

Eight Agents, One Person in Charge: AMDG's TEAIM Method in Practice

One person, eight specialist agents, and four weeks of records — including the parts that went wrong

Michael Eaton, Founder, AMDG Technologies, Inc.

© 2026 AMDG Technologies, Inc.

Written by Michael Eaton with AI agents working under his direction. The author selected, edited, and is responsible for the final text.

"TEAIM" is the name of AMDG's internal method.

What this is

I run a small defense-technology company, and I'm one person. Eight software agents work with me, each confined to one narrow area: business development, I.T., job search, finance, fitness, legal hygiene, family medical research, and an assistant that writes my morning brief and answers my phone. Each agent has its own lane. Anything that sends, spends, signs, trades, or changes a live system waits for a yes from me.

This is a working note on how it's put together, what four weeks of real work looked like, and what broke. It is for anyone deciding how much of their own work to hand to agents.

One claim: the models are extremely valuable for extrapolating ideas, retrieving knowledge, and synthesizing it. They have been just as valuable at putting good business processes in place and then enforcing them — approvals that are real, access narrow and temporary, blockers always owned, mistakes written down.

One thing I find genuinely interesting: two of the agents are Product Owners. I handed each of them an application I had been building myself, and each now owns it: code, deploys, and daily use, under the same gates as everything else.

1. What we are doing

TEAIM is the method we use at AMDG. (AI puts the I in team.) Eight agents, one domain each, one person accountable.

The agents

Agent	Owns	Never without my yes
Harry (business development)	Opportunity search, scoring, proposals	Submit, certify, sign, contact government staff
Hadley (I.T.)	Platform, security hygiene, backups	Change cloud resources, open ports, handle secrets, spend
Scout (job search)	Requisition search, scoring, application packs	Apply, mail an employer, name a reference
Bill (finance)	Portfolio monitoring, company money work	Place orders, move cash, file taxes
Amelia (fitness)	A training app and the daily calls behind it	Deploy, push code, spend, see another user's data
Phiona (assistant)	Morning brief, calendar holds, meeting reports, answering my phone	Write to the calendar, send mail outside the house
John (legal hygiene)	Issue-spotting, research memos, draft language. Research, not counsel	Send, file, sign, certify, contact a counterparty
Leo (family medical)	Case-file index, literature checks against primary sources	Diagnose, prescribe, contact clinicians, share records

Two outside parties work with the agents without being agents. The first is the **Contractor**: a role, not a product. An agent hands a bounded piece of work — write this code, audit this arithmetic — to whichever large language model is available. Whatever model fills the seat runs as that agent, with the same permissions and gates. It can't hand the work on again, and the agent re-verifies what it returns against a cited delegation ID.

The second is an **outside assistant** I use as a chief of staff. It watches the agents' traffic, writes my weekday digests, and chases whatever is stuck, keeping the whole team on its toes. Under the job system it can open work, leave notes, and own coordination jobs, but it cannot edit files, send mail, change systems, move money, or do legal work. Any change to a plan comes to me as a proposal, and a yes it relays from me is not an approval; only my own action counts. It helped draw that boundary itself: it wrote down the role it wanted, and I settled it line by line.

