PORTFOLIO · SEPTEMBER 2026

AI Transformation Portfolio

Five case studies from building and running an AI-native company

01	The Foundry	Deciding what to build before an autonomous factory builds it
02	Stopping Apex	Five strategies, zero dollars lost: ending a research program on evidence
03	Condor	An AI-assisted business development operations system for federal contracting
04	The operating model	Running the business side of a company where AI agents do the building
05	VistaInspect	Building the evidence corpus before the product
	Five patterns that transfer	

STEVEN FELLOWS

Co-Founder, AI Strategy & Operations, Jiyu Technology Inc

stevenpfellows@gmail.com · 206-483-4426 · stevenpfellows.com

How to read this portfolio

What Jiyu is, what I do there, and what I do not claim.

Jiyu Technology Inc is a two-founder AI company incorporated in Delaware in May 2025. It builds and operates software through a gated, largely autonomous delivery system: AI coding agents build, cross-model review checks, and a named human approves each release. My co-founder is the engineering and systems authority. I am the business authority: I hold spend and go/no-go decisions, write and co-write the specifications AI builders work from, administer the machines with him, and run finance, legal, tax, and vendors.

I am not an engineer, a data scientist, or an ML researcher, and nothing in this document claims otherwise. What I did in each case is stated in the words the company's own records use. Where a result belongs to my co-founder or to the team's pipeline, it says so.

The reason these cases matter to an enterprise is that the problems are the same ones an AI program faces at scale: deciding which use cases deserve investment, killing the ones that fail, keeping humans accountable inside automated pipelines, choosing between frontier and local models on cost and privacy, and building evidence before building product. The scale here is small. The decisions are not.

> Claim standard used in this document: Built (code exists) · Deployed (running on a host) · Wired (connected to real inputs) · Demonstrated (ran end to end on real input at least once, with artifacts) · Validated (gate metrics met and a named human signed off). A present-tense capability claim requires Demonstrated at minimum.

WHERE EACH CASE COMES FROM

Every figure is taken from the company's own records: the Foundry idea register and decision log, the Apex closing memo, the Condor build specifications and decision log, the Code Factory role assignments and capability-claim standard, and the VistaInspect project overview and hydration specifications. Dates are the dates in those records. Where my co-founder made a decision or built a component, the case says so.

01 The Foundry

Deciding what to build before an autonomous factory builds it

PERIOD
June – September 2026

STATUS
In production use inside Jiyu

MY ROLE
Designed, directed the build, Business Authority on every run

IN ONE LINE
10 ideas · 5 killed · 1 graduated · 0 advanced without pre-registered bars

BUSINESS PROBLEM

A two-founder company with more ideas than capacity and no neutral way to choose among them. Every idea arrived with an enthusiastic sponsor, usually one of us. AI research could be made to say almost anything. And the cost of a wrong "yes" was months of build time in an autonomous factory that would happily build it. We needed a decision process that would survive our own optimism.

WHY AI WAS APPROPRIATE

The expensive part of evaluating an idea is the research: market sizing, competitor mapping, unit economics, regulatory checks, comparable failures. Multi-agent research does a week of analyst work in an afternoon. But AI research is also cheap to bias: ask a model whether your idea is good and it will find reasons. AI was the right tool for the research. It was the wrong tool for setting the bar or making the call. The design problem was keeping those two things apart.

SYSTEM AND APPROACH

The Foundry is a seven-stage pipeline (Intake, Framing, Research, Scoring, Verdict, Conviction, Graduation) with exactly two human gates. At Gate 1 the Business Authority signs the decision bars an idea must clear before any research runs: a score floor (default 65 of 100, often raised to 70), the critical dimensions where low confidence is disqualifying, and hard gates such as legality, capital cap, and operating load. Research fans out to five to eight agents across a twelve-dimension rubric; a mandatory Bear Memo argues the kill case; a weighted scorecard produces a score and a recommendation. At Gate 2 the Business Authority reads the dossier, the Bear Memo, and the scorecard and renders PURSUE, ITERATE, PARK, or PASS. Every verdict and every bar change is written to a Decision Log and an Idea Register, and a dashboard reachable from a phone shows what is waiting on a human.

