BrightMeld

RESEARCH · WHITEPAPER · AUGUST 2026

Controlled AI: What the Evidence Says About Doing It Right

AI accelerates. Discipline decides.

MARC EPSTEIN · MANAGING PARTNER, BRIGHTMELD

~3,200 WORDS · 13 MIN READ

38 SOURCES · EVERY FIGURE VERIFIED AGAINST ITS PRIMARY

+26%

completed tasks with AI assistance — randomized trial, 4,867 professional developers, three enterprises.

MANAGEMENT SCIENCE · PEER-REVIEWED

−19%

slower with AI — randomized trial, 16 experienced maintainers in mature codebases, who believed they were 20% faster.

METR RCT · PREPRINT

BOTH RESULTS ARE TRUE. THE DISTANCE BETWEEN THEM IS THE SUBJECT OF THIS PAPER.

00 Two true results

Start with two findings that are both true: AI coding assistance measurably speeds developers up, and it measurably slows them down. Which result your organization gets is not luck — it is a matter of control.

A randomized controlled trial across 4,867 professional developers at Microsoft, Accenture, and a Fortune 100 company — peer-reviewed, published in *Management Science* — found that AI coding assistance produced a 26% increase in completed tasks.

A randomized controlled trial of 16 experienced open-source maintainers, working in mature codebases they knew deeply, found that AI assistance made them 19% slower — while they believed it had made them 20% faster.

Neither study is wrong. Your vendor quotes the first one. Your staff engineers feel the second one. And the distance between them is not noise — it is the most useful fact in the research.

This paper is an evidence guide to governing AI-assisted software development: what the research actually shows about where AI coding fails, what it shows about what works, and what a team can do with both on Monday morning. Every number in it is someone else's, published, and linked. None of it requires trusting the author.

The short version: AI reliably accelerates code production, and the research now maps where it fails when ungoverned — specification, self-assessment, and security. That map is good news, because a mapped failure is a manageable one. The best-supported response — and the way a leading model builder runs its own AI systems — is to put human judgment up front, ratifying intent before generation, and put machines behind it, verifying mechanically after. Use it up front; don't chase it after the fact.

01 The pattern in the results

The pattern across the research is this: benefits are easiest to capture where work is bounded and verification is cheap; costs surface as context, consequence, and verification burden rise.

Be clear about what that claim is and isn't. No single study isolates the cause of the gap between those two trials — the studies vary task type, developer tenure, codebase maturity, tool generation, and outcome measures all at once. And no controlled trial has compared different placements of human oversight for AI-generated work. What follows is the operating posture the evidence best supports, not a completed proof.

But the studies bend the same way.

Start with the gains, because they are real. A meta-analysis of 23 studies found a significant positive productivity effect (g = 0.33). The *Management Science* trial measured a 26% increase in completed tasks across three enterprises. Google's DORA research, which had found AI adoption negatively associated with delivery performance in 2024, measured the throughput relationship turning positive in 2025. Telemetry from 22,000 developers shows task throughput up 33.7% and epics completed up 66.2% at high AI adoption.

Now read where those gains live — not as asterisks, but as coordinates. The same meta-analysis found gains "tend to be larger in controlled experimental settings, while effects are smaller in open-source and enterprise contexts." The *Management Science* gains concentrated among less experienced developers. The most-quoted number in the industry — 55.8% faster — came from 95 freelancers building a toy HTTP server from scratch: a greenfield task with near-zero verification cost. The studies are not saying the upside is fake. They are telling you where it concentrates: wherever verification is cheap.

The costs are just as measurable, and they cluster at the other end. DORA's stability penalty persisted through both years — their own framing: AI is an *amplifier* of the engineering system it lands in. And the same 22,000-developer telemetry that shows the throughput gains shows incidents per PR roughly tripling, median review time quintupling, and 31% more pull requests merging with no review at all.

Organizations cannot control every variable in those studies — but they can control where and how verification happens. That boundary is what this paper is about.

