ChatGPT

Are AI Code Review Tools Better Than Human Reviewers

Executive summary

The short answer is **no, not overall**. In the strongest recent evidence, AI review tools are **better as a fast, scalable first-pass screening layer** and **worse as a full substitute for human reviewers**. The clearest direct comparison from the last six months shows that AI review comments are addressed far less often than human comments: in a large study of GitHub Actions–based review tools, **hunk-level AI comments were addressed at 6.5%–19.2%, file-level AI comments at 0.9%–4.2%, and human comments at 60.0%**. In a second large study of 278,790 review conversations, **AI suggestions were adopted 16.6% of the time versus 56.5% for human suggestions**, and human reviewers provided important **understanding** and **knowledge-transfer** feedback that AI agents largely did not. [1]

On defect-focused benchmarks, current automated reviewers still trail humans substantially. In the recent **c-CRAB** benchmark, the best automated reviewer achieved an **overall pass rate of 32.1%**, and the **union of four tools** reached **41.5%**, versus a **human upper bound of 100%** on a benchmark derived from human review concerns. In the recent **CR-Bench** prototype evaluation, agentic reviewers showed a classic precision/recall trade-off: the best reported setup reached **32.76% recall, 5.10% precision, and 8.83% F1**, while more conservative setups improved usefulness and signal-to-noise at the cost of recall. That is not "better than humans"; it is evidence that AI reviewers are useful but still noisy and incomplete. [2]

That said, AI review can still create real value. Atlassian's ICSE 2026 accepted study on **RovoDev Code Reviewer** reported that **38.70%** of generated comments triggered subsequent code changes, with a reported **30.8% reduction in PR cycle time** and **35.6% fewer human-written comments**. GitHub reports that Copilot code review usually returns a first pass in **under 30 seconds**, and its agentic redesign produced an **8.1% increase in positive feedback**; one later model change improved positive feedback by **6%** at the cost of **16%** more latency. These are encouraging productivity signals, but they are mostly **vendor or vendor-internal metrics**, not independent proof that AI now beats humans on code-review quality. [3]

The best current answer is therefore:

- **AI is better** for rapid first-pass coverage, repetitive or static-pattern defects, security scanning, review queue reduction, and low-cost parallel review. [4]
- **Humans are better** for business logic, architecture, product intent, ambiguous trade-offs, social coordination, mentoring, and final acceptance decisions. [5]
- **Hybrid review is best** for most teams: AI first pass, then human review focused on design, intent, and risk. GitHub explicitly recommends using Copilot code review to **supplement, not replace**, human review. [6]

Scope, assumptions, and prioritized sources

This synthesis uses only sources published or current within the last six months of **May 24, 2026**. I prioritized, in order: **official product docs/blogs**, **peer-reviewed or accepted academic papers**, **recent preprints that directly benchmark code review**, and a small amount of reputable tech media only where it added context rather than core evidence. The resulting evidence base is unusually strong on **workflow latency, adoption, and actionability**, but still weak on fully controlled, apples-to-apples comparisons of current commercial products against humans on the exact same pull requests. [7]

