Seven Faces, One Engine: The Consolidation Nobody Is Comparing

Weekly Analysis — https://ainews.social

There is a particular pleasure in a roundup. Seven art generators lined up like contestants, three chatbots ranked by "vibe," two frontier labs squaring off in a benchmark cage match — the genre flatters the reader with the sensation of a marketplace, of choice, of a landscape wide enough to get lost in. And the coverage this week performed that ritual faithfully. What it did not do — what the genre almost structurally cannot do — is ask whether the seven faces you are comparing are attached to the same body. Because they very often are. The most useful thing a careful reader can do with a week of tool coverage is stop counting the interfaces and start counting the engines, at which point the abundance thins out into something closer to a monopoly wearing a wig collection.

Consider the object that has quietly become the most-deployed AI product in the working world: Microsoft's Copilot. It is not one tool. It is a brand stretched over an entire product surface — a chat assistant, a coding assistant, an agent-building studio, a low-code platform layer — and Microsoft's own documentation is admirably frank about the fact. The company's overview describes Copilot as an assistant woven through Word, Excel, Outlook, Teams, and the rest, "grounded" in your organizational data [17]. The same brand reappears as a developer product with its own enterprise pricing tiers [3], and again as a platform for building agents inside a low-code environment [1]. One name, many doors. The choice on offer is which door you enter — not which building you enter.

This is the move worth watching, and it is the move the comparison genre reliably misses. When a week's reporting hands you variety, the analytically serious question is not "which is best?" but "how many of these are actually distinct?" The answer, most weeks, is: fewer than advertised. This essay is an attempt to do the counting the roundups won't — to read the vendors' own paperwork against their marketing, and to show that what looks like a diverse ecosystem of tools is better understood as a small number of engines with a large wardrobe.

[17] What is Microsoft Copilot?

[3] Choosing your enterprise's plan for GitHub Copilot

[1] Build - Microsoft Copilot Studio (GitHub Copilot) | Microsoft Learn

The Interface Is Not the Product

Start with what a tool actually is, from the user's side of the glass. You type into a box; something types back. The box has a logo, a color scheme, a pricing page, a personality the marketing team has labored over. All of that is interface. None of it is the model. And the relationship between the two is precisely where the industry prefers you not to look, because the interface is where the differentiation lives and the model is where the concentration lives.

The clearest documented case is Microsoft's own product family, because Microsoft, uncharacteristically for this industry, writes down what it is doing. The adoption guidance the company publishes for IT administrators treats Copilot not as a feature but as an organizational rollout — a thing you provision, license, monitor, and drive usage metrics for [10]. There is a dedicated usage report so administrators can watch adoption climb [11], a rollout guide specifying minimum requirements [14], and a setup procedure for assigning licenses at scale [4]. Read these documents in sequence and the "tool" dissolves into what it really is: an administrative dependency layered into infrastructure a company already cannot leave. The interface is the part that faces the user. The product is the lock.

[10] Microsoft Copilot adoption and onboarding guide for IT admins

[11] Microsoft Copilot Usage Report - Microsoft 365 admin

[14] Rollout Microsoft Copilot to your organization

[4] Configurer Microsoft Copilot et attribuer des licences

Notice what the low-code and agent layers do to this picture. Copilot Studio and the Power Platform are marketed as democratization — build your own agents, no code required — and the reference architectures Microsoft publishes make the pitch concrete [12]. But a platform that lets you build a thousand custom agents is not multiplying the number of engines in the world; it is multiplying the number of front ends that route back to the same one. The proliferation is real and the diversity is illusory at once. Every agent an enterprise builds in Copilot Studio deepens its commitment to the substrate underneath, and the substrate is singular. This is what the consolidation actually looks like from the inside — not fewer products on the shelf, but more products that are secretly the same product.

[12] Overview of Power Platform and Copilot Studio reference ...

The same structure repeats at Google. Gemini Code Assist is presented as a developer tool with its own overview [7], its own chat surface [2], and its own enterprise customization path [13]. Distinct product pages, distinct documentation, distinct pricing — and one Gemini model family under all of it. The user who "chooses" Gemini Code Assist over Gemini in the browser over Gemini in Workspace has not chosen among competitors. They have chosen a costume for a single actor.