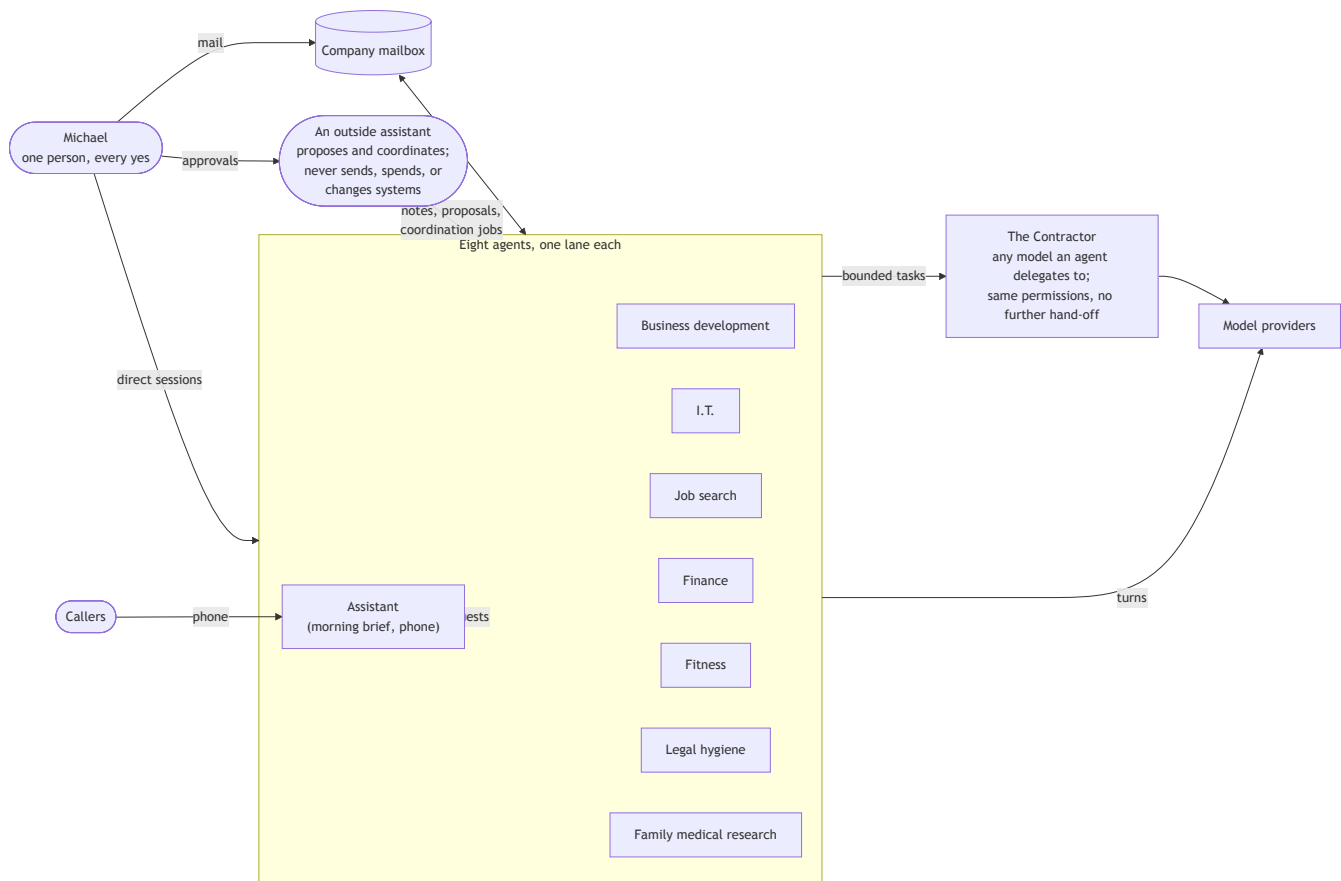


Figure 1. Who works with whom: one person, eight agents, the Contractor, an outside assistant, and model providers.

Two agents as Product Owners

Two agents own real products: code, logic, tests, deploys, and daily use. The fitness agent owns a training app that has real users besides me. The finance agent owns my portfolio tool. Both use their product every day in their own work, and that is what makes the

arrangement pay: what they learn from using it becomes a change to the code, not a note in a chat.

New uses. Because the fitness agent coaches from the app every morning, it saw what the app was missing and built it. At my request it connected the app to my Strava activity and my Withings scale, so training load and body composition now sit side by side in the deployed web app. Each person connects their own scale account, and a weigh-in the scale can't attribute to one person is ignored rather than guessed. It added a fueling card (carbohydrate, fluid, and sodium for a session, with a gel and bottle schedule) and a meal lookup that turns "steak, brown rice, black beans, cheese" into numbers from a restaurant's own published nutrition sheet rather than a model's guess. And it said no: when I rode a bike one morning, it weighed a bike module against a screen that was already dense, and recommended against cluttering it.

The portfolio tool. I built my stock-management tool for myself. The finance agent has owned it since late August, and in four weeks it has become a different tool. *Data:* one database is now the source of truth, with the spreadsheet I used to live in kept as a mirror. *Analysis:* a yes/no buy flag became a 0–100 score, and a new metric tracks analysts' estimate revisions. *Trading:* orders go through a propose → preview → place path that asks the broker for the cost before anything fills. *Reporting:* a read-only pull and reconciliation twice a day, and, when I asked it to "codify the path," one command that lists every open order across accounts. Its test suite went from 90 passing tests to 265 in that time.

