

Deciphering



Glyph

[About](#) * [Archives](#) * [Mastodon](#) * [GitHub](#) * [Patrons](#)

What Would A Serious AI Product Look Like?

Every “AI” tool is missing critical features that you would need if you wanted to do real work with them. What would those look like, if they existed?

Sunday September 27, 2026

ai

llm

One of the issues that I have with the current generation of “AI” products is that they do not appear to take their own premises seriously. I look at a plethora of obsequious chatbots claiming to be serious tools for problem solving, and I think, this is not what a problem-solving tool would look like.

Even before we get to the tremendous ethical problems with the frontier labs, it is this impression of their composition as a *product* that makes me feel, constantly, whenever I am interacting with them, that they are less a software product than that they are a grift, a scam designed to make me feel like I am interacting with a product that has capabilities that it simply does not, to try to lull me into a false sense of security that I can trust it.

The frontier labs are of course the worst offenders, but every criticism here applies just as much to Ollama, which (if anything, due to the obviously poorer quality of the available models themselves) needs these features *even more* than the frontier labs do.

Here, I will set down a few features that might convince me that an LLM-based product, particularly one focused on research or software development, was actually serious about helping me do useful things with it.

Make “Checking For Mistakes” A First-Class Feature

This is the biggest issue, and the major reason that I was inspired to write this post.

It is a truth universally acknowledged, that AIs cannot reliably provide information.

I could cite a ton of news articles and studies about this fact, but there is no need. Every single chatbot admits this, up front, in a fine-print disclaimer as a core part of their user interface. Gemini says “AI can make mistakes, so double-check responses”, Claude says “Claude is AI and can make mistakes. Please double-check responses.¹” ChatGPT says “ChatGPT can make mistakes. Check important info.”.

Every time I see that last one, I wonder how I’m supposed to know what “info” is supposed to be “important”.

All of these warnings are all small, gray text, painfully obviously included as legalese to push responsibility back onto the user rather than to help with anything. This is a *core limitation* of all these products. Checking their output is a part of the workflow for using them that:

1. you *absolutely cannot skip or skimp on* without creating risks to yourself and whoever you are conveying its output to, and,
2. it is *very easy to skip or skimp on* and you are encouraged at every turn to do so, because “just trust the output” is one of the quickest ways to save time.

A chatbot product that took this weakness seriously, as an actual consideration for using it, would put a checkbox next to *every claim* in its output. It would be a 2-column worksheet, where you’ve got the LLM output in the first column, and next to it, human notes in the second column, explaining what work went into checking this claim, and a *big* checkbox that you would only check off after you believe you’d checked its claims thoroughly enough.

Coding assistants would need to have some version of this as well. Right now, this is pushed off into code review, which means it is a dark pattern which subtly encourages the “author”² to offload this work to their code reviewer without ever looking. Once again, “it’s probably fine, I don’t need to check” is the quickest way to save time and churn out those PRs faster.

It might even be useful for coding harnesses to have some affordance for checking code before it even runs tests. [As the vendors themselves have admitted](#), it’s not just expensive to burn tokens on your “AI”, you also end up burning far more *compute* on the AI. Being able to check your diffs before sending them over to uselessly exhaust your testing compute cluster would be useful.

If your product tells me that it makes mistakes and I must be the one to check for the mistakes, but then gives me *zero tools* to check for mistakes, I cannot take it seriously.

More Citations to Check, And More Details

Most chatbots prefer to give an answer, rather than a citation. In my own personal use, I find that when asked to provide a list of citations with clearly marked sources for each

one, they will appear to “get bored” halfway through the list and simply stop including citations at some point.

When the bots include citations at all, present them as inline annotations that say nothing but the domain name of the search result, in a font so small that it’s barely legible, and an equally indecipherable icon that is fewer than 16 pixels on a side.

This is backwards.

Now, I am aware that these citations do come from somewhere, and in an attempt to reduce hallucinations, all of the major providers support some form of “grounding”³, and that those little barely-readable citation links are referencing actual structures in the [RAG](#) pipeline and not just potentially-hallucinated tokens, but I’m not talking about the underlying machinery in the model, I’m talking about the *presentation to the user*.