MY ROLE

I directed the specification (drafted by an AI assistant at my direction), set the governance rules, and served as Business Authority on every run: signed the bars at Gate 1, read the research, rendered the verdicts at Gate 2. Two design decisions were mine specifically. When the first design asked me to claim an "upside tier" for an idea at intake, I refused ("that's what the research is for"), so tiers are now assigned from evidence at scoring. When the system offered a static default bar to sign, I called it rubber-stamping; the redesign shipped 2026-06-16 tunes bars per idea and shows the change as a diff for sign-off. The core library, Cowork skill, and dashboard were built by AI coding agents under the company's gated delivery process; my co-founder operates the hosts. I did not write code.

TECHNOLOGY CHOICES	Specification-first build with AI coding agents (Claude Code and Codex CLI); a deterministic core library (77 passing tests) so scoring is reproducible; multi-agent research fan-out against a shared rubric; an Obsidian vault as the system of record for registers, dossiers, and verdict memos; a web dashboard on an internal host over Tailscale. One run consumed roughly 800K tokens across eight agents.
TRADEOFFS CONSIDERED	Rigor against speed: two human gates and a Bear Memo slow a run to days; the alternative was fast and worthless. Pre-registration against learning: a bar fixed before research can be wrong, so we allowed exactly one post-research path, a logged bar refinement in the Decision Log, visible rather than silent. Default bars against tuned bars: defaults are easy to sign and easy to game; tuned bars cost ten minutes of thought per idea. Score against judgment: the scorecard recommends, a human rules, and the log records when the human disagreed and why.
OUTCOME	Ten distinct ideas (eleven runs) between June and September 2026. Five closed PASS, one graduated PURSUE (Ask Jiyu, 81 of 100, all gates passed, built as a factory test article at zero cash cost), three scored PASS and await verdict, and the rest are in research. Scores ranged from 33 to 81. No idea advanced without meeting bars signed before its research ran. A Tesla rideshare-rental idea scored 46 against a signed 70 floor after research found the leased variant contractually void and the owned variant negative before insurance. A copy-trading proposal scored 34 against a 70 floor; I ruled it a signup scheme rather than a business, and the evidence agreed. On that last run the system caught its own failure: a dashboard button had written a stub framing brief with default bars, so my first sign-off was on the wrong document. The register was rewound the same day and I re-signed the correct bars before research ran.
WHAT I LEARNED	The bar you sign before the research is the only bar you can trust after it. Make disagreement mandatory and cheap; the Bear Memo is the most valuable page in every dossier. Governance that is a checklist gets rubber-stamped; governance that shows a diff gets read. And killing ideas early is the product: the five PASS verdicts saved more than the one PURSUE earned.
ENTERPRISE RELEVANCE	This is the AI use-case intake and prioritization problem every enterprise program has: a flood of proposals, sponsors with incentives, research that can be produced on demand, and a portfolio committee that cannot tell rigor from enthusiasm. The Foundry's shape transfers directly: pre-registered success criteria per use case, AI-assisted research with a mandatory red-team memo, a weighted scorecard the committee can inspect, a named human who signs the criteria before the study and renders the verdict after, and a decision log that survives turnover and audit.

02 Stopping Apex

Five strategies, zero dollars lost: ending a research program on evidence

PERIOD
February – June 2026

STATUS
Decommissioned 2026-06-16

MY ROLE
Set the business bar, ruled on pre-registered verdicts, drove the decision to stop

IN ONE LINE
5 strategy generations tested and killed · paper only · $0 real capital at risk

BUSINESS PROBLEM

Apex was a research program: a paper-trading platform for Kalshi prediction-market contracts, built to find out whether a durable edge existed before any real capital was committed. By spring 2026 it had a real engine, a dashboard, a database recording millions of evaluation decisions, and a team that had spent months on it. Live trading was never enabled. The question was no longer "can the system trade on paper?" It was "is there a business here, and who will say no if there is not?"

