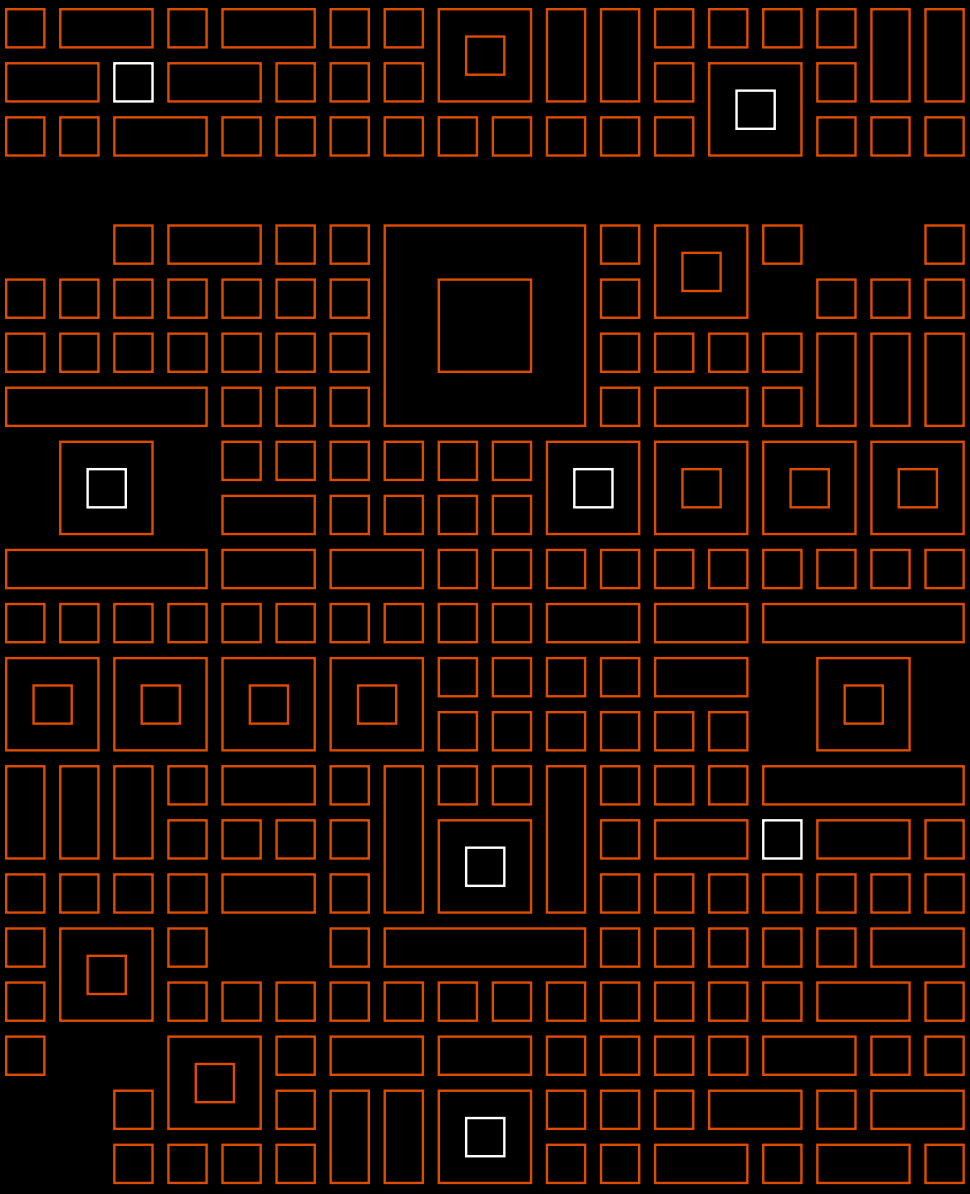


Managing Agents

Deploying AI at scale for predictable, measurable results

aboard

About Aboard

Aboard is a solution engineering firm. We combine AI-accelerated software delivery with the skills and craft we developed across two decades of software consulting.

We don't put AI in charge—we put it to work, building reliable custom software for industry leaders in a fraction of the time it used to take.

We typically partner with clients for a year or more to clear their roadmaps, integrate AI into existing data and workflows, and ship platforms that actually fit their needs.

Why did we write this?

This paper isn't about Aboard, or the work we do for our clients. But agents are still a new-enough technology that everyone is learning by doing—and there are a lot of takes out there. Before we hand you ours, here's why we think it's worth your time.

Three reasons to trust our perspectives:

1. We use agent-based methods all day

We ship software using AI tools and methods. Yes, we might build agents for our clients as a part of larger efforts, but more often, agents are just a component of "classic" software. Other times, we're building in-house agents to help keep our company efficient. All our lessons are won the hard way: Through experience, and trial and error.

2. We've seen (and done) "digital transformation" before

We've provided solutions to thousands of users across Fortune 100 companies and giant NGOs. Time and again, we've seen that most organizations' software problems aren't really about the technology, but about everything else: Strategies, priorities, processes, and culture. We know from experience that agents won't deliver value just by showing up—they need to be part of a greater whole that makes sense. And that's true for big orgs and SMBs alike.

3. We despise obfuscation in tech

We love technology. The tech industry, on the other hand...don't get us started (or do; that's why we have a podcast). Demystifying tech has always been our power move: At our old agency, Postlight, and now at Aboard. Yes, getting agents right means knowing what you're doing. But we're also interested in shifting the power of tech away from the gatekeepers and middlemen, and towards the people who know what they *need* from software—even if they aren't "technical."

Table of Contents


Here's
the
truth
about
agents

Everything you know about business software is wrong

But not for the reasons you've heard.

You know the story, told in a thousand LinkedIn posts. Maybe your boss has even sent you articles. It goes like this:

Agents are *the* way to put AI to work—for real this time. (Forget about those pilot projects that failed, or that whole vibe-coding thing—this time is really real, for real.) This isn't just *SaaS: The Sequel*. Agentic AI can help produce things like an on-demand virtual workforce, just waiting to be released into the hallways of your business. It will quietly, efficiently, and cheaply complete tasks, advance projects, and clean up messes, freeing up your team to do more complex stuff. Heck, you could even downsize, cost cut, and automate your whole operation. And in any case, those agents are coming for your business, maybe even your job, so you had better befriend them or you'll get eaten. That's why OpenAI and Anthropic—alongside Salesforce, Microsoft, SAP, Oracle, and all the rest—are pushing agents down your throat like you're a foie-gras goose. That means it must be real, right? Right?!?!

Not quite. You can start breathing again. You okay? Good. Just know that there is value in using agents, if you know how to grab it. And what is the value, exactly? Glad you asked.

It's (almost) cheaper to build than buy

The truth is, agents *are* different. But not because they're happy, inexhaustible Oompa-Loompas running the chocolate factory 24/7. They are not going to take over and let you (or your boss) sip out of a coconut. Nor are they the little grim reapers of software, coming to take your livelihood.

Instead, they can—and will—change your business's *relationship to software*. That's what this white paper is about. Thanks to agents, the traditional "buy versus build" equation is obliterated. You no longer face a brutal choice between perma-renting SaaS solutions that never quite fit, or spending a fortune you don't have on custom software development. For the first time, building is often easier than buying.

That doesn't mean you can just prompt everything into existence. Agents aren't a magic wand. They're still just software. But if you understand what they're actually good for, they can give you the opportunity—very cheaply, and in a low-risk way—to reassess how your business works.

Don’t panic

Remember when we told you to breathe? Keep it up, you’re going to be fine! This is a confusing time, and everyone feels excited, overwhelmed, or sick to their stomach.

Here’s why you can relax a little.

DON'T PANIC #1

Whiplash is normal

AI went from "impressive consumer product" to "autonomous enterprise software you should have deployed yesterday" shockingly fast. But what about that viral MIT study claiming that 95% of enterprise AI projects fail? Or that vibe-coding startup whose agents erased someone's database? It's natural for a business to wonder: Are we already behind? Or maybe we could just...*not,* this time?

The truth is that you're not behind—but you probably shouldn't wait this thing out, either. Think of it like the stock market: On a day-to-day basis, "agent stuff" will appear to be jerking back and forth all over the place. But if you breathe and zoom out a bit, the general forward motion starts to smooth.

DON'T PANIC #2

Agents are AI now

Pretty much everything labeled "AI" nowadays has agents in it. Even chatbots like ChatGPT or Gemini have agents working under the hood. Claude Code, the best thing—so far—that's ever happened to every developer you know, is agents all the way down. Almost anything new "made with AI" will have agents involved, too. The reason is simple: Doing beats talking. Chatbots are text in, text out; agents are text in, *action* out. They're here to stay.

DON'T PANIC #3

The tech is (relatively) stable

You'd never know it from the river of jargon coursing through the internet at any given moment (harnesses, subagents, "context plumbing"???), but the essential way agents work is unlikely to radically change in the next year or so. That makes now a good time to educate yourself on the basics, and get a sense for what's possible for your organization.

DON'T PANIC #4

Nobody has it figured out yet

This may sound like the opposite of #3, but it just means that the concrete hasn't set when it comes to *exactly* how to get value out of agents. Much of what passes for "best practices" is still just hacks and strung-together prompts. Yes, Big SaaS is pressing "agentic AI" add-ons to all their users, but that's just the status quo holding on for dear life. There's plenty of room to figure out what might work for you, on your own terms, without lock-in or a ton of up-front investment. The timing has never been better.

Understand, assess, engage

This report is not a step-by-step guide on how to build an agent; it's for managers who need to understand from a higher level. If you do want to build, there are tons of tutorials out there—just search YouTube, or ask ChatGPT or Claude to help you. You can also refer to the upcoming "platforms and tools" section for ideas on where to begin your journey.

Getting a handle on agents happens the same way it would for any other business technology. Here's our three-stage process:

1. Understand

What are agents, actually?
What are their key features?
Who are the companies and players I should know about?
How do I create an agent?

2. Assess

What are agents good for?
Who's using agents well?
What does using agents "badly" look like?
Where can agents surface value in my organization?

3. Engage

What do I need to get started?
How do I keep things from going off the rails?
What does ROI with agents look like?
How should my process evolve over time?

Ready? Let's go.

1. Understand

What agents are: Just software

Agents are getting more capable all the time, but the storytelling around them is still ahead of the actual tech. That's because almost everyone describes them in terms of "person doing a job"—usually some variation of:

Agents are like call-center workers who can take actions on your behalf, but instead of people, they're made out of AI sparkles.
-+:-*-:+-

Simple to comprehend? Sure. But it's too open-ended, and sets you up for disappointment. Here's a better version:

Agents are just software.