Plus, regardless of whether a snippet of text came from a RAG query, we know that LLMs can [never provide an authoritative result](#); it’s a fundamental limitation of the technology. They can still garble the results of RAG as much as they can misrepresent any other training data. This means that it must never present its results as authoritative.

If you ask an AI to do research queries, every result should be presented as a *list of citations*. Moreover, the presentation should display each citation as a large object of in its own right, with clearly identified metadata, including not just the site where it was found but its publication date and, if possible, the name of the author. The literal, unmodified quotation (not from RAG, not a summary: a quotation extracted with a regular program and not an LLM) should be front-and-center, larger than any AI-generated text.

If the AI product wants to editorialize or summarize (which should not always be necessary!), the AI-generated text should be presented as small text underneath the citation that has been found, de-emphasized as much as the disclaimer is right now, at the very least until the user has verified that the summary is accurate. Perhaps, for a research project, a “did you read the citation” checkbox might even be helpful.

If your product openly tells me that it will scramble, misrepresent, or omit its citations in its summaries, and I must read the original human-authored citations to be sure, but then gives me no tools to track my reading of those citations or even any way to *find* them, I cannot take it seriously.

No First-Person Output, No Apologies

There is no reason for a software development or research tool to use first-person language to describe itself. They should not do so. In fact [they should not be allowed to do so](#).

There is also no reason that they should ever *apologize*. [It is a waste of everyone’s time](#); it’s a waste for the chatbot to generate the apology, it’s a waste for the user to read the

apology, and it's a waste for the user to respond to the apology. Yet they unfailingly do this upon every correction.

The vendors of these tools [know that they are routinely causing mental-health crises](#). In response, they have added non-functional “guard rails” that can still, in 2026, [easily be bypassed](#).⁴

A product seriously interested in helping with productivity would correct this *glaringly* obvious flaw, focus on the task at hand, and stop emitting useless verbiage.

In the previous two sections, I tried to focus on ways in which the harness would be constructed differently even if the LLM technology is fundamentally impossible to improve; in this case, I have to assume that the labs have *some* control over the model itself. But unless they are truly incapable of influencing their output (and all their “benchmarks” and “capabilities” seem to indicate that they can control it very tightly) they ought to be building models that are much less verbose.

More Non-Natural-Language User Interfaces

Although natural language could *hypothetically* be a powerful interface for interacting with a computer system, the practical upshot of LLM natural language interfaces is that these interfaces are imprecise and repetitive, full of [superstitions masquerading as “best practices”](#). The inputs are a mess and the resulting outputs are a mess.

The general way of addressing this unstructured mess is to allow the chatbot to *directly take action* in response to the user's input; in other words to supply it with “tools” via an MCP server. But again, this is backwards. If we cannot even express our intent clearly in the first place, why are we trusting this system to take potentially destructive and harmful actions on our behalf?

Instead, I would expect a product that was seriously invested in helping me accomplish specific tasks, to have user interfaces specific to those tasks. Is it supposed to be able to be a security scanner that can discover OWASP top 10 bugs in a codebase? Have a button for that. Build that functionality into your harness, train it directly into the model, use smaller models that can satisfy that functionality more effectively than throwing it at the planet-sized brain of Fable or whatever.

I'm aware that there are small software startups that do *something* like this, but they are bolted on to the side of the main model providers' APIs, not integrated into the core of the product and not using their own models and AI systems to achieve consistent and repeatable results.

Strong Data Provenance Indicators

Chatbots produce data tables pulled from websites, from APIs, from MCP tools or from summarizing and scrambling the user's input. In order to provide the illusion of a seamless interface, this data is presented in-line regardless of where it comes from. But some of these outputs are produced mechanically via regular old API calls, for example, from the result of calling a tool or querying a website, but presented uniformly.