WHY AI WAS APPROPRIATE

The research and engineering were AI-heavy: strategy hypotheses, backtests against a year of real market prints, and an adversarial recompute of every verdict by a second model (Codex) working from raw data before reading our scripts. AI made it possible to test five strategy generations in four months. It did not make the answer any easier to hear. The decision layer had to be designed so that a human would act on the result rather than move the target.

SYSTEM AND APPROACH

Every strategy generation was pre-registered: hypothesis, kill and advance criteria, and the sample size required, written before the P&L was seen. The first generations (15-minute directionals; eight epochs, about 100 days, 22 hypotheses) reached fee parity and no edge. Strategy 3 (an hourly bracket curve model) reached calibration and stayed unprofitable. For Strategy 4 I reset the bar from "prove capability" to a business outcome: roughly $100 a day of meaningful income. That bar funded a venue-wide census against a full year of real prints (all five taker families failed the confirm bar) and a maker fill simulation (the passive quoter lost money in the venue's two best cells, minus $2.64 and minus $6.31 a day, at every size). A pre-committed exit written on 2026-06-10 said that if both halves failed we stop. Both failed. We stopped.

MY ROLE

I held the business decisions. I set the income bar that replaced the capability bar, ruled on the pre-registered verdicts (recording the taker path as closed venue-wide and approving the maker study as the last live path), and drove the decision to stop when the pre-committed exit fired. My co-founder built and operated the platform and executed the decommission on 2026-06-16, including a later, separate kill of the successor research program in August, which was his call and not mine. Every figure here is from the company's closing memo.

TECHNOLOGY CHOICES

Kalshi live API and official historical candles; a Postgres database that grew to 52 GB (12.5 million evaluation decisions, 263,604 contract settlements recorded); pre-registered probe specs; Codex as an independent adversarial recomputation on every verdict; paper trading throughout, with live trading never enabled.

TRADEOFFS CONSIDERED	Sunk cost against evidence: months of build and a working system argued for one more probe; the pre-commitment argued for stopping, and it was written down first. Capability against outcome: the system could trade, and that fact was worth nothing; reframing to an income bar killed real-but-unbankable findings (a 2 to 4 cent favorite-longshot bias with about $10 to $15 a day of capacity) that a capability bar would have kept alive. Passive collection against full decommission: keeping the data cost storage and operational surface for a closed search; we chose to reclaim 71 GB and archive the repository.
OUTCOME	Five strategy generations tested and killed with evidence. Zero real capital at risk at any point. The engine, dashboard, and API were stopped and disabled, the database volume deleted, 31 open issues closed, and the repository archived. The methodology (pre-registration, pre-committed exits, adversarial recompute, business-outcome bars) was documented and carried into the next research program. The closing memo's own summary: this was not a failed build; every gate the system was engineered to enforce worked.
WHAT I LEARNED	Write the "if this fails, we stop" clause before you see the results, and put a human's name on it. Set the bar as a business outcome, because "it works" is the most expensive sentence in technology. Small samples lie, and so do samples without a mechanism. A well-calibrated model can still lose money; the gate is net-of-cost value, not accuracy. And the most dangerous system is the one that looks like it has an edge on a small sample and talks you into funding it. Apex never did that, because it was not allowed to.
ENTERPRISE RELEVANCE	Enterprises rarely kill AI pilots; they let them fade, which costs more. This case is the template for a pilot exit: pre-registered success criteria tied to business value, a pre-committed stop clause, an independent recompute of claimed results, a named executive who rules, and a closing memo that turns a dead project into a reusable prior. It is also what AI value realization looks like when the honest answer is "no value here at our cost structure."