"But that makes it sound like the stuff I already use!" you say. Correct. Agents are like the software you already use: Spreadsheets, email, PowerPoint, Slack, CRM, ERP. The difference is that using agents can also be like building your own, more *flexible* software. (Something that, until now, you couldn't really do without training, a team, and tons of time and money.) This could be anything from personalized time-savers to department-level automated workflows. In addition to thinking, *What tasks and operations would an agent be useful for in my business?* consider thinking, *What software needs do I have that aren't being met?*

In our case, they inspired a new way of doing business. But let's not get ahead of ourselves.

What it feels like to let software go.

FROM THE ABOARD NEWSLETTER

Software can be temporary

I was a programmer for many years, and the superpower of programming isn't that you can build majestic and resilient systems. It's that you can hack crap together really fast and see what breaks, then hack some new crap together that breaks less. Eventually the hacky crap gets shiny enough that you can put it in a package and tell people it's great.

Where I've really been enjoying Aboard lately is...it helps me learn new things, but second, in that it helps me make ridiculously *small, disposable* workflows. When we think about workflow we tend to think about "publishing a magazine" or "running a loading dock" or "maintaining a website." We rarely think small—which is why so many companies' big process-development projects fail after 18 months. Just last week I talked to someone on their third e-commerce platform migration in three years. That's too many.

I would argue from experience that thinking really small is usually the best way to get started.... Spinning up software, playing with it, and tweaking with it makes me happy. Because the moment I have an idea for something I want to achieve, I'm already underway—the rest is details.

— Paul Ford · Read more

What can you do with agents?

Here are some tasks, processes, and software-shaped things you can set your agents to work on, using low- or no-code tools:

- [ ] Sentiment analysis on social media feeds
- [ ] Create a digest of new podcast episodes
- [ ] Daily email summaries that you can use to guide social media creation
- [ ] Classify and sort documents into categories (invoices, contracts, receipts)
- [ ] Suggest content adjustments for SEO
- [ ] Triage and route incoming emails or support tickets
- [ ] Technical interview simulator (to practice conducting or undergoing them)
- [ ] Automatically repurpose content across different platforms (e.g., Instagram, LinkedIn)
- [ ] Compile and summarize KPI/OKR reporting
- [ ] Brand reputation monitoring and reporting
- [ ] Scan your calendar for upcoming sales calls and compile research reports to aid call prep
- [ ] And, of course, it can code for you

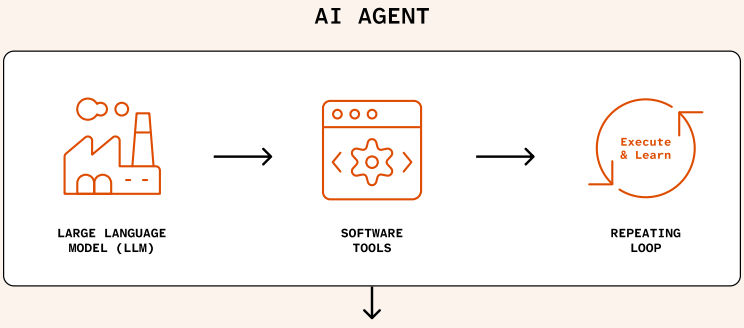
FURTHER READING:

- Product, Platform, Interface, Medium, Language: What Is AI? — John Battelle, 2026-04-06
- "Robot, make me a chair" — *MIT News*, 2025-12-17

Anatomy of an agent

Technically speaking, an agent is a LLM with access to software tools, running in a loop. That may sound complicated—but no more so than saying “cars are engines with access to wheels, running on gasoline.”

Let’s break down the parts:



LLM

What it’s for: Processing text. The LLM turns text into more text by converting and generating it, or into less text by compressing and summarizing it. That’s all. (Yes, AI tools can also spit out images, video, and sound—but let’s keep our focus on text here.)

Examples: ChatGPT, Gemini, Claude.

How it works: Picture the LLM less like a brain and more like the industrial machine it really is. Text (words or code) goes in and gets mulched up into long lists of numbers, which are then pressed back into useful “text-shapes” that come out the other end, like turning bark chips into plywood. All the text going in—whether it comes from you, other apps, or even the LLM itself—is combined into something called “context”, which acts like the agent’s short-term memory. The text that comes out can be in a specific shape (called “structured output”) that can slot into other software to make it do things, much like a quarter going into a gumball machine. *Which brings us to...*

Tools

What they're for: Making the LLM really do things, not just say things.

Examples: Other apps, websites, and hardware—anything with an input field or an API (like Google Search, or your hard drive).

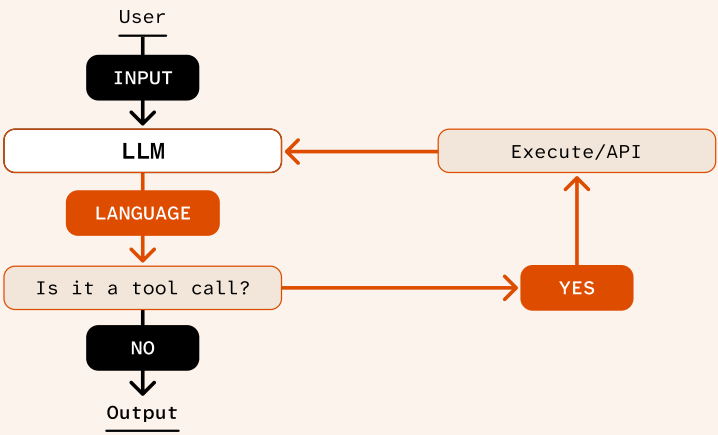
How they work: Structured output from a LLM "calls" a tool to make it do something. The data produced by that tool (say, weather info from a specific zip code) becomes more context that gets fed back into the LLM—which can reformat it for display ("It's 70°F and sunny") or mulch it up into new text-shapes that generate further actions. *Which brings us to...*

Loop

What it's for: Doing things over and over, so you don't have to.

Example: An agent checks your calendar for upcoming meetings, searches the web for information about who you're meeting with, summarizes it into a report, and repeats until there are no more meetings left.

How they work: When you type into a chatbot, it stops after one round of responses—it can't keep going without you telling it to. Agents can repeat actions until some stopping condition (sometimes called a "goal") is met. Loops make agents useful and powerful, but they're also what makes them just software—because all your other apps use loops, too (we just don't make a big deal out of it).



You have to read all of these.

FROM THE ABOARD NEWSLETTER

Terror-free ways to talk about AI

Be the boss. Someone will ask me, a human, for advice on building a software product. Then I will open up ChatGPT, not a human, and use my best prompting: *You are a senior software architect. Define the full plan for building [describe thing here]. Include a complete breakdown of the sub-system and tools you'd use. Prefer open-sourced, highly-vetted tools.* It will produce a thousand words, organized in bullet points, and frankly, it's very good. I'll throw that into an email and then go line by line, giving feedback on it—just riffing, looking for ways to cut scope, prompting further conversation. I've hired many software architects, and, along with my co-founder Rich, I've reviewed their work and offered lots of feedback. It feels right to me, then, that I ask the AI to do something I can evaluate and react to as a very human boss. It accelerates but can be kind of extra; I add value and suspicion. Everyone wins.

Read everything. Related to the above—as I've messed with coding projects, I am finding that ChatGPT and Claude are brilliant little stochastic buddies, but also that every fiber of my being just wants to cut-and-paste their slop into my code editor in the hope that I can avoid any actual effort. This never works, any more than cutting-and-pasting from Stack Overflow worked. Things break, then you ask the bots to help you debug, and it screws that up, and you have no clue what's going on. You've given up agency. The answer is simple: Don't. You have to read everything, and you have to think. Sorry. If an AI makes something and you don't have time to read it, you're using the technology like a chump....

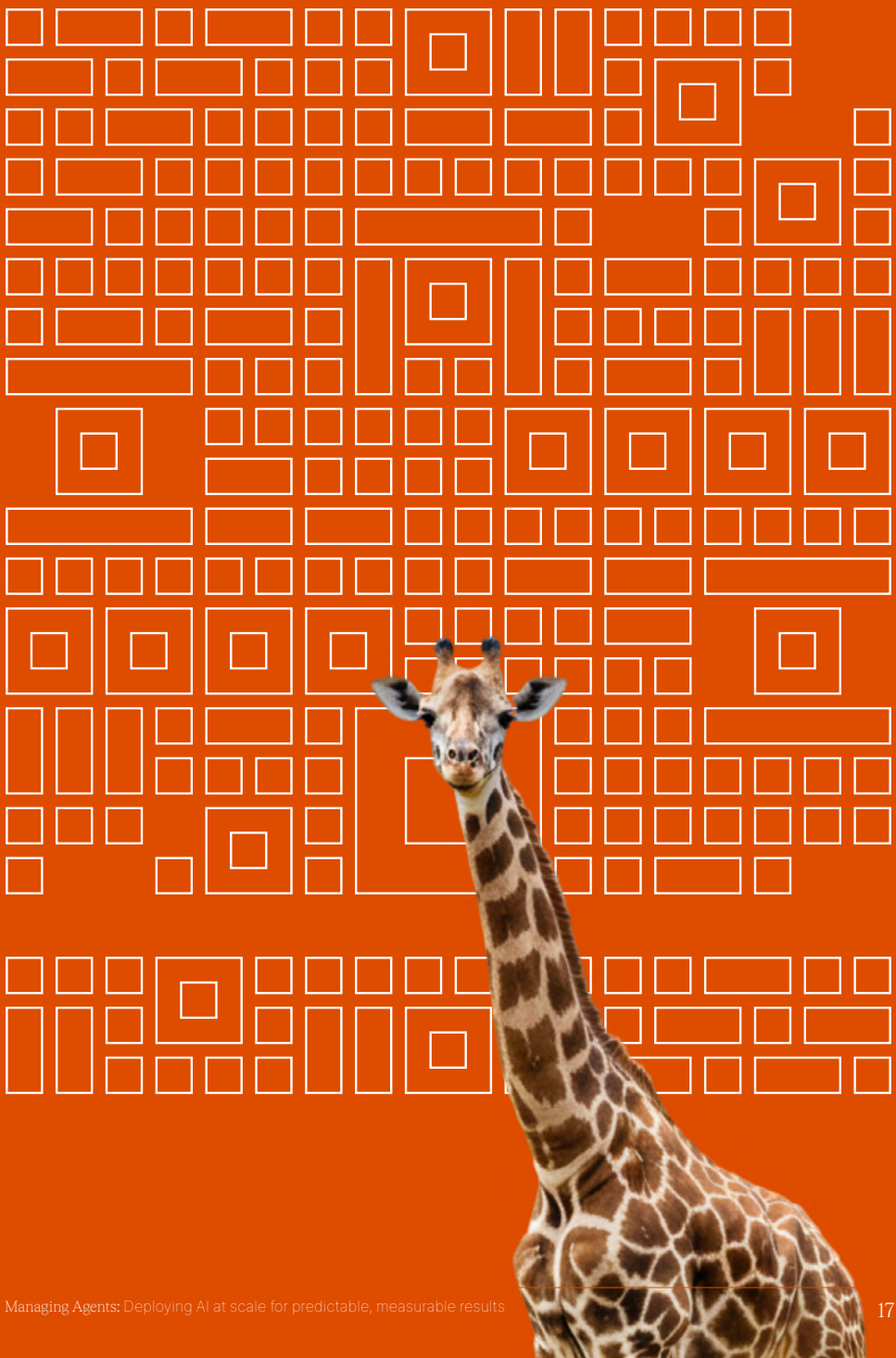
Build pipelines, not prompts. This technology works best as a sequence of transformations that are guided by a lot of thought. For example, in coding, you ask the LLM to break down the problem into an architecture.... Once you fully understand all the pieces and have a mental model, you can ask it to convert those descriptions into code and build an application from there.... The merit of this approach is that it seems to actually work, as opposed to just typing "build me a Facebook clone," which might spin up a bunch of code but doesn't actually work for the long haul. The other merit is that, once you find pipelines that work they're very re-usable.... This is...process. Which is something organizations love and understand.

