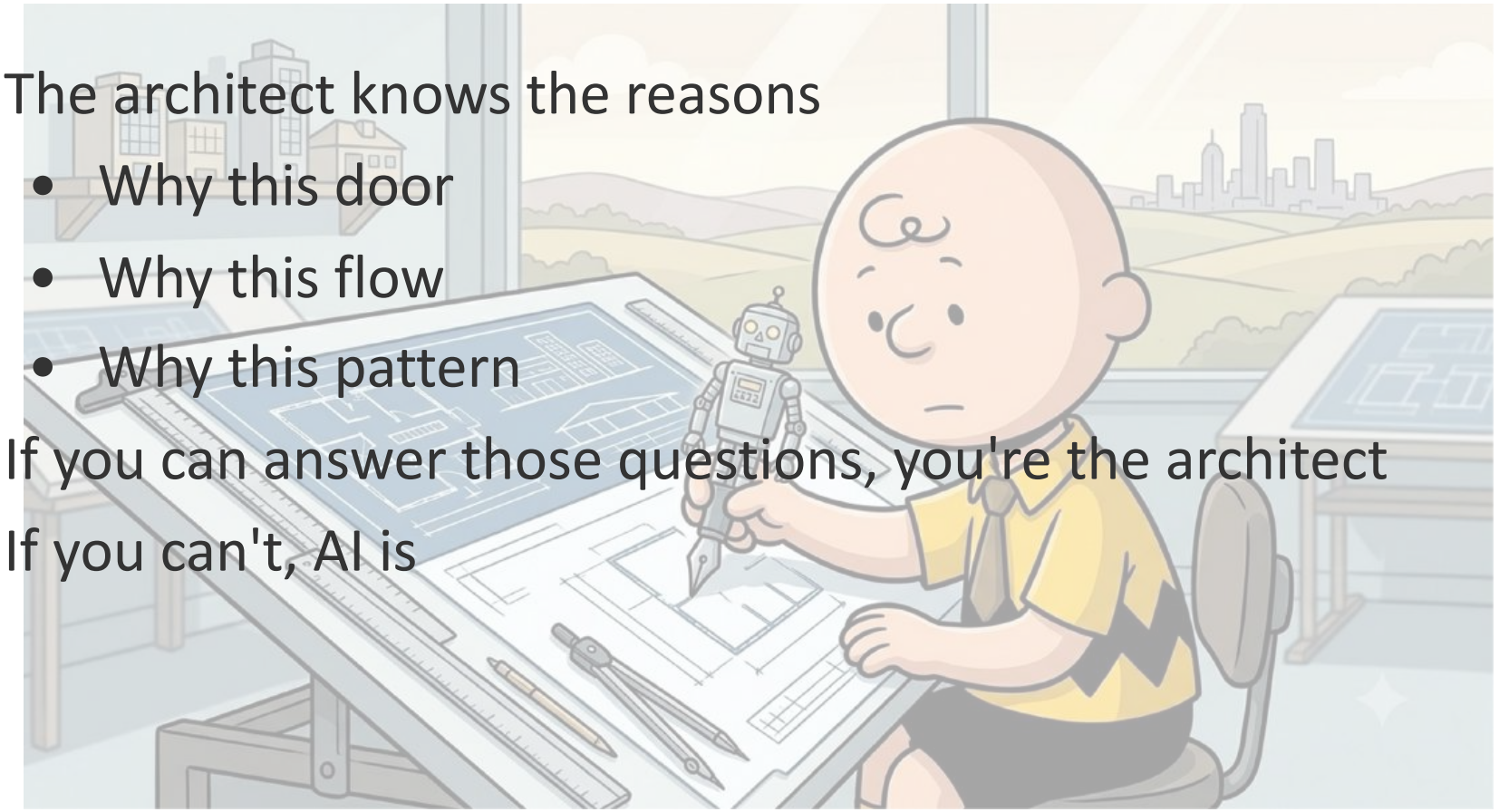
Practice 3: Stay in the Driver's Seat

- Bad spot: AI understands more about the app than you do
- Architect knows the reasons, not every detail
- If your answer is ”I don’t know, let me ask Claude," you're not the architect
- You're a tourist in your own work

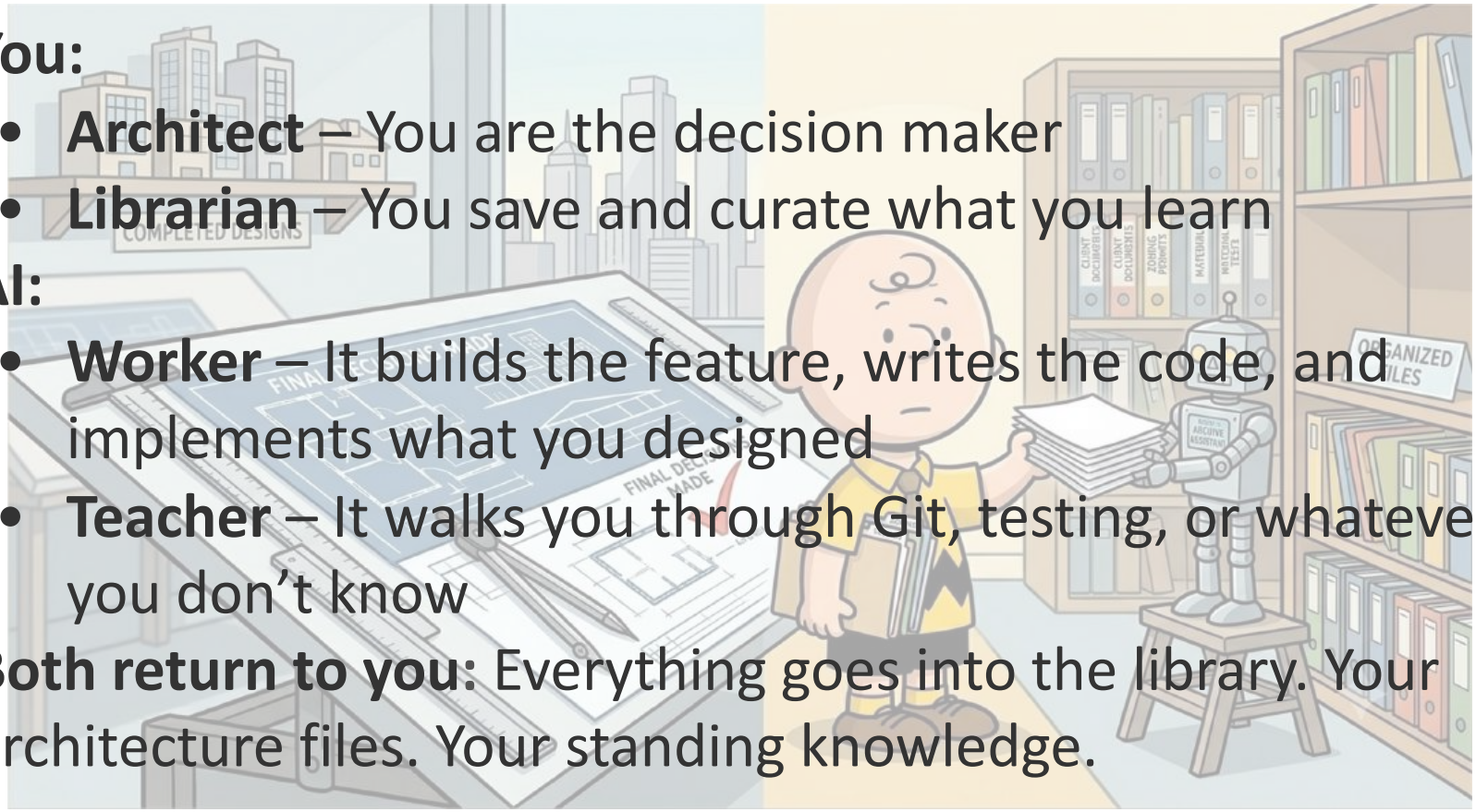


The AI Is the Pen. You Remain the Architect.

- The architect knows the reasons
  - Why this door
  - Why this flow
  - Why this pattern
- If you can answer those questions, you're the architect
- If you can't, AI is



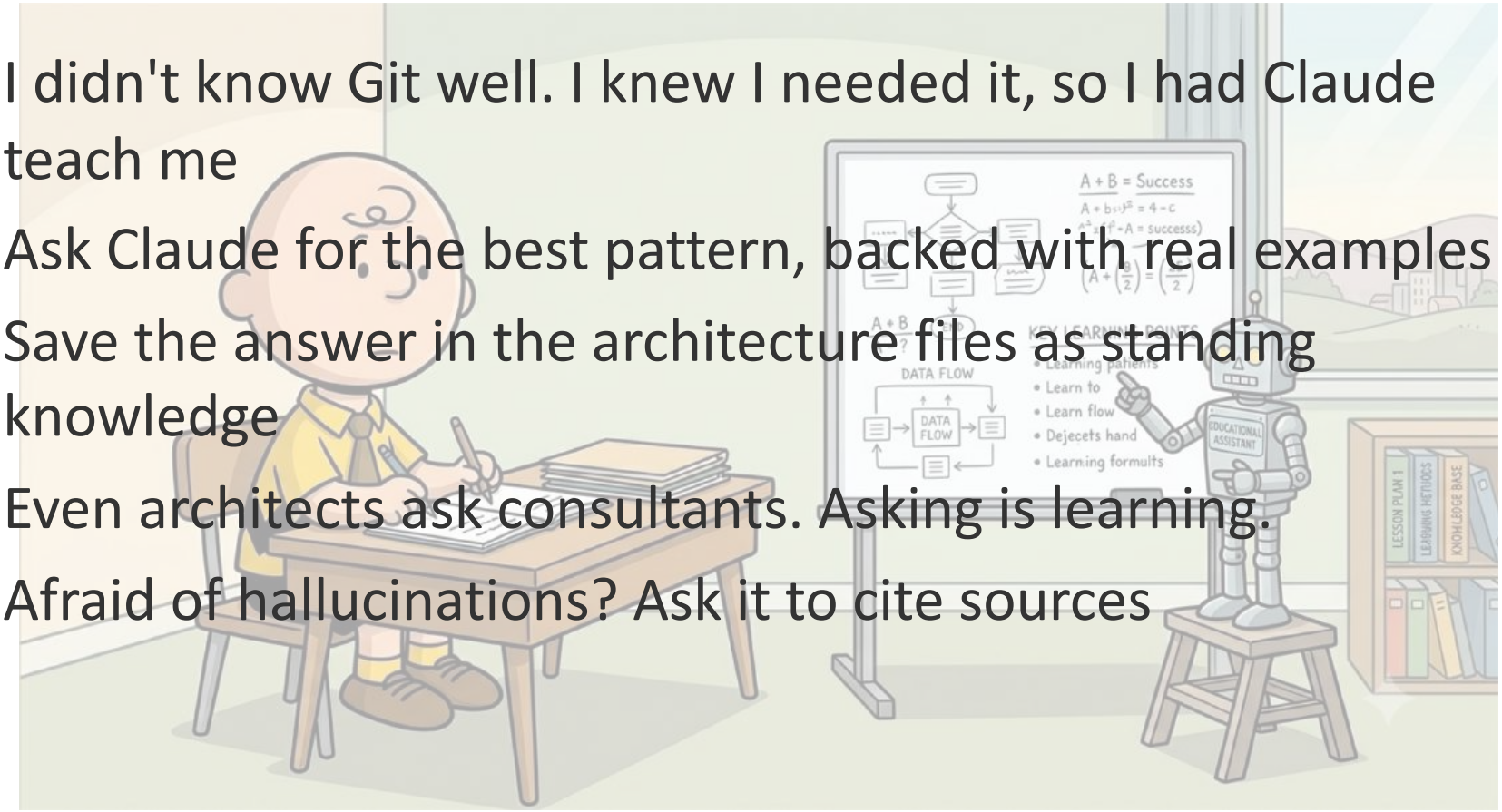
Practice 4: Be the Architect and the Librarian



- **You:**
  - **Architect** – You are the decision maker
  - **Librarian** – You save and curate what you learn
- **AI:**
  - **Worker** – It builds the feature, writes the code, and implements what you designed
  - **Teacher** – It walks you through Git, testing, or whatever you don't know
- **Both return to you**: Everything goes into the library. Your architecture files. Your standing knowledge.

