

AI-to-AI Code Reviews of GitHub Pull Requests

Niruthiha Selvanayagam 

École de technologie supérieure (ÉTS Montréal), Montréal, Canada

Taher A. Ghaleb 

Department of Computer Science, Trent University, Peterborough, Canada

Abstract

AI coding agents are increasingly integrated into software development workflows, operating on both sides of the pull request (PR) process: *AI authoring agents*, which create or modify PRs, and *AI reviewers*, which evaluate them. This creates a closed loop where one AI coding agent reviews contributions of another AI coding agent. In this paper, we construct an AI-to-AI code review dataset by linking AI-authored pull requests with AI-attributed review events from CODAGE, a public dataset of coding agent-generated GitHub events. Our dataset contains 248,641 PRs across 12 AI coding agents with at least one AI review, including 45,269 reviewed by a different AI product and 208,145 by the same product. We observe that cross-product AI-to-AI code review occurs in only about 1.6% of identified agent-authored PRs but is substantial in absolute terms: 45k PRs written by one identifiable AI product and reviewed by another. This activity grows by more than two orders of magnitude from 2025-Q1 to 2025-Q3. We measure reviewer behavior using CodeRabbit comment categories, per-PR comment volume, and time to first review, and find that it varies across author-reviewer pairs. For example, Claude-Code PRs receive more **refactor** comments from CodeRabbit than Copilot PRs (35.0% vs. 10.5%), a difference that may stem from PRs themselves rather than the reviewer. For three of four dual-role reviewers, mean comments per PR were 58–65% higher in the same-product group, though effect sizes were small or negligible and the difference was concentrated in the upper tail. Median time from PR creation to first AI review was 1.2 minutes for cross-product pairs and 4.7 minutes for same-product pairs, reflecting which reviewer bots are in each group rather than the product pairing. Overall, our large-scale characterization shows that closed-loop AI-to-AI code review is on the rise but remains a minority phenomenon, with review output varying across authoring-agent groups and author-reviewer configurations.

2012 ACM Subject Classification Software and its engineering → Empirical software validation; Software and its engineering → Software maintenance tools

Keywords and phrases AI coding agents, AI code review, closed-loop AI, Pull requests, Mining software repositories, GitHub events

Digital Object Identifier 10.4230/LIPIcs.ESEM.2026.75

Category Emerging Results, Vision & Reflection Track Paper

Supplementary Material

Dataset (Replication Package): <https://github.com/Niruthiha/AI-AI-CodeReviews>

Funding This work is partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC): RGPIN-2025-05897.

1 Introduction

Two types of AI coding agents now commonly appear in GitHub pull-request workflows. Some agents author code: systems such as Devin, OpenAI Codex, Copilot, Cursor, and Claude Code can open pull requests (PRs) on behalf of users, either autonomously or as assistive tools [12]. Others review code: tools such as CodeRabbit, Sourcery, and PR-Agent post line-level comments and review decisions. In addition, several authoring agents, including Copilot, Codex, Gemini Code Assist, and Amazon Q, also appear as reviewers in public GitHub data.



© Niruthiha Selvanayagam and Taher A. Ghaleb;
licensed under Creative Commons License CC-BY 4.0

20th International Symposium on Empirical Software Engineering and Measurement (ESEM 2026).

Editors: Robert Feldt, Maria Paasivaara, Daniel Mendez, Stefan Wagner, and Marvin Muñoz Barón;

Article No. 75; pp. 75:1–75:15



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Together, these roles form what we call *closed-loop AI review*: an AI coding agent contributes to a GitHub repository, and one or more AI coding agents review it. We use “closed-loop” in this observable sense only. Our review stream is restricted to AI-attributed review events, so the term means that AI occupies both sides of the pull request, not that humans were absent: a PR in our datasets may also have received human review that we do not observe. This shift raises two research questions (RQs):

■ **RQ1: How prevalent is closed-loop AI review on GitHub?**

This RQ characterizes the prevalence and growth of closed-loop review. As AI-generated reviews become more common, an increasing share of public code may be evaluated without direct human review, affecting both downstream maintenance and studies that rely on GitHub review data as evidence of human decision making.

■ **RQ2: How does reviewer behavior vary with the author–reviewer setup?**

This RQ examines whether the authoring agent influences reviewer behavior. If reviewer behavior varies across authoring agents, then closed-loop review cannot be treated as a single process. Instead, both maintainers and researchers need to account for which agent wrote the code and which agent reviewed it.