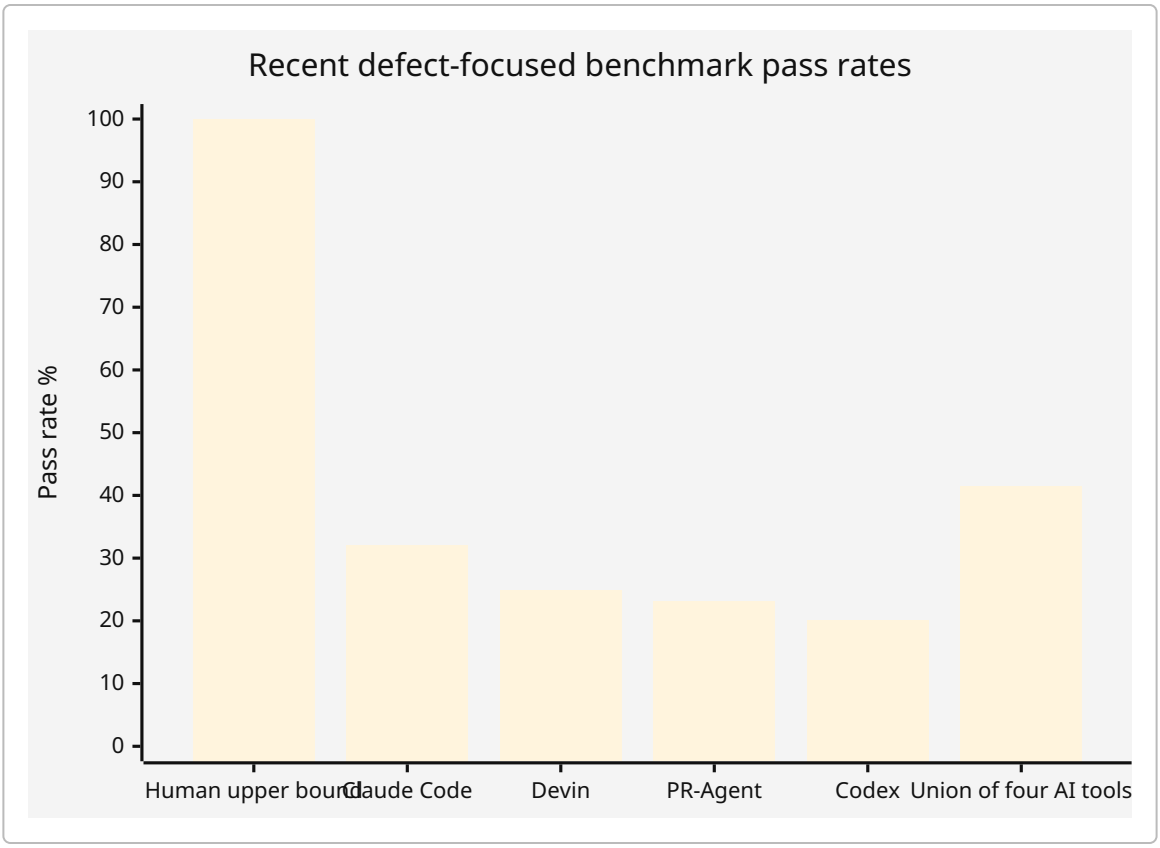
The report assumes a standard pull-request workflow in GitHub, GitLab, or a similar system; small-to-medium reviewable changes rather than massive monolithic PRs; mixed enterprise or OSS codebases; and no domain-specific regulation that legally requires human-only approval. Where your environment is **safety-critical** or **highly regulated**, the conclusions tilt even more strongly toward human final authority, because recent official docs from GitHub itself still warn about **missed defects, false positives, insecure suggestions, and biases** in AI review. [6]

The highest-priority recent sources were:

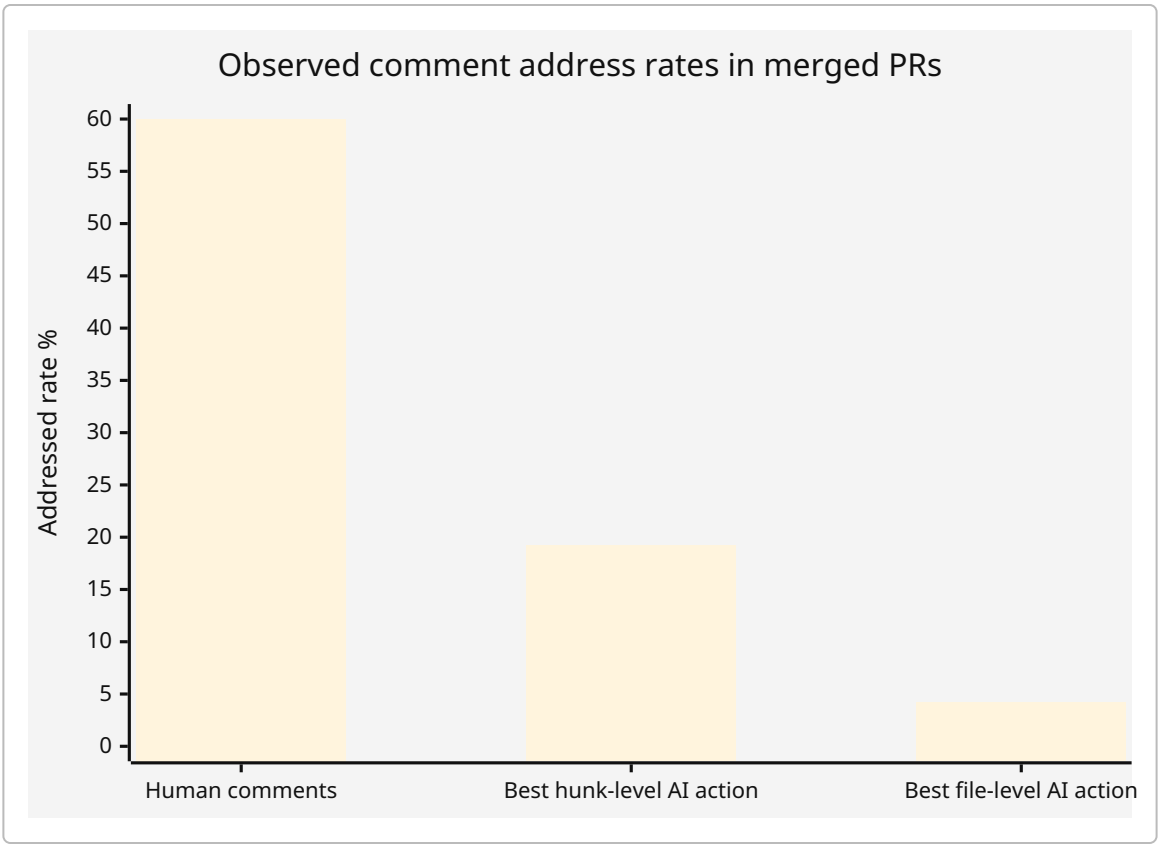
Source type	Most relevant sources	Why they matter
Accepted / peer-reviewed	**RovoDev Code Reviewer** accepted at **ICSE 2026 SEIP**; **Does AI Code Review Lead to Code Changes?** in **IEEE TSE 2026 early access**; **An evaluation study of LLMs for addressing code quality issues** in **Empirical Software Engineering** (April 2026) [8]	Strongest recent empirical evidence on actionability, workflow impact, and code-quality repair
Recent benchmark preprints	**c-CRAB** and **CR-Bench** (March–April 2026); **Human-AI Synergy in Agentic Code Review** (March 2026) [9]	Best recent quantitative evidence on detection coverage, precision/recall trade-offs, and human-vs-AI interaction patterns
Official vendor docs and changelogs	GitHub Copilot code review docs/blogs, GitLab Duo docs/blogs, CodeRabbit docs/changelog, Snyk docs/pages, AWS docs for CodeGuru Reviewer and Amazon Q Developer [10]	Best current sources for feature scope, latency, pricing model, privacy, deployment, and integration