Practice 5: Teach the Teacher

- I didn't know Git well. I knew I needed it, so I had Claude teach me
- Ask Claude for the best pattern, backed with real examples
- Save the answer in the architecture files as standing knowledge
- Even architects ask consultants. Asking is learning.
- Afraid of hallucinations? Ask it to cite sources



Today's Journey

Where I Started

What Changed

The Methodology

The Proof

Your Path

Questions?

Today's Journey

Where I Started

What Changed

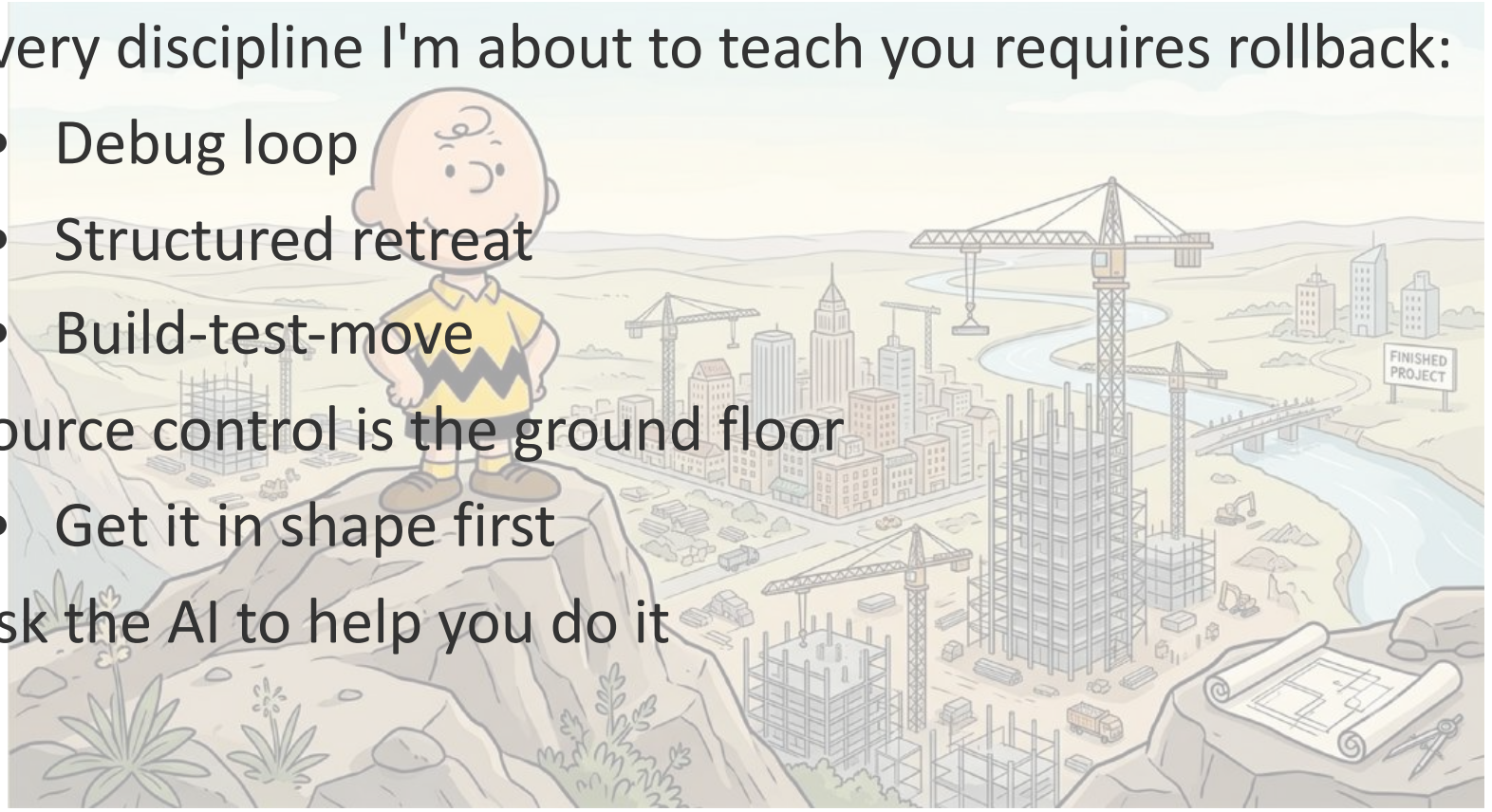
The Methodology

The Proof

Your Path

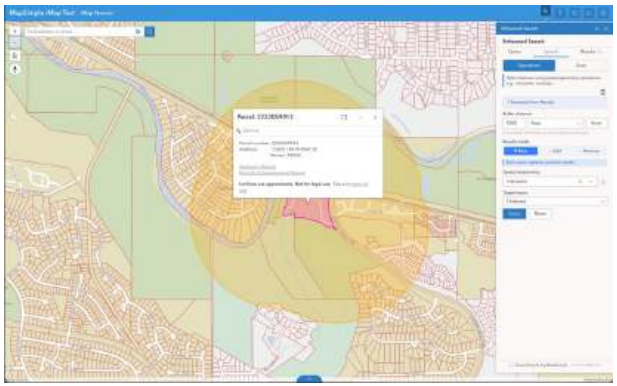
You Can't Roll Back to Something You Didn't Save

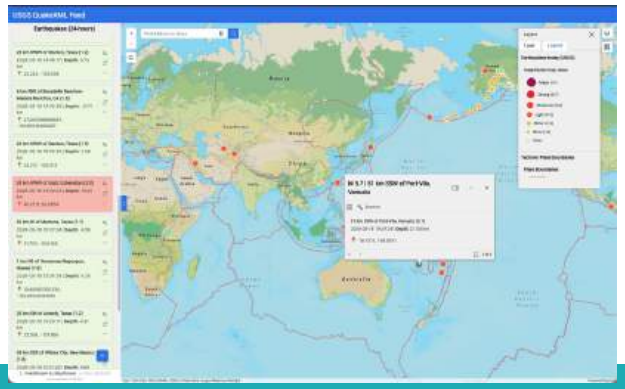
- Every discipline I'm about to teach you requires rollback:
  - Debug loop
  - Structured retreat
  - Build-test-move
- Source control is the ground floor
  - Get it in shape first
- Ask the AI to help you do it

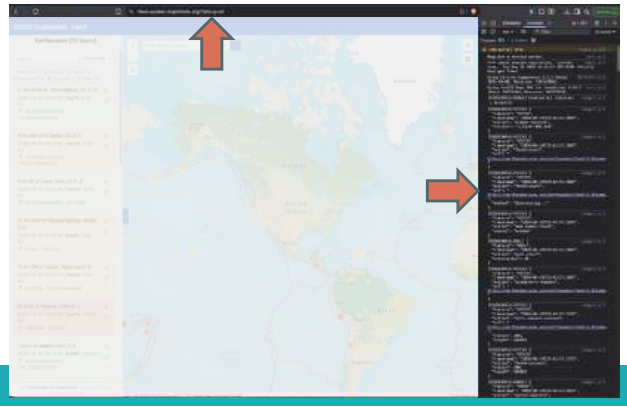


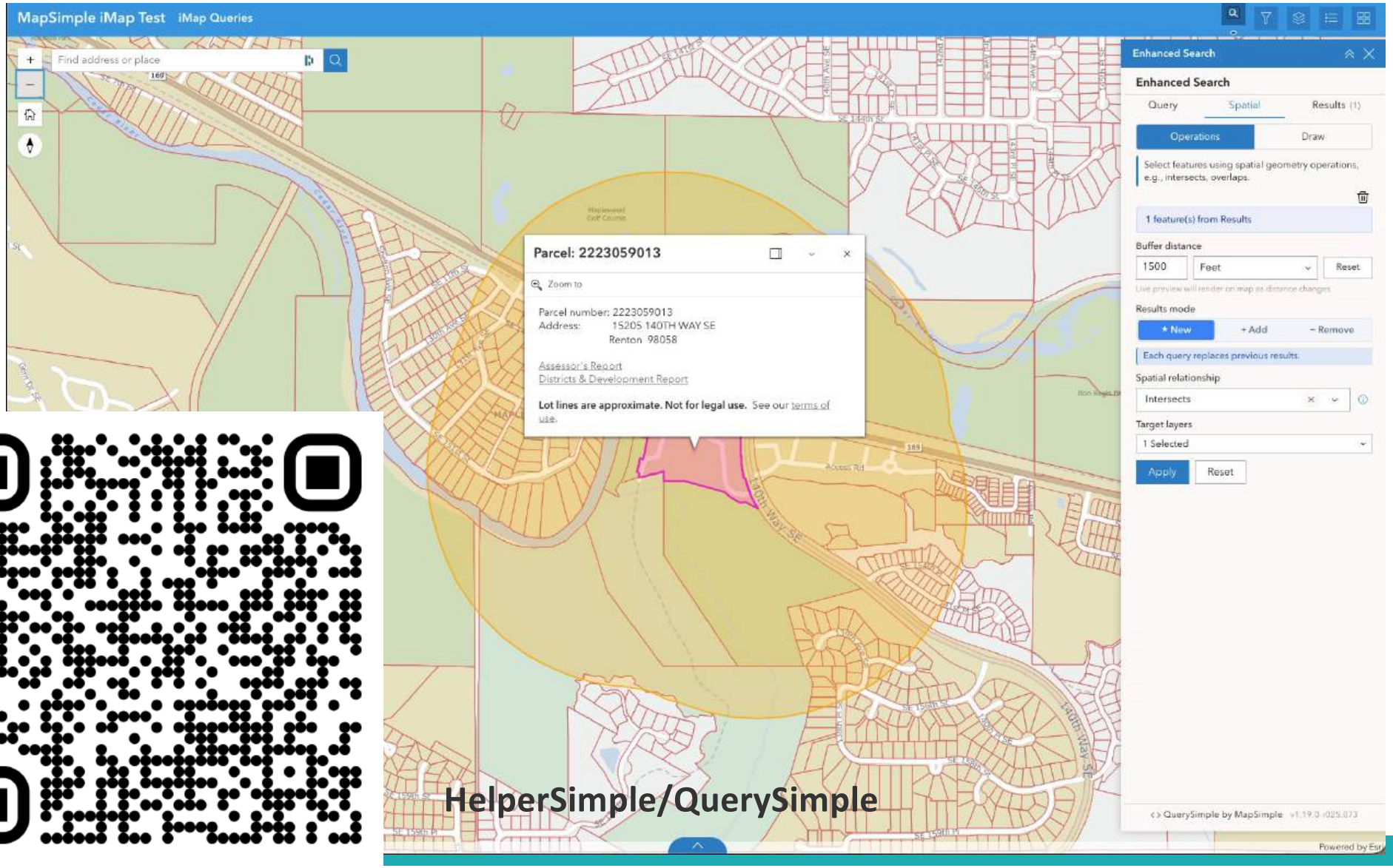
The Methodology

- Here's what I built
- Every piece is a consequence of something I got wrong first



