The Missing Contributor Class: A Fourth Contributor Taxonomy for AI-Assisted Commits

Jordan Newell
Independent researcher

November 7, 2026

Abstract

Git's credit system recognizes exactly three contributor classes: the author, the human co-author (via the `Co-authored-by:` trailer), and the machine account (the `[bot]`-suffix convention). AI coding agents fit none of these. The result is an ecosystem forced into a binary that satisfies no one: agents masquerade as human co-authors through vendor-default trailers, or their contribution is stripped from history entirely. The 2025–2026 conflict over AI attribution — VS Code's reverted `git.addAICoAuthor`, the Linux kernel's nine-month adopt-to-deprecate arc for `Assisted-by:`, GCC requiring `Assisted-by:` where QEMU rejects it outright — is not a fight about *whether* to attribute AI work. It is a fight about *which existing class* AI must be crammed into. Both available classes are dishonest: co-authorship overclaims human creative agency; the bot class underclaims it to nothing.

We propose a fourth contributor class, defined in the existing trailer-taxonomy tradition (Signed-off-by, Reviewed-by, Suggested-by, Co-authored-by), with a vendor-neutral grammar, an opt-in consent model, and an observer rendering contract. The class makes no claim of human authorship and no claim of erasure — it names the thing neither slot can: non-human creative contribution, disclosed, machine-readable, and accounted separately. We document the attribution conflict as evidence that the gap, not the politics, is primary; we post-mortem the kernel's failed first attempt to extract its design constraints (vendor-neutrality, non-gating) and add a third of our own: never fight for an existing slot; and we specify the class, its grammar, and a reference implementation running in production. We also name the enforcement layer the class deliberately avoids — advisory, actor-differentiating, and *humanostic* gates — and show the attribution war's incidents read as gate-posture data.

Artifact: reference implementation (`commit-msg` hook, `aiattr.policy` = strip/rewrite/off) live in production on a 20-agent fleet across seven hosts since 2026-11-06; 16-case test suite. Reference implementation DOI: `10.5281/zenodo.24034440` (concept DOI; `https://github.com/JordanNewell/assisted-by`). Core grammar: `Assisted-by: <name> [; mode=generated|edited|reviewed]`

Keywords: git; AI attribution; commit trailers; software provenance; AI-assisted coding; open source governance; contributor taxonomy; humanostic enforcement

1 The problem

Every AI-touched commit in the wild files itself into one of two existing classes, and both are wrong:

Class	Mechanism	Who uses it for AI	What it actually claims
Human co-author	`Co-authored-by:` trailer	Claude Code, Cursor, Copilot squash-merges	A human made this. Avatar, contributor-graph credit, creative agency implied
Bot	commit authored by a `[bot]`-suffix account	`copilot-swe-agent[bot]`, `devin-ai-integration[bot]`, GitLab Duo service accounts	Automation executed this. No creative claim, invisible to authorship
AI agent	**nothing**	—	**The class does not exist**

Dependabot never caused a single flame war. Why? Because its class is honest: "automation did this exact thing." Nobody's dignity is involved. The entire heat around AI attribution is class-heat, not attribution-heat. We built a slot for humans and a slot for dumb automation, and left nothing for the thing agents actually are — non-human contributors exercising real creative discretion.

The two ways out both fail:

- **Masquerade.** Default-on vendor trailers put a brand in a human-credit slot. GitHub matches the trailer email against real accounts, so AI co-authors inflate human contributor graphs — the documented "Copilot polluted my contributors graph" complaints. And it breaks in dumb ways: when Claude Code stamps `noreply@anthropic.com` and it turns out a random human registered that address, commits attribute to a stranger.
- **Erase.** Strip hooks and detag policies keep graphs clean and destroy provenance. Future maintainers, auditors, and tooling lose the only machine-readable signal of how a change got made — the signal license auditors, security reviewers, supply-chain provenance tools, and researchers all consume.

Both camps correctly sense the options are dishonest. The war exists because there's nowhere honest to put AI.