One more finding sets up everything that follows. In the METR trial, developers *felt* 20% faster while *measuring* 19% slower — and when METR tried to re-run the study with late-2025 tools, 30–50% of developers refused to submit tasks without AI, which METR says makes clean trials of this question increasingly infeasible (their new point estimates lean positive, with confidence intervals straddling zero). Read those two facts together: you cannot learn what AI is doing to your organization by asking people how it feels, and the research community is losing the ability to answer it for you. You will have to measure it yourself. Fortunately, that turns out to be the cheapest recommendation in this paper.

02 Where the costs accumulate

Three exhibits: models fail at knowing what to build, green checkmarks overpromise, and the human gate sits where it works worst. Each one ends with what that makes possible.

EXHIBIT A — THE FAILURES ARE UPSTREAM, AT INTENT

On VERINA, a benchmark of 189 verification tasks, the best general model wrote correct code 72.6% of the time, sound and complete formal specifications 52.3% of the time, and successful proofs 4.9% of the time. That's one benchmark, in one formal language — but the gradient it shows repeats everywhere the question is asked. At CMU, researchers found frontier agents produce unfaithful formal specifications 22–49% of the time — omitting input assumptions, accepting wrong outputs — and that using an LLM as the judge misses a quarter of those failures. On the informal side, a benchmark of requirements elicitation found models surface less than half of users' implicit requirements; its authors state the field's position plainly: "the bottleneck of LLM-based automated software development is shifting from generating correct code to eliciting users' requirements."

The field evidence matches the benchmarks. A study of 33,000 agent-authored pull requests found they merge best on documentation, CI, and build chores — and worst on bug fixes and performance work, the tasks that require understanding. An analysis of 3,843 agent work sessions found failures are predominantly *epistemic* — wrong beliefs about the task — that they "begin within the first few execution steps," and that they "remain hidden until recovery is no longer possible." And across 20,574 real-world coding-agent sessions, 91.49% of visible misalignments between what the developer wanted and what the agent did required explicit human correction. Agents rarely notice they've misunderstood you.

There is also one well-documented internal case of a company testing whether a model could own definitions outright. Anthropic — building its internal analytics system — tried bootstrapping its canonical metric definitions with an LLM. The output looked plausible and scored *net-negative* against a smaller, human-curated set: it encoded the very ambiguities it was supposed to eliminate. Their standing rule now: the model drafts the documentation; a human owns the

definition. That's governed analytics, not application code — but the mechanism transfers: plausible-looking model-authored definitions are precisely where quality dies quietly.

> **WHAT THIS MAKES POSSIBLE**
>
> A division of labor with measurement behind it. In the one four-condition comparison of requirements production judged against the ISO 29148 standard, the hybrid arrangement — humans collaborating, AI synthesizing — beat both pure-human and pure-AI conditions outright. The model drafts; a human ratifies — and the evidence gives no support for delegating that final ownership to the model. The job doesn't disappear with better models. It *is* the job.

EXHIBIT B — THE FALSE GREENS

A green checkmark is not what it appears to be — measurably, three different ways.

Passing tests isn't correctness. When researchers layered property-based checks over LLM solutions that had already passed their unit tests, 18–23% failed outright.

Correctness isn't security. On BaxBench, a benchmark of 392 backend applications, researchers didn't just scan the AI-generated code — they ran exploits against it, and succeeded against roughly *half of the functionally correct programs* each model produced. The code worked and was exploitable, at the same time.

And model progress isn't fixing it. Veracode has run the same security benchmark across more than 150 models over two years, through GPT-5-class and Claude-4-class releases: "Two years of 'revolutionary' model releases have moved the security needle from approximately 55% to... approximately 55%." The pattern inside that number is telling: models now ace the famous vulnerability classes (SQL injection: 82% pass) and remain near-blind to taint-tracking classes (cross-site scripting: 15%; log injection: 13%). Meta's own security benchmark found the sharper version: "models with superior coding abilities were more susceptible to suggesting insecure code."