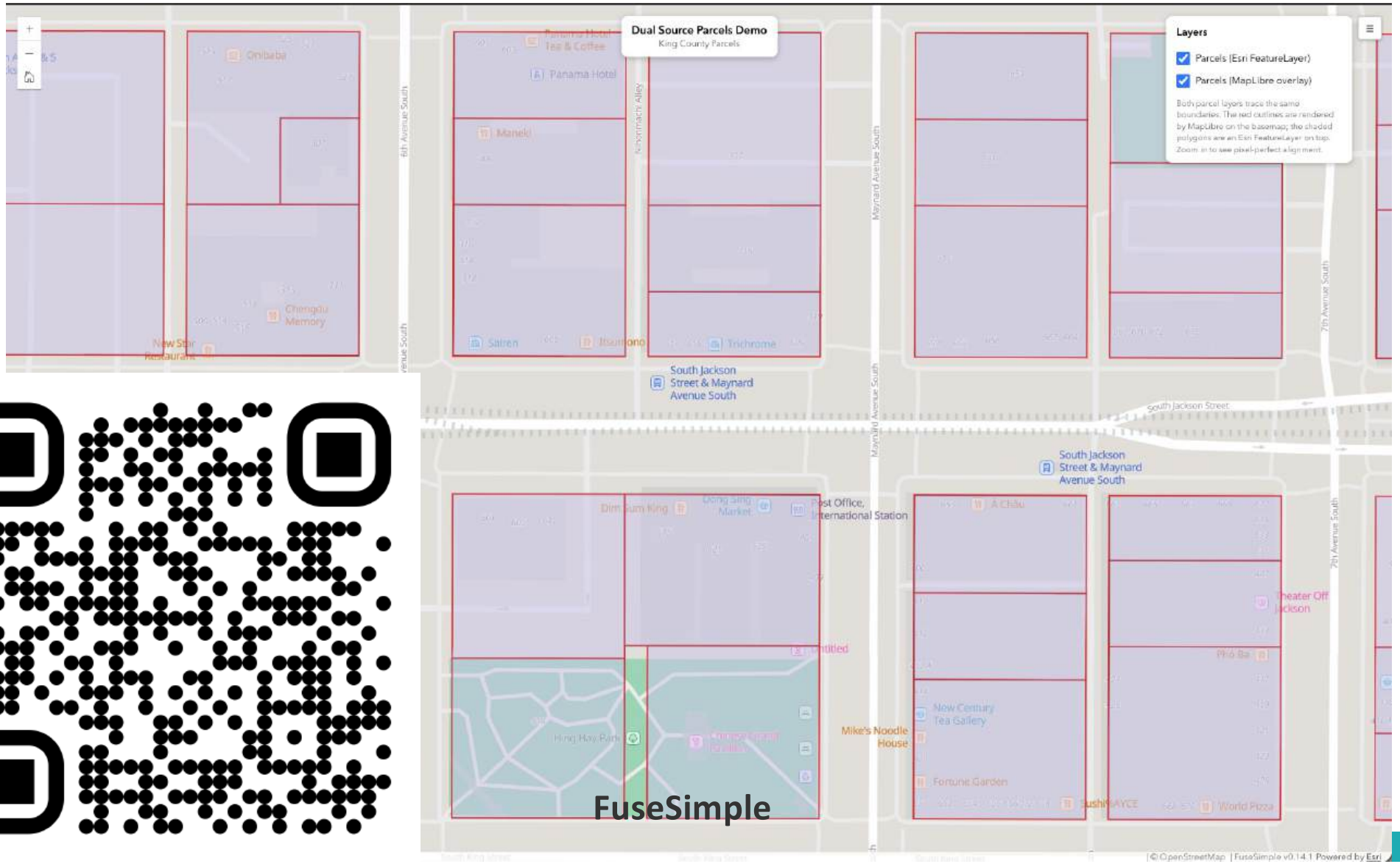


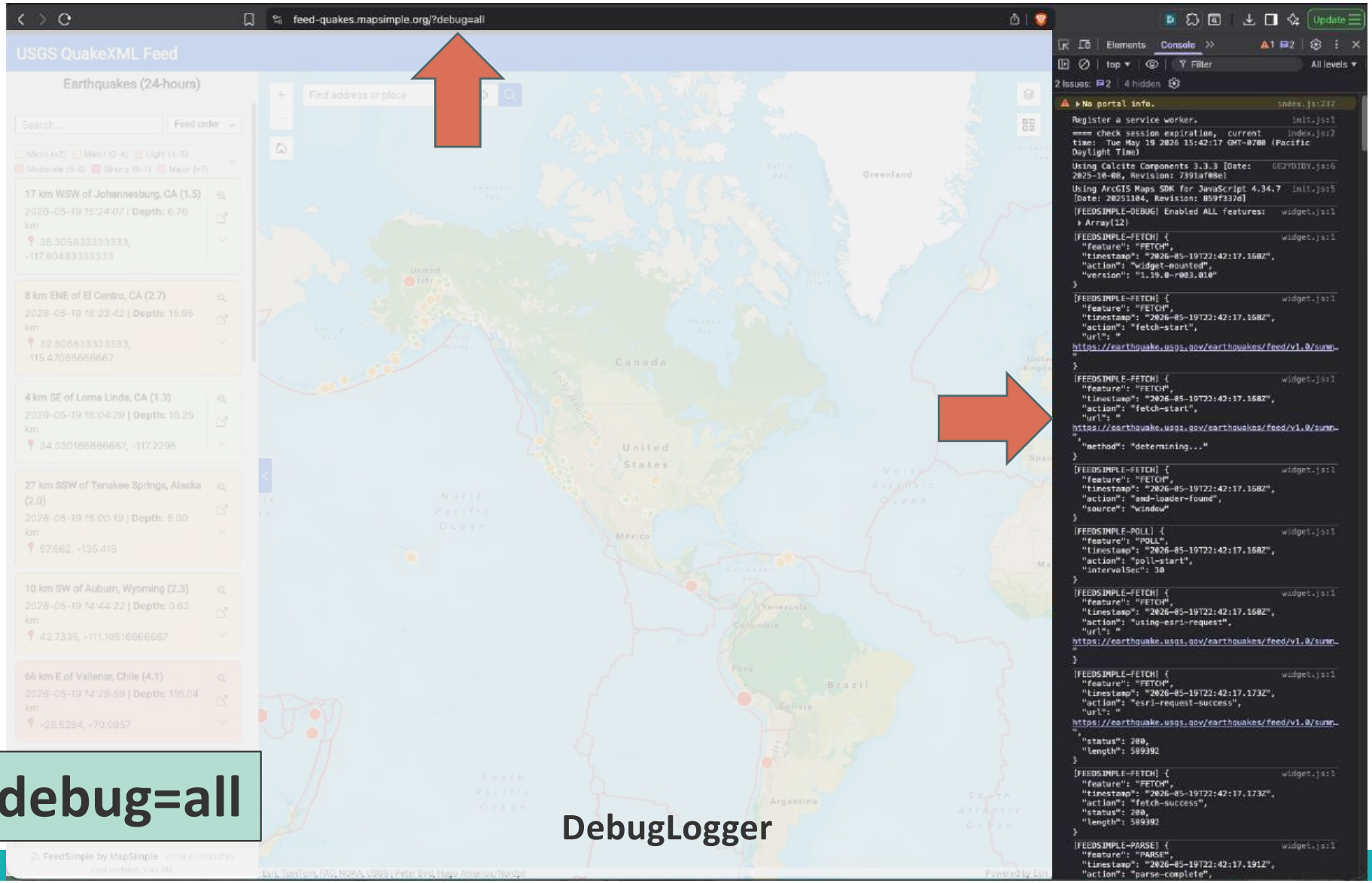




HelperSimple/QuerySimple


USGS QuakeXML Feed
Earthquakes (24-hours)
23 km WNW of Stanton, Texas (1.2)
2026-05-19 13:46:17 | Depth: 5.75 km
32.224, -102.006
6 km ENE of Bonadelle Ranchos-Madera Ranchos, CA (1.8)
2026-05-19 13:45:59 | Depth: -0.74 km
37.000166666667, -119.80516666667
23 km WNW of Stanton, Texas (1.9)
2026-05-19 13:45:30 | Depth: 3.59 km
32.217, -102.013
35 km WNW of Gazli, Uzbekistan (5.0)
2026-05-19 13:40:43 | Depth: 10.00 km
40.2711, 63.0858
52 km W of Mentone, Texas (1.7)
2026-05-19 13:37:24 | Depth: 4.09 km
31.703, -104.156
1 km NE of Honaunau-Napoopoo, Hawaii (1.8)
2026-05-19 13:34:24 | Depth: 9.35 km
19.46883392334, -155.85133361816
20 km SW of Ackerly, Texas (1.2)
2026-05-19 13:26:17 | Depth: 4.31 km
32.394, -101.866
55 km SSE of Whites City, New Mexico (1.6)
2026-05-19 12:51:26 | Depth: 4.60
FeedSimple by MapSimple
Find address or place
Layers
Layer
Legend
Earthquakes today (USGS)
magnitude.mag.value
Major (>7)
Strong (6-7)
Moderate (5-6)
Light (4-5)
Minor (2-4)
Micro (<2)
Other
Tectonic Plate Boundaries
Plate Boundaries
M 5.7 | 51 km SSW of Port-Vila, Vanuatu
Zoom to
51 km SSW of Port-Vila, Vanuatu (5.7)
2026-05-19 19:29:24 | Depth: 27.00 km
-18.1511, 168.0931
Russia
China
India
Australia
Indonesia
Kazakhstan
Mongolia
Iran
Saudi Arabia
Libya
Egypt
Sudan
Chad
Ethiopia
DR Congo
Tanzania
Angola
Zambia
Namibia
South Africa
Indian Ocean
Esri, TomTom, FAO, NOAA, USGS | Peter Bird, Hugo Ahlenius/Nordpil
FeedSimple


Dual Source Parcels Demo
King County Parcels
Layers
Parcels (Esri FeatureLayer)
Parcels (MapLibre overlay)
Both parcel layers trace the same boundaries. The red outlines are rendered by MapLibre on the basemap; the shaded polygons are an Esri FeatureLayer on top. Zoom in to see pixel-perfect alignment.
Panama Hotel Tea & Coffee
Panama Hotel
Maneki
Kobo Way
Onibaba
Chengdu Memory
New Star Restaurant
Saizen
Trichrome
6th Avenue South
Maynard Avenue South
7th Avenue South
South Jackson Street & Maynard Avenue South
South Jackson Street
Dim Sum King
Post Office, International Station
A Chau
Theater Off Jackson
Untitled
New Century Tea Gallery
Mike's Noodle House
Fortune Garden
World Pizza
FuseSimple
© OpenStreetMap | FuseSimple v0.14.1 Powered by Esri