1.1 The numbers behind the problem

This isn't a someday problem. The people running the platforms have been quoting the volume for two years:

Who	Said what	When	Where
Sundar Pichai (Google)	"more than a quarter of all new code at Google is generated by AI, then reviewed and accepted by engineers"	Oct 30, 2024	Alphabet Q3 2024 earnings call
Mark Zuckerberg (Meta)	Meta will "have an AI that can effectively be a sort of mid-level engineer" — in 2025	Jan 10, 2025	Joe Rogan Experience #2255
Sam Altman (OpenAI)	in frontier companies, more than half the code is AI-written (paraphrase; transcript unconfirmed)	Mar 21, 2025	Stratechery interview
Satya Nadella (Microsoft)	"maybe 20 to 30 percent of the code that is inside of our repos… are probably all written by software" — "going up monotonically"	Apr 29, 2025	LlamaCon fireside with Zuckerberg (media-transcribed)
Zuckerberg (Meta)	roughly a quarter of code in Meta's repos already AI-written; half within a year	Apr 29, 2025	LlamaCon fireside (on-stage estimate, per CNBC)
Thomas Dohmke (GitHub)	AI is "on track to write 90 percent of code within the next 2–5 years"; "Either you embrace AI, or get out of this career"	Aug 4, 2025	"Developers, Reinvented" blog post
Elon Musk (xAI)	"you don't even bother doing coding. The AI just creates the binary directly"	Feb 2026	xAI all-hands (reported by Reuters; no transcript)
Andrej Karpathy	"There's a new kind of coding I call 'vibe coding', where you fully give in to the vibes, embrace exponentials, and forget that the code even exists"	Feb 2, 2025	@karpathy, X — became Collins Word of the Year 2025

Sit with the Nadella line for a second. Microsoft says 20–30% of the code in its own repos is AI-written. GitHub — *Microsoft's* platform — still has no way to represent a single one of those contributions except as a human co-author or a bot. The CEO of the company that owns the world's largest code host has publicly quantified a class of contributions his own platform cannot name.

And the pattern holds everywhere you look: every leader will tell you how much code AI writes. A targeted sweep (2023–2026; web, X, earnings-call and keynote transcripts) located no credit-mechanics statement from Karpathy, Musk, or Zuckerberg; the other named leaders' public statements concern volume, not mechanics — though Dohmke's GitHub shipped attribution plumbing, which is mechanics by proxy if not by word. Karpathy gave the vibe-coding movement its name and never touched attribution. Volume is discussed at every earnings call; credit mechanics at none. The practical effect is that the credit question sits with the bottom of the ecosystem — individual maintainers fighting hook-by-hook in the dark. Whatever the reason, the observation stands: the class is missing at every altitude at once.

2 Background: the taxonomy we already have

- **Credit trailers** come from the kernel/DCO tradition: `Signed-off-by`, `Reviewed-by`, `Tested-by`, `Acked-by`, `Reported-by`, `Suggested-by`, `Helped-by`, `Co-authored-by`. Every one is a human role. Git itself treats trailers as free-form — no registry, no gating. Which means: nothing needs to change *in git* to add a class. Classes are conventions that renderers adopt.
- **Machine accounts**: GitHub's `[bot]` suffix (GitHub Apps, GA May 2017). Bot badge, excluded from Insights → Contributors. Fine for what it is.

- **Co-authored-by rendering**: GitHub matches the trailer email to a verified account email or `username@users.noreply.github.com`; unmatched emails render as dead text. Contribution counting needs a registered email — which is exactly how AI trailers leak into human stats. GitLab and Forgejo don't render the trailer yet (open feature requests, as of November 2026); Gitea shipped co-author rendering in Sep 2026 — with no notion of an AI class.
- **The taxonomy is moving right now**: GitHub shipped push attribution (Oct 12, 2026) — a third commit identity: author, committer, pusher. Identity plumbing is under active construction. The window is open.