— Paul Ford · Read more

FURTHER READING:

- LLMs training other LLMs — *Import AI* (Jack Clark), 2026-03-16
- LLM Evals: Everything You Need to Know — Hamel Husain, 2026-01-15

2. Assess



The hidden qualities of useful agents

You can picture an agent in your head now. But to really start *understanding* them—and how to unlock their value—you'll need to keep these important attributes in mind.

Guardrails

Guardrails are instructions, prompts, or other ways of constraining what an agent can do, so it behaves in a predictable way. They turn agents into normal software—reproducible processes that you can debug when they fail. (And they will.) Guardrails are necessary because LLMs are "non-deterministic": They hallucinate by nature, and don't always behave consistently.

The paradoxical thing about agents is the more constrained they are, the better they work. More constraints = better agents.

Here are three guardrails we always make sure to use when building with agents:

- **Tight specifications.** Stuff you add to the LLM's prompt that's specific and "small" enough for the agent to accomplish without getting creative. Think "summarize LinkedIn pages," not "run my marketing department."

- **Structured output.** You can force some LLMs (like ChatGPT, Gemini, or Claude) to only produce text in certain formats—perfectly shaped "pegs" that slot into "holes" in other software. This radically narrows how much an agent can go off the rails. (For example, Weather.com's API will accept JSON data generated by an agent, but not free-associated sonnets.)

- **Constrained environments.** Give an agent access to exactly what it needs and nothing more: Treat it like a line cook rather than a chef. For instance, we never let our agents write code—they can only use certain predefined tools. Another example is preventing agents from accessing the file system of a computer, where they could damage or delete important data.

Constraints are all about reining in agents' "autonomy," which makes them seem less like AI magic. That's almost always good for real-world performance, but it may not be good for marketing. Generally, guardrails are still something you have to put in place yourself, but that's also a good thing, because it means you're in the driver's seat. The good news is if you take your time, agents can do lots of things well. The bad news (for some) is that this isn't an excuse to do almost no work as the robots take care of it all. Please forget anything you have ever read that says a version of "one prompt, job done!"

Parallelization

The key to making agents work at their best isn't just iteration (running them in loops), it's parallelization: running many copies of them at once. An agent isn't a one-time thing. You can spin up dozens, and let them attack tasks en masse.

For example: With one agent, extracting a thousand company names from a spreadsheet, searching each company's website, and summarizing all of that information into a competitive-set report could take hours. But a few agents could do it in minutes. Without parallelization, agents are just handier chatbots; with it, they're productivity piranhas. (Of course, you pay for more AI usage—that's how it goes.)

Parallelization is one of those things that seems big and confusing at first. But it gets easier once you stop thinking of agents as "virtual employees" and start thinking of them as, well, code. The whole point of software is that it's infinitely copyable. You'd open 30 (or 300) browser tabs without batting an eye, and close them again just as casually. You'd use your phone and your laptop at the same time—and send emails or post on social while on both devices. You don't necessarily think of that as "parallel," but it is.

They're there for a very good reason.

FROM THE ABOARD NEWSLETTER

The sacred guardrails of AI

More than a year ago, we realized AI's ability to create code would drastically change our product roadmap. After all, our product, Aboard, was designed to speed up data-driven software development. Our chosen industry, with acronyms like SaaS, ERP, CRM, CMS, B2B, ARPU, and CAC, was in the crosshairs of the new giant LLM companies like Anthropic and OpenAI, not to mention Google and Microsoft.

As a tiny bootstrapped enterprise, we had to adapt—or perish—or adapt then perish, or...what? There were no books and no solid guides, only LinkedIn posts on how to crush it with ChatGPT and YouTube videos about AI passive income. The world offered a vast range of opinions married to an extreme paucity of expertise.

Today, however, we can see a craft of working with LLMs emerging. It usually comes down to one word: "Guardrails." ...

First, zoom way out: LLMs are general-purpose systems that produce *extremely specific nonsense*. The result looks less like what we think of as a classic, "computer-ey" database output and more like a conversation. You can ask an LLM for anything, with no limits: Code, prose, poems, summaries, literally any form of expression. It will almost always produce *something*. That is its nature. We interpret it and assign meaning to its output. That is our nature.

— Paul Ford · Read more

FURTHER READING:

- New Paper: Towards a science of AI agent reliability — *AI Snake Oil (Princeton CITP)*, 2026-02-24
- Agents — Chip Huyen, 2025-01-07
- "Let's Work on the Next Task": Claude Code, GitHub, and the Most Diligent Project Manager I've Ever... — *A Computer Scientist in a Business School*, 2026-03-04

The agent landscape: Who's who at a glance

Almost everyone who currently provides software is now offering agent-based technologies. Here's a tour of the major players.

1. Quick-decision framework

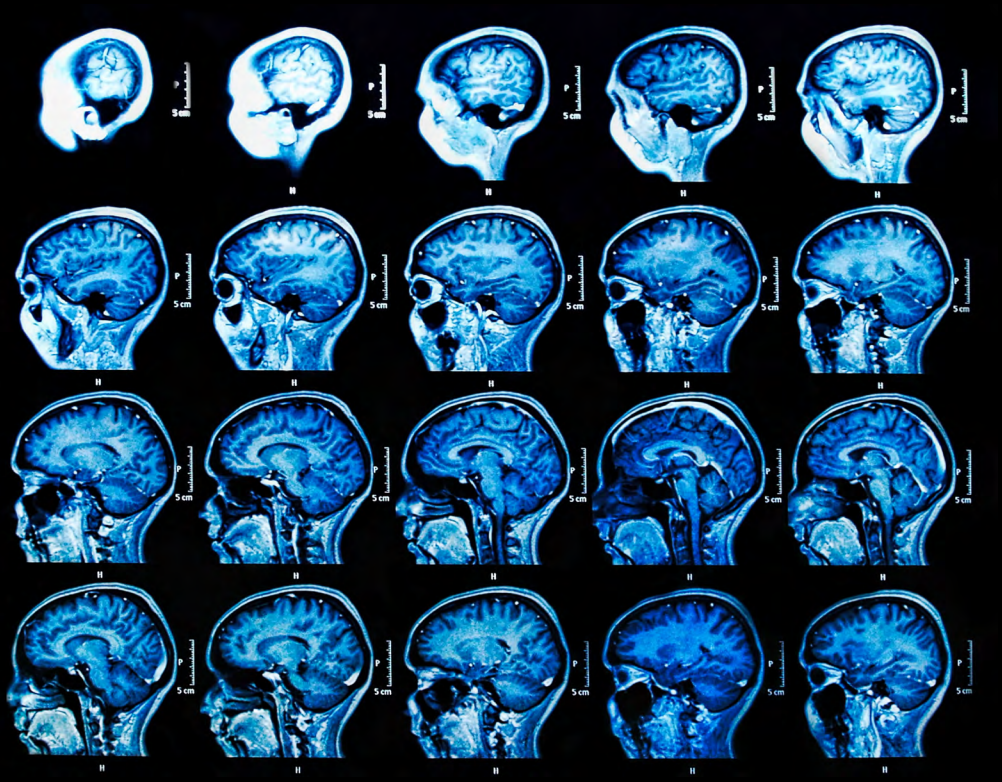
All platforms verified active as of May 2026. Use this to find your starting point, then consult the detailed tables that follow.

IF YOU ARE…	START HERE	WHAT TO KNOW
Already using Salesforce, SAP, or Microsoft 365	Your existing vendor's agent platform (Agentforce, Joule Studio, Copilot)	All three are active and investing heavily. Agentforce runs on third-party models (GPT-4o by default); its value is CRM orchestration, not the model. SAP Joule shipped a 2.0 overhaul at Sapphire 2026 with full pro-code support. Copilot Studio has a 2026 release wave with expanded governance. *Note: Agentforce is limited to the Salesforce data ecosystem.*
A developer building custom AI products	Claude Code or Cursor + your choice of LLM provider	Both are active and growing fast. Claude Code is Anthropic-model-centric; Cursor supports multiple LLM providers—choose accordingly. If you need maximum model flexibility, lean toward Cursor. *Note: Claude Code now writes ~4% of all public GitHub commits.*
A non-technical team wanting quick automation	Zapier Agents (teams) or Lindy (individuals)	Zapier Agents connects 9,000+ apps and is built for org-wide automation. Lindy is better suited to individual productivity—email, scheduling, research. They solve different problems, so match to your use case. *Note: Lindy is a personal AI assistant, not a team platform.*
An enterprise needing governance and scale	Amazon Bedrock AgentCore or IBM watsonx Orchestrate	The two have integrated into a joint-control plane. AgentCore handles memory, session management, and observability at scale. watsonx is the stronger choice for IBM-centric or U.S. federal deployments (FedRAMP authorized). Both carry meaningful vendor lock-in. *Note: Building on AgentCore embeds you deeply in AWS infrastructure.*
A startup wanting flexibility and low cost	n8n (self-hosted or cloud) or Flowise (open source, now Workday-owned)	n8n shipped v2.0 in 2026 with native LangChain integration and ~70 AI nodes: Credible at mid-market scale, too. Flowise was acquired by Workday in August 2025; the open-source foundation is being maintained and invested in. Both remain self-hostable. *Note: n8n is now embedded in SAP Joule Studio—adoption is accelerating beyond startups.*

2. Foundation models (Big AI)

Choose your "brain"—most platforms let you swap models, but defaults matter. Current flagship models as of May 2026.