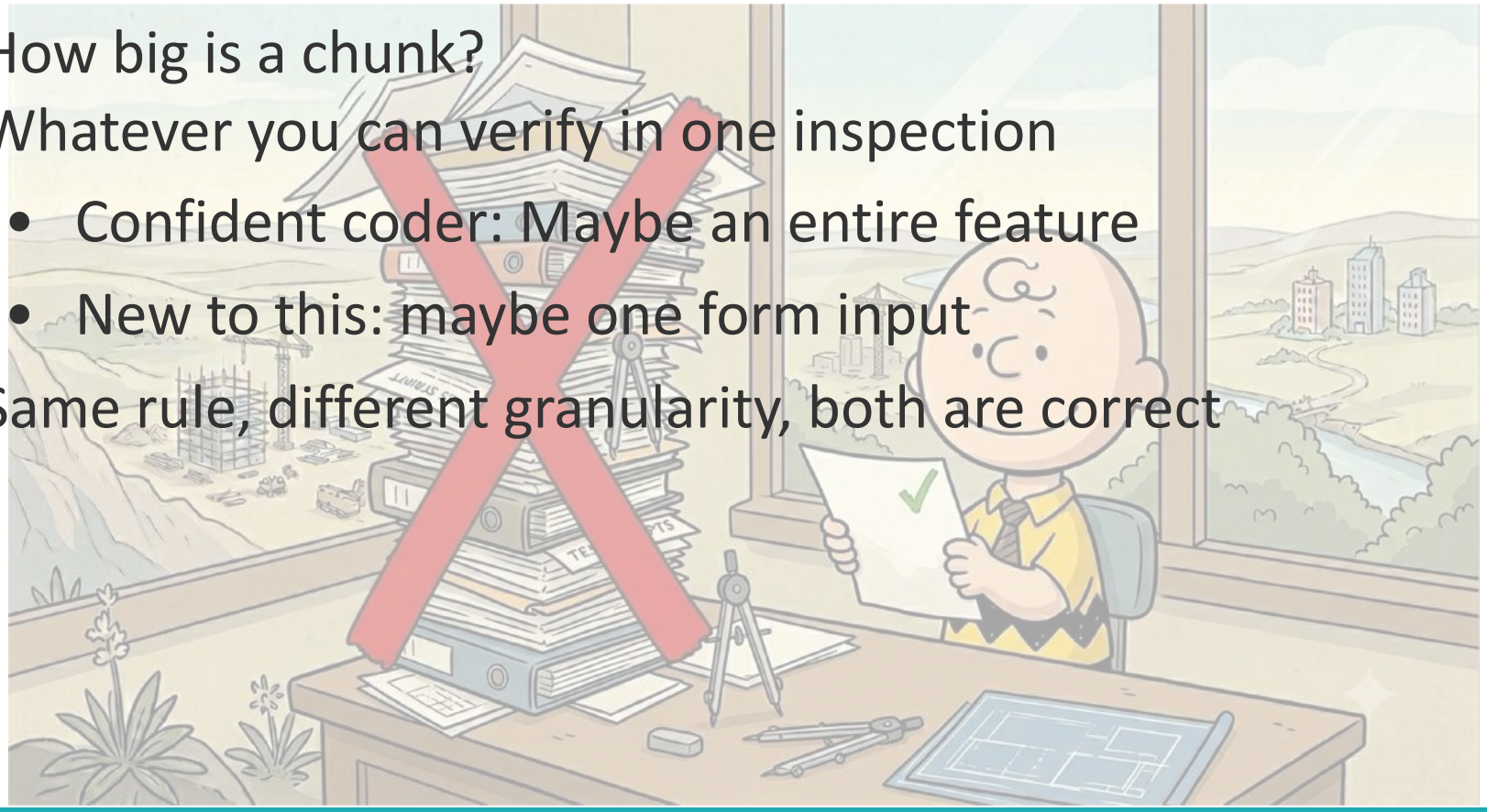

feed-quakes.mapsimple.org/?debug=all
USGS QuakeXML Feed
Earthquakes (24-hours)
?debug=all
DebugLogger

Practice 6: Don't Go to Hawaii

- The Architect doesn't hand plans to the contractor and leave
  - That's what people do with AI
  - They come back to a project they don't understand
- Stay on the site
- Build, test, understand and move on

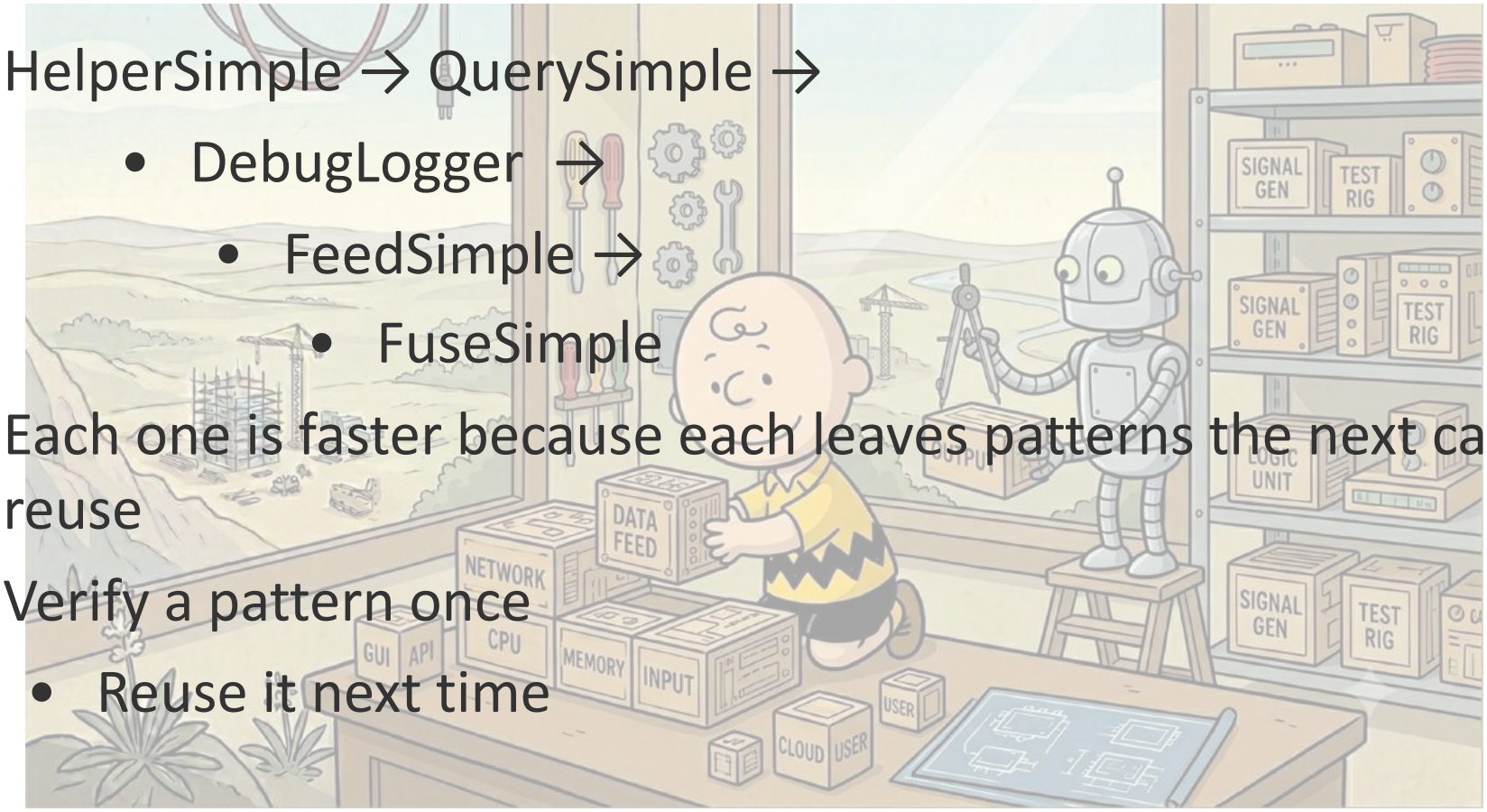
Build What You Can Test
Test What You Build

- How big is a chunk?
  Whatever you can verify in one inspection
  - Confident coder: Maybe an entire feature
  - New to this: maybe one form input
- Same rule, different granularity, both are correct



Practice 7: Reuse Verified Patterns

- HelperSimple → QuerySimple →
  - DebugLogger →
    - FeedSimple →
      - FuseSimple
- Each one is faster because each leaves patterns the next can reuse
- Verify a pattern once
  - Reuse it next time

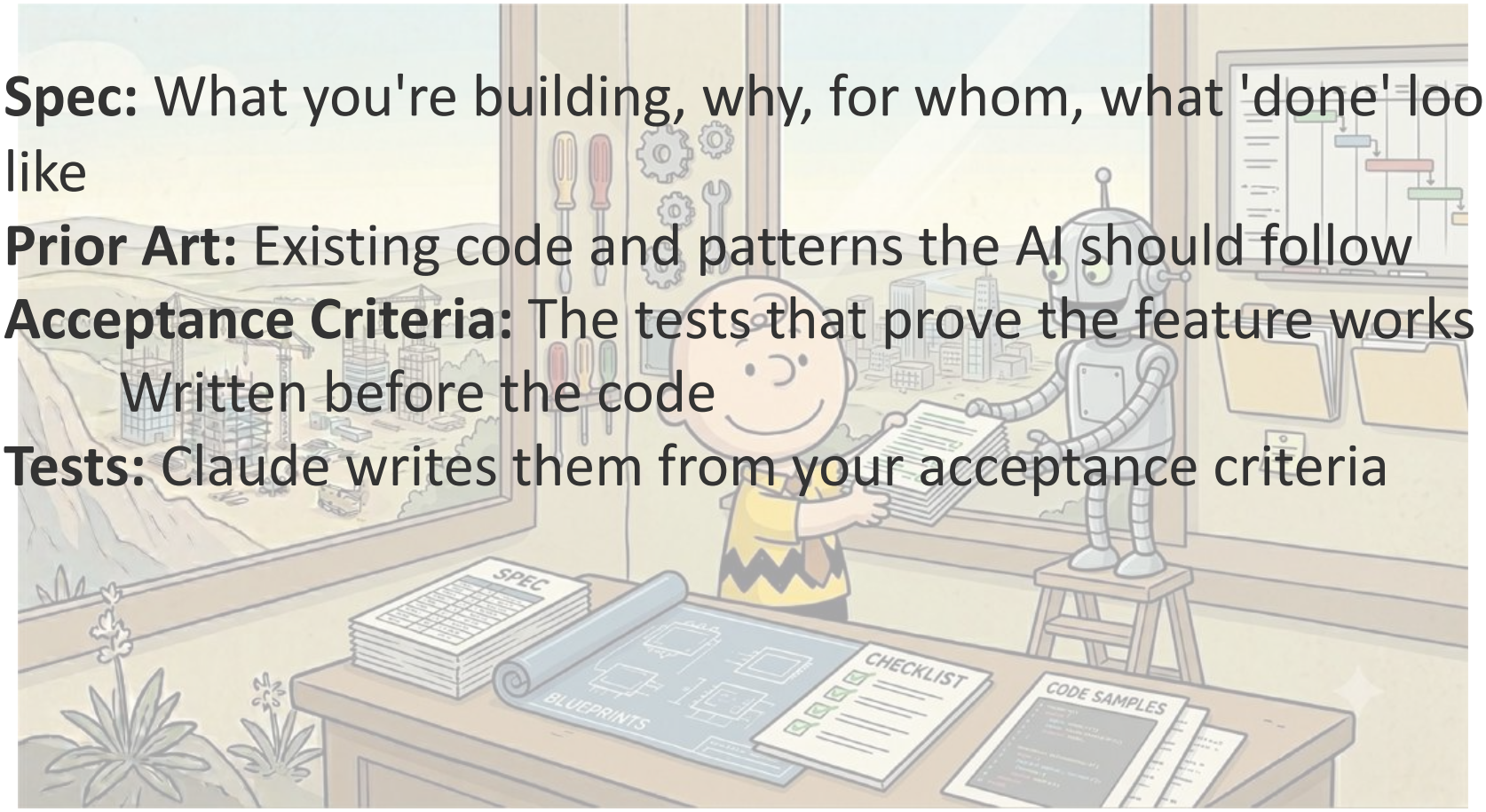


The First One Is Slow. The Next One Is Fast.