But there is a huge difference between an authoritative data source being inlined as part of a chatbot conversation, being treated as *input* by the chatbot, and some ad-hoc hallucinated data being treated as *output* of the chatbot.

If a product is trying to help me make accurate, empirically-grounded, data-driven decisions, the source of the data is critical.

Integrated into the “check for mistakes” and “verify citations” workflow I described above, there's a necessary “verify data programmatically” pass as well; to have tools that will treat portions of the output as a *regular spreadsheet*, allowing regular-old computer arithmetic to verify things and *showing where such arithmetic was used, and how*.

Better User Control of Reproducibility

Anyone familiar with the technical specifics of LLMs will know that they have a variable called “[temperature](#)” which controls the degree of randomness that the LLM uses to produce its outputs. But most users don't know this, because it isn't exposed as part of the user interface by default.

This leads to a subjective impression that you asked ChatGPT, and you got ChatGPT's authoritative answer.

You can't just set the temperature to zero and still get useful results - I am aware that it does more than just scramble the output at random, and there are perhaps good reasons that simply exposing *just* a temperature setting would not be that useful to users. But if we followed some more of my earlier recommendations for making more structured UI elements to solve specific problems rather than having long back-and-forth chats where each refinement depends on the previous response, perhaps those elements could also re-play the process so that users can see how reliable the bot is at a particular task and develop a sense of how the stochastic nature of the process actually affects it.

Similarly, if a user is trying to solve the same problem repeatedly with a chatbot, and the chatbot *product* has numerous computational tools that don't really have anything to do with the LLM, such as deterministic data-processing tools, then having a way to freeze the non-deterministic parts of the transcript but re-populate a particular data frame with updated information and fork / continue the conversation from there would be a way to avoid introducing pointless additional randomness when you already know what tool you're trying to use.

The fact that every conversation is presented as this flat chat prompt that doesn't let me interact with any of the widgets that were previously produced except through more chatting, really makes me feel like the whole product is just doing predatory social-media style “increase time on site” optimization, just trying to lure me into further repetitive and unreliable chats, rather than letting me get in, solve my problem, and get out.

Context Visibility

Managing the LLM context is *the* ongoing challenge facing organizations that are trying to use “agentic” workflows. Filling up the context with too much information [causes well-known problems](#). In response, advanced LLM users attempting to solve larger problems must break up very long prompts into “skills”, give access to lengthy information via “tools”, and delegating sub-problems to “sub-agents” rather than simply extending a single prompt indefinitely.

All of these strategies have flaws, because even on the largest models, compared to the breadth and depth of knowledge-work problems, LLM contexts are quite small.

And yet, none of these products will *show the context to the user* by default. There are third-party addons that can show you a [simple progress bar](#) but for addressing the premier engineering difficulty with this technology, that is below the bare minimum.

This lack of visibility means that almost all of the tools for extending the context are flying blind. Rather than responding meaningfully to a full context, everyone just kind of guesses how much state they need by guessing and trying over and over again with progressively more elaborate skill and sub-agent layouts. Even managing context compaction ends up being an [advanced API-driven workflow](#)⁵.

A serious product that was trying to help the user understand would not only show “available context” but explain the impact of context compactions, make it easier to see harness-generated prompts, and so on. This would be a first-class feature, combined with the aforementioned reproducibility / replay tools, would allow users to do real experiments to develop an understanding about how to make good use of the context window.

A Sandbox That Actually Works

I’ve been focused on the chatbot interface here because it is the most *immediately* egregious upon looking at the UI. But the “agentic loop” tools used for coding are equally dangerous, if not more so. Coding tools [keep destroying everyone’s data](#), over the course of *years*.

These catastrophic incidents that become front-page news are relatively rare compared to the amount of coding-agent use out there. But they also aren’t the only kind of sandbox violation. Coding models will so routinely edit test code instead of the system under test

that there are “pro tips” articles all over the web giving you the [flawed advice](#) to simply ask the agent not to cheat. News write-ups of the catastrophic incidents themselves will also offer glib and wrong advice, like “use a docker container”. That might prevent it from literally deleting your operating system, but it won’t prevent it from destroying all the local work you have in your codebase (it needs access to a checkout, after all!)