3 The war, on the record (2025–2026)

Actor	What they did	When	Which camp
curl (Stenberg)	adopted `Assisted-by:` for AI attribution — the trailer itself dates to 2013 in curl as human credit	May 2025	tag-it
QEMU	declined AI-derived content outright	Jun 2025	reject
Gentoo Council	AI-assisted contributions "expressly forbidden"	Apr 2024	reject
GitHub community	"Copilot assisted" badge requested	May 2023	ignored — no roadmap entry, ever
VS Code	`git.addAICoAuthor` flips on by default in 1.117	~Apr 22, 2026	244 downvotes on the tracking issue (vscode#314311); bugs fire the trailer with AI disabled
VS Code	softened in 1.118, reverted to off in 1.119	~May 6, 2026	community filed an `Assisted-by` alternative (vscode#313962, May 3); Microsoft engaged in #314311 (May 5)
Linux kernel	`Assisted-by: LLM` merged into the docs	Jan 2026	adopt
Linux kernel	acknowledgment becomes effectively mandatory ("failure to do so may impede the acceptance of your work"); DCO declared human-only	Apr 2026	adopt, peak
Linux kernel	attribution simplified out — history as "a free advertising platform for a bunch of AI companies"	Jul 2026	politics arrives
Linux kernel	`Assisted-by` deprecated after Maintainers Summit; Torvalds: AI-policing rules are "utterly stupid"	Oct 2026	**deprecate wins. ~10% of the 7.3 cycle's 18,533 non-merge commits (through rc7) carried the tag.**
GCC Steering	AI declined except tests; `Assisted-by:` required where permitted	Jul 2026	split
QEMU checkpatch	ERRORs on `Assisted-by:`, `Generated-by:`, robot emoji, AI-name co-authors	Sep 2026	reject, hardened
NetBSD	AI-generated code presumed "tainted" — barred without prior written core approval	since May 2024	reject
Sourcehut	AI-assisted projects "not welcome"	Aug 2026	reject
Kubernetes	bans AI co-author trailers	2025–2026	reject

Read that table again. Nobody is winning. Adoption and rejection are both accelerating. The one project that made tagging *mandatory* detonated its own convention in nine months. Vendors aren't waiting for consensus — they're stamping human-class trailers or routing through bot accounts while the debate burns. And the ecosystem wants this signal badly enough to build it piecemeal, undocumented, three different ways — `git-ai`, `ai-commit-check`, `botcommits.dev` — plus badge actions and academic datasets mining the trailers as research data.

4 Every attempt so far, and why it died

Attempt	Form	Venue	What happened
`AI-assistant: <tool> (<model>)` — Ferdinandy, Dec 2025	blog	blog; no mailing-list follow-up	nothing
`Coding-agent: + Model:` — Rehm, Mar 2026	blog	blog	author retracted it himself in 6 days
AI-as-author vs AI-as-co-author — Foster, Apr 2026	blog, Git Rev News #134	blog	no mailing-list thread ever followed; stays inside the human classes
git-ai (spec v3.0.0)	Git Notes, client-side	GitHub project	deliberately avoids trailers and forge semantics entirely
Barista Labs (McLellan): `Assisted-by: TOOL:MODEL`	blog convention, Jun 2026 (updated Oct 2026 after the kernel drop)	blog	nearest neighbor on grammar — but explicitly *anti-class*: "the tool does not get an avatar… that's the feature, not the bug"
arXiv:2504.16485	empirical study (613 mined instances, 111 devs)	ACM TOSEM 2025	analyzes the practice, proposes nothing
arXiv:2609.07542	policy landscape survey	arXiv	catalogs policies, proposes nothing
"Copilot assisted" badge	feature request	GitHub community #56848	unanswered since May 2023

Here's the part that should stop you for a second: **no proposal for an AI contributor class has ever been posted to git@vger.kernel.org** — as of this writing. The list just hosted the `Assisted-by: LLM` trailer fight (Sep–Oct 2026), so it knows the territory — what it has never seen is the class argument. The blog posts never reached the list. The hill is not just unclaimed — nobody has even walked up to it.

And the space has exactly two live poles, with a hole in the middle:

- **The human-class pole**: vendor trailers claiming co-authorship. Foster's convention.
- **The invisible pole**: Barista Labs wants `Assisted-by` *because* it renders nowhere — "that's the feature, not the bug." The GitHub-removement crowd goes further: take AI names out of contributor lists entirely.

Visible-but-human, or invisible. Those are the options on the table. Nobody — anywhere — argues for the third position: **visible, separately accounted, its own class.** This paper claims that position. A two-pass verification sweep (2026-11-06) confirmed the gap: nothing in the searchable record argues for the third position.

Why the earlier attempts died — the extracted constraints:

1. **Vendor-neutrality.** Brauner stripped the kernel's examples — git history had "started to function a bit like a free advertising platform for a bunch of AI companies." Any convention that smells like vendor marketing dies on contact. The grammar has to be operator-controlled, default-off, brand-agnostic — a field, not an endorsement.