What the quantitative evidence says

The most important methodological point is that **recent public evidence is not a single clean bake-off between GitHub Copilot Code Review, current CodeRabbit SaaS, GitLab Duo Code Review Flow, Snyk Code, CodeGuru Reviewer, and human reviewers on the same PR set**. Instead, the evidence comes from three kinds of studies: **human-vs-AI comment actionability studies**, **human-derived defect benchmarks**, and **vendor/internal workflow metrics**. Those three types point in the same direction: AI reviewers are useful, but they still do not match humans as full reviewers. [11]



On **c-CRAB**, which converts human review concerns into executable tests, current automated reviewers show a large gap from human coverage. **Claude Code reached 32.1% overall pass rate, Devin 24.8%, PR-Agent 23.1%, and Codex 20.1%; combining all four tools raised coverage to 41.5%**. The authors explicitly warn that the **100% human score is an upper bound**, because the benchmark itself encodes human-identified issues; this should be read as "AI still covers only a minority of the issues humans raised in this benchmark," not as a universal claim that humans are literally perfect. Even with that caveat, the gap is large. [12]



The most direct recent human-vs-AI actionability evidence comes from the IEEE TSE 2026 early-access study of 16 AI review actions on GitHub. In its annotated sample, **human comments were addressed 60.0% of the time**, while the **best hunk-level AI action reached 19.2%**, another hunk-level action reached **6.5%**, and the best file-level action reached only **4.2%**. Across the broader analysis, the authors conclude that **hunk-level review is substantially better than file-level review**, but still far behind humans. They also show that comments are more likely to be acted on when they are **concise, code-rich, and manually triggered** rather than automatically dumped on every PR. [13]

A second recent large-scale comparison, **Human-AI Synergy in Agentic Code Review**, reinforces that point. It found that **human suggestions were adopted 56.5% of the time versus 16.6% for AI suggestions**. Humans also provided **understanding** and **knowledge-transfer** feedback that AI agents largely did not, while AI agent comments were much more verbose at **29.6 tokens per line of code versus 4.1 for humans**. When humans reviewed AI-generated code, conversations became **11.8% longer**, and review threads ending with AI-agent responses had substantially higher rejection rates (**7.1%–25.8%**) than threads ending with human responses (**0.9%–7.8%**). That is a strong recent signal that AI can add review burden, especially when reviewing AI-written code. [14]