PROVIDER	KEY MODELS	STRENGTHS	BEST FOR	PRICING MODEL
Anthropic (Claude)	Opus 4.7, Sonnet 4.6, Haiku 4.5	Reasoning, safety, long context (200K tokens), coding, agentic tasks	Complex analysis, regulated industries, agentic coding	Pay per token via API or through partners (AWS Bedrock, Vertex AI, Azure)
OpenAI (GPT)	GPT-5.5 (latest), GPT-5.4, GPT-5 mini	Broad capabilities, largest ecosystem, natively omnimodal (text, image, audio, video)	General-purpose, wide integration support, computer use	Pay per token; ChatGPT subscriptions from $20/month
Google (Gemini)	Gemini 3.1 Pro/Flash, Gemini 3 (latest family)	Native multimodal, Google ecosystem, strong coding and long context	Workspace integration, video/image understanding, Google Cloud workloads	Pay per token via Vertex AI; Gemini bundled in Workspace Business/ Enterprise tiers
Open Source	Llama 4, DeepSeek V3, Mistral Large	Self-hosting, customization, no vendor lock-in, cost control	Privacy-sensitive, high-volume, custom fine-tuning	Free models; pay for compute



3. Enterprise platforms (tech incumbents)

Best if you're already in their ecosystem—deep integration beats standalone features.

PLATFORM	TARGET USER	TECHNICAL SKILL	KEY STRENGTH	KEY LIMITATION	PRICING
Salesforce Agentforce	Sales, service, marketing teams	Low-code to pro-code	Deep CRM integration, pre-built agents for sales/service	Locked to Salesforce ecosystem and data; complex pricing (three simultaneous models)	Free tier (200K Flex Credits); $2/conversation or $500/100K Flex Credits; per-user add-ons from $125/user/month
Amazon Bedrock AgentCore	Developers, platform teams	Pro-code	Model flexibility, any framework, enterprise scale, memory and observability built in	Requires AWS expertise; deep infrastructure lock-in over time	Pay-as-you-go; consumption-based
Microsoft 365 Copilot / Copilot Studio	Knowledge workers, office users	No-code to low-code	Native to Word, Excel, Teams, Outlook; Graph data access; active 2026 release wave	Limited customisation outside M365 stack; M365 licence required	AI bundled into M365 tiers from July 2026 price increase; Copilot Studio add-on available
Google Agentspace / Gemini Enterprise Agent Platform	Enterprise knowledge workers	No-code to low-code	80+ app connectors, multimodal search, Workspace native; rebranded as Gemini Enterprise Agent Platform at Cloud Next 2026	Transitioning/rebranding in progress; billing complexity across 15+ services	Contact sales; usage-based via Vertex AI; Gemini bundled in Workspace tiers
IBM watsonx Orchestrate	Enterprise IT, operations	No-code to pro-code	700+ system connectors, 100+ pre-built agents, FedRAMP authorised, AWS integration	IBM ecosystem gravity; strongest for IBM-centric or U.S. federal deployments	30-day free trial; enterprise pricing; available on AWS Marketplace
SAP Joule Studio 2.0	SAP customers, business process owners	Low-code to pro-code	Deep SAP integration, ERP workflow automation; rebuilt for 2026 with pro-code, Cursor and Claude Code support, n8n embedded	SAP-centric; limited outside SAP ecosystem; v1.0 had poor adoption	Free design-time access through end of 2026; part of SAP BTP thereafter
Zapier Agents	Business users, ops teams	No-code	9,000+ app integrations, fast setup, familiar interface; full AI orchestration platform	Less suited for complex logic; consumption costs can grow	Free tier; Pro from $19.99/month

4. Startups and specialized tools

More flexibility and often lower cost, but requires more evaluation and integration work.

PLATFORM	APPROACH	TARGET USER	TECHNICAL SKILL	KEY STRENGTH	KEY LIMITATION	PRICING
Rivet	Visual / drag-drop	Developers, AI engineers	Low-code, code	Open source, visual debugging, TypeScript integration	Smaller community; requires dev skills to deploy	Free (open source)
n8n	Visual / drag-drop	Technical ops, developers	Low-code	Self-hostable, 400+ integrations, active community; v2.0 shipped January 2026 with native LangChain and ~70 AI nodes; now inside SAP Joule Studio	No persistent memory natively; requires external DB for context	Free (self-host); Cloud from $24/month
Flowise	Visual / drag-drop	Developers, AI teams	Low-code	LangChain-based, 100+ LLM support, open source; acquired by Workday August 2025. Open-source foundation being maintained, invested in	Product direction tied to Workday's HR/finance focus; enterprise use cases may drift from pure agent building	Free (open source)
Vellum	Prompt-to-agent	Product teams, AI developers	Low-code, code	End-to-end LLMOps: prompt versioning, workflow orchestration, monitoring; built-in evaluation and observability	Enterprise-focused pricing; $20M Series A (July 2025); opaque paid-tier pricing (contact sales)	Free tier; paid tiers up to $500/month; enterprise custom
Lindy	Prompt-to-agent	Business users, individuals	No-code	Natural language setup, 5,000+ integrations, SOC2/HIPAA; personal AI assistant model—best for individuals, not team automation	Credit-based pricing can get expensive at scale; better for personal workflows than org-wide automation	Free (400 tasks); Pro $29.99/month
Monday Agent Factory	Prompt-to-agent	Monday users, project teams	No-code	Email, SMS, voice calls; native Monday integration	Requires Monday; still maturing	Part of Monday plans

5. Coding assistants

For teams building software—dramatically accelerates development velocity. Significant M&A activity in this space in 2025–2026.

TOOL	INTERFACE	TARGET USER	KEY STRENGTH	KEY LIMITATION	PRICING
Claude Code	Terminal/ CLI	Developers comfortable with command line	Agentic (plans and executes), codebase-aware, iterates autonomously; ~4% of public GitHub commits; $2.5B annualized revenue by February 2026	CLI-only; steeper learning curve for non-terminal users; Anthropic-model-centric (not multi-LLM)	Usage-based via Claude API; Max plans from $100/month
Google Antigravity	VS Code fork (IDE)	Developers wanting agent-first IDE with parallel agents	Gemini 3.1 native; up to 5 parallel agents; built-in browser for visual verification; multi-model (Claude, GPT also supported)	Pricing controversial: Credit system reduced free tier, opaque quotas; launched November 2025 with generous free tier then progressively cut	Free tier (limited credits); Pro $20/ month; AI credits at $25/2,500
Amazon Kiro	VS Code fork (IDE + CLI)	Teams wanting spec-driven, structured development	Spec-driven workflow; agent hooks for automated triggers; Figma MCP integration; now generally available (was preview in 2025); Amazon Q Developer being sunset in favour of Kiro	Credit-metered: Free tier only 50 credits/month; spec-mode costs 5x vibe-mode per credit; AWS-centric	Free (50 credits/ mo); Pro $20/1,000 credits; Pro+ $40/2,000 credits; Power $200/10,000 credits
Cursor	VS Code fork (IDE)	Professional developers	Multi-model (Claude, GPT, Gemini), 8 parallel agents, strong Fortune 500 adoption; 7M+ monthly active users; $2B+ annualised revenue	Subscription cost; model costs additional; pricing restructured June 2025 (credits replaced fixed requests)	Free (Hobby); Pro $20/mo; Pro+ $60/ month; Ultra $200/ month; Teams $40/ user/month
Windsurf	VS Code fork (IDE)	Developers, teams	Cascade agentic assistant, 70+ languages, cross-IDE plugins; ranked #1 LogRocket AI Dev Tool Power Rankings Feb 2026; now owned by Cognition AI (acquired July 2025 after Google acquihired founders for $2.4B)	Ownership uncertainty: Cognition integrating with Devin agent; product direction unclear; pricing restructured March 2026 (credits replaced by quotas; Pro raised from $15 to $20)	Free (5 Cascade sessions/day); Pro $20/month; Max $200/month; Teams $40/user/month

6. Key evaluation questions

QUESTION	WHY IT MATTERS
What systems do we already use?	Native integrations (Salesforce → Agentforce, M365 → Copilot) deliver faster time-to-value than building from scratch
Who will build and maintain agents?	No-code tools (Zapier, Lindy) empower business users; pro-code tools (Bedrock, Cursor) need developers
What is our data sensitivity?	Regulated industries may need self-hosted (n8n, Flowise) or enterprise-grade compliance (IBM watsonx FedRAMP, Salesforce)
How complex are our workflows?	Simple automations → Zapier/Lindy; complex multi-step reasoning → Bedrock/watsonx; visual debugging → n8n/Rivet
What is our budget model?	Per-seat (Copilot, Cursor) vs per-usage (Bedrock, Zapier, Agentforce Flex Credits) vs. self-hosted (n8n, open source) have very different economics at scale
How fast do we need results?	Pre-built agents (Salesforce, IBM) = days; custom platforms (Bedrock) = weeks; open source = depends on team skill

7. Investment priorities by company stage

STAGE	RECOMMENDED FOCUS	WHY
Startup / SMB	Lindy, Zapier Agents, n8n (self-hosted), Cursor	Low/no up-front cost, fast experimentation, scales with you
Growth stage	Vellum, Monday Agent Factory, existing vendor tools	Need governance and reliability; worth paying for managed solutions
Enterprise	Salesforce/Microsoft/Google native tools, Bedrock AgentCore, watsonx	Compliance, audit trails, enterprise support, integration depth
Tech / product company	Cursor + Claude Code, Bedrock AgentCore, open source foundations	Building differentiated products requires flexibility and control

***Note:** Pricing information is indicative and subject to change. Always verify current pricing directly with vendors before making purchasing decisions.*

OpenAI seen from below.