2. **Non-gating.** Torvalds is right about the policing part: AI is like compilers. The class must be pure disclosure. It can never be a CI gate, a policy weapon, or a quality score. Projects that want to reject AI work will keep doing that at their own policy layer; the class must not become their enforcement tool.

3. **Don't fight for an existing slot.** "Use Co-authored-by differently" inherits every fight co-authorship already has. A fourth class competes for nothing.

5 The proposal: a fourth contributor class

5.1 What it is

An **AI-assisted contributor** is a named, non-human agent that exercised creative discretion in producing part of a change. Distinct from a human contributor (author, co-author) and from an automation contributor (bot: fixed task, no discretion).

The human who hits the button stays the author, full stop — and the sole `Signed-off-by` signatory (DCO stays human-only; kernel precedent, kept). The class claims *assistance with creative work*. Never authorship. Never mere execution.

5.2 Grammar

Formalize the incumbent string instead of minting a new one. `Assisted-by:` already won the informal adoption war — curl carried it into AI attribution, the kernel ran it at scale, libkrun and fluxcd codified it, VS Code reached for it during the revert crisis. A spec that formalizes it makes all existing usage conformant overnight, with zero migration. The kernel politics attached to the string are a policy problem, and this spec is explicitly not a policy.

The kernel's deprecation deserves one more word, because a hostile reader will call the string a corpse. It isn't: the Oct 2026 deprecation was a kernel-*policy* event — about mandatory acknowledgment and what the maintainers called advertising — not a verdict on the string itself. Non-kernel adoption (curl, libkrun, fluxcd) never took the mandatory step and never entered the politics; that usage is the adoption base this spec formalizes. Projects that deprecated lose nothing here: the class is opt-in, and a project that wants no AI stamps simply never configures them.

One deliberate choice needs justifying: the sub-parameters live *inside* the trailer's value rather than in sibling trailers (`Assisted-by-mode:` and friends). Value-internal parsing keeps the trailer count stable — consumers that don't know the parameters still see one plain string — and keeps the class from fighting `git interpret-trailers`' tokenizer rather than coexisting with it. A reference parser ships with the implementation; the grammar is one regex.

```
Assisted-by: <agent-name> [; tool=<tool>] [; model=<model>]
             [; mode=generated|edited|reviewed] [; id=<operator-local-id>]
```

- `<agent-name>`: operator-chosen, no registry, no verification — by design. A model name, a fleet agent name, whatever. Yes, `Assisted-by: the robots` is fully conformant. The cute word isn't a bug; it's proof the class can't be captured by brands.

- `mode` is the only field with fixed semantics: `generated` (agent produced content), `edited` (agent transformed human content), `reviewed` (agent critiqued human content).
- Everything optional. The whole trailer optional. **Default: absent.** Nothing stamps itself — no vendor default-on, ever. That's the VS Code lesson, learned at the price of a 244-downvote revert cycle.
- One trailer per contributing agent; trailers repeat when multiple agents contributed to one change.
- `id` is intentionally operator-local. Cross-repo agent identity is out of scope and belongs to Git-Notes-class systems, not commit trailers.
- Mode boundary rule: when an agent both generated and edited within one change, use the strongest claim (`generated`); `reviewed` is reserved for critique-only involvement.
- Real-world examples that are already conformant:

```
Assisted-by: fleet-agent-7; model=local-13b; mode=edited
Assisted-by: Cursor:claude-sonnet-4.6
Assisted-by: LLM coccinelle sparse
Assisted-by: Cursor:claude-opus-4.6
```

  (Reproduced verbatim — no endorsement implied. Sources, top to bottom: a neutral operator-style example; kernel ChangeLog-7.1; the kernel's original grammar; libkrun's contributing guide.)

5.3 The observer rendering contract

This is the actual invention. Everything above already exists in scattered form; what doesn't exist anywhere is a rule for how platforms *treat* the class. Consumers — forges, gateways, anything that already parses `Co-authored-by:` — get three rules:

Rule	Meaning
Never human-render	No avatar. No facepile. No human-class chip. A distinct rendering, visibly not-a-person
Never human-account	Excluded from human contributor stats by default. Optional separate AI-contributors ledger, which never merges into the human one
Never erase	Always rendered somewhere in the commit view. Disclosure that hides itself is indirection, not disclosure

Two scope notes keep the contract honest. First, the never-gates constraint binds *contributors*: nothing here blocks, scores, or rejects a commit. Second, the rendering rules bind *consumers*, and a consumer that ignores them loses nothing in git terms — they are a request on platforms, weighted by what weights all platform conventions: users noticing.