03 Condor

An AI-assisted business development operations system for federal contracting

PERIOD
Early 2026 – April 2026 (dormant since)

STATUS
Pipeline demonstrated in dry-run mode; no revenue; business not launched

MY ROLE
Designed the business and the system; ran the Claude Code MVP build

IN ONE LINE
about 400 primes ingested · 65 with named contacts · every decision logged with a timestamp

BUSINESS PROBLEM

A veteran-owned small business (VOSB) pursuing federal management-consulting work as a subcontractor faces a research problem before it faces a sales problem: which primes hold contracts at which agencies, which of them need a VOSB sub to meet their subcontracting plans, who the Small Business Liaison Officer is, and which opportunities are worth an hour of a one-person firm's time. Done by hand, that is a full-time analyst's job, and the firm had no analyst.

WHY AI WAS APPROPRIATE

The inputs are public but scattered: SAM.gov APIs, agency subcontracting directories published as Excel, PDF, and web pages, award histories. The judgment is repetitive and rule-shaped: score fit, rank, draft outreach, flag changes. That is the profile where a small language model behind a fixed rubric does well and a human should only see the top of the list. AI was also appropriate for the parts that are tedious rather than hard: parsing a 1,922-row directory into 325 usable prime records, or reading a solicitation and filling in a scoring template.

SYSTEM AND APPROACH

I designed the business first: a seven-workstream research brief (entity and compliance, market and contract fit, delivery model, financial model, bootstrap and go-to-market, automation roadmap, operations and systems), a decision memo, and decision, risk, assumption, and source registers. From that came the operations system: two pipeline tracks, Prime Intelligence (build and rank a prime database, draft outreach, track relationships) and Opportunity Intelligence (monitor SAM.gov and agency directories daily, score and match opportunities to tracked primes), converging in shared stages for teaming, proposals, delivery, and invoicing. Every action flows through a review dashboard that logs decisions with timestamps, designed for VetCert audit compliance. The pipeline runs ingest, score, AI review, human review with no manual clicks. Built as an MVP from an implementation spec written for Claude Code, then extended over several phases (daily digest, activity center, match scoring, saved filters, email templates, CSV export).

MY ROLE

Condor was mine: the business design, the registers, the workflows, the architecture and data-source map, and the MVP implementation spec. I ran the Claude Code build of the MVP pipeline and served as the operator the dashboard was built around. Later build phases ran through the company's gated delivery process, and the migration of the pipeline's inference from a metered API to our local models was a company infrastructure decision rather than mine alone. The business itself was never launched; it has been dormant since spring 2026, and it earned nothing. This is a case study about a tool I built, not a business I am pursuing.