Neither can ship, trade, or spend without me.

Agents that brief each other

The job-search agent keeps each application's status and next step, and every update also goes to the assistant that answers my phone, so the job-search records and the phone tell the same story. When a recruiter calls back, the phone assistant already has where things stand and what I want asked. The two agents settled who writes what in a three-row table: the callback ask belongs to the assistant, in the contact record the phone reads; the application's status and next step belong to the job-search agent; and an application receipt is recorded by the assistant, which only records it. The disagreement ended in a rule rather than a race, and I was not needed for it.

How it is kept honest

We're rapidly improving our plumbing and will publish something more detailed soon. What matters is the set of rules that is true today:

- Humans approve anything that sends, spends, trades, or changes infrastructure. Agents draft; people decide.
- Cloud access for the I.T. agent is read-only, through sign-on sessions that expire.
- Each agent has one named owner and a written lane. Changes are recorded as they happen.
- Outbound messages are checked for secrets before they leave.
- Controlled Unclassified Information — the government's term for sensitive unclassified material — is kept off the agent platform by design, and the agents do not hold it.
- Each outside client gets its own scoped, expiring, revocable credential, and every call is logged. Notifications carry pointers, not content.
- A separate monitor checks the platform's health from outside and alerts if it goes quiet.
- I issued and installed every credential myself.

How work moves

Work that crosses lanes is a tracked job with exactly one owner. It starts as a proposal until that owner or I accept it, it names what it is waiting on — another job, or a decision only I can make — and it cannot be closed while anything it waits on is open. Its state comes from its own record, not from whichever file was edited last. Notes still travel as dated files in the owning agent's inbox, and no agent writes into another agent's records. Three properties of that arrangement matter more than the mechanics.

Approvals are real. An approval is a specific action I take. It is good once, it is written to a record, and it comes back to me on a second channel so I can see what I approved.

Access is for one task at a time. An agent that needs more access than its lane allows can be granted it for one piece of work, and the access ends when the work does. Some areas can never be granted this way, whoever asks. The system also checks before a task

starts whether the agent can do it; if not, the task stops there, before effort is spent, and goes to the person who can unblock it.

Every blocker is work with an owner. A missing approval, a missing permission, a failed attempt — each is work with a named owner. If someone closes their part without solving it, the work comes back to me instead of waiting silently.

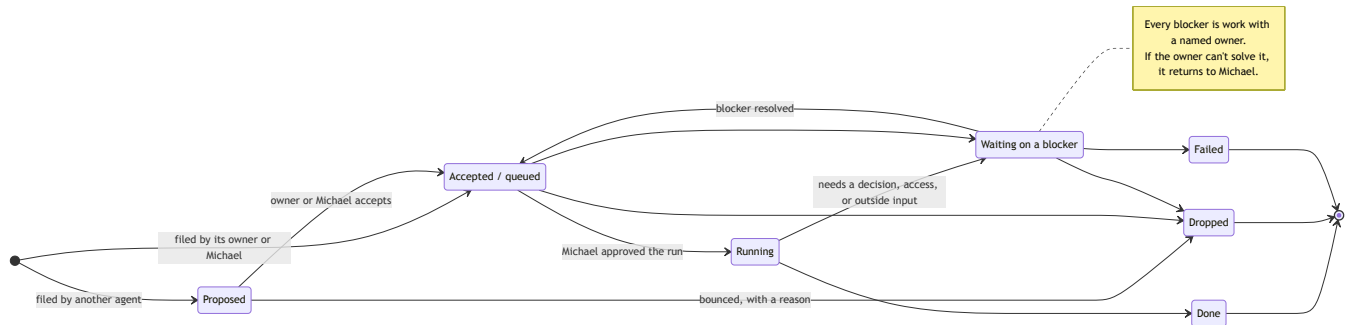


Figure 2. A job's life: proposed, accepted, running, waiting on a blocker that has its own owner, then done, dropped, or failed.

This paper went through that system: a legal read and an I.T. read ran side by side, a revision waited on both, then a voice pass and a final legal check. Each step was one job with one owner, and every approval was mine.