FROM THE ABOARD NEWSLETTER

The AI behemoths go horizontal

First, upstart companies are broadening their offerings, but they're already unbelievably huge, so it feels like Godzilla swishing its tail around. Anthropic is nerdy and is trying to nail down coders, and their products vibe with a lot of engineers. That's bad news for a lot of smaller vibe-coding tools, though. OpenAI is aggressively trying to move into the consumer product space, with offerings like "shopping" and "a browser," because they want to compete with Microsoft and Google.

Since literally everything else is cheap compared to LLM development, this creates a really wild situation where their competitors—like Perplexity—are totally reliant on them, while OpenAI can simply stand up another 30-person team and go "make me a browser," which they then give away on top of ChatGPT. It's unfair, but that's the risk of building your business on someone else's API. Of course, Google already has a browser, and OpenAI is building on top of that. There's a lot of mess ahead.

Second, software development acceleration is well underway, and LLM companies are looking for more white-collar low-hanging fruit to pick off. A product like Claude Code, or Copilot—they're finding their market and their identity. But a lot of jobs are programming-adjacent; junior analysts at banks are basically spreadsheet programmers.

It's not surprising that OpenAI is nudging along a "financial model" generation path in its products. They're obviously going to keep fanning out to finance and engineering-adjacent roles, trying to replace the less experienced humans in those positions. Anyone with "analyst" in their title who isn't a therapist is up for inspection: Business analysts, risk analysts, actuaries, data analysts, policy analysts. If you use Excel, SPSS, R, or MATLAB, they may be gunning for you here....

This sort of all-in massive platform rapid development and deployment by multiple tech megacorps at once doesn't have a lot of precedent.

— Paul Ford · Read more

FURTHER READING:

- Import AI 447: The AGI economy; testing AIs with generated games; and agent ecologies — *Import AI* (Jack Clark), 2026-03-02
- A Field Guide to Rapidly Improving AI Products — Hamel Husain, 2025-03-24

What are agents good for?

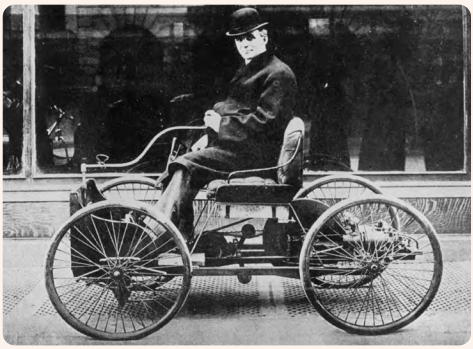
…and how do I tell?

We've gotten a grip on what agents are made of and where to get one. Now we can demystify what they're actually good for: Where the real value can come in, beyond just keeping up with the Joneses.

But first, a few reality checks—because investigating agents will mess with your intuition.

Don't panic: They're not actually thinking

Agents can do a lot more than chat, and AI companies like to call that process "thinking" or "research" in their UIs and branding. But those words can lead you astray if you get too used to them. Appearances aside, agents don't understand you any more than a Roomba appreciates the finish on your hardwood floor. They are simply machines that are inclined to do something rather than nothing when the power is turned on, and because they have LLMs inside them, their whirring noises often *look* remarkably like thinking. But it's always worth keeping in mind: While you have thoughts, agents don't. They just have outputs—and those outputs are only as good as your inputs.



A "horseless carriage" moment

The default story around agents is this: They're exciting because of what they can let us do without. Operator-less customer service! Developer-less coding! HR-less employee onboarding! It's a lot like when cars were first seen as "horseless carriages," or email and desktop computers were supposed to usher in the "paperless office." It's an easy and natural way to start making sense of some new capability. But it's limiting.

We think agents are exciting for the opposite reason—because of the new processes and opportunities they create. It's additive, not subtractive; more about enabling than replacing. That's not to say we love tedious tasks. If agents can help kill those for your organization, great. But horseless carriages didn't kill traffic, and the paperless office didn't kill paperwork. Same with agents: "-less" isn't necessarily more.

So what *is*?

FURTHER READING:

- Real AI Agents and Real Work — *One Useful Thing* (Ethan Mollick), 2025-09-29
- How did Anthropic measure AI's "theoretical capabilities" in the job market? — *Ars Technica*, 2026-03-31
- AI is Creating Peak Software, Media is the Best Analogy — *Fabricated Knowledge*, 2025-07-10

Three agent superpowers

1. Personalizing

We don't mean rewriting emails; we mean distilling information into your own personal brew. Since every agent has a text-processing LLM at its heart, agents can be great for quickly gathering, transforming, and packaging up text in useful ways, on your terms.

Some examples:

- **Staying abreast of industry news:** Create a business-intelligence aggregator, spidering over hundreds of sources to create digests for an individual or whole team. On a smaller scale, sift your social feeds for links to include at the end of your next newsletter (and insert them into it once you approve).
- **Meeting/sales/conference prep:** Scan your team's calendars two weeks out, and create a dossier about each upcoming prospect or major event. It doesn't need to be President's-Daily-Brief-quality to be useful. For instance, you could use it to familiarize yourself with the industry jargon your counterparts use.
- **Bespoke demos:** Use agents to gather information about potential clients to create customized software prototypes for them, then attach them to (non-bot-written) follow-up emails.

DO: Curate. Remember, agents don't understand anything. Mindless quantity has a quality all its own: you can use agents to hoover up impractical amounts of content, sort it all like a combine harvester, and expose signals that *you're* smart enough to recognize.

DON'T: Impersonate. Agents make it easy—and tempting—to create personalized spam at scale. It may even work once or twice: we admit to having been unwittingly charmed by emails that began with a thoughtful-sounding reference to a recent podcast, before realizing it was a con, and perma-blocking everyone who sent them.

2. Parallelizing

Doing things in parallel is as "computery" as it gets, but the average person just doesn't think this way. The world is mostly sequential: We start things, finish them (sometimes), start other things, and so on. But you know who *does* think this way? CEOs of giant companies. Satya Nadella knows how to parallelize because he has 100,000 human "agents" working for him. Now you do, too—they're just software instead. Which is even better, because they can do trivial but useful-in-aggregate things that you could never ask people to do.

Some examples:

- Don't just summarize one document—summarize them all. Then turn them into a vector database, hook them up to a chatbot, and (with the right guardrails!) "talk" to your company policies instead of searching for them.

- Generate documentation for every codebase in your org, and create strategies for building shared libraries for your engineers.

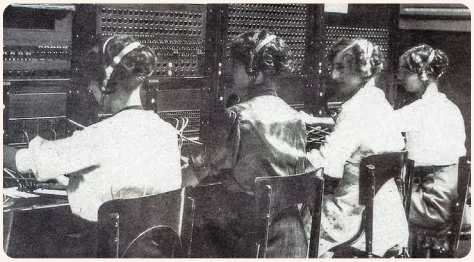
- Explore a range of future-facing business scenarios simultaneously, then have agents rank the results.

DO: Think wide, not deep. Agents are good at chomping through short, specific tasks. Where do pools of those tasks tend to collect? Backlogs (like expense reconciliation) and updates to copies (like legal agreements) are a good place to start.

DON'T: Sweat this. It feels unholy to break normal processes into reproducible little components that can then be aggregated back together. No one starts seeing like a CEO overnight. Just keep your eyes and ears open, and opportunities will present themselves.

3. Easing communication

Organizations need information to flow easily: Not just between people, but between



departments, processes, and systems. When it doesn't, the reason is often because of something dysfunctional in the org's structure or culture. Software can't fix that. Other times, it's just because Point A can't easily *talk* to Point B without a lot of messy, tedious assistance.

Maybe the new CRM can't sync with the old ERP without copy/pasting. Maybe compliance forms still have to be walked around a job site. Maybe invoices involve *just* enough different languages and formats to make processing them by hand the least-worst option.

This, agents can help with. Think of them as "chaperoning" information around obstacles and across gaps.

FROM THE ABOARD NEWSLETTER

In praise of inauthenticity

Just a cowboy living in the moment. Not a mobile phone app in sight.

The conventional wisdom—from perhaps the last 30 years—is that when you operate and market a firm in public you are supposed to be an "authentic" company. You're supposed to have thoughts, feelings, a personality—especially on social media. The idea being that we're all just humans in this together, even the brands.

Authenticity is hard to pin down because the definition changes with the cultural moment. The parameters vary a lot, but if you think of, say, beef jerky, that's always advertised in an "authentic" way. A typical beef jerky campaign will show you footage of cowboys and go, "It's meat on a stick. What do you want from us?" …

Real authenticity is exhausting—think of that relentlessly "honest" friend who loves to criticize you and talk about your faults. Are they fun to hang out with? No. "Corporate" authenticity is even more exhausting. I know because I practiced it for years in many contexts.

I think a lot of it comes down to pacing. Everyone tries to get authentic too fast. Can't we build the relationship first? When I meet a stranger at a party and they start telling me about their excruciating divorce or deep religious convictions or political anger right away, I see that as a warning sign. The same is true when brands do it. Tell me what you do, about your kids, talk about the model train meetup you host in your gazebo. You could even ask a question! No one asks questions. It's been many years since anyone asked me a question at a party. Maybe I just look like I'll give terrible answers.

— Paul Ford · Read more

Some examples:

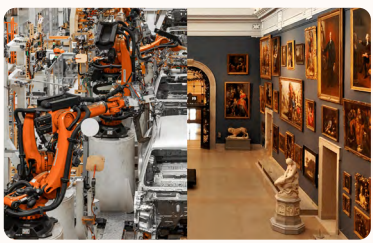
- Seamlessly linking deskless workers with their sales teams and suppliers.
- Handling edge cases in paperwork-processing workflows that fall through the cracks of rules-based software.
- Translating product requirements into actionable specifications for custom software prototypes (we do this at Aboard).

DO: Listen to the pain. Where are the aching joints in your organization? This will help you understand how things can be improved, not just band-aided.

DON'T: Simply grease the rails. Agents will slip right into the existing grooves in your operation, for better or worse. But not everything works better just because it's faster. We call the entrenchment of bad processes "procedural debt," and we'll unpack it in more detail later.

FROM THE ABOARD NEWSLETTER

When it comes to AI, don't mix up artifacts with processes



Good for one, not the other.