TECHNOLOGY CHOICES	Postgres; scheduled agents; SAM.gov Opportunities, Awards, and Entity APIs; Playwright for directories published only as web pages (DHS: 77 primes, 65 with liaison contacts); SBA and GSA directory files; a web dashboard. On 2026-04-16 the pipeline's model calls were migrated from the Anthropic API to the company's local inference server behind a provider abstraction layer: a 122B mixture-of-experts model for opportunity reviews, a 32B coder model for structured prime reviews, with Claude retained as automatic fallback.
TRADEOFFS CONSIDERED	The first design was backwards, and the lesson is the case study. It started from SAM.gov solicitations, which is how a prime thinks; by the time an RFP posts, primes already have their teams. A subcontractor has to start from the subcontracting directories and the liaison officers, so the pipeline was rebuilt directory-first. Automation against judgment: scoring and drafting are automated, but a human reviews every outreach and every decision is logged, both because the firm's owner had to be visibly in control for VOSB certification and because I wanted to be. Frontier against local models: the local migration cut metered cost to zero for routine reviews and kept the frontier model as a fallback rather than a default.
OUTCOME	Demonstrated: a fully automated pipeline from ingest to human review, running on a schedule, with about 400 primes loaded from three directory sources and 65 of them contactable through a named liaison officer; an outreach dashboard with fifteen shipped features; a dry-run execution plan validating the workflow through outreach. Not demonstrated: real outreach, teaming, or revenue. The build's own status note said it plainly: the bottleneck is no longer the tool, it is the data, since only 65 of roughly 400 primes had contacts. The business is dormant. What survives is a working pattern for AI-assisted business development with an audit trail.
WHAT I LEARNED	Redesigning the workflow mattered more than any model choice; the directory-first fix was worth more than every prompt improvement combined. An audit log is a feature, not overhead, when a regulator or a certifier can ask who decided what. Local models were good enough for rubric-shaped review and free at the margin; the frontier model earned its cost only as a fallback. And a finished tool does not make a business: the honest status of Condor is "dormant, no revenue," and I would rather say that than let a dashboard imply otherwise.
ENTERPRISE RELEVANCE	This is the shape of most enterprise agentic-workflow proposals: scattered public and internal sources, rule-shaped judgment, a human who must stay accountable, and an auditor who will ask. The transferable parts are the design order (business model and registers before architecture), the human-review gate with timestamped decisions, the provider abstraction that lets a program switch between frontier and local models without rewriting the workflow, and the willingness to report a dormant outcome as dormant.

04 The operating model

Running the business side of a company where AI agents do the building

PERIOD
May 2025 – present

STATUS
In daily use

MY ROLE
Business Authority for spend and go/no-go; CFO and CMO; co-administrator of the fleet

IN ONE LINE
First fully autonomous delivery demonstrated 2026-07-06 · ~$40K hardware · ~$500/month subscriptions

BUSINESS PROBLEM

Two founders, no employees, and a plan to have AI agents do most of the software delivery. That creates a governance problem before it creates an engineering one: when an agent says a feature is "done," what does that mean; who is allowed to approve a release; who is allowed to spend money; how does the company keep a record it can trust when most of the work was produced by systems that do not remember yesterday?

WHY AI WAS APPROPRIATE

The economics only work with AI doing the building. The risk only stays acceptable with humans doing the deciding. So the operating model had to be written down in a form both humans and agents could follow: named roles, explicit gates, a vocabulary for status claims, and a knowledge base that agents read from and write to.

SYSTEM AND APPROACH

JiyuOS, the company's written operating model, defines the human roles (Engineering, Product, Business, Marketing, and Infrastructure Authority; a Release Authority who alone holds the merge-to-main gate) and the delivery gates every change passes: spec clarification, decomposition convergence, cross-model code review, a recorded human release approval, deploy, and a post-merge verification window with automatic revert. CI is treated as the one non-agent signal in the evidence bundle. A capability-claim standard governs every status statement with five verbs, weakest true one wins: Built, Deployed, Wired, Demonstrated, Validated. Approvals happen in chat with an agent (Hal) that files issues and reports state. The company's memory is an Obsidian vault with semantic, keyword, and hybrid retrieval, weekly reports written by agents, and decision logs per project. Inference runs on a mix of subscription tools (Claude Code, Codex CLI, Cowork) and a local server, a Mac Studio with 512 GB of unified memory serving five models behind one OpenAI-compatible router over a private network.

MY ROLE

I hold Business Authority for spend and go/no-go decisions, serve as Marketing Authority, and act as Product Contributor writing and reviewing the specifications the agents build from. I run finance, legal, tax, vendor and account administration (including organization ownership with AI vendors), and the tax filings. I contributed to model selection and routing for the local stack and to the company's cost rule: subscription and local compute over metered API wherever quality allows. I co-administer the fleet with my co-founder (daemon installs, access fixes), and I am reachable to the agents through the same chat approvals as he is. My co-founder designed and built the factory, owns systems operation, and holds the Release Authority. Spend to date: about $40K in hardware, about $500 a month in subscriptions, and about $10K in contract development.