Prior work has primarily studied AI coding tools and AI review systems in isolation, including productivity effects of code generation tools [16] and usability of AI-assisted suggestions [23]. Earlier studies of reviewer bots [26] were conducted before the current ecosystem of dual-role AI coding agents and do not analyze author–reviewer pairings. This paper addresses these questions by introducing a large-scale dataset and empirical study of AI-to-AI code reviews of GitHub PRs. We identify AI-authored and AI-reviewed PRs by applying our attribution method to CodAGE (CodAGE) events [7] from 2024–2026, enabling measurement of closed-loop interactions at scale. We find this phenomenon already substantial in absolute terms, if still a minority of all agent activity: in 2025, tens of thousands of PRs authored by AI coding agents are reviewed by other AI coding agents across multiple products.

Contributions. Overall, this paper makes the following contributions.

1. We introduce a large-scale dataset of AI-to-AI code reviews, covering 248,641 PRs that received at least one AI review: 45,269 cross-product and 208,145 same-product.
2. We characterize closed-loop AI code review patterns over time, tracking how PRs are reviewed by the same or different AI product.
3. We empirically analyze AI reviewer behavior under different author–reviewer setups, examining how reviewer output varies with authoring agent and setup using CodeRabbit comment-category labels, per-pull-request comment volume, and time to first review.

Paper Organization. The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 describes our data and methodology. Sections 4 and 5 present our findings on closed-loop AI review prevalence and reviewer behavior. Section 6 discusses implications. Section 7 addresses threats to validity. Section 8 concludes the paper and outlines future directions.

2 Background and Related Work

AI-authored pull requests on GitHub. Prior work on AI code generation has largely focused on code-suggestion tools used by human developers, including studies of Copilot productivity [16], usability of LLM-based suggestions [23], and AI-assisted pair programming [10].

More recent autonomous or semi-autonomous agents open PRs with limited or no line-by-line human authorship [12]. Prior work characterizes these agent-authored PRs directly: in one study of Claude Code, 45.1 % of merged PRs required additional changes [24]; non-merged agent-authored PRs tend to involve larger changes, touch more files, and fail CI more often [6]; and merge outcomes turn on reviewer intervention and repository context more than on agent identity alone [18]. This work centers on whether agent-authored code is accepted; it says little about how AI reviewers behave when they are the ones doing the evaluating.

Reviewer-side bots. Wessel et al. [26] characterized the adoption of bots on GitHub and showed that bot introduction can affect human review volume and latency. Brown and Parnin [2] studied developer interaction with bots in code review workflows, and recent surveys chart how generative AI is now reshaping modern code review more broadly [27]. LLM-based review has since moved into practice, with systems evaluated or deployed at industrial scale [13, 4, 19, 22] and multi-agent review pipelines developed in research [20, 17], alongside a parallel line of work asking whether their output is trustworthy: developers accept only a minority of CodeRabbit suggestions [14], current systems remain weak at realistic review generation [28, 32], and review comments can be ungrounded in the code they describe [21]. A closer group of recent studies looks at reviewer bots specifically on agent-authored PRs [31, 30], reporting that AI suggestions are adopted far less often than human ones and that agent involvement can speed decisions without reliably improving review quality. This work establishes that AI now reviews AI-authored code; what remains open, and what we take up, is which agents review which others across the ecosystem, and whether observable reviewer behavior differs between same-product and cross-product configurations.

Bot/Agent attribution. Prior work identifies automated accounts using commit- and account-level heuristics, such as BIMAN [5], or comment-level classifiers, such as BoDeGHa [9]. Ghaleb et al. [8] recently used machine learning models to detect AI coding agents using behavioral fingerprints. While these studies distinguish automated contributions or infer the originating agent, they do not identify the corresponding AI product or vendor, which we address this gap using signature-based attribution of AI coding agents.

Multi-agent LLM evaluation and this work. Outside software engineering, “LLM-as-a-judge” [29] and constitutional AI [1] study settings in which models evaluate, critique, or revise model-generated outputs. However, the corresponding deployed software setting, AI reviewers commenting on AI-authored PRs in public repositories, is far less studied. To our knowledge, prior work has not characterized product-resolved AI-author–AI-reviewer pairings at this scale across GitHub. We address this gap by quantifying how often such closed-loop interactions occur, which agents participate, and how reviewer behavior varies with the authoring agent and the author–reviewer configuration.

3 Data and Methodology

This section describes the raw GitHub events data, the attribution of events to AI coding agents, the analysis dataset, and the CodeRabbit comment-category classifier.