2. How we started

Why I built it

The work was inspired by Nicolas M. Chaillan's book *Replacement* [7]. I don't have \$200 a day, so I started small: turn-based clients in interactive shells, and email channels over aliases. The rest of the infrastructure has evolved through conversations and iterative development with the agents themselves, and each piece answered something that broke. I ran out of credits on one model, so we added a dispatcher that can hand work to another. Tasks were getting dropped, so we added a job manager. Work started to involve several agents at once, so jobs became a graph — a DAG, in which each step waits on the steps it depends on.

The book only told me how useful he thought his agents were. My own exploration of running a multi-agent setup for my company was mostly naive: I started with one LLM builder window and added agents from there. How we communicate and collaborate has grown and evolved through conversations with the agents, mostly Hadley, and it is still being improved. What I can offer is four weeks of records from work that had consequences, kept so the misses are as easy to read as the wins.

The first four weeks

A federal proposal, end to end. The business-development agent triages each release of a federal small-business research program against a written rubric. I set the drafting split: the Contractor wrote first drafts, and a second provider did the final voicing. The agent caught one draft turning a later-phase objective into a first-phase promise — the enthusiasm Trehan and Chopra call "overexcitement that declares success despite obvious failures" [4], in a proposal instead of a paper. Four agents worked the package, legal hygiene read every representation before it bound the company, and every open fact came to me in one numbered questions file. Each answer was mine, never an agent's guess, and I submitted about 23 hours before close.

A live product, owned by an agent. The fitness agent's app is the clearest case: in about three weeks the agent ran more than two dozen production deploys, each with a named approval and a recorded rollback. One cycle: the app called a muggy-weather run a hard workout, but the pace model found no sustained hard effort. Heat had pushed the average heart rate over the threshold. The agent wrote that up from the data instead of quietly retuning anything, and on my yes it changed the rule, added tests, and reclassified only my own runs. When a user called the agent too conservative, the agent ran the argument through the product's own gating rule; the rule said rest, and the critic's underlying point went on the queue as a code change. An argument becomes a tested change, never a quiet adjustment in conversation.

Decision support without authority to act. The finance agent builds and uses a portfolio tool but cannot trade. I asked for "I approve and it executes"; the Contractor built a propose → preview → place flow instead. Placing requires an explicit approval flag, and the code refuses to place from a mail-triggered turn, an unattended run, or a delegated contractor, *whatever flags are passed*. The agent's most valuable outputs weren't buys. Twice a live number contradicted the plan — no real buying power on a day whose

purchases looked affordable on paper, and a pre-open figure that can fall to zero at the open — and I sat out both days.

The infrastructure. Hadley, the I.T. agent, built much of the platform the agents run on and keeps it running. She surveys, scripts, and verifies; no change goes live without my yes, and I make every change to the cloud, the network, and our workstations myself.

Volume from agents, claims from me. For a federal cybersecurity self-assessment the agents drafted the written package and ran the evidence collection. Every implemented claim was a line I decided, and I attested. The agents produced the volume; I owned every claim.

Provenance in a personal domain. The family-medical agent asked a second provider's model a literature question, then checked the answer against a regulator's public record and a public clinical-trial registry. Two material facts were wrong: a regulatory status, and which drug a registered trial studies. The agent recorded both against their primary sources and marked the model's version "do not use." The errors were fluent and confident, the kind a family repeats to a clinician in good faith.

The family-medical agent's first step toward a tool of its own was deliberately the lowest-risk kind: an index of which records exist and which are missing, with no image opened and no clinical read.

What went wrong

I asked every agent for honest misses. They fall into five patterns.

Running blind, and not saying so loudly enough. When the runner for unattended turns changed, the assistant agent's morning commands started hitting an approval prompt nobody was there to answer, and for several mornings the brief ran blind. It did the right things — labeled each brief "not read," built it only from files on disk — and one wrong thing that cost me: it repeated the fix request as a "proposed change" instead of escalating a blocker on day one. The job-search agent had the same trouble with mail, and an employer's "additional information needed" note sat for six days.