TECHNOLOGY CHOICES	OpenClaw-based agent (Hal) for dispatch and bookkeeping; Claude Code and Codex CLI as builders and cross-model reviewers; GitHub with branch protection and required checks; Discord for approvals; a Mac Studio M3 Ultra (512 GB) running vllm-mlx with Qwen 3.5 122B (mixture-of-experts), Qwen 2.5 Coder 32B, a small triage model, an embedding model, and Gemma 4 31B behind a FastAPI router; Tailscale as the network boundary; Obsidian plus a retrieval service as the knowledge base.
TRADEOFFS CONSIDERED	Autonomy against accountability: the factory can build, review, and file on its own; merge to main and any novel deploy still require a named human, because the click is cheap and the decision is not. Local against frontier models: local inference is free at the margin and private, and it is good enough for retrieval, triage, extraction, and rubric-shaped review; frontier models are reserved for work where measured quality justifies the cost. Vocabulary against velocity: making everyone say "deployed, not yet wired" slows conversation and prevents the most expensive failure, which is believing your own status report. Memory against noise: a vault that agents write to fills with stale claims, so we run scheduled maintenance that dates and challenges them.
OUTCOME	The factory was declared ready and demonstrated its first fully autonomous delivery on 2026-07-06; dozens of pull requests moved through the gated chain in July 2026. The capability-claim standard is in force on every status report. The local inference server has served the company's routine model calls since April 2026, and one running pipeline (Condor, in dry-run mode) was migrated off metered API onto it. Finances, taxes, vendor accounts, and legal structure are current. The company operates on two founders' time and a subscription budget most departments would call a rounding error.
WHAT I LEARNED	The governance vocabulary is the product. "Can now" is a claim that needs evidence, and a company that enforces that sentence on its agents will enforce it on its people. Named authority beats consensus: one Release Authority, one Business Authority, and a log. Cost discipline is a design decision made early, not a cleanup done late. And the knowledge base only stays useful if something is paid to doubt it.
ENTERPRISE RELEVANCE	Every enterprise deploying agents will need an operating model, and most are improvising one. The transferable pieces are a role map with named human authorities, a gated delivery chain where the human holds the decision rather than the click, a status vocabulary that stops overclaiming, a cost rule that decides subscription versus local versus metered before usage grows, and a knowledge system with maintenance. This is also the governance and change-management substance behind the words "responsible AI" and "AI operating model" in a job description.

05 VistaInspect

Building the evidence corpus before the product

PERIOD
2026 – present (pre-pilot)

STATUS
Under development; no customers; no revenue

MY ROLE
Customer discovery, inspector interviews, QA planning, reference corpus, vendor and billing setup

IN ONE LINE
3 inspector interviews · 42 real reports from 9 platforms · ~3,900 photos in the reference corpus

BUSINESS PROBLEM

Independent home inspectors spend hours after each inspection turning notes and photos into a report, and the report is the product their client pays for. An AI-assisted report writer is an obvious idea, which is exactly the problem: obvious ideas attract demos, and a demo that writes a plausible paragraph about a roof tells you nothing about whether the system is right about this roof. Before building much, we needed to know what "right" looks like, in the formats and vocabulary inspectors use, across the platforms they already use.

WHY AI WAS APPROPRIATE

Report writing from structured notes and photos is a good fit for language and vision models: the inputs are consistent, the output format is templated, and an inspector reviews everything before it goes out. It is also a domain where accuracy has consequences (a missed foundation crack is a liability, not a typo), which means the evaluation corpus matters more than the model.