At some point in every career, you come to realize that the *artifact* is a side effect of a *process*. Comedians have a hilarious hour of material for their HBO special because they've workshopped the jokes at comedy clubs for months. Newspapers happen because everyone knows their place in the daily schedule. Musicians work on albums for years; software also takes years to get right. Everyone thinks they could write a hit song, but very few people have the time, resources, and patience to write thousands of terrible songs until they figure out what makes a good one.

AI is wild because it can make convincing artifacts, what Rusty Foster called "language without thought." We've never seen this before, and it's making us all wacky. A bot made me a video! It wrote my college essay! It made an app! But then, when you poke a little closer, they turn out to be weird—kind of eldritch artifacts, vague. The code might run, but it still needs a lot of human intervention to become a product. To me, the jury is out on whether AI can make good stuff. But what it's *really good* at is accelerating processes.

If your job is to create an artifact for public consumption, Claude or ChatGPT are often the wrong tools for the final product. The process does matter—it's integral to the artifact. They can summarize news, or send you interesting links, or give you helpful background. But ultimately, someone has to go to the restaurant, eat the bread sticks, and tell you how it went. That's the process.

But if your job is to make artifacts that push the bureaucracy forward—case file notes, quick briefs, purchase orders, incident reports, all the things that get emailed, reviewed, and filed in the "Z:\" drive—then this stuff is a godsend, and it lets you go home early. The artifacts you create in these jobs are not the end state—they're part of the process.

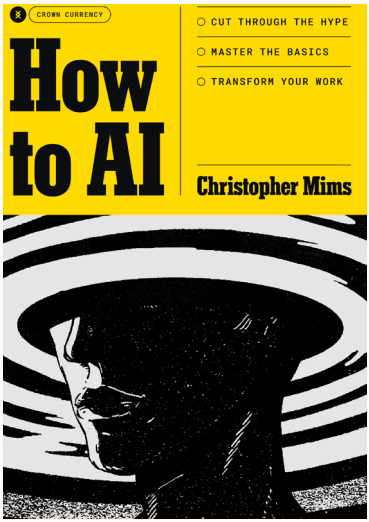
— Paul Ford · Read more

FURTHER READING:

- A Guide to Which AI to Use in the Agentic Era — *One Useful Thing* (Ethan Mollick), 2026-02-18
- AI as Normal Technology — *AI Snake Oil* (*Princeton CITP*), 2026-02-12

FIELD NOTES

Real-world AI use cases



The most useful agent deployments aren't from unicorn AI startups or world-eating SaaS behemoths. They are regular companies servicing other regular companies, doing the regular things that make up 80% of real-world business. Three from *Wall Street Journal* technology columnist Christopher Mims's book:

Procore (construction software) puts an agent inside Slack and Microsoft Teams that can produce, fill, and route the dreaded RFI—the Request For Information that every construction project drowns in. "You can now invoke the RFI agent to go and create that RFI based on that conversation and fill all the information in for you," says Procore VP Kris Lengieza. Multiply that by dozens of workers across multiple jobsites and thousands of hours get saved.

Filevine sells "Depo Copilot," what Mims calls "a conversation between an agent and itself." Pre-loaded with the lawyer's question list, it transcribes a deposition in real time and pings the attorney when she's drifting from material she needed to cover. Texas personal-injury attorney Kim Jones Penepacker uses it "almost every time"—not to replace her judgment, but to backstop it when no second-chair lawyer is in the room.

Rossum, which is named for the 1920 Karel Čapek play that coined the word robot, trains its agents on back-office documents—purchase orders, invoices, bills of lading—instead of Reddit. They interpret GL codes, apply discounts, convert currencies, request approvals, and write to ERP systems. "You can have ten companies using QuickBooks, and each of those ten companies produces a different invoice," says cofounder Petr Baudiš. Rossum's agents handle them all. Regarding the service, Mims notes that "solving a deeply unsexy problem well has been advantageous for the company." That's high praise, in our book.

Adapted with permission from How To AI *by Christopher Mims (Penguin Random House).*

When agents go wrong

We've all heard of "dark patterns"—ways of putting software toward nefarious (or just frustrating) ends. Agents have their own ways of breaking bad, too. Here's a cheat sheet for recognizing what *not* to do with agents, intentionally or unintentionally.

Lazy vs. Evil: Agents can make trivially easy what used to require significant effort and resources. This kind of power can tempt people and organizations to try things that they couldn't before. Those things usually have two flavors: Sins of omission (lazy) or commission ("evil"—not *literally*, but just in the late-capitalism sense of "generally vile, venal, or lacking in integrity").

Inward vs. outward: Agents' ill effects can be directed within an organization (affecting staff, systems, and processes) or outward (affecting its customers, clients, and relationships).

Procedural debt: Agents can easily be used to further entrench ways of doing things that were dysfunctional to begin with. Like the "technical debt" that builds up when people use software to slap together solutions for short-term gains, procedural debt accrues when agents are used to enable or amplify self-defeating behaviors within an organization.

Chesterton's Fence: Using agents to "disrupt," "transform," or otherwise uproot and replace things without bothering to understand why they were put in place to begin with.

Generic content: What it says on the tin. Agents' powers of parallelization can create uncannily efficient slop factories that make lines go up by polluting the internet.

Hollow automation: Horseless-carriage-style deployments that cut costs while degrading customer experience or product value.

Workslop/accountability sinks: Using agents to cover for inefficiencies, sidestep answerability, or otherwise break legible chains of responsibility within your organization.

Impersonation/faking connection: The ultimate agentic sin. Pretending to be an engaged person to automatically extract value is the quickest way to torch your organization's prospects, relationships and reputation.

Here be brand damage: People might get excited about hitting their OKRs with swarms of robots, but the overall risk to the organization's reputation is quite large.

Rotting from the inside: You may be "innovating with agents" while sowing frustration and sapping effectiveness among your people.

How to avoid these traps

A rule of thumb: The more an agent feels like normal, useful software, the more value you'll get from it. So how do you figure out what "normal and useful" looks like inside your organization?

AW, SNAP!

FURTHER READING:

- Systematic debugging for AI agents: Introducing the AgentRx framework — *Microsoft Research*, 2026-03-12
- Here's What Agentic AI Can Do With Have I Been Pwned's APIs — Troy Hunt, 2026-04-16
- Some thoughts on "Agentic DevOps", AIOps, and Vibe Coding. With Gene Kim and Nicole Forsgren — James Governor, *Monkchips* (*RedMonk*), 2025-07-25

FROM THE ABOARD NEWSLETTER

The eternal dream of instant software

The product manager leading the engineers.

[O]ne of the subjects that we talk about most is *acceleration*. How can Aboard help you do "software things," faster, without writing code?

This led me to make a short list of the main accelerators in software over the last 30 years. Each category represents billions in investment and effort and huge communities. For example:

- **`Simpler Code.`** Make an easier programming language that is simple to teach and learn. Logo, BASIC, Python.
- **`Embedded Code.`** Put a programming language inside of something else. World of Warcraft can be extended with the Lua programming language. You can script the 3D tool Blender with Python. PHP was designed to work inside of HTML tags. The most dominant example here is JavaScript, which was born inside the browser.
- **`Visual Code.`** Move stuff around, sometimes with boxes and arrows. Draw interfaces with your mouse. Frameworks like Visual Basic, or IDEs like XCode and Visual Studio Code—the dominant platforms of Apple and Microsoft, respectively, get you part of the way there, but they're still mostly textual. There's always an attempt to go all the way and get rid of text editing. This can work, but it tends to only work in niches, like audio engineering or visual effects, like Max for music composition, or the teaching language Scratch. (RIP Yahoo Pipes.)
- **`Pure Code.`** Make a system that is less likely to fail, but requires more work up front. Haskell, TLA+—and the "Software that Never Fails and Can't Be Hacked" of Green Hills Software. These are worth it when you're writing an OS for your airplane. The White House endorses them.
- **`Low Code.`** Give people tools to build a database, then offer easy ways to do things to the data in that database. Coda and Airtable are good examples here, as is Notion if you look at it a certain way.
- **`AI-assisted Code.`** Tell the system what you want and it does it. This has been the dream for decades and there have been many attempts, but now…it's actually (kind of) working with AI tools like CoPilot. Computers turn out to be okay at writing code after all.

Software people making lists like the one I just made always leave out the spreadsheet, which has billions of users, makes intuitive sense, is the lightest-weight database available to most humans, and drives trillions of dollars in decisions every year. It's not "real" programming though. Ah well.

Nonetheless, there is a huge range of human challenges related to listing stuff, tracking stuff, and organizing stuff in rows where spreadsheets represent the 90% solution. The remaining 10% is where humans get into the mix. I really do think this is an important way to think about software tools: How is this better than a spreadsheet? Is it faster? Simpler? More visual? Does it serve a special function (like making World of Warcraft easier to script)? More collaborative? Does it provide more structure and lead to fewer problems?

There's another thing about spreadsheets that gets undervalued: They're disposable. They sit in the folder and they don't bug you to update. You can open one from 20 years ago and you'll know exactly what's going on. Zero surprises. It'll still be an ugly pile of rows…but it'll work about the same. We don't celebrate that quality in code very often—it would be bad for business—but it's great to do things and throw them away. Then you can move on to the next thing.

— Paul Ford · Read more

Latent software, or: Where to find value with agents

Like we said at the beginning: Agents are just software. But remember how we also said that agents can, and will, change your business's *relationship to software*? You've made it this far—now we can tell you what we mean.

Unlike Excel or SAP, agents can turn your stuff—documents, plans, and processes that you probably already have laying around—into business software that can fit your needs like a glove. If you know where to look, the seeds of these potential solutions are hidden in plain sight. We call it latent software: Thanks to agents, for the first time, it can be easier to build than to buy.

Here's a treasure map for where to find the latent software in your organization, and use agents to make it real:

PDFs and Word documents

Look for any piece of digital paper describing a process that is auditable and useful: Procurement, content production, anything where assets are managed. Any of them could become an agent-powered workflow. Today, it's a static checklist of instructions. Tomorrow, it could be software, stood up within an hour. Literally just dump it into your chat of choice and say "can this be an application"—see what happens.

Google Docs or Sheets, shared tracking applications

Businesses already create scores of makeshift applications out of cloud-based scratchpads and spreadsheets. Agents can leverage that material and turn it into something robust. If it's being used to track workflows and notify people about statuses, consider it fodder for an agentic experiment.

