Handwave ai

HANDWAVE AI · A NOTE ON BUILDING WITH AI

Where your business keeps its memory

Nobody loves governance. We care about it because the decisions the business runs on end up somewhere nobody can read.

Aaron Nelson · Founder & CEO, Handwave AI
aaron.nelson@handwaveai.com · handwaveai.com

We all had a Sam

They were the person who had everything in their brain, years of organically developing code perfectly designed to their job. Logic carefully curated to fulfill exactly one use case, and it provided that stability day over day.

Then Sam moves on. Or maybe Sam finally got that promotion. Or perhaps the system grew to the point that Sam could no longer hold it all in their head.

Then the pain comes. Do we unwind the things? Do we need to build new? How do we relitigate every decision in the library upon library of code?

Most places pick a fourth option, which is to keep running code nobody understands and hope nobody asks.

What it costs

> The thing Sam built over the past year is going to cost half a million dollars to rebuild properly.

> A regulator, a lender or an auditor changes what a number has to mean, and Finance needs that definition changed in reports now used across the organization. Nobody can say where all of them are.

> Sam, reached on the phone, cannot remember the equations behind the investor reporting from five years ago.

> Marketing quietly builds their own version in a spreadsheet, and now there are two numbers and a standing argument about which one is real.

> You stop asking new questions, because every new question means a new build; nothing gets reused, everything gets made again from scratch, and the build is never small.

Now give Sam’s job to the thing writing code today

The logic lives in the repo and in the “head” of a model, which may or may not have memory of the equation it placed in your pipeline, or of the calculation it rendered on your website just so.

Sam was where the business kept its memory. The model is where it keeps it now, except this memory resets between questions. Sam could be asked, and Sam answered the same way twice. Ask again next quarter and the model may decide differently, and nothing about the output will tell you that it did.

> Sam’s logic was undocumented and at least consistent. The new arrangement is undocumented, inconsistent, and faster than anyone can review.

Maybe Sam helped onboard the model, with carefully crafted prompts and documents; the business rules, the edge cases, the reasons behind the odd ones. Context helps the model decide. It doesn’t keep a record of what it decided. Afterward there is no way to know which part of Sam’s handoff the model used, which it inferred around, or whether what it produced reflects what Sam meant or what it guessed from how the data was laid out. What comes out is code, and the code shows what was built, not why.

Which sounds like this

> Why did metric X swing so hard with the last production release?

> The quarterly went out, and a board member is asking why one line does not match what they remember from last year. Nobody changed it on purpose. That is worse than an error, because an error has a cause you can name.

> We're paying someone for a week to find every place metric Z appears, because investor relations needs it changed.

> We only found it because somebody remembered the old number from Sam's report.

And all of that is what happens after the problem is found. Sometimes nobody finds it at all.

02 MAKE THE DECISION VISIBLE

A decision anyone can read

There's one fix I'd fund first. It won't fix your tools or your staffing, but it's the cheapest insurance an AI budget can buy.

Decide what a number means once, in the open, with a person's name on it; then keep that decision somewhere anyone can read without being technical.

Start with the handful of numbers the business actually runs on, which is a shorter list than anyone expects. Settle what each one counts, what it excludes, which date it belongs to, and who gets to change it, in a sentence a person outside the technical team can argue with. It can be as plain as this: *Net Sales = Gross Sales less Voids and Comps. Owned by the CFO. Changed only through her, and the change is recorded.* A general manager can disagree with that sentence; nobody can disagree with a query.

Then keep it where it can be seen. A decision that sinks back into the code is gone again, no matter how carefully it was made the first time, and when the code and the definition disagree, the code is wrong by default.

03 LET THE SYSTEMS FOLLOW

And the machine gets better work

Give it the decision instead of asking it to guess. It does the work; a person made the call. Reports carry the definition they were built from, and pipelines follow it.

The powerful part of bringing the logic out of the codebase is that the context is no longer stuck inside the code. Validation can look at where that number shows up across the business, in every report and every system that touches it, and ask whether it still tracks to reality. That is work no person can do at the speed it needs doing, and it is what the machine should be spending its attention on once it is no longer busy reinventing what the number was supposed to mean.

The machine can do the work. It cannot own the meaning.

© 2026 Handwave AI, Inc.