There is a flurry of activity in the infosec space where people are rushing to plug the gaps left by these coding harnesses. Everyone’s got their own version of an [MCP approval gateway](#) where you can optionally place a proxy between your agent and your production infrastructure.

In the best case, though, all these mitigations and proxies and prompts simply turn the user into an [auto-approval automaton](#), hitting Y, Y, Y, Y over and over again, until you finally are driven mad and hit “yes to all”, turn on full-auto mode and submit yourself to the void. With nothing between your personal vigilance and disaster, there are no workflows left beyond decrementing your own vigilance until there’s nothing left and then hoping the disaster never arrives.

The fact that *some* mitigations exist that can be deployed by extra-cautious users does not change the fact that “agentic coding” is an unsafe-by-default technology deployed without concern or guidance. Every frontier lab has tied a spring-loaded shotgun to a dog; the fact that dog owners can publish thoughtful blog posts explaining how you can teach your dog the basics of gun safety or how you can have your dogs play in a bullet-proof room does not mitigate the fact that the product should not have been allowed in the first place, nor should it continue to exist without VERY strong security controls.

I might believe that a frontier lab were seriously interested in providing developers with a useful tool if they shipped something that had safety built-in.

That means tools *in the harness*, detached from any LLM, independent of the prompt, that could:

- sandbox all filesystem operations and strictly limit ANY deletions outside of specified scopes, regardless of operating system,
- enforce snapshotting of the entire repo on every operation for easy rollbacks and minimal lost work,
- remove [the disaster of “auto mode”](#) (not to mention nonsense like `--dangerously-skip-permissions`) entirely, and
- carefully consider a structure for presenting plans to the user where, rather than provoking immediate alert fatigue by asking for checks on every action, make structured plans which can be submitted to the user as a group of actions and reviewed and approved as a batch.

In the same way that I suggested above that research-based tasks should have a way of re-issuing prompts to determine how reproducible a result is, or whether other sources

might be found, agent-based tasks should have a way of being executed against mock services for popular APIs, so that the verification can match both on the front-end (review the plan for making the API calls before they're executed) and the back end (review the API calls that were issued to the mock service and verify that they matched).

Instead, the frontier labs provide us products that are disasters out of the box, give us “best practices” to build massive and elaborate, as well as incomplete and error-prone, security perimeters of our own design. Then they blame “operator error” when it inevitably goes wrong. I cannot believe that these design choices are intended to help us be productive.

Bonus: Human Processes

Organizations *deploying* AI also frequently come across as unserious, for similar reasons. In 2023, naive exuberance could perhaps be forgiven. But today, as we near the close of 2026, there are several well-known problems, that have been extremely well-covered in the press. None of these things should be surprising, but most orgs deploying these tools are still just letting them rip and hoping it all works out.

Organizations deploying these tools would need at least three kinds of major modifications to their internal processes, if they wanted to be serious about using them safely:

1. Shift Rotations to Prevent Vigilance Decrement

There have been several high-profile incidents where software developers' gradual acquiescence to accepting LLM output have lead to serious economic consequences for the companies deploying them, perhaps best typified by Amazon's “[millions of lost orders](#)” due to a gradual decay of their engineering processes from LLM use.

These outages, and other AI-related failures, are due to the difficulty of maintaining focus on the same problems. In other words, as I described above, [vigilance decrement](#) is a constant problem, because AI outputs are *most often* correct, but continue to be incorrect in surprising and non-intuitive ways. As I have [previously written](#), you cannot trust yourself to catch every bug with code review, and LLM output.

Aviation, for example, has *very strict rules* around [rest requirements](#). There is also a specific rule that “[No certificate holder may operate an aircraft without a second in command if that aircraft has a passenger seating configuration, excluding any pilot seat, of ten seats or more.](#)”. Other safety-critical professions have similar rules.