One failure class deserves special mention because ordinary diff review is poorly suited to catch it: models hallucinate package names — 5.2% of the time for commercial models, 21.7% for open-source, with 43% of the fake names recurring predictably across runs. A researcher registered one commonly hallucinated name as an empty package; it drew more than 30,000 authentic downloads in three months. (Demonstrated exploitability; no attributed breach in the wild yet — but the import statement looks exactly like a real one.)

WHAT THIS MAKES POSSIBLE

Each of these failure classes has controls that catch what unit tests and diff-reading miss — property tests for logic, exploit-grade and static gates for security, registry verification for dependencies. None is complete; their value comes from overlap, mechanically enforced. Waiting for model scaling to make them unnecessary is the one strategy with two years of flat data against it. The toolbox, meanwhile, exists today.

EXHIBIT C — THE MISPLACED GATE

The default control in most organizations — a human reading AI-generated diffs at the end of the pipeline — is the one control the evidence says is failing. Three measured ways, none of them the reviewers' fault.

First, verification quietly became the job. Detailed instrumentation of AI-assisted programming sessions found developers spending 51.4% of their time verifying suggestions — while most organizations still budget review as a step at the end.

Second, it doesn't scale. In the Faros telemetry, median review time rose 441% — or the gate simply gets skipped, with 31% more PRs merging unreviewed. The field outcome, from a study of 302,000 verified AI-authored commits: more than 15% of commits from every AI assistant introduced at least one issue, and roughly a quarter of those issues were still alive at the repository's latest version.

Third — and most interesting — plausibility defeats vigilance. In the Stanford security study, developers using an AI assistant wrote less secure code *and* were more confident it was secure. In Sonar's 1,149-developer survey, 96% of developers said they don't fully trust AI-generated code to be functionally correct — and only 48% say they always check it before committing. The top AI frustration in Stack Overflow's 49,000-developer survey: answers that are "almost right, but not quite" (66%) — which is precisely the property that makes vigilance expensive. Reviewers even feel more positive toward AI-authored pull requests while those PRs carry more redundancy than human ones.

The tempting countermove — have AI review the AI — inherits a correlated and exploitable version of the same problem: simply framing a change as bug-free cuts LLM reviewers' vulnerability detection by 16–93%. A reviewer drawn from the same distribution as the generator checks the code against itself, not against intent.

None of this is new to engineering as a discipline. Two decades of clinical automation-bias research — the medical field's version of this exact problem — reached a settled conclusion: over-reliance on a mostly-right assistant is mitigated by gate *design* — accountability, workload, placement — and never by asking humans to be more vigilant.

> WHAT THIS MAKES POSSIBLE
>
> This is a placement problem, not a people problem. Which means it's fixable by decision, not by hiring.

03 What the evidence supports doing

What works is not mysterious: checkable governing artifacts, structured workflows, mechanical enforcement, deliberate governance. Here is that playbook in three tiers of evidentiary strength — labeled, because the difference between evidence and inference is worth protecting.

DIRECTLY DEMONSTRATED — SOMEONE MEASURED IT

- *Machine-checkable governing artifacts carry the signal.* In the largest benchmark of specification-governed generation (12,504 formal specs), adding natural-language prose beside the formal spec produced no significant improvement — the checkable artifact does the work. Separately, feeding architecture documents and an explicit implementation plan to the generator as first-class input measurably improved architectural conformance and functional correctness.

- *Ratified definitions beat model-bootstrapped ones, and enforcement must be mechanical.* Anthropic's net-negative bootstrapping result, above — and its sequel: their system's offline accuracy decayed from ~95% to ~65% within a month until documentation maintenance stopped being a norm and became a merge gate; roughly 90% of their data-model changes now carry the documentation update in the same

diff. One internal case, in analytics rather than application code — but it is a rare measured decay curve, and the mechanism (discipline decays; gates don't) is not domain-specific.

- *Hybrid beats both extremes at requirements.* The ISO-judged four-condition comparison, above.

