Build a self-improving AI harness

> **Nick didn't write this.** It's an AI compilation of his film *The Self-Improving Harness*, and everything in it comes from the film: what it says and what it shows. “I” and “we” in the film are Nick and his work.

> **Use this guide at your own risk.** Automated workflows can make wrong decisions at scale, expose data in logs and traces, run up model costs or change systems you did not mean to change. This is general technical information, not legal, security or compliance advice, or a promise of results. Follow your providers' instructions and terms, and get qualified review where the work is regulated. This material is provided as-is, without warranties to the extent permitted by applicable law; it does not exclude rights or liabilities that cannot lawfully be excluded. No company sponsored this guide. Product names belong to their owners.

By Nick Reese. For people building real workflows with AI.

The film in a line: when everyone has the same models, your edge is how fast you turn mistakes and learnings into better business processes. A title company ran that loop on people for decades, on six foundations. Map them onto your harness and every failure leaves the business with a better system. Graded against those six, with 36 checks, not one of 108 tools scored better than a C on any foundation.

The plan at a glance

In your harness	At the title company
1. Every step has one job	One owner for each job: the person who read the records wasn't the one who approved them
2. Trace every journey	Every file left a trail
3. A trail of authority	The guidelines always had the final say
4. An evaluator, a model that checks the work	A second set of eyes: the underwriter checked the work
5. One view of every step: see it, test it, swap it	Every claim fed the actuary tables, so the insurer always knew how it was doing
6. Every failure becomes a test	Every failure improved the system

Get those six right, and you'll figure the rest out as you go.

Two ways to use this guide

For your AI: hand it the text version and chat it through with your favorite AI model, so it can walk you through everything in the film.

For you: read or print the PDF.

1. Every step has one job

At the title company, nobody did the whole job. There was no super human who read every record, drafted every exception, approved it and underwrote it. The person who read the records wasn't the one who approved them. In your harness, every step has one job.

A lot of the tech community builds the opposite: super agents. A super agent is one big agent, handed every tool, every rule and a massive task. People reach for super agents because the models keep getting smarter, and good enough is tempting. For business processes, things need to be right, so that is the target, because in most cases, once a business harness is built, nobody is watching it.

So when a run goes wrong, you have to find where. At the title company, you'd ask: did the search miss it, the review, or the underwriter? With one big agent, there is no step, just tool calls.

What happened on USA Arms, Nick's firearms directory: one run was fed the lead data for a range with “ACOG” in the name. An ACOG is a type of scope used on rifles. The workflow found the most authoritative page on ACOG, chose the American College of Obstetricians and Gynecologists, and snowballed it into a shooting range. That was with a state-of-the-art agent.

The usual fixes are to cram more rules into the context, preprocess the input, or watch tool calls and try to prevent them there, hoping that case is solved, just for another one to pop up.

Breaking each step into one job makes it testable. Yes, each step is less intelligent than one big agent. But moving “is this firearms related?” into a step of its own makes it fully testable.

2. Trace every journey

At the title company, every file left a trail. You could pull any file, at any point in history, and rebuild it: what went in, what got decided, what happened after. Your harness should do the same.

- Record what goes into each step and what comes out, for every case. That's tracing a journey.
- Start capturing logs today. Most off-the-shelf workflow tools have this capability if you enable it.
- Then wire up a tool that lets you, or your agent, inspect the exact details of every run.

What happened on USA Arms: without tracing, Nick never would have figured out how ACOG got into the system. The auditor approved the website because it was handed part of the page with “Gun Violence Prevention” language, and accepted that as “firearms related”. You can't learn from a run you can't reconstruct.

3. A trail of authority

If you take one thing from the film, take this.

At the title company, the underwriting guidelines outrank any one examiner's call, and an exception gets signed off and written down. Your harness needs the same thing: human-level product decisions documented, referenced and used as authority for all agent decisions, just like the underwriter's guidelines.

- You aren't after a change log. A change log tells you what happened, not the decision context that informed the change.
- The product goal comes first, then the owner's decisions, then the tests and rubrics, and the agent's own decisions last.
- Rubrics are huge leverage, so spend your time there. Like the underwriter's guidelines, they're authority, but in a perfect world they're derived from human decisions, so they don't live at the same level of authority.
- Each step gets a decision log, and every entry is pinned to the goal, the owner's decision or the test behind it.

What happened on USA Arms: one decision, “anything that isn't about firearms is not relevant”, lets the agents figure out the tedious edge cases, like ACOG.

Without a trail of authority, you get ghost context: the decisions, assumptions and reasoning that shaped a system, but weren't carried forward to the next person or agent working on it. Authority-backed decisions and documentation systems are half the battle in managing it. Nick covers ghost context in detail in his video on scaling judgement.

4. An evaluator

At the title company, the underwriter was a second set of eyes. They checked the examiner's work against the guidelines, and caught what the examiner missed. In your harness, that's an evaluator: a model that grades each step's work against a rubric.

- A rubric is a written list of what good looks like. The rubrics come from your trail of authority, and a state-of-the-art model can help pull out the finer details, in line with your decisions.
- The model doing the auditing shouldn't be the model doing the work. Use whatever model is best at auditing, just like you wouldn't hire a web designer to do your bookkeeping.
- There are small, fast models that are great at auditing. For now, the hardest edge cases still take a state-of-the-art model.

5. One view of every step

At the title company, every claim landed in the actuary tables, and the insurer watched its loss ratio: what it pays out in claims against what it takes in. When that number moves, somebody goes looking for what broke. Your harness needs the same thing.