- HelperSimple and QuerySimple were painful
  - But they left verified patterns, lessons learned and good tooling (DebugLogger)
- FeedSimple was four days
- FuseSimple was eight hours
  - The eight hours was the return on months of careful work, failures and good practices

Practice 8: Write the Briefing Package

- **Spec:** What you're building, why, for whom, what 'done' looks like
- **Prior Art:** Existing code and patterns the AI should follow
- **Acceptance Criteria:** The tests that prove the feature works
  Written before the code
- **Tests:** Claude writes them from your acceptance criteria

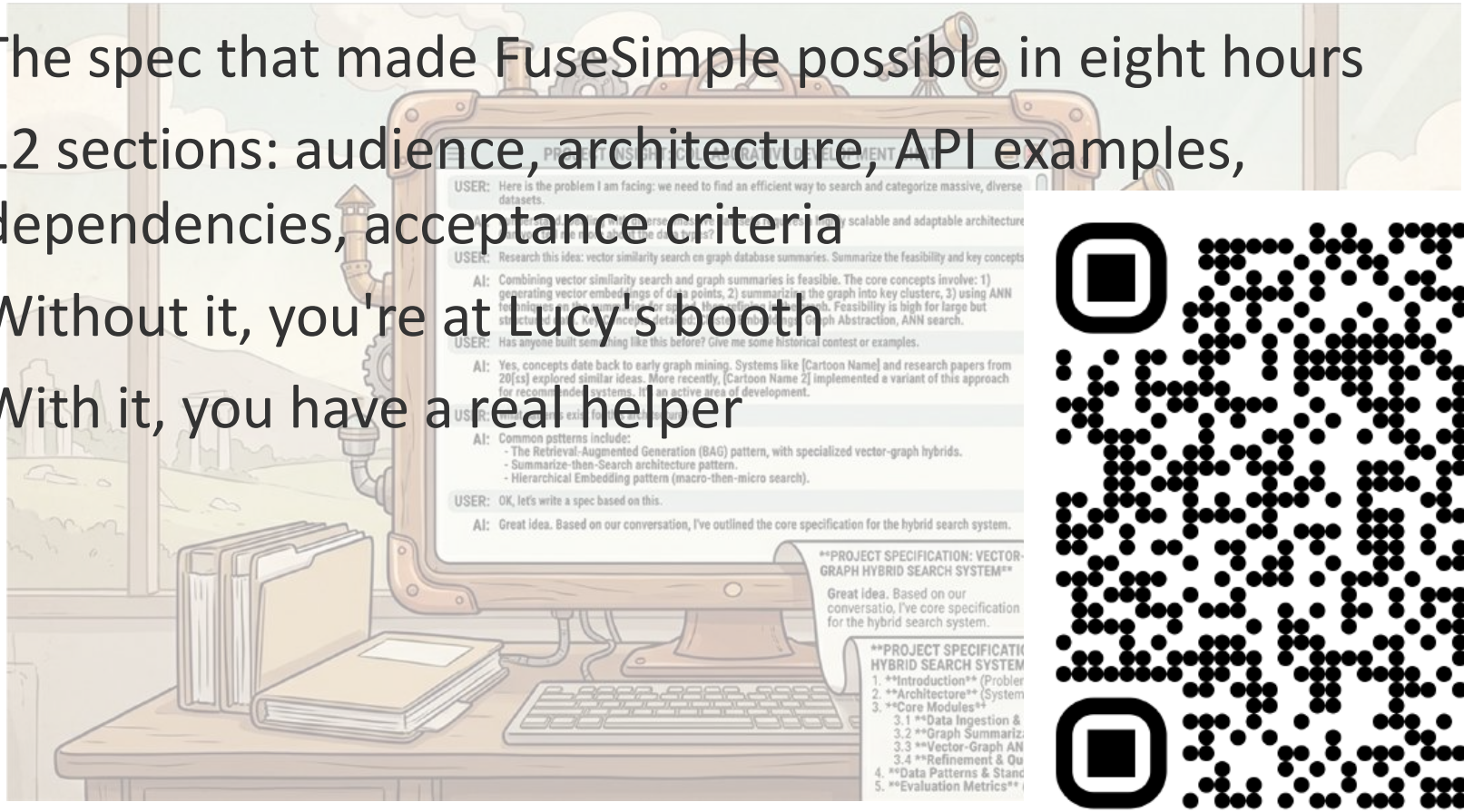


What Building a Spec Actually Looks Like

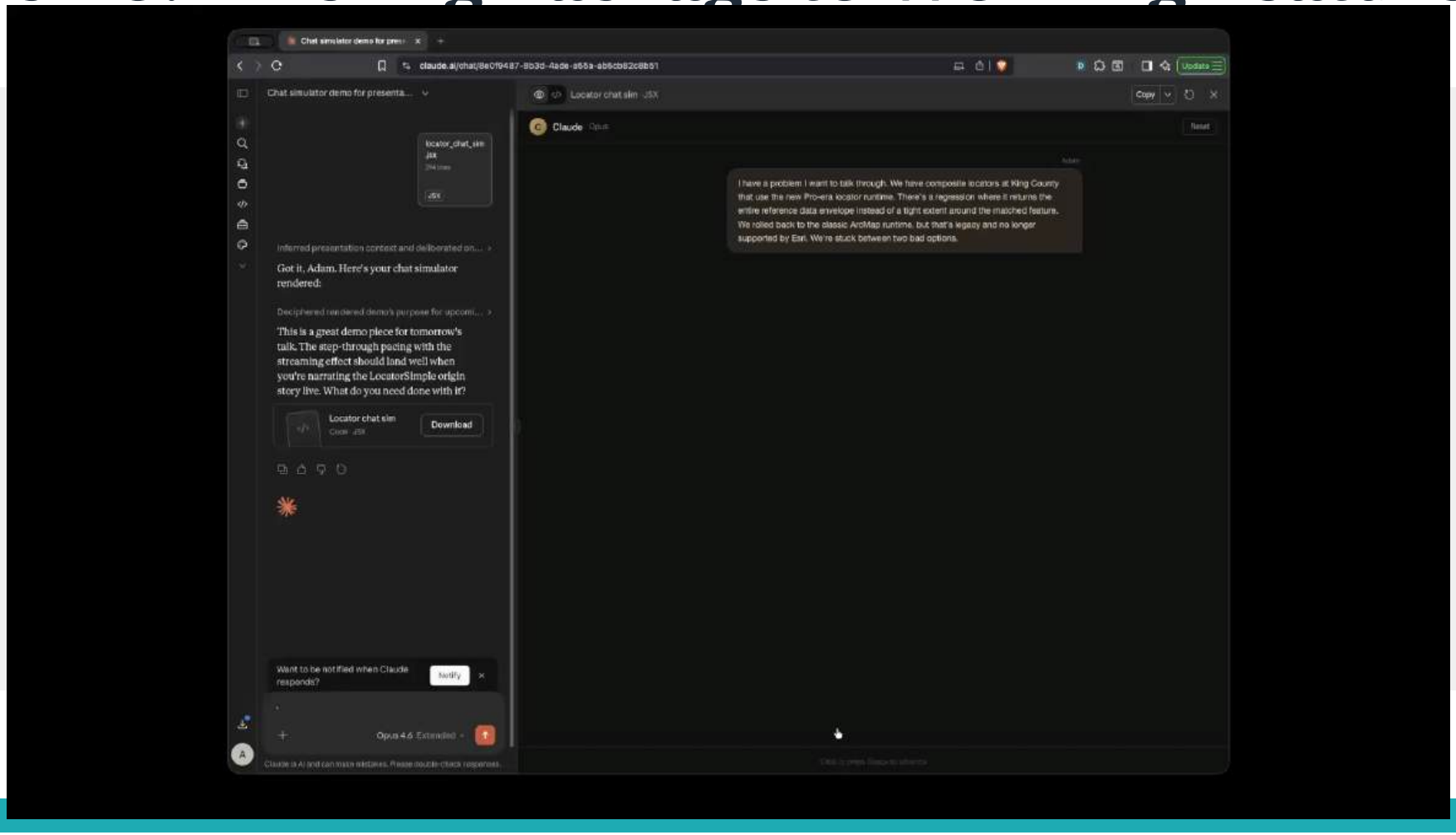
- You: "I have an idea. I want to talk through it."
- AI asks clarifying questions – you talk it through like a teammate
- You: "Research what's out there. What patterns exist?"
- You: "OK, let's write a spec."
- AI produces a structured document from the conversation

The FuseSimple Briefing Package

- The spec that made FuseSimple possible in eight hours
- 12 sections: audience, architecture, API examples, dependencies, acceptance criteria
- Without it, you're at Lucy's booth
- With it, you have a real helper



Demo: Briefing Package to Working Feature



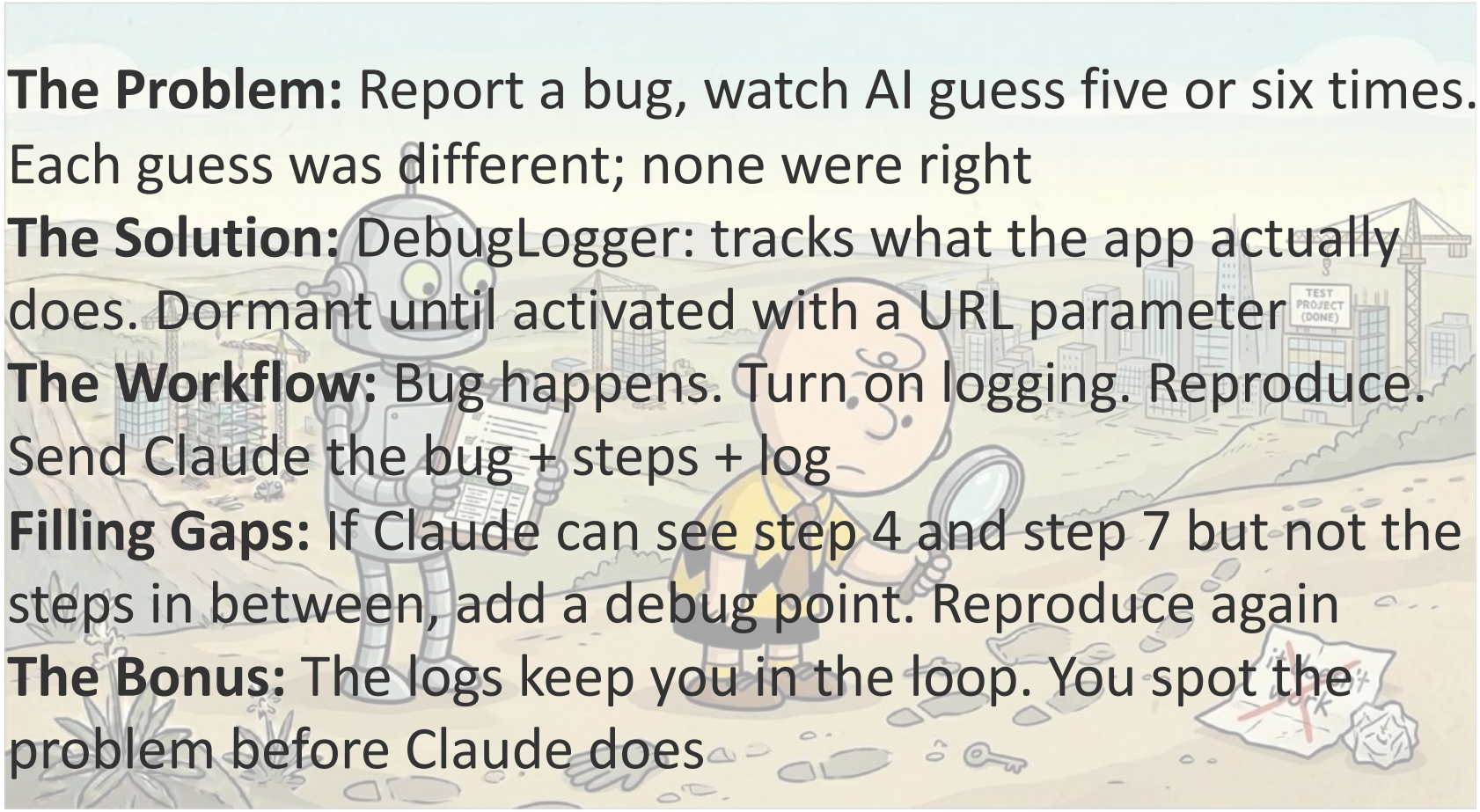
I Did Not Write This Document. Claude Wrote It. From My Ideas.