STRONGLY SUPPORTED — CONVERGING EVIDENCE, NO CONTROLLED TRIAL

- *Structure over free-running autonomy.* In the one rigorous mapping of the adjacent model-driven-engineering field — 86 studies — 96.5% of systems used structured, non-agentic workflows, and the authors conclude "the workflow itself, rather than the autonomy of the LLM, is where most of the methodological complexity in current approaches is concentrated." That is prevalence, not a head-to-head trial — but the near-unanimity of where serious work concentrates is itself information.

- *Every source that measures governance finds it associated with the outcome.* The one peer-reviewed churn study to date (IEEE Transactions on Software Engineering) found *no general increase* in code churn in 151 repositories that deliberately manage how AI is used — the degradation other studies see is not fate. A CMU synthesis of 3,100 practitioner opinions puts it directly: "review is the control point through which a coding agent's effect on software is decided... AI does not fix the sign of that effect: the team sets it." DORA finds two years running that control systems — automated testing, mature version control, fast feedback — are associated with stability under AI volume, and its ROI research finds gains compound where those foundations exist.

REASONABLE ENGINEERING INFERENCE — LABELED AS SUCH

- The full gate placement described next. No one has tested it end-to-end as a package. It is assembled from the demonstrated and supported findings above — and it is consistent with how a leading model builder governs its own AI systems internally.

A note on what is *not* on this list: spec-driven development tooling. The flagship products in that category ship with no benchmarks and no comparative evaluation, and the most-quoted efficacy statistic in that market — "60–80% fewer rework cycles" — traces to no study at all.[1]

04 Move the gate

Three placement rules, applicable Monday: humans on intent, machines on verification, and human review made scarce and sharp.

1. **Upstream, on intent.** Humans ratify the specifications, contracts, and risk decisions — before generation. These are small, high-leverage artifacts; reviewing them is the work automation bias grips least, and they are the exact artifact class the evidence says not to delegate. The model drafts; a human owns the definition.

2. **Downstream, mechanized and layered.** Executable checks matched to failure class, enforced as merge gates — because in the one measured case, discipline decayed within a month, and human diff-vigilance is the one control that measurably fails at volume.

3. **Human review retained — risk-stratified and adversarial, not flat.** Triage review depth by blast radius. Assign accountability per merge. Prompt reviewers to refute, not confirm. These are the mitigations the code-review and clinical-automation literatures independently converge on.

THE MAP FROM FAILURE CLASS TO CONTROL

FAILURE CLASS	THE EVIDENCE	CONTROLS THAT TARGET IT
Wrong or incomplete intent	Specs sound 52.3% vs code correct 72.6%; <50% of implicit requirements elicited	Human ratification of spec/contract before generation
Logic errors behind passing tests	18–23% of test-passing solutions fail property checks	Property/behavioral tests layered on unit tests
Exploitable-but-correct code	Exploits executed on ~half of correct backends; flat ~55% security line across 150+ models	Exploit-grade and static security gates in CI
Hallucinated dependencies	5–22% hallucination rates; 43% recur predictably	Lockfiles, registry allow-lists, verification before install
Duplication and debt drift	Copy-pasted lines now exceed refactoring moves (vendor telemetry); >15% of AI commits introduce issues (302k-commit study)	Composition metrics tracked; refactoring budgeted
Review overload and rubber-stamping	Review time +441% or bypassed; confidence inversion	Risk-stratified adversarial review; accountability per merge
Silent decay of curation	~95% → ~65% in one month (Anthropic case)	Spec/doc updates enforced in the same diff (merge gate)

None of this says “less AI.” The throughput the trials measured is real. Teams that move the verification boundary to where the failures actually are put themselves in position to make it compound; teams that don’t tend to watch that gain reappear as review queues, rework, and incidents. Use it up front; don’t chase it after the fact.

05 Measure the result

Here is the good news promised earlier: whether AI is paying off in your organization is a question you can answer yourself, in about a quarter, with data you already collect. The one method the evidence rules out is sentiment — developers felt 20% faster while measuring 19% slower.

Six outcomes, most of them already in your systems: lead time for changes; review time per change; escaped defects; reversion and rework rate; incidents per change; unreviewed-merge rate. Baseline them, move the gate, and compare quarters.