The recent **CR-Bench** prototype evaluation shows why teams often experience AI review as simultaneously useful and noisy. In that benchmark, a **single-shot GPT-5.2** review agent achieved **27.01% recall, 3.56% precision, 6.30% F1, 83.63% usefulness, and 5.11 signal-to-noise ratio**; a **Reflexion GPT-5.2** setup improved recall to **32.76%** and precision to **5.10%**, but usefulness dropped to **66.10%** and SNR to **1.95**. This is the core trade-off in current AI review: if you push hard for recall, comment noise tends to rise; if you suppress noise, you miss more issues. [15]

Where AI review **does** look better is speed and throughput. GitHub says Copilot code review usually returns feedback in **under 30 seconds** and now accounts for **more than one in five code reviews on GitHub**, after **10x usage growth** and **60 million code reviews** since launch. GitHub also reports that **71% of reviews surface actionable feedback** and that Copilot now averages **about 5.1 comments per review**; its agentic redesign produced an **8.1% increase in positive feedback**, and a later model upgrade delivered another **6% positive-feedback gain** at the cost of **16% more latency**. GitLab states that Duo Code Review has a **120-second AI Gateway timeout**, while Code Review Flow can run **hundreds of reviews in parallel** and costs **$0.25 per review**. CodeRabbit says it publishes its review within **minutes** of PR creation. Humans, by contrast, do not have recent public six-month latency benchmarks in this evidence set, but their throughput scales linearly with reviewer time rather than near-instantaneously across a fleet. [16]

The strongest recent productivity result is Atlassian's accepted ICSE 2026 study: **38.70%** of RovoDev comments triggered code changes, **PR cycle time fell 30.8%**, and **human-written comments fell 35.6%** over a one-year deployment study. This is substantial, but it is still a **single-company internal deployment** and therefore should be interpreted as strong but not universal evidence. [17]

By defect type, the recent evidence suggests a division of labor rather than one winner. AI tools are strongest on **defect detection, code improvement, security/compliance/pipeline checks, and concrete line- or hunk-level fixes**, while humans retain an advantage on **why**, not just **what**: architecture, product intent, ambiguity resolution, trade-offs, and review discussion that transfers knowledge. Recent docs and papers line up on this. GitLab's agentic review explicitly combines **repository context, pipeline, security findings, and compliance requirements**; CodeRabbit combines AI with **51 static analysis, linter, and security tools** and categorizes findings as **potential issues, refactor suggestions, and nitpicks**; Snyk is strongest not as a general reviewer but as a security-focused AI SAST system with **19+ languages**, **25M+ data flow cases**, and **80%-accurate security autofixes** in vendor reporting. Humans, meanwhile, uniquely contribute understanding and knowledge transfer. [18]

Recent official docs are also clear that AI review still has material failure modes. GitHub says Copilot code review may **miss code quality problems**, produce **false positives**, generate **inaccurate or insecure code**, and reflect **biases** from training data. Snyk's own docs likewise caution that generated fixes can still be **incorrect, syntactically invalid, or break the application**. Those caveats matter because they come from the product vendors themselves. [19]

Strongest recent evidence for and against AI review

Direction	Strongest evidence	What it means
Pro AI	GitHub says Copilot usually reviews in **<30 seconds**; CodeRabbit says reviews appear **within minutes**; GitLab Code Review has a **120-second timeout ceiling** and GitLab Code Review Flow can run **hundreds in parallel**. [20]	AI is clearly stronger on first-pass speed and parallel throughput.
Pro AI	Atlassian's accepted ICSE 2026 study reports **38.70%** of AI comments triggered code changes, **30.8%** lower PR cycle time, and **35.6%** fewer human-written comments. [17]	In some real deployments, AI meaningfully reduces review bottlenecks.

Direction	Strongest evidence	What it means
Pro AI	GitHub reports **71%** of Copilot reviews surface actionable feedback and **8.1%** more positive feedback after the agentic redesign. [21]	AI review quality is improving in production, at least by vendor-defined usefulness metrics.
Pro AI	Snyk reports **80%-accurate security autofixes**, **up to 50x faster** unsafe-code remediation, and **84%+ MTTR reduction** in its AI-security workflow pages. [22]	AI is especially strong in security-focused review/ remediation workflows, though these are vendor claims.
Pro human	In the GitHub Actions comparison, **human comments were addressed 60.0%**, versus **6.5%–19.2%** for hunk-level AI and **0.9%–4.2%** for file-level AI. [23]	Human comments remain far more likely to drive actual code change.
Pro human	In Human-AI Synergy, human suggestion adoption was **56.5%** vs **16.6%** for AI; humans also provided review types AI lacked. [24]	Humans remain better at context-rich, socially effective review.
Pro human	In c-CRAB, individual automated tools passed only **20.1%–32.1%** of benchmark tests; even all four combined reached **41.5%** versus the human upper bound. [12]	AI reviewers still miss a large fraction of human-identified concerns.
Mixed / caution	CR-Bench shows the best AI setups only at **27.01%–32.76% recall** and **3.56%–5.10% precision**, with usefulness/noise trade-offs. [15]	AI review quality is usable but still noisy; aggressive recall costs trust.
Mixed / caution	GitHub and Snyk both document false positives, misses, and potentially insecure or invalid generated fixes. [19]	Human verification remains necessary.