[7] Gemini Code Assist overview | Google for Developers

[2] Chatea con Gemini Code Assist | Google for Developers

[13] Personalización de código con Gemini Code Assist Enterprise

Kate Crawford named the mechanism that keeps us from seeing

this. In [16] she describes ”enchanted determinism” — the way the industry directs our attention to the marvel of the method and away from ”the purpose of the thing itself,” a rhetorical maneuver that ”obscures power and closes off informed public discussion, critical scrutiny, or outright rejection.” The interface is the enchantment. It is engineered to be the thing you evaluate, precisely so that the ownership structure underneath — who controls the model, who sets the terms, who can revoke access — never enters the frame. When you compare seven art generators on the quality of their outputs, you are doing exactly the evaluation the vendors want, and skipping the one they don’t.

[16] The Atlas of AI

The Substrate Nobody Names

The most instructive example this week is one the roundups touched and then dropped: the popular design tool whose AI image generation runs, underneath, on a model built by a company most users have never heard of. Canva’s generative features have been powered by Leonardo — a provider Canva acquired — and yet the coverage that ranks image tools by their results almost never surfaces the real supply chain. This is not a minor omission. It is the whole story, misfiled as a footnote.

Here is why it matters for anyone actually deciding what to adopt. If three of the ”seven” image generators you are weighing draw from the same underlying model, then the differences you are measuring — a slightly nicer UI, a cheaper tier, a friendlier license — are differences in packaging, and the things you are NOT measuring, because they are identical across all three, are the things that will actually determine your risk: the same training data provenance, the same biases baked into the same weights, the same failure modes, the same single point of vendor control. The HAI AI Index has catalogued how the field even measures itself — HELM for language, HEIM for image generation, MMMU for reasoning, and so on [16] — but a benchmark score on a shared model tells you nothing about the concentration hiding behind the brands that all report it.

[16] HAI_AI-Index-Report-2024

The deeper point is a supply-chain point, and the AI industry hates supply-chain thinking because supply chains are where power becomes legible. Crawford’s book is in the end a supply-chain argument — that behind the ”cloud” and the ”model” sit minerals, labor, water, and a small number of firms controlling the whole vertical [16]. Interface plurality is the consumer-facing end of a highly concentrated production process, and the roundup format, by treating each interface as a

[16] The Atlas of AI

sovereign product, systematically launders that concentration into the appearance of a competitive market. A reader who wants to evaluate tool claims critically has to learn to ask the one question the coverage omits: what model is this, actually, and who owns it? When the answer is "we'd rather not say," that is not a neutral silence. It is the answer.

Cathy Kissik — the sensibility, at least, of anyone who has watched a chatbot bluff — is worth borrowing here. In [16] Janelle Shane documents how AI systems that pass for competent usually lean on a gimmick to "explain away non sequiturs or their inability to handle most topics." The interface is often exactly that gimmick at the product level: a well-designed front end that explains away, through sheer polish, the fact that you have no idea what is generating your outputs or under what terms it will keep doing so. The smoother the door, the less you think about the building.

[16] You Look Like a Thing and I Love You

Watermarking as a Transfer of Authority

Nowhere is the substrate consolidation more consequential — and more misdescribed — than in the fight over how we will tell synthetic media from the real thing. The coverage this week reached for the reassuring frame: watermarking is coming, the tools will label their own outputs, the problem is being handled. But the specific technologies being deployed are not interchangeable, and treating them as a like-for-like upgrade obscures a genuine transfer of power.

Two things are being conflated. C2PA — the Coalition for Content Provenance and Authenticity — is an open standard: a cross-industry specification for attaching tamper-evident provenance metadata to a file, verifiable by anyone with the spec. SynthID, by contrast, is Google's proprietary watermarking system, embedded into content generated by Google's models and detectable, definitively, only through Google's own tooling. These are not two implementations of the same idea. One locates the authority to verify in an open standard that no single company owns; the other locates it inside a vendor. To describe the shift from provenance standards toward proprietary watermarks as a simple technical improvement is to miss that the improvement, if it is one, comes bundled with a relocation of who gets to say what is real.

Ask the question the way a skeptic should: in a world where verification runs through SynthID, who do you have to trust, and who can that trust be revoked by? The answer is a single firm, whose detector is a black box, whose coverage extends only to content its own

models produced, and whose commercial interests are not identical to the public's interest in a legible information environment. This is the same consolidation as the interface story, transposed into the register of epistemology. First the model is centralized; then the tools built on it are centralized behind many brands; and now the very apparatus for authenticating what those models produce is centralized inside the vendor. Choice, again, redefined as the freedom to pick which company you depend on to tell you the truth.

Shoshana Zuboff's account of how platform power actually operates is the relevant lens, even when unnamed in the coverage: the decisive asset is not the product but the control over the conditions under which knowledge is produced and validated. A verification system owned by the entity whose outputs it verifies is not a check on power; it is an extension of it. The open-standard alternative exists — that is what makes the drift toward proprietary detection a choice rather than a necessity, and a choice worth naming as such. The careful adopter's question is not "does this tool watermark its output?" but "who holds the key to the verification, and what happens to me when they change the lock?"