The research community, by its own account, is losing the ability to run clean trials on this question. Your organization doesn't need them to. The era of settling this by survey — or by vendor slide — is over.

06 What remains unknown

Stated plainly, because the credibility of everything above depends on it.

No controlled comparison of upstream versus downstream gate placement for AI-generated work exists. Nobody has benchmarked spec-driven tooling against iterative prompting — including the vendors shipping it. The strongest volume telemetry is vendor-produced (flagged where cited); its direction is consistent with the peer-reviewed core, which is why the picture holds. And silent failure — output that is wrong, plausible, and used without objection — is an open problem everywhere, including at the best-resourced AI companies, who say so themselves.

07 The invitation

What the evidence supports is an operating posture, not a formula: ratify intent up front, mechanize verification behind, and measure your own outcomes. Teams that adopt it position themselves where the evidence says the upside concentrates. Teams that don't are re-running the experiment the pessimistic studies already ran — at production scale, with their own codebase as the test article.

AI accelerates. Discipline decides.

[1] The claim circulates in SEO content citing other SEO content; the trail terminates in a product review containing no such evaluation. GitHub's own Spec Kit — the flagship of the category — publishes no benchmarks.

References

In order of first appearance. Evidence type labeled per entry.

01 Cui, Demirer, Jaffe, Musolff, Peng & Salz. "The Effects of Generative AI on High-Skilled Work: Evidence from Three Field Experiments with Software Developers." *Management Science*, 2025. PEER-REVIEWED

02 Becker, Rush, Barnes & Rein (METR). "Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity." arXiv:2507.09089, 2025. PREPRINT · RCT

03 Maier, Gunzenhäuser, Schweisthal, Schneider & Feuerriegel. "A meta-analysis of the effect of generative AI on productivity and learning in programming." arXiv:2605.04779, 2026. PREPRINT

04 DORA (Google). "Accelerate State of DevOps Report 2024." INDUSTRY SURVEY

05 DORA (Google). "The 2025 DORA Report: State of AI-assisted Software Development." INDUSTRY SURVEY

06 Faros AI. "The Acceleration Whiplash: 2026 Engineering Report." INDUSTRY TELEMETRY

07 Peng, Kalliamvakou, Cihon & Demirer. "The Impact of AI on Developer Productivity: Evidence from GitHub Copilot." arXiv:2302.06590, 2023. PREPRINT

08 METR. "We Are Changing Our Developer Productivity Experiment Design." metr.org, Feb 2026. RESEARCH ORG

09 Ye, Yan, He, Kasriel, Yang & Song. "VERINA: Benchmarking Verifiable Code Generation." ICLR 2026; arXiv:2505.23135 (v3). PEER-REVIEWED

10 Agarwal, Parno, Welleck et al. "Verus-SpecGym: An Agentic Environment for Evaluating Specification Autoformalization." arXiv:2605.26457, 2026. PREPRINT

11 Jin et al. (Peking University). ReqElicitGym — benchmark of LLM requirements elicitation. arXiv:2602.18306, 2026. PREPRINT

12 Ehsani et al. "Where Do AI Coding Agents Fail? An Empirical Study of Failed Agentic Pull Requests in GitHub." MSR 2026; arXiv:2601.15195. PEER-REVIEWED

13 Zhao, Li, Barr, Sarro, Ye et al. "Failure as a Process: An Anatomy of CLI Coding Agent Trajectories." arXiv:2607.09510, 2026. PREPRINT

14 Tang et al. "How Coding Agents Fail Their Users: A Large-Scale Analysis of Developer-Agent Misalignment in 20,574 Real-World Sessions." arXiv:2605.29442, 2026. PREPRINT

15 Chang, Peng, Leder, Jiao & Cherry (Anthropic). "How Anthropic enables self-service data analytics with Claude." claude.com, June 2026. CASE STUDY

16 Salgado Neto, Araujo & de Souza Santos. “Collaborative and AI-Supported Requirements Elicitation: An Empirical Study.” arXiv:2606.24060, 2026. PREPRINT