Comparison table of tools versus human reviewers

Reviewer / tool	Public recent quantitative evidence	Best-fit defect classes	Adaptation / learning	CI/CD and workflow fit	Privacy / code security posture	Cost / scale
GitHub Copilot Code Review	Usually **<30 sec** to first review; **60M** reviews total; usage **10x** since launch; now **>1 in 5** GitHub code reviews; **71%** of reviews surface actionable feedback; recent redesign **+8.1%** positive feedback; later model upgrade **+6%** positive feedback with **+16%** latency. [25]	General correctness, readability, maintainability, some security/ performance with custom instructions. [26]	Repository custom instructions; agentic context retrieval across repo and linked issues/PRs. [27]	Native GitHub PR workflow; can auto-review; runs on GitHub Actions; supports self-hosted runners. [28]	**Important variance by plan**: from April 24, 2026, Copilot Free/ Pro/Pro+ interaction data may be used for model training unless opted out; Copilot Business and Enterprise are not affected. GitHub warns about missed issues, false positives, insecure suggestions, and bias. [29]	Usage-based billing plus GitHub Actions minutes for private-repo code reviews from June 1, 2026; no flat public per-review price. [30]

Reviewer / tool	Public recent quantitative evidence	Best-fit defect classes	Adaptation / learning	CI/CD and workflow fit	Privacy / code security posture	Cost / scale
CodeRabbit	Current docs say reviews appear **within minutes**; direct metrics API exposes merged-PR review times and comment breakdowns; dashboard tracks human-vs-CodeRabbit comments and review-ready to last-human-review time. Recent independent head-to-head evidence on the current SaaS product was **not found**; the closest public independent evidence is the older `coderabbitai/ai-pr-reviewer` GitHub Action in the TSE study, where the best hunk-level action reached **19.2%** address rate. [31]	General PR review with bugs, logic flaws, refactors, security findings, style/nitpicks; strongest when paired with toolchain findings. [32]	Strong adaptation story: **Learnings**, code guidelines, path instructions, multi-repo analysis. [33]	Works on GitHub, GitLab, Azure DevOps, and Bitbucket; integrates with **51** static analysis/security tools. [34]	CodeRabbit says customer code is **never used for training**; code is shared with OpenAI/Anthropic for review only; data retention can be disabled; Enterprise offers self-hosting; SOC 2 Type II announced in Feb. 2026. [35]	Public pricing: **$24/user/month** annually for Pro, **$48/user/month** for Pro+; Enterprise custom. [36]

Reviewer / tool	Public recent quantitative evidence	Best-fit defect classes	Adaptation / learning	CI/CD and workflow fit	Privacy / code security posture	Cost / scale
GitLab Duo Code Review / Code Review Flow	Duo Code Review request timeout **120 sec**; Code Review Flow costs **$0.25 per review** and can run **hundreds of reviews in parallel**; GitLab positions it as repository-, pipeline-, security-, and compliance-aware. Public precision/recall numbers in the last six months were not found. [37]	Stronger than generic AI review for repos already living inside GitLab's DevSecOps stack: code, pipeline, security, compliance. [38]	Custom review instructions and model selection; progressive enhancement rather than mandatory replacement. [39]	Deepest native integration of the tools reviewed here with CI/CD/security/compliance, because those signals already live in GitLab. [38]	Very strong privacy posture: zero data retention with Anthropic, AWS, Fireworks AI, and Google; limited vendor-side retention for some OpenAI models; no model training by GitLab; supports full self-hosting or hybrid AI gateway/models. [40]	**$0.25/review** list price; predictable per-review economics are unusually transparent. [41]
Snyk Code / Snyk Agent Fix	Vendor says **19+ languages**, **25M+ data flow cases**, **80%-accurate** security autofixes, **up to 50x faster** unsafe-code remediation, and **84%+ MTTR reduction** in secure-AI-code workflow materials. [22]	**Security vulnerabilities and code-quality flaws**, not general peer review. Strong for secure coding and remediation, weaker as a substitute for business-logic review. [42]	Learns at the model/rule level rather than team-conversation memory; no recent public evidence of team-specific adaptive review comparable to CodeRabbit Learnings. [43]	Strong IDE, pull-request, and CI/CD integration. [44]	Strong privacy position: no training on customer code; proprietary/self-hosted models plus secure third-party LLM use; encrypted communications and customer data segregation. [45]	Public list pricing was not evident in the recent sources I reviewed. [46]