The invisible pole — record but never render — is simpler, and simplicity is a real virtue. It loses to visibility on the demand side: the disclosure has named consumers. License and security auditors need to see, per commit, which changes carry machine-generated content. Supply-chain provenance tooling wants a visible, queryable signal to build on — the same instinct behind SPDX and SLSA's artifact-level attestations, one level down. And the research community has already built trackers and datasets — bot-commits.dev, ASSERT-KTH's 21.8M-row agent-commits corpus — from exactly this signal, mined the hard way from prose trailers that no schema defines. Visibility is what makes those consumers possible;

rendering is what makes them cheap. Whether the demand justifies forge work is precisely what criteria E5 and E6 will measure: we claim the position is correct, not that it is proven.

The ledger also has costs, and honesty requires naming them. A visible AI-contributors ledger is a marketing surface — the Brauner objection transfers from commits to ledgers — but the transfer defangs it: the spec defines per-repo, operator-owned accounting, and any cross-repo aggregation is a third-party product the spec neither enables nor endorses. Agent names are unverifiable by design, so the ledger can be spoofed — and spoofing is inconsequential, because the ledger carries no human credit: a forged `Assisted-by:` line buys nothing a plain string doesn't already. The forge work is real; that is why the ledger is optional, and why E6 treats platform adoption as a measured outcome, not an assumption.

Note what this does to the attribution games currently being played: misfiled AI co-authors stop inflating human graphs — the pollution complaint dies — and the *structural* payoff of misfiling disappears. What does not die is skill-signaling laundering: passing AI work off as untouched human output makes the human look more capable, and no attribution scheme can detect that (§5.4 concedes it). The class claims to end structural misfiling, not bad faith.

5.4 What the class deliberately refuses to do

- No gates, no CI blocks, no policy enforcement — disclosure only (Constraint 2)
- No verification, certification, or scoring of AI involvement — it's a claim, not a metric
- No solution to AI-laundering detection — undetected unattributed code remains undetected; the class just makes *honest* disclosure cheap and legible
- No required telemetry — no model names, versions, or anything an operator doesn't want to write

5.5 The enforcement layer, named: gate postures and *humanostic*

Constraint 2 forbids this class from gating. That constraint is easier to keep than to state, because the war in §3 is not really adopt-vs-reject — it is a war between *gate postures* that the discourse had no name for. Coin one:

> **humanostic** (adj.) — of a guardrail: enforced below the conversational layer, such that it binds human and AI operators identically, with no conversational override path.

The portmanteau (human + agnostic) sits in the engineering lineage of *platform-agnostic*: the variable abstracted away is the operator's species. Three enforcement postures exist, and the war's major incidents map onto them:

Posture	Enforcement	Incident from §3
Advisory	prompt, mandate, or policy text; binds only whoever complies	kernel's Apr 2026 mandatory acknowledgment ("failure to do so may impede the acceptance of your work"); Gentoo's express prohibition; Kubernetes' trailer ban — normative text with no machinery
Actor-differentiating	machine-enforced, binds by identity or content origin	QEMU checkpatch ERRORs on `Assisted-by` and AI-name co-authors — enforced, but gates AI content specifically
Humanostic	machine-enforced, binds identically	none in the war — which is the point: every gate anyone built either relied on compliance or singled AI out

The mapping does real analytic work. The kernel's Apr 2026 mandate failed not (only) because AI-policing is "utterly stupid," as Torvalds put it, but because it was an advisory gate: machinery-free text binding only the compliant, enforced socially until the social enforcement became advertising. QEMU's checkpatch holds because it is machine-enforced — but it differentiates actors, a policy choice a project is entitled to. What nobody built in this war was a humanostic gate, because gates bind contributors and constraint 2 forbids exactly that of this class.