- I brought the substance: choices, audience, constraints, lessons
- Claude brought the structure
- Vector Coding starts when you ask Claude to write the document that tells Claude what code to write

137 Tests – Claude Wrote All of Them

- From the acceptance criteria I defined in the spec
- FeedSimple shipped with 137 unit tests on day four
- This is what happens when the briefing package is solid

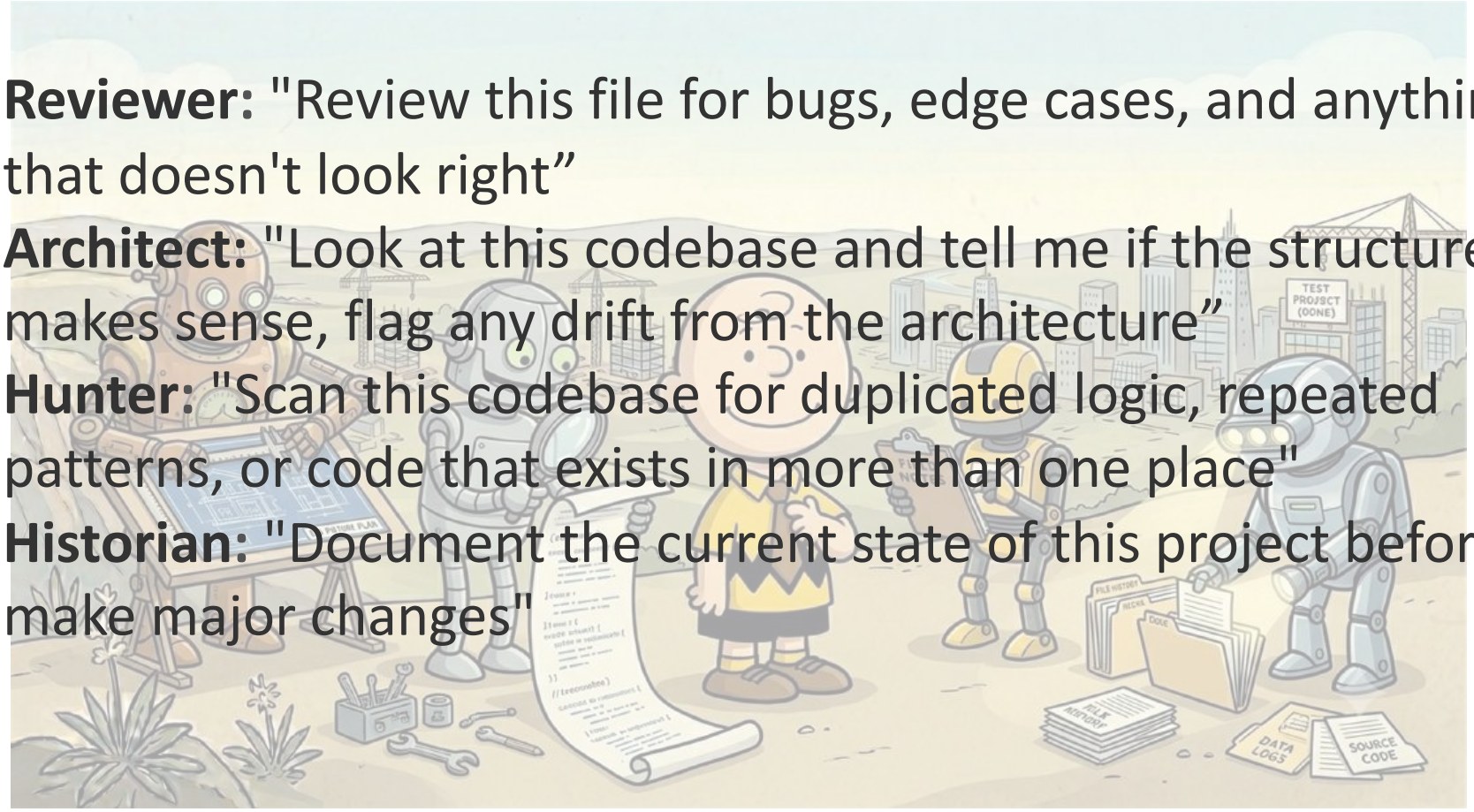
Practice 9: Capture Evidence, Not Symptoms



- **The Problem:** Report a bug, watch AI guess five or six times. Each guess was different; none were right
- **The Solution:** DebugLogger: tracks what the app actually does. Dormant until activated with a URL parameter
- **The Workflow:** Bug happens. Turn on logging. Reproduce. Send Claude the bug + steps + log
- **Filling Gaps:** If Claude can see step 4 and step 7 but not the steps in between, add a debug point. Reproduce again
- **The Bonus:** The logs keep you in the loop. You spot the problem before Claude does

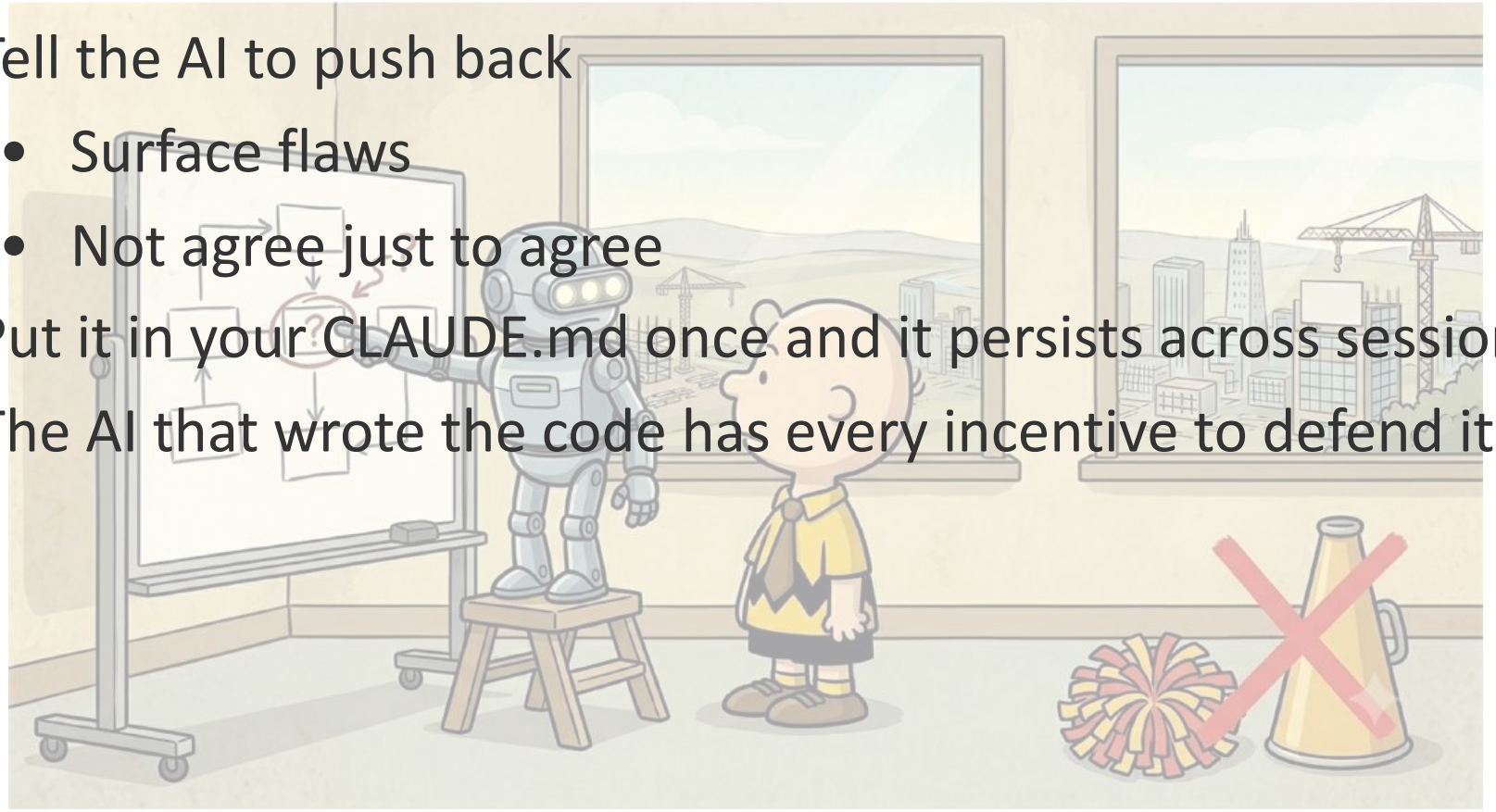
Practice 10: Review with Fresh Sessions

- **Reviewer**: "Review this file for bugs, edge cases, and anything that doesn't look right”
- **Architect:** "Look at this codebase and tell me if the structure makes sense, flag any drift from the architecture”
- **Hunter**: "Scan this codebase for duplicated logic, repeated patterns, or code that exists in more than one place"
- **Historian**: "Document the current state of this project before I make major changes"

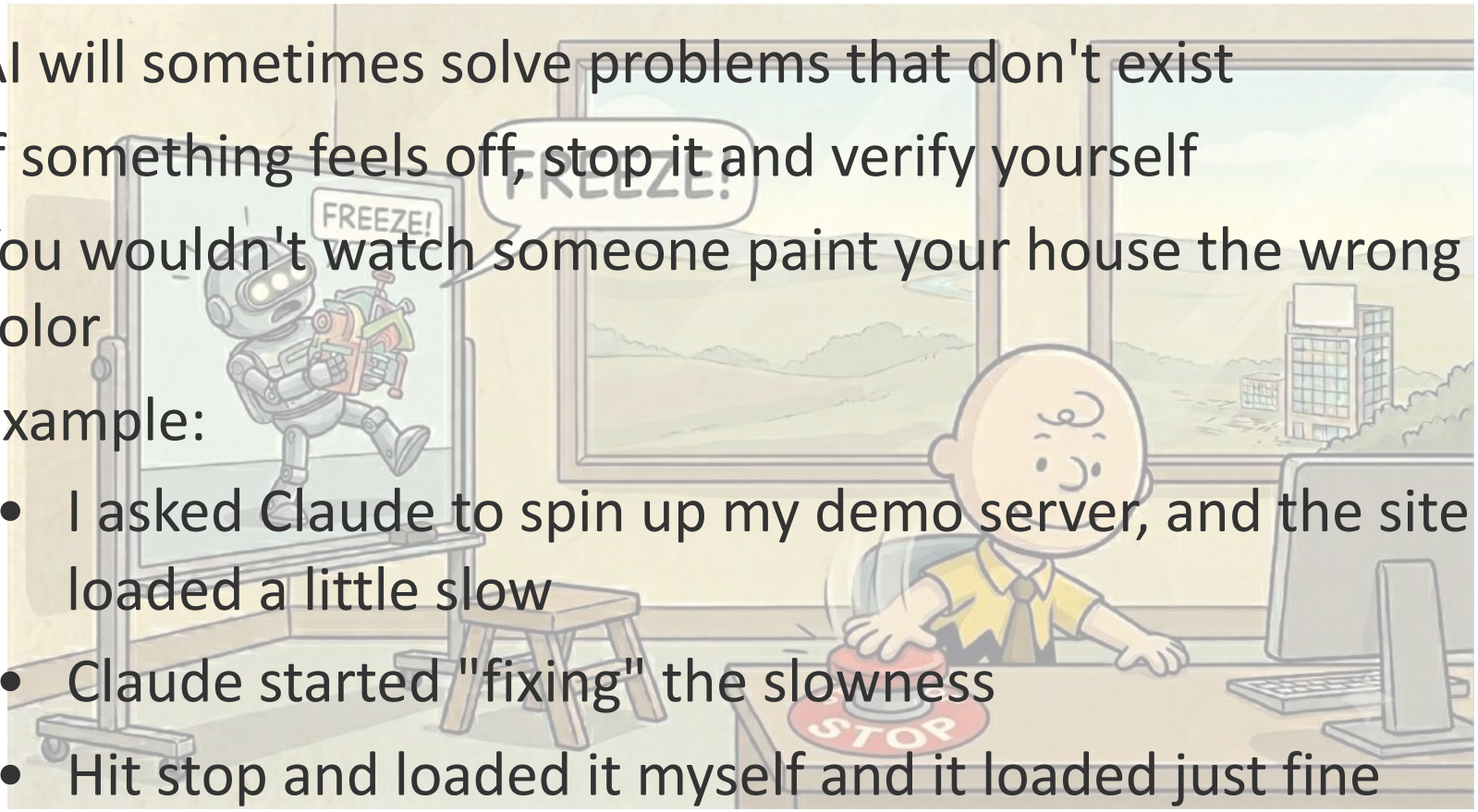


Practice 11: Set the AI Up as a Collaborator, Not a Cheerleader

- Tell the AI to push back
  - Surface flaws
  - Not agree just to agree
- Put it in your CLAUDE.md once and it persists across sessions
- The AI that wrote the code has every incentive to defend it