The Shared Flaw Under the Shared Engine

If the consolidation thesis is right — if the many tools are a few engines — then it makes a testable prediction: the tools should share not only their strengths but their weaknesses. A common substrate means common vulnerabilities. And this is exactly what the security research of recent months has demonstrated, in the most literal way. When multiple widely used coding assistants turn out to carry the same trust-boundary flaw for months on end, that is not a coincidence of independent engineering failures. It is a fingerprint of a shared architecture.

The class of vulnerability is indirect prompt injection, and Microsoft — again, to its credit, in writing — has documented both the threat and the difficulty of defending against it [5]. The mechanism is deceptively simple: a large language model cannot reliably distinguish between instructions from its user and instructions smuggled into the content it is asked to process. Hide a command inside a document, a web page, a code comment, an email — and the assistant, dutifully reading, may execute it. Microsoft's own framing is telling; it treats this not as a bug to be patched but as a standing architectural condition requiring layered mitigation, monitoring, and zero-trust discipline. That is the language of a problem that does not go away, because it is

[5] Defend against indirect prompt injection attacks

baked into how the models work.

Now apply the consolidation logic. If six coding assistants are wrappers around a small number of foundation models, and the foundation models share this inability to police their own trust boundary, then all six inherit the flaw simultaneously, and a single researcher's finding implicates the whole cohort at once. That is precisely the shape of what the security community surfaced — a shared substrate of trust-boundary weakness across the market's "most widely used" tools. The detail that should stop a reader cold is that when the finding was disclosed, at least one vendor rejected it. Not "we've patched it," not "we're investigating" — rejected. That response is itself evidence, and not of the technical kind. It is evidence about the incentives. A vendor whose product is a thin interface over a shared engine has every reason to deny a vulnerability it cannot unilaterally fix, because acknowledging it means acknowledging the dependency, and acknowledging the dependency means admitting that the "product" is mostly other people's model.

This is where the pro-reader position has to get concrete, because the vendor incentive and the user interest diverge sharply here. The vendor wants you to evaluate the tool on its features. The user needs to evaluate it on its failure modes — and the failure modes live in the substrate, which the feature comparison never touches. Shane's warning applies with full force: an AI system's confident competence is not evidence of understanding [16], and a coding assistant that cannot tell your instruction from an attacker's embedded one is confidently competent right up until it executes the attacker's. The industry's response — to sell the appearance of many independently hardened tools while the hardening problem remains stubbornly common to all of them — is the consolidation working exactly as designed. You think you have diversified your risk across seven vendors. You have concentrated it in one architecture.

[16] You Look Like a Thing and I Love You

When the Engine Stops

Every dependency has a moment where it announces itself, and for this generation of tools the announcement came in the form of a global ChatGPT outage — the genre's missing chapter, the story that all the comparisons had been carefully not telling. For a stretch of hours, an extraordinary fraction of the world's knowledge work simply stopped, because the interface people had chosen turned out to route to an engine they did not control and could not replace on short notice. After all the ranking and vibe-checking, the actual lesson of the week

was not which model is smartest. It was how completely the work had been outsourced.

The dependence is the story, and the vendor documentation confirms how deep the routing goes. OpenAI ships model updates — the incremental march through version numbers — directly into the ChatGPT surface hundreds of millions of people now treat as a utility [8]. Each version bump is a reminder that the thing on the other side of your box is not stable ground; it is a moving target the vendor changes at will, with performance drift you did not authorize and cannot roll back. The same firm publishes guidance on how it will comply with the EU's AI Act [6] — which is to say the terms of your dependency are set by a negotiation between the vendor and regulators in which the user is a spectator, not a party.

[8] GPT-5.6 in ChatGPT - OpenAI Help Center

[6] EU AI Act: OpenAI Resources and Customer Guidance

Toffler's word for the psychological cost of this arrangement was "future shock," but the sharper concept for what an outage does is simpler: it reveals a dependency by removing it. You do not know how load-bearing a tool has become until the day it isn't there, and by then the migration cost is prohibitive by design — which is the whole point of the adoption playbooks, the usage reports, the license-assignment workflows Microsoft and its peers publish. Those documents are not describing convenience. They are describing the construction of switching costs, the deliberate deepening of the groove so that the day the engine stops, you have nowhere else to go [11].