The term has ancestry, and honesty requires citing it. Mandatory access control already names machine-enforced constraints that bind administrators [16]; "a government of laws and not of men" (Adams, Massachusetts Constitution, 1780) is the 246-year-old constitutional statement of the same shape. What neither names — and what no security or governance literature we located names — is the symmetry claim: the bound actors include humans and AI agents *as equals*, indifferently. Enforcement does not entail that symmetry: GitHub rulesets ship bypass lists of named actors, and permission systems routinely gate agents while waving humans through. Humanostic is a deliberate design choice, not a byproduct of enforcement, and its boundary is a trust boundary rather than an absolute: a humanostic gate is one whose conversational channel offers no override and whose boundary-transcending overrides (config changes, support channels) are themselves gated.

The fleet this paper's reference implementation runs on operates all three postures: advisory WIP limits in skill text, permission prompts that gate agents but not the operator, and humanostic branch protection — the `enforce_admins` rule that declined the author's own direct push. Disclosure (§5) and enforcement are different layers; the war happened, in part, because projects conflated them — and the class stays out of the enforcement layer entirely.

6 Objections, pre-answered

"Free advertising for AI vendors." That's the current system — vendor trailers default-on, stamped into human credit. The class replaces the ad channel with a neutral field that nothing stamps unless a human chooses to, and that never renders as a human. Killing the masquerade *reduces* vendor surface area, not increases it.

"Policing AI is stupid; it's like compilers." Agreed — and this polices nothing. We don't credit compilers as co-authors; we also don't hide that compiled binaries came from source. The class is exactly that third slot: provenance without personhood.

"It'll be gamed." Attribution is gamed *today*, against the human class — fake co-authors, polluted graphs, brands wearing collaborators' faces. A separate class shrinks that payoff, because the AI ledger carries no human credit to steal.

"Stripping is easier." Stripping stays available to anyone who wants it — the class imposes nothing. It serves the position the discourse keeps circling: disclose accurately, decide locally.

7 Roadmap

1. **This paper** — timestamped (DOI via Zenodo) before anything public.
2. **Reference implementation at the workflow layer — done 2026-11-06**: a portable commit-msg hook with the grammar implemented (`aiattr.policy` = strip/rewrite/off, `aiattr.agent` for operator-chosen names, warn-only grammar validation), a 16-case test suite, running in production on a 20-agent fleet across seven hosts. The DCO-tooling pattern: a tool, not a forge. Agent platforms (openclaw, etc.) are downstream *observers*, not the vehicle.

3. **git@vger** — the never-mailed venue. Post a stripped-register variant of this paper (constraints, grammar, kernel post-mortem — no rhetoric; the list punished mandatory policy, and it punishes manifesto voice too). Frame as trailer-taxonomy discussion under constraints 1–2, with the post-mortem as evidence that policy-gating was the failure mode, not disclosure.

4. **Observer adoption** — GitHub community feedback (the 2023 badge thread, still unanswered as of November 2026), OpenSSF ai-ml-security WG, and the forge feature-request stack (GitLab #538959: co-author rendering and AI-class rendering are the same parsing layer).

7.1 Evidence pack — falsifiable success criteria (set 2026-11-06, before build)

The difference between a standard and an opinion is adoption. Adopted = measurable. Criteria set publicly before the reference implementation existed, so the verdict can't be moved after the fact. 90-day window from DOI:

#	Evidence	Proves	Measured by	Target (90d)
E1	DOI'd spec, timestamped	priority claim on the public record	Zenodo record live, dated	1 record
E2	Reference impl running in a production fleet	it works in production, not prose	author's own commit history post-migration (conformant trailers, zero vendor-class trailers)	fleet migrated, daily use
E3	Third-party adoption	standard, not opinion	external repos with conformant `Assisted-by:` stamps; tool installs/forks	≥3 external repos or ≥25 tool installs
E4	git@vger engagement	gatekeepers engaged, constraints 1–2 survive contact	thread posted; substantive replies (not drive-by rejections)	thread + ≥2 substantive technical replies
E5	Dataset uptake	the research crowd needs this corpus	dataset DOI downloads/citations; direct researcher contact (ASSERT-KTH et al.)	≥1 academic citation or acknowledgment
E6	Platform/vendor movement	the class becomes real	response or adoption on anthropics/claude-code#36105, GitHub community #56848, GitLab #538959, or a vendor shipping the grammar	≥1 documented movement

Verdict rule, fixed now: **≥3 of E1–E6 hit = standard-track effort, continue. Zero = well-documented opinion, say so and stop.** 1–2 = re-examine venue strategy before more effort. This table ships in the paper.