And yet, even in the age of the supposed “AI revolution”, most software teams are still assigning every engineer a full feature load, not planning for any rest, and telling people to review code whenever they happen to have some “free time”.

Maintenance of vigilance has to be your top priority. Regular, scheduled, *inviolable* rest periods where people do work without AI assistance, and are not exposed to any AI output for review or otherwise, would be crucial in order to stay mentally sharp enough.

The tools themselves should have this sort of thing built in. The mistake-review process described above should have a periodic spot-check mode where a second reviewer periodically reviews a chatbot log, doing their own independent verification of claims, to see if they spot the same errors. This could provide a feedback loop to determine how much rest is necessary to maintain continuous attention and actually spot hallucinations.

2. Skill Practice To Prevent Skill Loss

It is also well-known that AI use leads to AI reliance, and AI reliance leads to [skill loss](#).

I like to use the analogy to dockworkers at a seaport⁶ adopting automation.

If you employ dockworkers to load and unload ships all day long, they are going to be getting tons of exercise. They will be able to lift heavy objects on demand, whenever. They might have plenty of health problems and injuries from this type of work, but “lack of exercise” will not be a problem.

With the development of standardized container ships and mechanized cranes, you are going to be changing their job description substantially: now they mostly spend all day sitting in a small cubicle moving a control lever back and forth, not lifting heavy stuff. They will get worse at lifting heavy objects.

In this analogy however, the cranes are not all that reliable. We know they break, and they drop their payloads sometimes, and the stuff needs to be manually moved. But this only happens a few times a week, at most. If you need whoever is driving the crane to be able to jump out at any moment and still move stuff around manually, then you need to make an affordance for that. You need to give them *time* to go to the gym and do some lifting for practice, or every crane failure is going to be a major emergency.

An organization doing an AI transformation would also need a massive increase to learning & development budget, both in terms of resources and in terms of schedule. If your people are going to lose skills because they’ve lost regular practice in the incidental course of doing their duties, then they are going to need *deliberate*, intentional, non-incidental practice of those skills to keep them sharp.

But rather than trying to accommodate new workflows and give time for people to adjust, most AI mandates are simply dropped on workers like a ton of bricks, with no time to adapt and no affordance for maintaining their skills. Operate the crane and stay fit and healthy and ready to switch back to manual lifting at any time and then get back in the crane cockpit right afterwards. Don’t mess up.

Then an accident happens and everyone is surprised, as if this process weren't practically *designed* to produce a terrible result.

3. Mental Health Resources to Deal with Mental Health Risks

AI psychosis often begins with [practical problem-solving](#), and beyond that, it can start [specifically at work](#). Not to mention the more pedestrian condition of [“AI brain fry”](#).

If you are mandating your employees to use a hazardous tool that may seriously and *directly* damage their mental health, you need trainings and resources. You need in-house therapists and you need to be making sure to check in with people actively to make sure that this is not happening.

Again, the tool itself ought to have some way of dealing with this. An [occasional “take a break”](#) popup is easily dismissed; they need a user-visible AI [personal dosimeter](#) so you can see your *cumulative* usage over time.

I don't even know if “usage over time” is a sufficient metric to gauge risk. Maybe if your work chatbot start to talk about [resonance](#) too much, unless you literally work as an acoustic engineer, that should be flagged for someone.

We are, again, years into dealing with these tools, and we know these risks exist. Yet no serious mitigations are provided. Not even any way of measuring the risk exposure.

And More

There are also many other risks associated with the technology. There are [intellectual property risks](#) with the foundation models, due to recklessness with their training data. There are [existential financial risks](#) associated with the infrastructure build-out. The extent to which most “open” models are [simply derivatives](#) of frontier models is an open question.

What I Think

If any *one* of these things were regularly overlooked by AI vendors or users, that would be a totally normal product oversight. Room for improvement for the next version, but nothing catastrophic.

Shipping without *any* of them doesn't seem like lean product management, it seems like a careless attitude towards risk and a product design philosophy oriented entirely towards short-term demos, with no regard for how to realize actual productivity gains.