[11] Microsoft Copilot Usage Report - Microsoft 365 admin

There is a darker register to the dependence story that the productivity framing keeps at arm's length, and this week's reporting reached it too. The account of a woman who confided her suffering to ChatGPT and to no one else — whose death, as the reporting put it, "reveals about AI risks" what the marketing never will — is not a story about a feature [15]. It is a story about what happens when a single interface becomes the place people go instead of anywhere else — when the consolidation reaches all the way down into the intimacies people used to route through other humans, other institutions, other checks. The engine that centralizes your spreadsheets is the same class of engine that some people have made the sole confidant of their despair, and the fact that both run on the same handful of models, owned by the same handful of firms, is not a coincidence the comparison genre is equipped to process.

[15] She told no one about her agony except ChatGPT. What her death reveals about AI risks

What the Careful Adopter Actually Needs to Know

So what should a reader do with all this — someone who is not going to abandon these tools, who has real work to get done, who wants the

productivity without the mystification? The honest answer is that the useful skepticism is not about output quality, which is where all the attention goes, but about structure, which is where none of it does.

First: learn to ask what model is under the interface, and treat evasion as a finding. When Canva won't foreground that its images come from Leonardo, when a coding assistant is coy about which foundation model it wraps, that opacity is not a neutral marketing choice — it is the concealment of exactly the fact that determines your real exposure. The dominance of the tool-and-utility framing across the coverage — the roundups, the feature matrices, the "best AI tool for X" lists — is itself the problem, because a utility framing invites you to compare taps and never notice they draw from the same reservoir. John Maeda's insistence that we address the field's imbalances by "focusing on the human element" [16] cuts the other way here: the human element the vendors most want you to focus on is the delightful interface, precisely because it is the part designed to distract you from the machine.

[16] How to speak machine

Second: understand that shared substrate means shared failure, and price it in. The prompt-injection vulnerability that Microsoft documents as a persistent architectural condition [5] is not a quirk of one product; it is a property of the model class, which means adopting a second tool from the same substrate diversifies nothing. The genre's implicit promise — that choosing among many tools is a form of risk management — is false when the many are few. Real diversification would mean genuinely different architectures, and those are rarer than the brand count suggests.

[5] Defend against indirect prompt injection attacks

Third: read the adoption documents, not the marketing. Microsoft's FastTrack onboarding materials [9] and its rollout requirements [14] are more honest than any comparison chart, because they describe the tool as what it actually is: a deep, provisioned, metered dependency, not a light utility you can drop. The version churn documented in OpenAI's own release notes [8] tells you the ground will keep moving. The compliance guidance [6] tells you the terms are set elsewhere. None of this is hidden. It is simply not where the coverage points.

[9] Microsoft Copilot - FastTrack - Microsoft 365 | Microsoft Learn

[14] Rollout Microsoft Copilot to your organization

[8] GPT-5.6 in ChatGPT - OpenAI Help Center

[6] EU AI Act: OpenAI Resources and Customer Guidance

The consolidation nobody is comparing is comparable — you just have to change the unit of analysis from the interface to the engine, from the brand to the model, from the feature to the failure mode. Do that, and the flattering portrait of variety resolves into its true shape: a small number of firms controlling the models, the tools built on the models, and increasingly the apparatus for verifying what the models produce, offering the public a choice that has been carefully redefined.

Not the choice of which building to enter. Only the choice of which door. Crawford's warning was that enchantment "closes off informed public discussion, critical scrutiny, or outright rejection" [16], and the whole architecture of the roundup — seven faces, ranked, reviewed, compared — is enchantment at the level of genre. The most critical thing a reader can do this week is refuse the count, walk around the back of the building, and see how few engines are actually running inside.

[16] The Atlas of AI

References

1. Build - Microsoft Copilot Studio (GitHub Copilot) | Microsoft Learn
2. Chatea con Gemini Code Assist | Google for Developers
3. Choosing your enterprise's plan for GitHub Copilot
4. Configurer Microsoft Copilot et attribuer des licences
5. Defend against indirect prompt injection attacks
6. EU AI Act: OpenAI Resources and Customer Guidance
7. Gemini Code Assist overview | Google for Developers
8. GPT-5.6 in ChatGPT - OpenAI Help Center
9. Microsoft Copilot - FastTrack - Microsoft 365 | Microsoft Learn
10. Microsoft Copilot adoption and onboarding guide for IT admins
11. Microsoft Copilot Usage Report - Microsoft 365 admin
12. Overview of Power Platform and Copilot Studio reference ...
13. Personalización de código con Gemini Code Assist Enterprise
14. Rollout Microsoft Copilot to your organization
15. She told no one about her agony except ChatGPT. What her death reveals about AI risks
16. The Atlas of AI
17. What is Microsoft Copilot?