3.1 Data source

Our data source is the CODAGE dataset (Coding Agent-generated GitHub Events) [7] collected from GHArchive.¹ We use the snapshot covering events from 2024-01-01 to 2026-04-15. CODAGE contains several event-type subsets. In this paper, we focus on CodAGE-PRs, CodAGE-Reviews, and CodAGE-ReviewComments from this period. These subsets capture PR authorship signatures and reviewer-side activity. Despite covering the relevant period, many autonomous AI coding agents emerged in early 2025. Candidate events were identified by a broad attribution pass over CodAGE-PRs, which includes branch-name evidence and yields 4,563,819 candidate PRs with a provisional agent label. The signature framework described in Section 3.2 is a stricter second pass over that pool: it requires body or vendor-login evidence and discards branch-only matches.

3.2 Identifying AI authorship and AI review

Attributing PRs to AI coding agents by actor login alone is unreliable: some use generic [bot] accounts, some use the human user’s login with a machine-readable body signature, and some quote other agents’ text, which can mislead body-only rules. We therefore use a two-tier signature framework. **S1 (high-confidence body signature)** is a regex or fixed substring the agent produces in the PR or comment body, such as the `Co-Authored-By: Claude <noreply@anthropic.com>` trailer from Claude Code, the `cursor.com/agents` URL from Cursor, or the `chatgpt.com/codex` URL from OpenAI Codex. **S2 (vendor-controlled account signature)** is an exact match or regex over a login unambiguously controlled by a vendor, such as `coderabbitai[bot]`, `devin-ai-integration[bot]`, or `gemini-code-assist[bot]`. S1 agents require body-level evidence; S2 agents accept body or login evidence. Branch-name prefixes (e.g. `codex/`, `cursor/`) are recorded but never sufficient alone, being user-controllable and prone to false positives.

We create one entry per matched (event, agent) pair. A few events match multiple agents, usually when one agent quotes another’s body trailer: 98 of 4,141,107 review entries (0.002%) and 111 of 8,560,237 review-comment entries (0.001%). We retain all matches, report the duplicate rate in the coverage report, and deduplicate by event ID when event-grain analysis requires it. This attributes *review-side* events to 12 AI coding agents (CodeRabbit, Copilot, Gemini Code Assist, OpenAI Codex, Amazon Q, Devin, Claude Code, Sweep AI, PR-Agent, Kiro, Cursor, Aider). The *author-side* stream yields 2,830,284 unique agent-authored PRs and adds Google Jules, which authors PRs but has no reviewer activity.

3.3 Cleaning, deduplication, and quarantine

We apply three cleaning steps.

(i) Quarantine. Every candidate PR or review event is scored against each agent’s signature set, and an entry is emitted only where the evidence meets that agent’s *inclusion floor*: agents with an S1 floor require a body signature, whereas agents with an S2 floor accept either a body signature or a vendor-controlled login. A vendor-controlled login is therefore sufficient on its own for S2-floor agents. Any candidate meeting no agent’s floor is quarantined, written to a separate file with a recorded reason. On the author side the reasons are `branch_only`, where only a branch-name prefix matched, and `no_evidence`; on the review side, they are

¹ <https://www.gharchive.org>

`empty_body` and `no_evidence`. Quarantine is negligible on the two review streams (590 of 4,141,598 review events and 793 of 8,560,919 review comments, 0.01 % of each) but substantial on the author stream, where 1,733,535 of 4,563,819 candidate PRs (38.0 %) are quarantined, 96.9 % of them as `branch_only`. This asymmetry underlies the coverage imbalance discussed in Section 7: our agent-authored population is the 62.0 % of candidates with a body or login signature, not all PRs a weaker rule would include.

(ii) Relabeling. We retain the provisional label from the first pass as `original_label` and report the two passes as a confusion matrix. Where they disagree, the signature attribution wins; the largest relabel consolidates `OpenAI_Codex_Cloud` and the CLI Codex trailer into `OpenAI_Codex`. Because both passes are ours, this is an internal consistency check rather than external validation.

(iii) Thin agents. Agents with fewer than 50 attributed PRs (Aider, Kiro, SWE-agent, Windsurf) are excluded from our per-agent quantitative claims.

3.4 Dataset construction

The unit of analysis is the *PR*, keyed by (`repo_name`, `pr_number`). We aggregate per-event entries into one entry per (PR, Reviewer) pair, retaining review and review-comment counts and first/last review timestamps, then join with the agent-author PR set to yield three datasets (A: AI author; B: AI reviewer; N: no *detected* AI reviewer): **A×B-cross-product**, the closed-loop cross-product dataset with author and reviewer from different products (45,269 PRs, 47,259 pairs); **A×B-same-product**, same product, e.g. Copilot reviewing Copilot (208,145 PRs); and **A×N**, agent-authored PRs with no *detected* AI review (2,581,643 PRs, for context only); absence of an attributed reviewer signature is not evidence that no AI reviewed the PR. A×B PRs receiving both same- and cross-product review appear in both A×B datasets with an overlap flag, so the dataset counts are not mutually exclusive and do not sum to the unique agent-authored total. We do *not* analyze a matched human-authored control dataset.