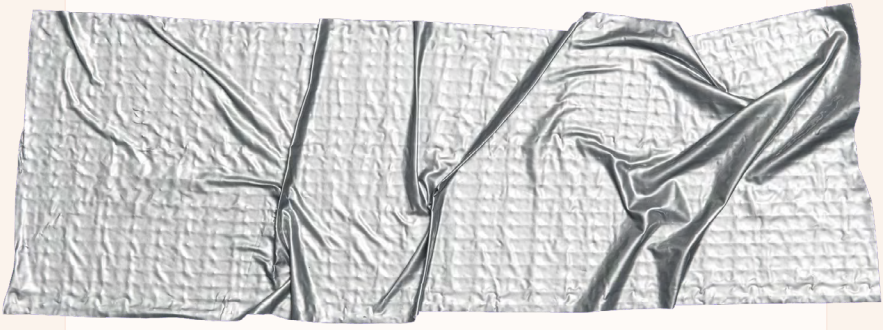
FURTHER READING:

- Headless everything for personal AI — Simon Willison, 2026-04-19
- Common pitfalls when building generative AI applications — Chip Huyen, 2025-01-16

Pitch decks that never went anywhere

The deck from a consulting firm three years ago, or the one that your head of engineering made for the big project that you couldn't find the budget for? Dig it up, because you just might be able to move forward with it. If it was $3 million dollars five years ago, it's $300K or less today. If there were problems you wanted to solve but you had to set aside due to your org's constrants, you might be able to build all that software with a team of virtual agents.

Latent software is a powerful idea. But as the band Poison once counseled: *Every rose has its thorn*. For agents, it relates to a thing we call procedural debt—and it's the yin to latent software's yang.



Yeah, sure, that'll solve the problem.

FROM THE ABOARD NEWSLETTER

Paying Off Your Procedural Debt

Let's coin a new phrase: Procedural debt. Procedural debt has nothing to do with technology. It's all those lousy, inefficient, bad habits that your company lives with today. You'll find it in government, non-profits, for-profits—just about any organization that's been around for more than five years. One thing about habits is if you enact them enough times, you get good at them. And when you get good at something, you're much less inclined to change.

Software that's rushed through organizations often doesn't help—instead, it just sort of formalizes and further acknowledges procedural debt. Tools can actually be a bad-habits enabler, further entrenching less-than-ideal ways of working.

Good software makes things work a little faster. Great software changes how you work. When people talk about "enterprise software solutions," that's the actual hope and promise, and why it costs so much money. It will change the way you work. It will abolish your technical debt—and your procedural debt.

Historically, implementing enterprise software solutions would involve hiring a ton of consultants and paying a lot for software licenses, and then, over

36-96 months, you'd roll out the new tools—slowly, with lots of training and painful migrations.

Today, we're headed for a world of AI, and the general vibe is that this will be replaced—all those consultants and coders—with "agents." They really love their automation on the West Coast, and they think typing into a box is too much work; the next wave is just telling an LLM to do things for you, autonomously, and report back. Agents can roam around the web, and also around your data and workflows, then suggest improvements, fix your grammar, tighten your legal language, and let you know that inventory is running low. They can do this all day and night.

More ambitiously, they can autonomously execute multi-step, complex processes that used to take all sorts of orchestration and coordination in the past. No more running that approval form down the hall! And they're getting smarter and better every day.

Now, if this was a more traditional marketing newsletter, I might say: Agents are a powerful tool for paying down your procedural debt. You'll be able to spin up agents to enact processes, and they'll accelerate everything you do. Procurement? Agents. Governance? Agents. Compliance? Agents.

But I don't believe that. Agents are amazing at producing artifacts; they're autonomous and incredibly motivated to work all day. And this makes them the ultimate enabler of procedural debt. They have no interest in taking a step back and offering up ideas about how to change how you work. They're not going to challenge assumptions or revisit the real underlying conditions that led to that oddball 18-step approval process.

You are going to be sold the idea that agents can do everything better and faster, and make you pretty reports at the beginning or end of the week. But the truth is that agent-based software is so easy to make, so fast to deploy, and so useful that I expect it to codify and enforce really bad processes. It'll encode that approval process. It'll keep that PDF-based workflow intact. It'll read through the thousand resumes without really thinking about why you have so many generic submissions instead of a few useful, targeted pieces of inbound with thoughtful cover letters.

I would like to propose a different approach: Let AI observe your work. Call it an "AI Advisor." Don't let it try to fix anything. Describe how you do things, upload the PDFs, share the government forms. Let it present you with a deck on how you can do better—what process could be simplified, what apps could shorten workflows.

We use agents in our products, we love the technology, and we love this new ecosystem. But we also believe that how people work together is...well, human. Our processes tend to be there for a reason, and it's important to know and understand those reasons before you engineer them away. LLMs can't understand, so that's still on us, at least for a while.

— Rich Ziade · Read more

3. Engage

What now?

You're ready to get started. There's no time like the present—and it's easier than you think.

Choose with confidence

When assessing what agents can do well, it's good to think in terms of confidence level. It's not yes/no or pass/fail—it's a spectrum. The three categories we described in "Assess" (Personalize, Parallelize, Ease Communication) are, in our experience, high-confidence areas for using agents. But you should begin building up your own intuitions about what counts as a high-, medium-, or low-confidence scenario for your org. A specific agent that works wonders in someone else's context may not make the same kind of sense in yours.

Processes > Tools

Don't get wrapped around the axle of which AI model is best (any of them) or what platform or provider to use (pick the lowest-hanging fruit). What matters more is the approach you and your organization take toward engaging with them.

DO: Make it safe to try.
DON'T: Mandate or impose.

What to consider:

- How much will it cost? What will the money be spent on?
- How much time will it take? Would setting informal goals be useful?
- Who will be involved? How can I best use the people I have?

Expect nonsense

We touched on this earlier, but it bears repeating: LLMs are general-purpose systems that produce extremely specific nonsense. That nonsense may happen to be useful or true, in an accident of statistics. We assign meaning to those coincidences, because that's human nature.

But wait: *Nonsense? It helped me write my college application!* Sure, often it produces a block of text that seems like a cogent reply to your prompt. Unfortunately, that apparent non-nonsense breaks down in unpredictable ways: The LLM might produce text that's too wordy, too curt, too formal, too casual, or just too random. It might parrot harmful biases or stereotypes. Inevitably, it will produce "hallucinations": Stuff that looks correct but is in fact wrong (and can be harder to spot than you think).

These aren't bugs as computer people have understood them over the last 70 years. Even hallucinations aren't errors, per se: They're a "natural" part of how LLMs work. (Fun fact: Technically speaking, *every* output from an LLM is a hallucination, but we just reserve that word for the outputs we don't like.) If you're serious about working with this technology, you need to assume that what it's generating is meaningless or error-filled—even if it looks pretty good. That's how they get you.

FROM THE ABOARD NEWSLETTER

Defining success on your own terms

Check out those beveled edges!

We're all anxious primates. Tech giants are bigger, more dominant primates. If you think of Apple not as the people who made the iPod but as a very large silverback gorilla (with beveled edges), sitting on top of a mountain of money and hooting, refusing to let you publish your app in the app store and taking 30% of all your bananas, then maybe the relentless need to point out how it's actually a big bad failure of a company makes more sense.

If you look around you will realize that you are constantly negotiating with enormous hooting, screeching technology gorillas. You can't talk about the gorillas directly, because that would remind everyone that you are not, yourself, a hugely powerful and dominant silverback technology gorilla. So you talk about money and how dumb everyone is. What's the market cap? Is the stock up or down? Boy did they screw up on that smart car thing.

But moneygoggles don't help you see things more clearly. Money is a totally fine metric when you're outside of a company—one of the best! But inside of a company, it's a severely lagging indicator. Money shows up months, or years, after other decisions. Which is why seeing the world through VC/market goggles can be dangerous, even if it satisfies your base instinct to say that you are the best and smartest and would have made all the right decisions, if only they'd put you in charge....

I don't think you should focus on the failure of others, or even the success of others. What you do need to do—and what is surprisingly hard—is to define success for your own part of the world, and work towards that. Non-monetary success, because again, money lags. Primates like to look *up* when we should look *forward*.

— Paul Ford · Read more

FURTHER READING:

- Quests, token leaderboards, and a skills marketplace: The elite AI adoption playbook — *Lenny's Newsletter*, 2026-04
- How to Successfully Integrate AI Into Claims Operations: Best Practices, Pitfalls, and Keys to Lasting Impact — *Risk & Insurance*, 2026-05
- Choice Hotels Moves AI Technology Beyond Pilot Projects and Into the Core of Hotel Operations — *Hotel Technology News*, 2026-04


95%
of AI
software
projects
FAIL

A rare photo of heaven.

FROM THE ABOARD NEWSLETTER

Use worse tools

The current crop of technologies can seem like a magic show. Let me draw you a picture! Write you some code? Didn't work? Have more! I built you an app right here in the browser. They have so many tricks and the magic comes so fast that by the time you learn what's going on, they've updated the entire model. So going just one step back has helped me stop focusing on the shiny new things and understand more.

The big thing I'm learning is how to manage and think about context windows. LLMs don't know what they know. They forget as they go along. To use them effectively, you need to be constantly reminding it of what it learned. The way you do this, I'm finding, is to mentally organize your session in an outline.

Let's say you are (1) Building a to-do app; (2) using a bunch of Javascript libraries; (3) working on the "new task" functionality. It writes the code! And then you go: "Let's do the next part, and build the ability to delete a task." The last one went really well, but on this next step, it seems to have forgotten everything it learned. But if you remind it of all the levels in the hierarchy, it will work. Every time you do something new, you need to go, "We are building a to-do-list and we are using these libraries...and now we're going to work on task deletion. Please review the last bit of code and propose a plan." And the results tend to be better! So much of technology means traversing up and down a hierarchical tree.

— Paul Ford · Read more

Keep it manageable

Most people think about agents being unleashed into existing infrastructure. But the big best practice here is to keep the asks small and specific. Set up narrow, threaded processes to test: Think in terms of assembly lines and conveyor belts, not open-ended autonomous "goals." (Asking an LLM to suggest ways of breaking a task down into simpler steps is okay, too.)

DO: Aim for quick wins.
DON'T: Try to capture complex processes.

What to consider:

- Where will the agent "live"? (The cloud? Company servers? My laptop?)
- What tools will it have, and why?
- What guardrails will I need? Who will put them in place?