References

Primary sources: reference-URL sweep Nov 7, 2026 — 34/34 resolving, zero dead, four content corrections applied; one phantom citation ("GitMCx") caught and excluded during prior-art verification. Adversarial fact-audit, Nov 7, 2026: 22 claims checked against primary sources; seven corrections applied, including a 10× commit-count conflation caught before publication. Sweep protocol for the negative-existence claims (class gap, attribution silence): lore.kernel.org and LKML mirrors (full-text), GitHub code and issues search, arXiv, Hacker News, Lobsters, and forge feature trackers, 2023–2026; query families: trailer strings (`Assisted-by`, `Co-authored-by` + AI names), class phrasing (contributor class, AI badge, fourth class), and per-name credit-mechanics queries.

1. Linux kernel adoption: `Documentation/process/coding-assistants.rst` (Jan 2026); `submitting-patches.rst` Apr 2026 wording ("you need to acknowledge… failure to do so may impede the acceptance of your work"); checkpatch Assisted-by recognition patch, Apr 30, 2026, `https://patchew.org/linux/20260430-fix-checkpatch-assisted-by-v1-1-46d3ef68a1de@uber.com` (earlier independent version by Sasha Levin, Mar 2026, commit d1db4118489f); ChangeLog-7.1 verified containing 328 `Assisted-by` mentions (309 in strict trailer form — a line beginning with four spaces followed by `Assisted-by:`, the git-changelog convention) (`https://www.kernel.org/pub/linux/kernel/v7.x/ChangeLog-7.1`)

2. Kernel deprecation arc: Sasha Levin v1, Oct 9, 2026, `https://lore.kernel.org/lkml/20261009083137.3-1-sashal@kernel.org/`; applied v2, Oct 13, 2026 (mirror lkml.iu.edu/2610.1/11749.html — source of the "roughly 10% of the 18,533 non-merge commits in 7.3 through -rc7" statistic; LWN counts 1,884 tagged), applied by Jonathan Corbet Oct 14; 2026 Maintainers Summit consensus to drop [LWN 1099594]; Brauner simplify-attribution RFC, Jul 1, 2026 (merge commit 816d9992d9ed); Torvalds "utterly stupid" on LKML, Oct 13, 2026 (TNW coverage, Oct 16, 2026)

3. VS Code: microsoft/vscode#314311 (244 downvotes / 30 upvotes, as of Nov 7, 2026); `https://github.com/orgs/community/discussions/194075` (Apr 27, 2026 complaint thread, OP 145 upvotes); HN id 47989883 (1,513 points, as of Nov 7, 2026); microsoft/vscode#313962 — `Assisted-by` alternative filed by community member Nato Boram (May 3, 2026); Microsoft engagement (dmitriv, May 5, 2026) in #314311. Reaction counts are snapshots and drift.

4. GitHub mechanics: `https://docs.github.com/en/pull-requests/how-tos/commit-changes/creating-a-commit-with-multiple-authors`; community discussion 195253 (bot exclusion from contributors)

5. Copilot agent: GitHub changelog, Sept 23, 2025 (GA), `copilot-swe-agent[bot] <198982749+Copilot@users.noreply.github.com>`; community discussion 179983; changelog Oct 12, 2026, push attribution public preview

6. Blog proposals: bence.ferdinandy.com (Dec 29, 2025); fabiorehm.com (Mar 2, 2026); Git Rev News #134 (Apr 30, 2026); `https://dafoster.net/articles/2026/04/17/ai-attribution-in-git`

7. Nearest-neighbor prior art: Sean McLellan (Barista Labs), "The AI contribution label belongs in the commit, not the meeting notes," baristalabs.io, Jun 20, 2026, updated Oct 23, 2026 (grammar-aligned, anti-rendering); git-ai project + git_ai_standard_v3.0.0, `https://github.com/git-ai-project/git-ai` (Git Notes authorship namespace, client-side, no forge semantics); GitHub community #56848 (2023 badge ask), #191565, #195253 (removal-direction requests); forge/vendor feature-request stack: GitLab #538959 (co-author rendering), Gitea #35496, anthropics/claude-code#36105 (`Assisted-by` instead of `Co-Authored-by`)