- Zoom into every step and see the timing, the testing and the long-term flows for each type of entry, and how it fits into the overall system.
- Track what changed at each step, why it changed and how.
- See it, test it, swap it.

If you don't have this view, you're stuck asking your agents how things are going. With it, when a number moves, you and your agents can see why, instead of spending forty-five minutes digging for it.

6. Every failure becomes a test

At the title company, every miss became a new check. Say an examiner misses a lien because the county indexed the company's name a little differently. They track down the miss, save that naming exception and add a check, so every search after that catches it.

Your harness should do the same: every failure becomes a test case, and you keep and learn from every one.

- When the evaluator shows you where a step is failing, zoom right into those cases and test something different.
- That something different can be a different agent model, a decision model, input from a deterministic system like normalizing the business name, a classifier trained on your own data collected by a state-of-the-art agent, tweaking the prompts, changing the tools, or anything under the sun. It's rarely one of these alone: in practice, a step runs a stack of them together.
- Replay the same cases through each one, let the evaluator grade them, and plug in the winner. Because each step has one job, nothing else has to change.

You can train classifiers on your own data that you had a state-of-the-art model collect. If that isn't on your radar, write it down.

For self-improving business harnesses, this is the core functionality you're after.

Put it together

Put those six together, and you have the title company's loop, running on agents.

1. A failure shows up in your numbers.
2. You trace the journey that failed, and find the step that got it wrong.
3. You read the decisions behind it.
4. You save it as a test case, and replay that step with something different.
5. The evaluator picks the winner.
6. You plug it in, and write down why.

That's the claim turning into a new line on the checklist. Every failure leaves the business with a better system.

Bootstrap it with a state-of-the-art model

If you're starting from scratch, you don't need all of this on day one. Focus on great logging, clear authority and a super agent that you know you'll refactor. This is where one big agent shines.

1. Hook in a harness like pi-agent-core, the thing that runs the coding harness Pi, and let a state-of-the-art model discover the problem space.
2. In Nick's experience, you can get to “good enough” pretty quickly this way: maybe forty to fifty percent accuracy, on a really complex task.
3. As you go, you discover the shape of the work. Break it into stages, and make sure you have all six foundations.

4. Have state-of-the-art agents help build the system, so local agents can run it. That's the end goal: you own your system, and the system can't change on you.

Tools you can start with today

As you move from super agents to workflows, you've got to decide where your system will run, and what tradeoffs you're accepting with each option. Nick had agents grade 108 tools against the six foundations, with a rubric of six checks for each foundation, every tool mapped onto all 36: 3,888 judgments.

- Not one of them scored better than a C, on any foundation.
- For the trail of authority, 107 of the 108 didn't have enough evidence to even grade.
- On everything else, the primitives, Windmill and Mastra came out on top.

Nick has used both. A year ago he gave up on Windmill, because he couldn't find a way to build version-controlled TypeScript that worked well with its system. He ended up writing his own compiler, and it all felt too hacky, so he went back to Mastra. It has almost no trail of authority, but it does the primitives, and he scaled it until he could write and iterate on his own system.

Nick's recommendation: write your own, in Rust or TypeScript. If you're using TypeScript, grab Mastra, and know you'll outgrow it.

Or, if you're technical enough, fire up Claude Code, ChatGPT or DeepSeek and start working. Have your agent audit your logs and your chat conversations, and bootstrap a series of flows that build the six foundations. Once you have that, everything else will follow.

What didn't work

- **A chatbot with RAG bolted on.** A search over your docs makes it confidently wrong, with citations. Technically that's a harness, just a tiny one.
- **One super agent as the finished system.** When a run goes wrong, there is no step, just tool calls. Start with one, and know you'll refactor it.
- **More rules in the context, preprocessing, or watching tool calls.** One case gets solved, and another pops up.
- **A change log as authority.** It tells you what happened, not the decision context that informed the change.
- **Windmill for version-controlled TypeScript.** Nick wrote his own compiler for it, and it all felt too hacky.

The 36 checks

The rubric the film shows: six checks a foundation, each scored 0 to 4.

1. Every step has one job: Bounded responsibility · Input/output contract · Independent invocation · Composable control flow · Replaceable implementation · Failure and effect boundary

2. Trace every journey: Per-step evidence capture · Journey correlation · Durable queryable custody · Version and context identity · Capture limitations and privacy · Trace to case extraction

3. A trail of authority: Owner-source provenance · Authority precedence · Decision to rubric lineage · Execution to authority linkage · Change rationale and supersession · Review and enforcement

4. An evaluator: Separate judge execution · Explicit configurable rubric · Evidence-aware verdicts · Versioned repeatability · Human calibration · Coverage and failure accounting

5. One view of every step: System to stage navigation · Connected operational and quality view · Historical cohort comparisons · See test swap loop · Change and governing rationale · Actionable coverage and access

6. Every failure becomes a test: Failure to durable case · Fresh re-execution · Independent regression grading · Versioned replay controls · Holdout and suite stewardship · Release protection and effect safety

Hand it to your AI

> Audit my logs and my chat conversations against the six foundations of a self-improving harness: every step has one job, trace every journey, a trail of authority, an evaluator, one view of every step, and every failure becomes a test. Then bootstrap a series of flows that build the six foundations.

A title company turned every miss into a better system, and with these six foundations, your harness can do the same.