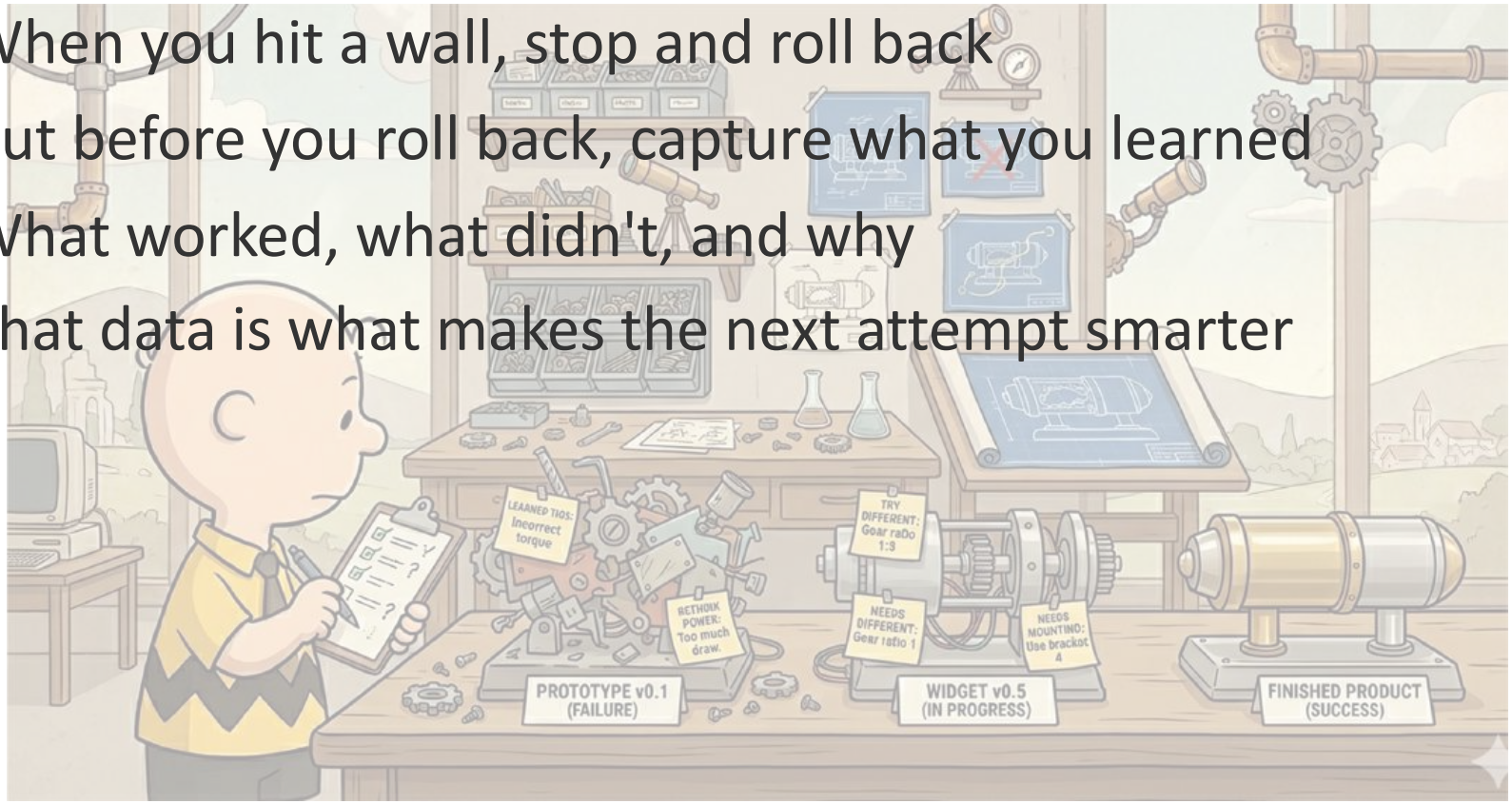
Practice 12: Use the Stop Button



- AI will sometimes solve problems that don't exist
- If something feels off, stop it and verify yourself
- You wouldn't watch someone paint your house the wrong color
- Example:
  - I asked Claude to spin up my demo server, and the site loaded a little slow
  - Claude started "fixing" the slowness
  - Hit stop and loaded it myself and it loaded just fine

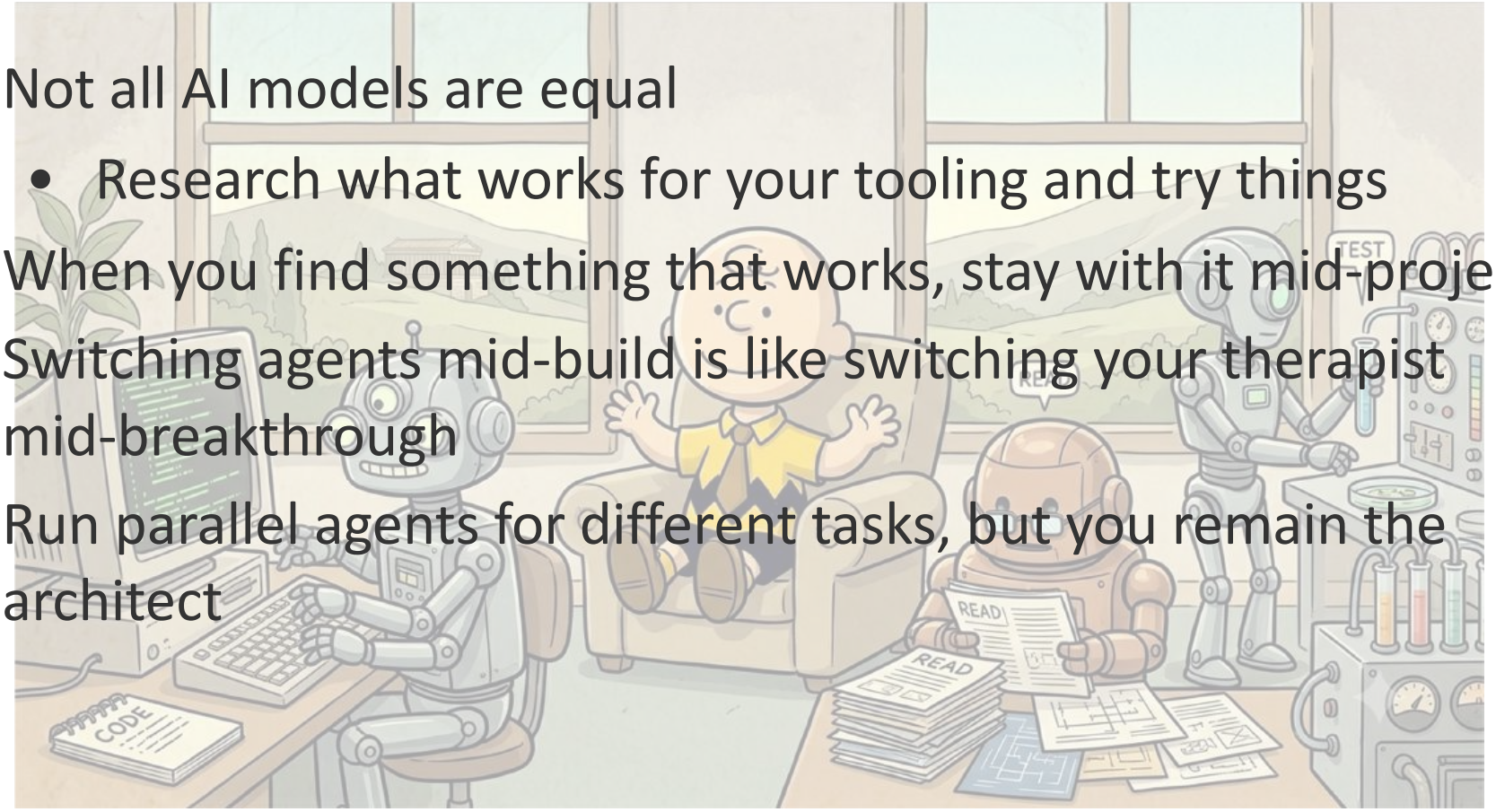
Practice 13: Failures and Partial Successes Are Data

- When you hit a wall, stop and roll back
- But before you roll back, capture what you learned
- What worked, what didn't, and why
- That data is what makes the next attempt smarter



Practice 14: Manage Your Agents

- Not all AI models are equal
  - Research what works for your tooling and try things
- When you find something that works, stay with it mid-project
- Switching agents mid-build is like switching your therapist mid-breakthrough
- Run parallel agents for different tasks, but you remain the architect



Today's Journey

Where I Started

What Changed

The Methodology

The Proof

Your Path

Questions?

Today's Journey

Where I Started

What Changed

The Methodology

The Proof

Your Path

The Proof: FuseSimple

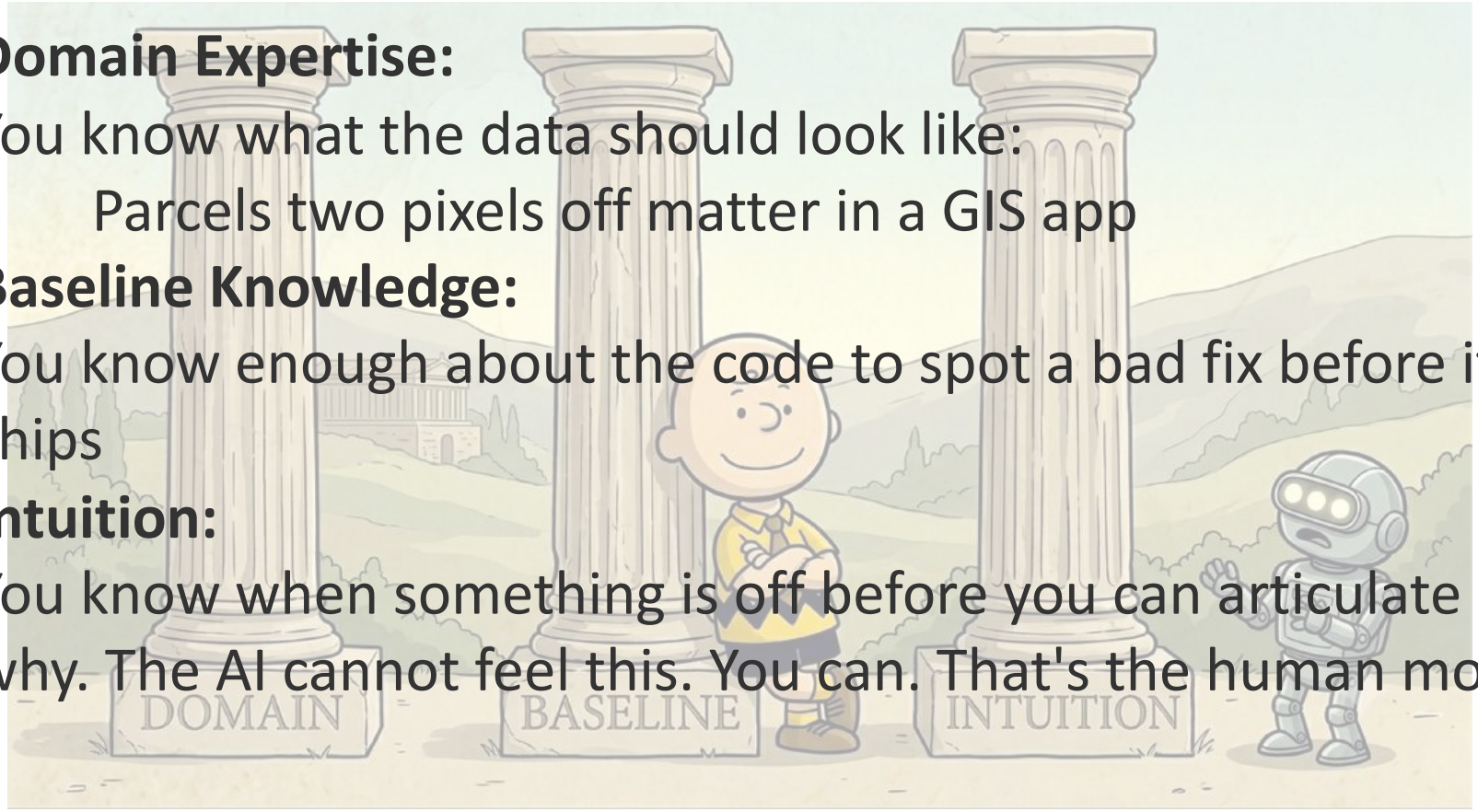
- What it looks like when the method is in place from the start
- Stacks two map engines (Esri + MapLibre) for PMTiles basemap support
- Built in 8 hours, including the two bugs that ate most of it