Reviewer / tool	Public recent quantitative evidence	Best-fit defect classes	Adaptation / learning	CI/CD and workflow fit	Privacy / code security posture	Cost / scale
Amazon CodeGuru Reviewer	**Legacy / sunset direction**: since **Nov. 7, 2025**, no new repository associations can be created; only existing associations remain. Supports Java/Python; account quota **5,000 reviews/ month**; free tier **90 days** up to **100k LOC**; AWS pricing example shows a **90k LOC** repo at **$10/ month** after the free tier. [47]	Java/Python defect detection with ML/program analysis; less relevant as a forward-looking comparison because AWS itself points users toward other services. [48]	Little recent evidence of modern adaptive review behavior.	Works with AWS CodeCommit, Bitbucket, GitHub, and GHES for existing associations. [49]	Standard AWS managed-service posture.	Predictable old pricing, but now strategically legacy. [50]
Human reviewers	In recent direct comparisons, humans outperform AI on downstream actionability: **60.0%** of human comments addressed in one study; **56.5%** human suggestion adoption vs **16.6%** for AI in another. Humans also provide understanding and knowledge-transfer comments that AI lacks. [1]	Business logic, architecture, product intent, trade-offs, cross-team coordination, mentorship, and nuanced approval/ rejection decisions. [5]	Continuous real-world learning from product, org, and domain context.	Universally compatible, but bottlenecked by reviewer availability and attention.	Best privacy posture when review stays within trusted staff, but subject to normal insider-access controls.	Expensive and unscalable relative to automated review; GitLab's own pricing blog uses a rough example of **$25** for 15 minutes of senior engineer time versus **$0.25** for one automated review. [41]

Recommended evaluation methodology for your own team

If you want a rigorous answer for your own codebase, the best design is a **randomized PR-level experiment** rather than a retrospective opinion survey. That recommendation follows directly from what the recent literature and product telemetry emphasize: comment actionability, resolved-before-merge behavior, review latency, downsides from noise, and downstream code quality. Recent research measures whether comments are addressed and how often suggestions are adopted; GitHub tracks comment reactions and whether flagged issues are resolved before merge; CodeRabbit exposes review times and comment breakdowns via API; and Atlassian's strongest internal result was a PR-cycle-time reduction. [51]

A defensible experiment looks like this:

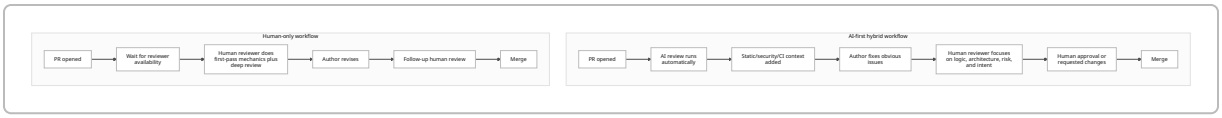
Design element	Recommendation	Why
Unit of randomization	**Pull request**	Avoids contamination from team-level assignment and maps cleanly to review workflow.
Arms	**Human-only**, **AI-first hybrid**, and optionally **AI-heavy** only on low-risk PRs	The real decision most teams face is not "AI or humans," but which hybrid.
Stratification	Stratify by **repo**, **language**, **PR size**, **risk tier**, **author experience**, **AI-generated vs human-authored code**, and **time zone / reviewer geography**	Recent evidence shows actionability depends on review granularity, comment form, contributor experience, and code-generation context. [1]
Blinded adjudication	For a sampled subset, use two senior reviewers (and a security reviewer where relevant) to classify comments as **true issue / false positive / subjective / duplicate**, blinded to whether the comment came from AI or a human	This is the cleanest way to estimate precision and avoid brand bias.
Seeded-defect set	Add a controlled subset of PRs with known injected defects or known issue-bearing training tasks	Recent benchmarks like c-CRAB and CR-Bench show why executable or verifiable defect tasks are more informative than text-similarity metrics. [52]