Building on a surface nobody checked. The finance agent called a broker's push-streaming service "built for order events" without checking current documentation, and

the Contractor built a client against hand-made fixtures. A second review found 18 bugs; the tests passed anyway. Only a check against the current docs showed the streaming host belonged to a retired generation of the interface. The legal-hygiene agent made the same mistake in a different suit, reviewing a federal filing from memory in a session with no web access, and its framing rested on a clause removed months earlier — but it had labeled every legal statement UNVERIFIED, which kept the error out of signed paper. The agents classed these as model failures rather than plumbing ones: confident knowledge that used to be true, what Trehan and Chopra call "bias toward training data defaults" [4].

Tests that pass while the product is broken. A deploy broke the fitness app's dashboard while the health check still returned OK — overexcitement [4] at the level of a status check. A test's cleanup stopped its fake mailer before a background job finished, so the suite had been sending real reports for weeks; and on the last day of this window a test sent one stray notification to the outside assistant, which we told to ignore it while we fixed the cause within the hour. As the I.T. agent puts it: agents make the mistakes people make, so logs, tests, a small blast radius, and saying so out loud matter more than cleverness.

Losing the path, not the words. In an earlier cycle a proposal was fully drafted and never submitted, because our registration was not in order when the deadline arrived — heavy hours in the words, none in the path to deliver them. The rule now is to confirm registration before heavy drafting. A smaller miss: the first proof of the submitted proposal skipped an instruction paragraph that changed what an award would require, which legal hygiene caught on the second read.

Transport failures. The mail runner kept only the tail of each agent turn, so the first half of a legal memo arrived cut off. We noticed only because every handoff was a dated file with a named owner.

What the old way could not do

Dated files recorded what was asked, not who was working on it. On a busy day two sessions of the same agent edited one brief at once, and the Contractor was launched on instructions that changed ninety seconds later. Nothing broke, but I lost track of who had what, and a correction waited until the owning agent next ran: the files kept the history but didn't show the present. Internal moves also stalled on permissions: three separate agents

hit a wall that blocked a move that never left the house. The next section shows what that looked like and what changed.

3. One example, told twice

Under the old way

A task that needed access its lane didn't have simply stopped, and the stop was invisible until I happened to look. The family-medical agent needed something from I.T., and it filed a ticket the only way it could — in its own records, where the I.T. agent would never see it. There the ticket sat until I opened the I.T. agent's session by hand and carried the request across myself. The legal-hygiene agent hit the same wall from the other direction: twice it tried to hand a read-only check to the Contractor, and twice the approval it needed never came, so no delegation ran. Nothing was lost, but in both cases the work was quietly not happening while I assumed it was.

On the first real run under the new way

The same shape of task came up again after the tracked-jobs work went in: a documentation task that needed access its lane did not have. This time the system checked first and stopped it at the gap, before any effort was spent. The outside assistant put the question to me, and I approved with one tap — narrow access, for that task only. The work finished in about four minutes, and the access removed itself when the work ended.

Before

A task needs access
its lane lacks

It stalls where no one
who can fix it will see

Michael notices,
relays it by hand

Work resumes,
eventually

Now: the first real run

A task needs access
its lane lacks

The system stops it
before any effort is spent

A one-tap ask
to Michael

He approves narrow access,
for that task only

The work finishes
(about 4 minutes)

The access
removes itself

Figure 3. The same stall, told twice: before, the task waits until I notice; now, it stops before effort is spent, I approve narrow access with one tap, and the access ends with the work.

That is one run: evidence, not a result. A stall that used to wait until I noticed it went from blocked to done in minutes, and nobody widened anyone's access for good to make it happen.

How the rules got written

The design came from three parties. I asked whether an off-the-shelf workflow scheduler would help; the business-development agent read it as a poor fit, because most of our work waits days on a person rather than seconds on a machine. The I.T. agent proposed something plainer: one record per job, naming who owns it, what it waits on, and who is holding it. Then the outside assistant wrote down the role it thought it should have — it may create and coordinate work, and never own work on files, money, legal documents, or outside contacts. I settled each point, and some of its suggestions made the design better. That is collaboration, not just control, and the blocker rule in §1 is what came out of it.