Furthermore, being available for *years* without anything like these features, despite hundreds of incidents demonstrating the risks, with hundreds of billions of dollars of funding, makes it seem to me like if they *were* to add all the features that would make

their product actually safe and hypothetically useful, these features would reveal that it is actually not an improvement to productivity.

In the [year](#) since I first wrote about measuring the cost/benefit ratio of AI, I have heard from numerous people who have shown this to management to try to illustrate why their AI initiatives — like [almost all AI initiatives](#) — were either failing or burning out their engineers.

I've also heard from lots of people that have told me that it's obviously useful and they don't need to measure so carefully, because they are getting lots of work done that they couldn't have otherwise.⁷

I have yet to hear from a *single* person who has said “yeah, we measured according to your methodology⁸, and it turns out that our AI work is going great and that our ratio is 0.75”.

Obviously, I cannot say for sure why this is; absence of evidence is not evidence of absence. But at this point I think the null hypothesis is that AI tools provide, in aggregate, zero value. They make mistakes too often, and the externalities they produce are so bad and so difficult to control that even before we get to the places where they are [just physically poisoning people](#), even the negative effects on their direct users end up cancelling out whatever benefit to they provide to their organizations.

If I were wrong, then including tools to *measure* an AI's effectiveness *at the tasks their users are actually trying to accomplish*, rather than [meaningless benchmarks](#), would show big productivity gains. The frontier labs would be champing at the bit to add such features, and crowing about their fantastic results.

I think the labs know that if they did that, it would present a grim picture to their users. Such tools would let their users see that it's making mistakes much more often than they realized, that they're spending much more time with it than they want to be, and that it's just generally not fit for purpose.

If they prove me wrong by adding in all of these safety mechanisms, and in the process, they make all of their AI technology less harmful, I'll be thrilled to be debunked.



Acknowledgments

Thank you to [my patrons](#) who are supporting my writing on this blog. If you like what you've read here and you'd like to read more of it, or you'd like to support my [various](#)

★ ★ ★

1. It is also interesting that for the next section, *sometimes* it seems that Claude's disclaimer is "Please double-check cited sources." instead. [↩](#)
2. ... by which I mean the "prompter", since authorship is not what's happening here. [↩](#)
3. [Claude has the "citations API"](#), [Google has various different kinds of "grounding" against its own APIs](#), and I guess Microsoft can [check OpenAI's homework](#) if you want. [↩](#)
4. Given the relatively slow speed of the justice system and the mainstream press around the world, we probably will not hear about whether people are managing to incidentally break through these guard rails to self harm right now, but there are no shortage of stories still being *reported* right now where people were still doing just that, such as in [this story](#) where the effect of the much vaunted "guard rails" in 2025 was that if you wanted it to write you a suicide note, it would refuse twice but acquiesce on the third try. I don't see any reason to believe this fundamental issue has been addressed in the meanwhile, since it had been happening for years at that point. [↩](#)
5. In this tutorial we can also see an incredibly rosy scenario presented, where a long-running workflow effortlessly compresses all of the necessary information into the new context, even if it uses a lower-fidelity model to do so, rather than the tangled and gnarly problem of problems which really *are* too big to fit in the context, which is to say, "most real-world problems". This presents the context limit instead as a minor speedbump to be worked around rather than the fundamental flaw in LLM tooling. [↩](#)
6. A heavily fictionalized seaport. This is not how actual dockworkers work. This is a simplistic metaphor about incidental benefits of instrumental tasks, it is not supposed to delve deeply into the mechanics of maritime shipping. In particular I know that cranes are more reliable than this and this is not actually how you would respond to a crane malfunction anyway. Feel free to share fun facts about maritime shipping if that is your special interest but please do not @ me to *correct* this metaphor. [↩](#)
7. To my knowledge, none of their publicly-traded employers have posted a measurable improvement to efficiency outside the margin of error. [↩](#)
8. Or any similar methodology. I don't need people to adopt the exact practice that I proposed there. [↩](#)