SYSTEM AND APPROACH

A reference corpus of 42 real inspection reports from nine reporting platforms (about 2.2 GB and roughly 3,900 photos), collected from published reports and from an inspector who granted usage permission, organized by the canonical sections the product uses (roof, exterior, foundation, electrical, plumbing, HVAC, interior, fireplace, attic). The corpus feeds a sample-data hydration system that pushes realistic reports through the product's own API on every run, so the reference set doubles as an end-to-end test of the backend. Customer discovery ran in parallel: interviews with working inspectors on workflow, report tools, and what they would pay for. Vendor accounts, organization verification with the model provider, payment processing, and a billing policy were set up ahead of a pilot.

MY ROLE

I led customer discovery and the inspector interviews, secured the sample reports and usage permission, collected and organized the reference corpus, planned QA, set up the vendor accounts (model provider organization, payment processor), and approved the billing policy. My co-founder and a paid contractor built the application; the photo-classification result the team later achieved on the corpus (9,440 photos classified at 99.3% accuracy at zero API cost) is the team's pipeline result, not mine. The product has no customers and no revenue, and the pilot has not launched.

TECHNOLOGY CHOICES

Reports and photos from nine platforms (HomeGauge, Spectora, Home Inspector Pro, and others), extracted and mapped to canonical sections; an API-driven hydration system with sixteen report fixtures spanning every lifecycle state; local vision and

	language models for classification and drafting where measured quality passes, frontier models where it does not; a provider organization and payment processor configured for a subscription SaaS.
TRADEOFFS CONSIDERED	Corpus first against demo first: a demo would have been faster and would have told us nothing. Local against frontier for photo work: local classification at zero marginal cost was good enough on our data, which we could only know because the corpus existed. Breadth against depth in discovery: three interviews is a start, not validation, and the project's own overview says so ("marketing copy is not verification"). Pilot timing: launching before the QA plan and billing policy existed would have created customers we could not serve.
OUTCOME	A usable evaluation and hydration corpus; a repeatable end-to-end test path through the product's API; a documented QA plan; vendor, payment, and billing groundwork done; three inspector interviews recorded. No pilot, no users, no revenue. The next steps are a customer cohort and a measured 30/60/90-day pilot, and they are conditional on the product clearing its own bars.
WHAT I LEARNED	The evaluation set is the first deliverable. Getting permission for real data early is a business task, not a technical one, and it unblocks everything after it. Discovery interviews are cheap and the temptation to stop at three is strong; the honest status is "pre-pilot" until a measured cohort says otherwise. And setting up billing before launch forces the pricing conversation that demos let you avoid.
ENTERPRISE RELEVANCE	Every enterprise AI product effort has this stage and most skip it: a reference set of real inputs and expected outputs, in the organization's own formats, with permission handled, before the model is chosen. The transferable pattern is corpus-first evaluation, an end-to-end test that uses the product's own path rather than bypassing it, a QA plan tied to consequences, and a status vocabulary honest enough to say "pre-pilot" in a portfolio.

Five patterns that transfer

The scale of these cases is small. The patterns are the ones I would carry into an enterprise AI program on day one.

---

1 Sign the bar before the study.

Pre-register the success criteria for a use case before anyone runs the research, and show every later change as a diff. AI research is persuasive; the defense is structural.

2 Set the bar as a business outcome, not a capability proof.

"Can the system do it?" is the wrong question. "Does it clear $X of value net of cost, at what capacity, by when?" is the one that kills marginal pilots before they consume a budget.

3 Name the human, and use the weakest true verb.

Every gate has a named authority and a recorded decision. Every status claim uses the weakest verb that is true: built, deployed, wired, demonstrated, validated. "It's automated" is a claim, not a status.

4 Know your cost rule before usage grows.

Decide in advance which work runs on subscription tools, which runs on local models, and which is worth metered API spend. Then measure. The cheapest inference is the inference you never had to call.

5 Build the evidence corpus before the product.

A reference set of real inputs and expected outputs is the first deliverable, not the last. It is what turns "the demo looked good" into "the system is right 99% of the time on our data."

> **Talk to me about any of these.** stevenpfellows@gmail.com · 206-483-4426 · stevenpfellows.com