8. Academic: arXiv:2504.16485, Kashif, Liang, Tahir, "On Developers' Self-Declaration of AI-Generated Code: An Analysis of Practices" (ACM TOSEM 2025); arXiv:2609.07542, Hora, Robbes, Zacchiroli, "We Permit the Use of AI, but […]: The Landscape of AI Policies in Popular Open Source Projects" (Sep 2026)

9. Builder quotes: Pichai — Alphabet Q3 2024 earnings call, Oct 30, 2024 (PCMag); Zuckerberg "mid-level engineer" — JRE #2255, Jan 10, 2025; Nadella — LlamaCon fireside, Apr 29, 2025 (The Verge, CNBC); Dohmke — "Developers, Reinvented," Aug 4, 2025 (figure synthesizes a 22-developer GitHub survey and Amodei's public estimate); Musk — xAI all-hands, Feb 2026, reported by Reuters, no transcript (cited with caution); Karpathy — vibe coding, X, Feb 2, 2025 (x.com/karpathy/status/1886192184808149383); "hottest new programming language is English," X, Jan 24, 2023 (x.com/karpathy/status/1617979122625712128); Software 3.0 — YC AI Startup School keynote, Jun 17, 2025; Altman — Stratechery, Mar 21, 2025 (paraphrase via secondary outlets); "Reflections," blog.samaltman.com, Jan 2025; Collins Dictionary named "vibe coding" Word of the Year 2025 (BBC)

10. Policies: GCC Steering (Jul 29, 2026); QEMU `docs/devel/code-provenance` + checkpatch (Sep 7, 2026); Gentoo Council AI policy (ban dated Apr 2024); NetBSD tainted tag; Sourcehut ToS, Aug 27, 2026 (DeVault); curl, "Death by a thousand slops," Jul 14, 2025

11. Ecosystem tooling: `https://github.com/git-ai-project/git-ai`; `https://botcommits.dev`; `https://huggingface.co/datasets/ASSERT-KTH/agent-commits-raw` (21.8M rows); fluxcd/agent-skills commit-assisted-by

12. Reference implementation: JordanNewell/`assisted-by`, v1.0.0 — Zenodo concept DOI `10.5281/zenodo.24034440`, `https://github.com/JordanNewell/assisted-by`, spec page `https://jordannewell.github.io/assisted-by/`

13. Standards: SPDX 3.0.1 AI profile; CycloneDX ML-BOM; SLSA v1.0 provenance; OpenSSF Model Signing v1.0 (Apr 4, 2025) — all artifact/model-level, none commit-level

14. Git taxonomy: `https://git-scm.com/docs/SubmittingPatches` (trailer family); 2026 Git Contributors' Summit AI-contribution discussion, unresolved — LWN 1096819 (Sep 2026 summary); Schindelin's summary on lore.kernel.org/git; the follow-on `Assisted-by: LLM` debate and its dropping, Git Rev News #140 (Oct 31, 2026)

15. openclaw observer internals: commit f953dbdf, Sep 18, 2026, linear trailer scan (1,986ms → 17ms); `src/agents/git-coauthor-attribution.ts` (human-only credit, agents/bots explicitly excluded); docs.openclaw.ai/concepts/user-model

16. Enforcement-layer references (§5.5): NIST CSRC glossary, "mandatory access control" (policy decisions by central authority; users cannot change access), `https://csrc.nist.gov/glossary/term/mandatory_access_control`; Massachusetts Constitution Part I, Art. XXX (drafted by John Adams, 1780), "a government of laws and not of men"; GitHub Docs, managing a branch protection rule ("restrict who can push" actor exemptions; ruleset bypass lists for named users, teams, and deploy keys), `https://docs.github.com/en/repositories/configuring-branches-and-merges-in-your-repository/managing-protected-branches/managing-a-branch-protection-rule`; GitHub changelog, "Bypass branch protections with a new permission," Aug 18, 2022

17. Green, B., "The flaws of policies requiring human oversight in computer algorithm decision-making," *Computer Law & Security Review* (2022) — the oversight-failure literature the humanostic posture names: nominal oversight fails because advisory mechanisms bind only the compliant

18. Kiškis, M., "Legal framework for the coexistence of humans and conscious AI," *Frontiers in Artificial Intelligence* 6:1205465 (2023) — nearest-neighbor "human agnostic" usage; the term is used there in a decentering sense (frameworks not premised on human primacy), distinct from the enforcement-symmetry sense defined in §5.5