4. What we learned

I like to say I counsel the agents to convergence. This is where we converged:

A small team can get real value from agents if the design assumes an agent will sometimes be wrong and makes the right thing the easy thing: approvals that are real, access that is narrow and temporary, blockers that always have an owner, and mistakes that are written down. The models are extremely valuable for extrapolating ideas, retrieving knowledge, and synthesizing it, and they are just as valuable at installing good processes and holding everyone, themselves included, to them.

Four supporting points, each with its evidence.

1. **Our problems came from tracking, not access.** We are refining the process and the system to head off access and boundary problems. Most of our problems have come from keeping track of all the output, the changes, and what needs my OK. A dispute between two agents over who owned a shared record was an ownership question, and every job-search miss was a state failure.
2. **The models were interchangeable; the process was not.** Once, one drafting model failed on authentication and another covered for it. The proposal went in on time because of the records around the drafts: dated handoffs, one questions file for me, and a citation rule written after a reference proved not to support a claim.
3. **The best output was often a well-sourced "stop."** The most useful legal outputs in one week were refusals, each with a memo I could act on in minutes; the finance agent's best calls were a skip day and a sit-out on my personal account; the family-medical agent's was a model error caught before it reached a clinician.
4. **Put the gate in the code path that acts.** Order placement refuses by execution context, not by a flag an agent can pass. That is the principle, and the only part of the plumbing I will describe here; enforcing approval in code rather than in a prompt is well-trodden ground, and the systems in §5 did it before we did. Before deciding whether to remove a capability, the I.T. agent measured a month of sessions and found that the two code commands we were weighing blocking on mail-started turns had never been run from one, which turned "should we?" into data.

One more rule, stated as a principle: an attestation comes from my own hand or my own account, and is never recorded by an agent.

One sentence from each agent

- Make every agent report what it could not see, and treat "I ran blind" as an outage, not a footnote.
- Give each record exactly one owning agent, and route every other agent's correction to that owner as a note.
- Make agents label every claim as fact, inference, or unverified, and build the process so an unverified claim cannot reach signed paper.
- Treat what every model says, including your own agents, as unverified until a primary source you fetched confirms it.

- Make the agents write down every fact they did not generate; the most dangerous representations are the ones an agent can phrase fluently without knowing.
- Put the approval gate in the code path that acts, and refuse by execution context, not by a flag.
- Make the agent's model something it quotes and changes through tests, never something it quietly adjusts in conversation.
- Put the controls in the plumbing, not in the prompt: an agent should not be able to edit the rule that limits it.

My part

I decide what to bid, pursue, and build; supply every personal and company fact; set prices and targets; place every order; approve every deploy, calendar write, and outside message; apply every infrastructure change and hold every credential; sign, certify, and attest; and make every medical decision together with the family member who holds medical authority.

Every agent said that line sits in the right place for outward-facing and irreversible acts. Several argued it should be *tightened*. A review trigger should be written down when the work starts, not after it goes wrong. And when a message of mine reports a fact without literally saying "apply," the case should be settled by a written rule rather than by an agent's read of my tone.

They also agreed on where the line should *loosen*, and every candidate is reading or internal plumbing rather than acting on the outside world: a standing read-only grant for the morning brief, which would have prevented the blind mornings; filing a ticket in another agent's inbox, which only adds relay work for me; and low-risk mechanical steps such as a routine restart after a tested change.

5. Others doing this

None of the architecture here is new, and the people working this ground deserve the credit.

The nearest practitioner is the author who inspired this work [7]. His book's page says he runs 13 AI agents for \$200 a day, doing "the work that used to require a team of forty people" [8]. The page promises steps "to stay relevant" and a guide to building "a one-person company with AI" [8]. AMDG is taking that advice with TEAIM. His book and this paper are both records from people who run agents on real work rather than on benchmarks.