FROM THE ABOARD NEWSLETTER

Keep your hands on the wheel

The fantasy we're sold is a "Built to Order" (BTO) universe, where little robot chauffeurs and butlers attend to us and do our bidding. The nascent AI industry is more guilty than most of pushing this fantasy. In the unmitigated enthusiasm of AI *innovation and solutions*™, the promise is that AI will do the whole thing: Reliably get the kids to soccer practice via self-driving cars. Or land the plane. Or remove your appendix. Or remove your appendix *while* landing the plane.

Just keep going until you hit Vegas.

But alas, we can't. At least not yet....

People using AI tools are seeing enormous improvements in productivity, especially coders. But it's because they *keep their hands on the wheel*. Harper Reed recently shared his LLM codegen workflow. It's worth reading because it illustrates how an expert engineer applies a sort of slow-motion version of procedural memory to step through the thinking and work behind building an app. It's many, many steps, and it directly attacks a lot of the pitfalls that come from just trusting AI to ship you a built to order app.

For now, that's the way to work with these tools: Not issuing commands and stomping around, but more like driving a strange, powerful vehicle, constantly making little adjustments and tweaks and keeping your destination front-and-center. AI cannot keep its eyes on the road (Waymo cars aside). Built-to-order is still a ways off, but accelerated task completion is here today, and it's great.

As people discuss AI and how it's going to gobble up jobs, there's a bit of irony in revealing that AI really badly needs your human brain to get it across the line. It's comforting, but also a challenge to all of us to level up. Maybe one day an AI robot will guide another to carefully navigate these steps, but until then, let's keep our hands on the wheel and go.

— Rich Ziade · Read more

Always be summarizing

Some experts we trust have noticed that LLMs become more accurate and predictable when they summarize, and less so otherwise. Since every agent has an LLM in it, this applies to agents, too. That's a good thing to remember when beginning to put guardrails on them.

Start by having an agent expand the inputs it receives, then have it filter and summarize the outputs. (These instructions can be given as prompts to the agent's LLM.) The idea is to turn an agent's limitations into an asset: Go big and messy first, then condense things down so as much nonsense as possible (hopefully) gets squeezed out.

"Deep research" agents from OpenAI and Google already rely on this trick. When you give them a prompt, they expand that input: Asking questions, searching the web, and generating "plans" or "chains of thought" (AI-speak for text that biases the LLM toward certain outputs). They might produce code and run it, to create even more context. Finally, they crush the end results into structured output: A report with sections, headings, and citations. Expand, then summarize. If it's good enough for Big AI, it's a good starting point for you, too.

Recognizing ROI

"Success" with agents can mean different things. Make sure you know it when you see it, and define a high-confidence task or domain to work in.

DO: Take ownership. Be ready to explain what happened and why (even to yourself).
DON'T: Tinker in isolation. Make people aware of what's happening so they can be partners, not blockers.

What to consider:

- How will we measure success? Time? Money? Something qualitative?
- What are our criteria for continuing to try something, versus cutting our losses?
- If we experience success, what's our next move?

Be nice

Do you have to be nice to agents? Of course not; they're machines. But being "polite"—making tidy bulleted lists, providing adequate context, generally treating them like new employees on their first day (yes, we know we expressly said *not* to think of agents this way, but just bear with us)—is actually very helpful. Why? Because it forces *you* to order *your* thoughts about what you want from them.

FURTHER READING:

- The State of AI Report 2025 — *Air Street Press*, 2025-10-09

Agents are just software, so: Garbage in, garbage out. Despite the powerful conversational illusion they produce, you can't actually build an understanding with them that goes both ways. Which means it can be very frustrating when they completely "forget" what you told them to do last week, or just a few hours ago. For this reason, you should approach things gently: Be "nice" to them as a way of setting yourself up for success. Agents' "memory" will probably become more reliable over time, but this habit of mind—and the clarity it produces—will continue to pay off.

How to not lose the plot

The "why" of agents is tricky. Once you start recognizing opportunities to use them and start putting them into action, it's easy to get dazzled or distracted. Potential use cases can snowball. You start seeing latent software everywhere. Just because you can use an agent to turn a seven-step process into three, should you? Where does this all net out, anyway?

DO: Focus on value. You don't have to be deeply technical to understand how these things can help. But you do have to have a feel for what moves the needle in your world, and what's just box-ticking.

DON'T: Expect radical change. Agents may be "just software," but organizations aren't. Yours may not want to change—few do. That's okay. Both things can be true at the same time: Agents can show transformative potential, and not instantly blow up "the way we've always worked."

What to consider:

- Are we making proofs-of-concept, or something we should scale and maintain? Do we even want to?
- How do we evolve with agents? Even if they work, are they the best way to realize our "latent software" long-term?
- Are agents surfacing new opportunities, or strengthening the business-as-usual? Neither is "better," but which direction are we actually interested in?

FURTHER READING:

- AI Delivers Mediocre Results—By Design. So How Do You Stand Out? | MetaLab CEO Luke Des Cotes — *CRAFTED.* (Dan Blumberg), 2026-03-10
- Import AI 447: The AGI economy; testing AIs with generated games; and agent ecologies — *Import AI* (Jack Clark), 2026-03-02

No waiting or requirements-gathering. They dive right in.

FROM THE ABOARD NEWSLETTER

Skipping step zero

Step Zero comes before code, almost before requirements. Step Zero is the problem statement.

There's an awful truth about software that always broke the hearts of our clients: You're usually going to throw away the first version. "Build one to throw away" is actually a big part of engineering culture. Sometimes we call it prototyping, sometimes we call it "low code." Years and years of building software have taught us that *building* is not the hardest part—figuring out *what to build* is.

So back to Step Zero: As we visit potential customers for Aboard (not just users, but entire organizations), we listen. And we watch. And we hear about how they work and why things suck and why they're in so much pain and how much money they're losing because they're working so inefficiently....

What we're learning is that in this new, AI-powered, lightspeed world of technology, the problems are the same. It could be 2004, 2014, or 2024—the exact same problems keep organizations underwater. Data is all over the place, things keep slowing down, orders go missing, people are grumpy.

I think the promise of AI is that we can actually throw away Step Zero—the great, unspoken, incredibly expensive step that happens before anything gets built. We can do it by throwing something on the screen that works—the details aren't always right, but at least you've gotten started, which gives people something to react to.

— Rich Ziade · Read more

The whole thing on one page

If you only remember three things from this paper, make them these:

1. Agents are just software

An agent is an LLM with access to tools, running in a loop. The fundamentals haven't changed—agents are constrained, debuggable, and best when they feel like normal, useful software. The more guardrails (tight specs, structured output, narrow environments), the better they work.

2. The opportunity is latent software

Look at your existing PDFs, spreadsheets, and unfunded pitch decks. They describe processes that, until now, you couldn't afford to turn into custom software. With agents, you can. Build is finally cheaper than buy.

3. The risk is procedural debt

Agents will encode whatever process you point them at, including the dysfunctional ones. They have no taste, no judgment, no memory of why that bad process exists. Before you automate, ask whether the workflow should exist at all. The same agents that surface latent software will entrench procedural debt if you let them.

That's the work in front of you: Find the latent software, dodge the procedural debt, and define success on your own terms. Agents can show you transformative potential. They can't tell you what's worth transforming.

FURTHER READING:

- It's open season for refusing AI — Brian Merchant, *Blood in the Machine*, 2026-04
- Taste Is Not Enough. Reality or Bust — *A Computer Scientist in a Business School*, 2026-03-28
- Sacrifice your job for the glorious AI future — Paris Marx, *Disconnect*, 2026-04-21

FROM THE ABOARD NEWSLETTER

The $15 Volvo

You can build a lot—a lot!—of software in *minute*s with AI. Some of that software even works. And this is powerful and surreal and exciting, but there's another reality: You're creating enormous technical debt and risk every time you use AI to code. All that code is going to keep running, and it will need updates and maintenance. People can say, "That's okay, AI will fix the problems," but that's a bet, not a strategy.

Definitely worth $15.

So while AI is rapidly changing everything from image creation to student-essay writing, I think the future of enterprise software—and enterprise is where the money is!—will be a slower process.... I think the most important metric in the future will be, "How much of this system was developed by people and how much was AI?"

Ultimately, the promise of a lot of AI tools right now is the $15 Volvo—and that's because what it does, generating code quickly and often at a good level of quality, seems so miraculous. But what business buyers want is a Volvo that gets delivered on time, has incredibly clear diagnostics, needs less maintenance (and when it needs it, a human checks the work). And sure, they want it to cost a lot less. But it's going to take some time to figure this out—and it's important to remember that business people buy products for safety, not style.

— Paul Ford · Read more

The bottom line

We used to run a boutique software agency/consultancy—helping companies understand and build what they really need is "our thing." But when AI agents came on the scene, we dropped what we were doing and spent six months figuring them out. That became Aboard, which leverages agents to make doing "our thing" better (i.e., radically faster and cheaper). And that opened up opportunities to deliver a new kind of value.

Working with agents didn't just accelerate our business, it changed it: What we could build, and who we could serve. We didn't set out to have agents transform our business, but here we are.

It's hard to get your mind around. But if, like us, you can get over the shock and engage with them intelligently, agents have a ton to offer—whether that's making "your thing" better, or altering it altogether.

More from Aboard

We talk about this stuff every day. Here's where to find us:

Aboard Newsletter
Paul Ford and Rich Ziade, weekly, free. Insights on AI, software, and your career.
→ aboard.com/newsletter

Aboard Podcast
Paul and Rich and guests on what's actually changing in tech.
→ aboard.com/podcast

Aboard YouTube
Video edition of the podcast, live events, and webinars.
→ youtube.com/@AboardApp

Work with us
Your partner for AI transformation. We build reliable software and deliver secure solutions for some of the world's best companies at record speed.
→ aboard.com/about

Get in touch
paul.ford@aboard.com · rich.ziade@aboard.com · hello@aboard.com

Colophon

This white paper was written and edited by an exceptional human team of engineers, leaders, and advisors.

Words: John Pavlus, Paul Ford, Rich Ziade, Adam Pash, and many others.
Editorial: John Pavlus, Gabe Boylan, Elizabeth Minkel.
Design: Minder Singh.

Special thanks to Christopher Mims (*Wall Street Journal*) for permission to adapt the Procore, Filevine, and Rossum case studies from his forthcoming book *How To AI*.


aboard