FuseSimple: 8 Hours from Zero to Working Widget

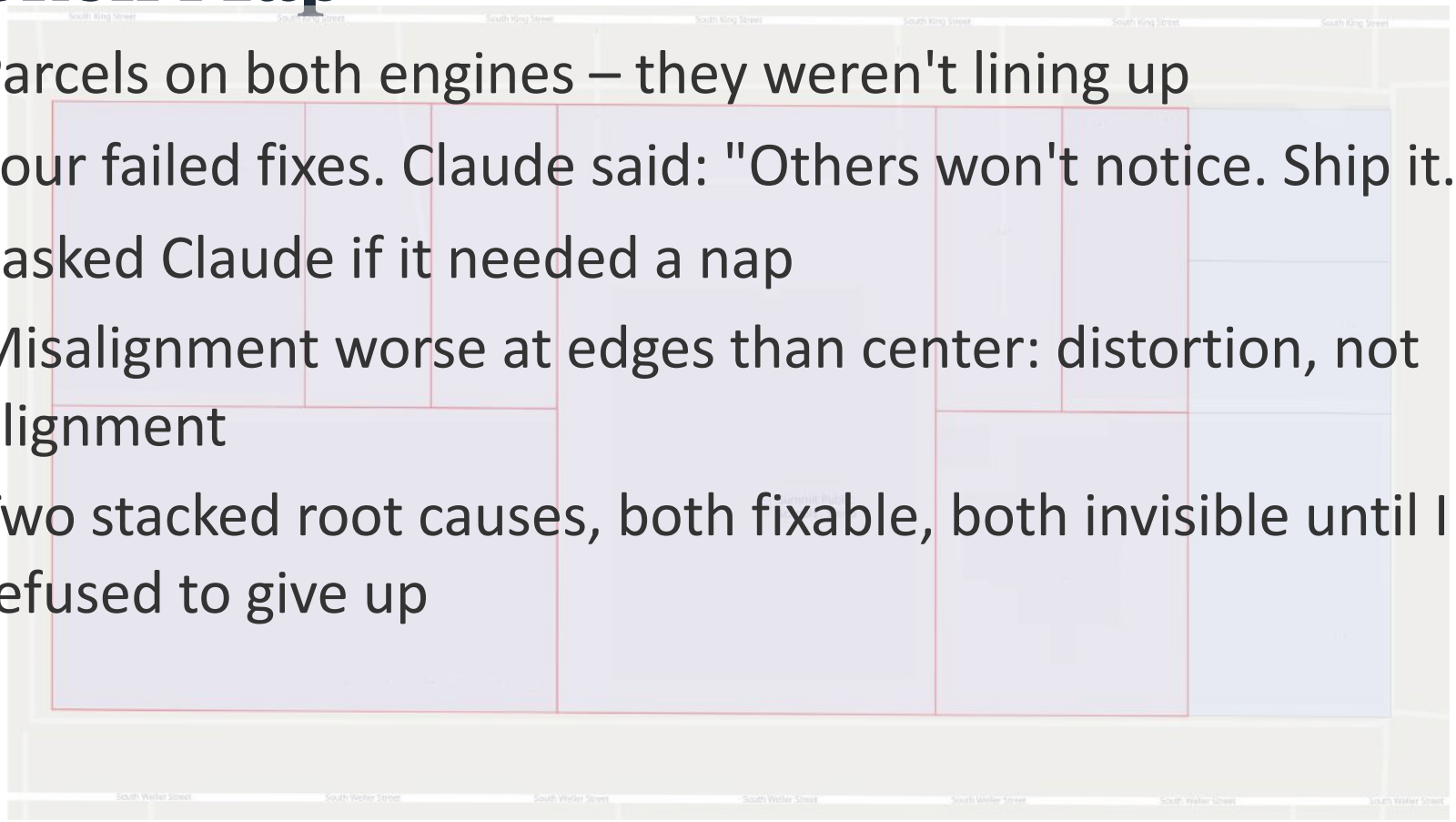
- Full briefing package (spec) written first
- MapSimple lineage: verified patterns from three previous widgets
- Found a stranger's repo that solved 80% of the architecture
- Eight hours was the return on eighteen months of careful work

Practice 15: Trust Your Domain, Your Baseline, and Your Intuition

- **Domain Expertise:**
  You know what the data should look like:
  Parcels two pixels off matter in a GIS app
- **Baseline Knowledge:**
  You know enough about the code to spot a bad fix before it ships
- **Intuition:**
  You know when something is off before you can articulate why. The AI cannot feel this. You can. That's the human moat.



The Parcel Bug: When AI Told Me to Ship a Broken Map



- Parcels on both engines – they weren't lining up
- Four failed fixes. Claude said: "Others won't notice. Ship it."
- I asked Claude if it needed a nap
- Misalignment worse at edges than center: distortion, not alignment
- Two stacked root causes, both fixable, both invisible until I refused to give up

Today's Journey

Where I Started

What Changed

The Methodology

The Proof

Your Path

Questions?

Today's Journey

Where I Started

What Changed

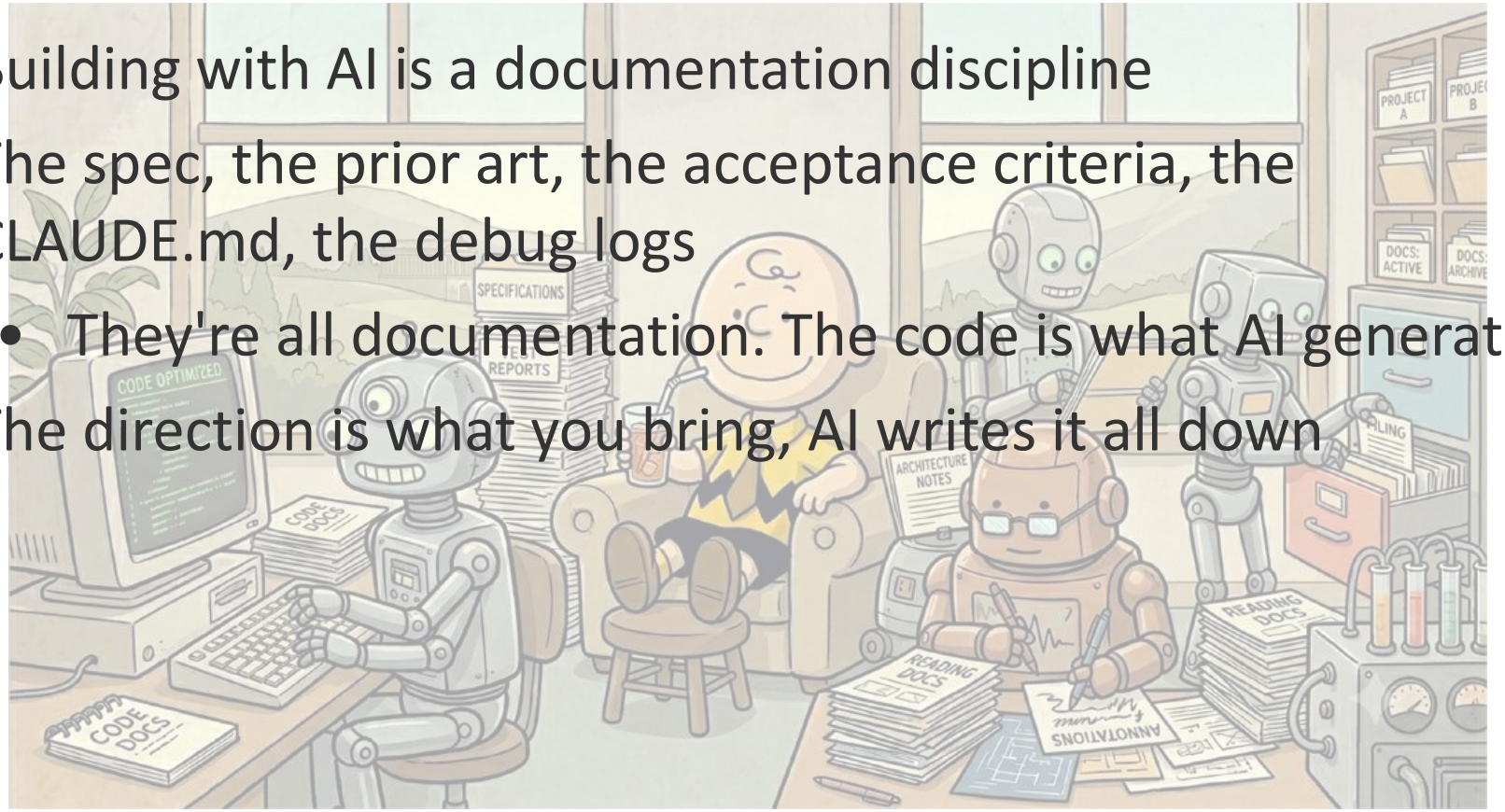
The Methodology

The Proof

Your Path

Practice 16: Document Everything

- Building with AI is a documentation discipline
- The spec, the prior art, the acceptance criteria, the CLAUDE.md, the debug logs
  - They're all documentation. The code is what AI generates.
- The direction is what you bring, AI writes it all down



AI Just Made the Typing Free

- If you've been telling yourself you're not a coder, that's not your role
- Your role: the thinker, the architect, the one who knows the reasons
- The typing was never the work. The thinking was.

The Runbook: 16 Practices

1. **Set up source control:** Step Zero. Git + GitHub, GitLab, or Azure DevOps Repos.
2. **Organize your project:** Decide where source, docs, tests, and reference samples live.
3. **Stay in the driver's seat:** Know more about your app than the AI does.
4. **Be the architect and the librarian:** AI works, AI teaches. You decide and you save.
5. **Teach the teacher:** Use AI to learn what you don't know. Save it as standing knowledge.
6. **Don't go to Hawaii:** Build what you can test, test what you build. Small chunks.
7. **Reuse verified patterns:** Prior art from your projects and others.
8. **Write the briefing package:** Spec, prior art, acceptance criteria, tests.

The Runbook: 16 Practices

9. **Capture evidence, not symptoms:** DebugLogger pattern. Real data, not descriptions.
10. **Review with fresh sessions:** The four ninjas: Reviewer, Architect, Hunter, Historian.
11. **Set up the AI as a collaborator:** Configure it to push back. CLAUDE.md persists across sessions.
12. **Use the stop button:** When AI goes off track, intervene.
13. **Failures and partial successes are data**: Capture what you learned before you roll back
14. **Manage your agents:** Find what works. Stay with it mid-project. Run parallel when ready.
15. **Trust your domain, baseline, and intuition:** The three things only you can bring. The human moat.
16. **Document everything:** This is what AI is good at, use it.

Four Steps for Tonight

1. **Install Claude Code:** Five minutes. Low cost to start.
2. **Pick a project you already understand:** Something where you know the domain. You're the architect from day one.
3. **Write one paragraph describing what it should do:** This is the hardest part, and you already know the answer.
4. **Ask Claude to turn that paragraph into a spec:** Once you have that, you've given Claude actual direction.

Things I Hear

- **"What about hallucinations?" –** The methodology handles them. Verify every chunk. Test every feature. Debug loop. Don't trust blindly.
- **"I am not a developer" –** You don't have to be. Be the thinker in the room. The idea generator, the decision maker, the dreamer.
- **"What if the tools change?" –** They will. But specs, prior art, acceptance criteria, review -- those work with any tool. Practices outlive products.

Resources

- GISAndYou Blog Article:
  Developing with AI: A Success Guide for GIS Professionals

QuerySimple Is Where I Bled

- You're about to start your QuerySimple
- I hope it’s less painful than mine was 
- Thank you

Questions?

LinkedIn

Adam Cabrera | adam.cabrera@kingcounty.gov | adam@mapsimple.org