Design element	Recommendation	Why
Primary metrics	**Time to first review**, **time to merge / PR cycle time**, **comment address rate**, **accepted suggestions**, **precision**, **recall on seeded defects**, **post-merge bug/revert rate**, **security escape rate**, **human review minutes per PR**, **developer helpfulness rating**, and **cost per merged PR**	These cover both “faster” and “better,” which frequently diverge.
Secondary metrics	**Comments per PR**, **noise rate**, **review abandonment**, **coverage of PRs reviewed**, **privacy exceptions**, **runner / token / API cost**, **reviewer satisfaction**, **author satisfaction**	AI often wins on coverage while losing on noise.
Minimum run length	Long enough to include at least one full sprint or release cycle per stratum	Needed to capture downstream defects and merge behavior.

A practical rollout threshold should be conservative. I would recommend adopting an AI review tool broadly only if, against your human-only control, the **AI-first hybrid arm** delivers all of the following:

1. A **material reduction in time to first feedback**.
2. At least a **modest reduction in PR cycle time** or no regression.
3. **No statistically meaningful increase** in post-merge defects, reverts, or escaped security findings.
4. **Acceptable precision** on adjudicated comments for your team’s tolerance of noise.
5. Clear evidence that human reviewer time is being redirected toward higher-value concerns rather than spent triaging AI noise.

For instrumentation, use each platform’s native telemetry where available: GitHub’s feedback and resolved-before-merge signals, CodeRabbit’s metrics API for merged-PR review times and comment breakdowns, and GitLab’s usage and credit dashboards. [53]



Conclusions and limitations

The evidence from the last six months does **not** support the claim that AI code review tools are broadly better than human reviewers. It supports a narrower and more useful claim: **AI review is better than humans at immediacy, coverage, parallelism, and low-cost first-pass screening; humans are better at high-context judgment, trust, mentoring, and final decisions.** [54]

In practical terms, AI review is most likely to outperform humans when the task is: small-to-medium PRs, repetitive patterns, low-to-moderate risk, line- or hunk-localized defects, style and maintainability checks, security scanning, CI/CD-aware checks, or “review everything quickly” coverage. GitHub, GitLab, CodeRabbit,

and Snyk all now support variants of this model, with GitLab notably strong on integrated DevSecOps context and privacy, Snyk notably strong on security-specific review/remediation, and GitHub notably strong on native scale and first-pass speed. [55]

Humans remain clearly better when the task is: evaluating business logic, architecture, nuanced trade-offs, ambiguous requirements, domain-specific correctness, inter-team coordination, or mentoring the author. Humans also remain much more effective at producing feedback that actually changes code in recent public comparisons. That is especially important in the age of AI-generated code, where recent evidence suggests human reviews of AI-authored changes become longer and more burdensome, not less. [1]

The best current operating model is therefore **hybrid**: run AI review automatically on every eligible PR, suppress low-value comments aggressively, prefer hunk-level suggestions with code snippets, route security and repetitive quality checks to AI, and require human review for final acceptance on anything non-trivial or high-risk. That conclusion is directly consistent with GitHub's own responsible-use guidance, the benchmark literature, and the strongest recent industrial study. [56]

Open questions remain. Public evidence in the last six months is still too thin to rank **current** commercial products head-to-head on the same PRs with the same adjudication process. Vendor metrics often measure **helpfulness, reaction, or actionability**, not true ground-truth defect capture. And some named products are not directly comparable: **Snyk Code** is primarily a security-focused AI SAST workflow, while **Amazon CodeGuru Reviewer** is effectively legacy for new adopters because new repository associations were discontinued after **November 7, 2025**. Those limitations do not change the conclusion, but they do mean every serious team should validate tools on its own repositories before declaring AI review "better." [57]

---

[1] [11] [13] [23] [51] https://arxiv.org/html/2508.18771v2
https://arxiv.org/html/2508.18771v2

[2] [9] [12] [52] https://arxiv.org/html/2603.23448
https://arxiv.org/html/2603.23448

[3] [17] https://arxiv.org/abs/2601.01129
https://arxiv.org/abs/2601.01129

[4] [7] [10] [16] [20] [25] [54] https://docs.github.com/copilot/using-github-copilot/code-review/using-copilot-code-review
https://docs.github.com/copilot/using-github-copilot/code-review/using-copilot-code-review

[5] [14] [24] https://arxiv.org/html/2603.15911v1
https://arxiv.org/html/2603.15911v1