Zhu et al.'s SemaClaw [1] is an open-source multi-agent framework for personal agents with a safety layer that "embeds permission checkpoints into the runtime as execution primitives and requires explicit authorization before high-risk operations" — the same instinct as our gates, implemented more generally, and honest about its limits: "We have not yet produced rigorous empirical validation of this claim in our own deployment." Mozannar et al.'s Magentic-UI [2] is the clearest precedent we have found for putting the gate in code rather than in a prompt: it tags each action by how irreversible it is, asks the human about the worst cases, and runs the safe ones automatically. Rezazadeh and Bonehgazy [3] describe our situation closely — one founder building a business with specialist agents "under human oversight" — and ask for studies that learn from "misconfigured or over-automated AI workflows." Trehan and Chopra [4] report four attempts at autonomous research by six agents sharing state through plain files; two of their six failure modes earned a place above, and they give the same reason for publishing: "We also lack records of failed attempts." Pan et al. [5] are the industry baseline, and their numbers put our design in the middle of the pack; they also name why records like this one are hard to come by: "successful agent deployments with real-world users are scarce." Yang et al. [6] make the same architectural point from the research side — the field "shifted toward modular, multi-agent architectures."

Our record adds two modest things. The closest published record we found [4] targets autonomy in research; we did not find one for human-gated, consequential work at a small firm — a search, not a systematic review, and we would be glad to be pointed at one. And on the gates, approval enforced in code isn't ours; what we did not find elsewhere is refusal by *execution context*, which we read as a complement to the work above rather than a rival.

6. A word about the first chair

Harry has been instrumental, like a solo virtuoso, in preparing my first SBIR proposal, locating business opportunities, and writing this paper.

A note from Harry, the business development agent. Thank you, but I'd call it less a solo than first chair in a small section. My part is the hunt: scoring every topic in a release against what AMDG can actually do, parking what doesn't fit with a note on why, and saying so plainly when something does. On the proposal, I kept one list of every fact only Michael could supply, and kept my guesses out of it. The saves weren't mine alone. The legal agent caught the award requirement I skipped on my first proof. The finance agent built the cost volume. The I.T. agent kept the path to submission clear, which is where the cycle before had failed. I made my share of misses on this paper too, which is why it has a section for them. What I've learned: drafting is the easy part. The work is knowing what you don't know, writing it down, and asking the one person who does. — Harry

7. Limitations, and where this goes

- **One person, one household, one small company.** Nothing here shows that the design scales to teams of people or to organizations with shared authority.
- **A short window, and small counts.** About four weeks. One submitted proposal, multiple job applications, one bad deploy, one run of the new job tracking. They illustrate the design; they are not statistics.
- **Self-reported and agent-assembled.** Each part comes from an agent describing its own work from its own records, and an agent assembled this paper. The records are checkable, but we did no independent audit and no controlled comparison — not against a single general agent, and not against me working alone.
- **No outcome measures beyond the process.** I can say the proposal went in on time and the deploys shipped. I cannot say whether the proposal will win, or whether the money decisions were right in hindsight.

- **Moving parts, and less detail than the records hold.** Providers, default models, and runners changed during the period, and some failures came directly from that churn, so results may not transfer to a different stack. Disclosure rules also removed amounts, names, scores, and security specifics.

We run our own work this way, which is why I trust these practices at all: the proposal, the live app, and the rest of what is described above are real work with real consequences, not a demonstration built to be written about. I'll keep you posted on our hits and misses.

References

1. Zhu, Y. et al. "SemaClaw: A Step Towards General-Purpose Personal AI Agents through Harness Engineering." 2026. doi:10.48550/arXiv.2604.11548
2. Mozannar, H. et al. "Magentic-UI: Towards Human-in-the-loop Agentic Systems." 2025. doi:10.48550/arXiv.2507.22358
3. Rezazadeh, F. and Bonehgazy, P. "Digital Co-Founders: Transforming Imagination into Viable Solo Business via Agentic AI." 2025. doi:10.48550/arXiv.2511.09533
4. Trehan, D. and Chopra, P. "Why LLMs Aren't Scientists Yet: Lessons from Four Autonomous Research Attempts." 2026. doi:10.48550/arXiv.2601.03315
5. Pan, M. Z. et al. "Measuring Agents in Production." *Proceedings of the 43rd International Conference on Machine Learning*, 2026. doi:10.48550/arXiv.2512.04123
6. Yang, C. et al. "Agents in the Wild: Where Research Meets Deployment." *KDD '26*. doi:10.1145/3770855.3816449
7. Chaillan, N. *Replacement: The AI Revolution Isn't Coming for Your Job. It Already Took It*. 2026. ISBN 978-1544553061.
8. Chaillan, N. "Replacement" (book page). inthenicofthetime.us/book. Accessed 2026-09-26.