17 Bose. “From Prompts to Properties: Rethinking LLM Code Generation with Property-Based Testing.” FSE Companion 2025. PEER-REVIEWED · WKSHP

18 Vero, Mündler et al. “BaxBench: Can LLMs Generate Correct and Secure Backends?” ICML 2025; arXiv:2502.11844. PEER-REVIEWED

19 Veracode. “2025 GenAI Code Security Report” and “Spring 2026 GenAI Code Security Update.” INDUSTRY

20 Bhatt et al. (Meta). “Purple Llama CyberSecEval: A Secure Coding Benchmark for Language Models.” arXiv:2312.04724, 2023. INDUSTRY RESEARCH

21 Spracklen et al. “We Have a Package for You! A Comprehensive Analysis of Package Hallucinations by Code Generating LLMs.” USENIX Security 2025; arXiv:2406.10279. PEER-REVIEWED

22 Lanyado (Lasso Security). “AI Package Hallucinations.” lasso.security, March 2024. PROOF OF CONCEPT

23 Mozannar et al. “Reading Between the Lines: Modeling User Behavior and Costs in AI-Assisted Programming.” CHI 2024; arXiv:2210.14306. PEER-REVIEWED

24 Liu, Widyasari, Zhao, Irsan & Lo. “Debt Behind the AI Boom: A Large-Scale Empirical Study of AI-Generated Code in the Wild.” arXiv:2603.28592, 2026. PREPRINT

25 Perry, Srivastava, Kumar & Boneh. “Do Users Write More Insecure Code with AI Assistants?” ACM CCS 2023; arXiv:2211.03622. PEER-REVIEWED

26 Sonar. “State of Code Developer Survey Report 2026.” n=1,149, fieldwork October 2025. INDUSTRY SURVEY

27 Stack Overflow. “2025 Developer Survey — AI section.” ~49,000 respondents. INDUSTRY SURVEY

28 Huang et al. “More Code, Less Reuse: Investigating Code Quality and Reviewer Sentiment towards AI-generated Pull Requests.” MSR 2026; arXiv:2601.21276. PEER-REVIEWED

29 “Measuring and Exploiting Confirmation Bias in LLM-Assisted Security Code Review.” arXiv:2603.18740, 2026. PREPRINT

30 Goddard, Roudsari & Wyatt. “Automation bias: a systematic review of frequency, effect mediators, and mitigators.” *JAMIA* 19(1), 2012. PEER-REVIEWED

31 Bursuc et al. “A benchmark for vericoding: formally verified program synthesis.” Dafny 2026 workshop @ POPL 2026; arXiv:2509.22908. PEER-REVIEWED · WKSHP

32 Cervantes, Kazman & Cai. “Improving LLM-assisted code generation through the use of architectural documents and implementation plans.” Designing 2026 @ ICSE 2026. PEER-REVIEWED · WKSHP

33 Zhang et al. “Large language models in model-driven engineering: a systematic mapping study.” *Empirical Software Engineering* 32(3), online July 2026. PEER-REVIEWED · JOURNAL

34 Xiao et al. “Self-Admitted GenAI Usage in Open-Source Software.” *IEEE Transactions on Software Engineering*, 2026. PEER-REVIEWED · JOURNAL

35 Agarwal et al. (CMU). “3100 Opinions on Code Review in an AI World.” arXiv:2607.07980, 2026. PREPRINT

36 DORA (Google). “The ROI of AI-assisted Software Development.” 2026. INDUSTRY

37 GitHub. “Spec-driven development with AI: Get started with a new open source toolkit.” September 2025. INDUSTRY

38 GitClear. “AI Copilot Code Quality: Evaluating 2024’s Increased Defect Rate.” 2025. INDUSTRY TELEMETRY

Marc Epstein is Managing Partner and Co-Founder of BrightMeld. This paper was researched and drafted with AI assistance; every citation was verified against its primary source by the author’s process, and the irony of that sentence is the point of the paper.