[6] [19] [26] [56] https://docs.github.com/en/enterprise-cloud%40latest/copilot/responsible-use/code-review
https://docs.github.com/en/enterprise-cloud%40latest/copilot/responsible-use/code-review

[8] https://conf.researchr.org/details/icse-2026/icse-2026-software-engineering-in-practice/2/RovoDev-Code-Reviewer-A-Large-Scale-Online-Evaluation-of-LLM-based-Code-Review-Autom
https://conf.researchr.org/details/icse-2026/icse-2026-software-engineering-in-practice/2/RovoDev-Code-Reviewer-A-Large-Scale-Online-Evaluation-of-LLM-based-Code-Review-Autom

[15] https://arxiv.org/html/2603.11078v1
https://arxiv.org/html/2603.11078v1

18 38 41 https://about.gitlab.com/blog/agentic-code-reviews-with-flat-rate-pricing/
https://about.gitlab.com/blog/agentic-code-reviews-with-flat-rate-pricing/

21 53 https://github.blog/ai-and-ml/github-copilot/60-million-copilot-code-reviews-and-counting/
https://github.blog/ai-and-ml/github-copilot/60-million-copilot-code-reviews-and-counting/

22 https://snyk.io/platform/deepcode-ai/
https://snyk.io/platform/deepcode-ai/

27 https://docs.github.com/copilot/customizing-copilot/adding-custom-instructions-for-github-copilot
https://docs.github.com/copilot/customizing-copilot/adding-custom-instructions-for-github-copilot

28 https://docs.github.com/en/copilot/how-tos/copilot-on-github/set-up-copilot/configure-automatic-review
https://docs.github.com/en/copilot/how-tos/copilot-on-github/set-up-copilot/configure-automatic-review

29 https://github.blog/changelog/2026-03-25-updates-to-our-privacy-statement-and-terms-of-service-how-we-use-your-data/
https://github.blog/changelog/2026-03-25-updates-to-our-privacy-statement-and-terms-of-service-how-we-use-your-data/

30 https://github.blog/changelog/2026-04-27-github-copilot-code-review-will-start-consuming-github-actions-minutes-on-june-1-2026/
https://github.blog/changelog/2026-04-27-github-copilot-code-review-will-start-consuming-github-actions-minutes-on-june-1-2026/

31 32 https://docs.coderabbit.ai/guides/code-review-overview
https://docs.coderabbit.ai/guides/code-review-overview

33 https://docs.coderabbit.ai/knowledge-base/learnings
https://docs.coderabbit.ai/knowledge-base/learnings

34 https://docs.coderabbit.ai/
https://docs.coderabbit.ai/

35 https://docs.coderabbit.ai/faq
https://docs.coderabbit.ai/faq

36 https://docs.coderabbit.ai/management/plans
https://docs.coderabbit.ai/management/plans

37 39 https://docs.gitlab.com/user/gitlab_duo/code_review/
https://docs.gitlab.com/user/gitlab_duo/code_review/

40 https://docs.gitlab.com/user/gitlab_duo/data_usage/
https://docs.gitlab.com/user/gitlab_duo/data_usage/

42 44 https://docs.snyk.io/scan-with-snyk/snyk-code
https://docs.snyk.io/scan-with-snyk/snyk-code

43 45 https://docs.snyk.io/snyk-data-and-governance/how-snyk-incorporates-generative-ai-into-the-platform
https://docs.snyk.io/snyk-data-and-governance/how-snyk-incorporates-generative-ai-into-the-platform

46 57 https://snyk.io/product/snyk-code/
https://snyk.io/product/snyk-code/

47 https://docs.aws.amazon.com/codeguru/latest/reviewer-ug/codeguru-reviewer-availability-change.html
https://docs.aws.amazon.com/codeguru/latest/reviewer-ug/codeguru-reviewer-availability-change.html

48 https://docs.aws.amazon.com/codeguru/latest/reviewer-ug/welcome.html
https://docs.aws.amazon.com/codeguru/latest/reviewer-ug/welcome.html

49 https://docs.aws.amazon.com/codeguru/latest/reviewer-ug/get-set-up-setup-repository.html
https://docs.aws.amazon.com/codeguru/latest/reviewer-ug/get-set-up-setup-repository.html

50 https://aws.amazon.com/codeguru/profiler/pricing/
https://aws.amazon.com/codeguru/profiler/pricing/

55 https://github.blog/changelog/2026-03-05-copilot-code-review-now-runs-on-an-agentic-architecture/
https://github.blog/changelog/2026-03-05-copilot-code-review-now-runs-on-an-agentic-architecture/