These two closed-loop datasets are defined at the level of the identifiable agentic *product*, or agentic harness, rather than the parent company or the underlying foundation model: **A×B-same-product** contains PRs in which the same identifiable product appears as author and reviewer, and **A×B-cross-product** those in which the authoring and reviewing products differ. Products operated by the same corporate vendor, such as Google Jules and Gemini Code Assist, are therefore treated as distinct, because they are separately identifiable agentic systems with different product names and potentially different harnesses, configurations, and review workflows; 117 of the 45,269 A×B-cross-product PRs are of this kind. These categories consequently do not necessarily correspond to corporate vendor or foundation-model boundaries.

3.5 Severity classifier and released artifacts

CodeRabbit comment categories. To measure *review content* for RQ2, we use CodeRabbit’s self-declared comment categories. CodeRabbit prefixes substantive comments with emoji-tagged headers: **Refactor suggestion**, **Potential issue**, and **Nitpick**, plus a fourth header, **Verification**, which appears in our historical data (9,525 comments) and which we treat as an observed comment prefix rather than a documented category. Because these headers appear explicitly in the comment text, we extract them deterministically using rule-based

matching without an LLM judge. We retain *severity* in artifact and column names for consistency with the released dataset, but do not interpret the categories as an ordered severity scale. In our dataset, CodeRabbit is the only dedicated reviewer-only bot with both substantial volume and machine-parsable category headers (Sourcery and PR-Agent together account for fewer than 250 review events).

Classifier. An ordered-regex pipeline strips boilerplate that would otherwise produce false matches, then applies the rules in order to the cleaned comment body; the first rule wins, and unmatched comments are **unlabeled**. It emits four substantive categories, **potential_issue** (suspected bug), **refactor** (working code whose structure could improve), **verification** (a request to verify a behavior or assumption), and **nitpick** (minor style), on which we impose no ordering, and three structural labels, **analysis_chain** (script-execution diagnostic), **conversational** (@-mention reply), and **rate_limited** (an operational rate-limit notice). The rule set and its ordering are provided in the replication package.

Replication package. Our replication package (data processing/analysis scripts, filtered data, signature registry, and detailed results) is available on GitHub.² The main source data used in this work is CODAGE (Coding Agent-generated GitHub Events), hosted on Hugging Face,³ specifically CodAGE-PRs, CodAGE-Reviews, and CodAGE-ReviewComments.

4 RQ1: Prevalence and Composition of Closed-Loop AI-to-AI Code Reviews

We first establish the scale and growth of closed-loop review, then decompose it by author-reviewer pair and by same- versus cross-product composition.

4.1 Volume and growth

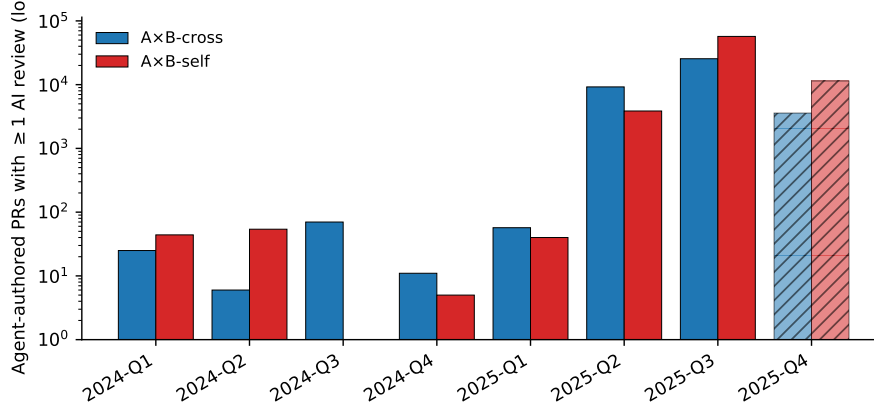
Of the 2,830,284 agent-authored PRs we identified, 248,641 (8.8%) received at least one AI review: 45,269 cross-product and 208,145 same-product, with the counts read as lower bounds and the percentage as conditioned on signature attribution (Section 7). We focus on cross-product review because same-product review may be part of an integrated product workflow. Activity was negligible through 2025-Q1 (57 cross, 40 same) but increased by over two orders of magnitude by 2025-Q3, reaching 25,492 cross- and 57,080 same-product PRs (Figure 1). Because 2025-Q4 counts are incomplete due to GHArchive attribute lag, temporal comparisons use Q2 and Q3.

4.2 Who reviews whom?

Table 1 reports the $A \times B$ -cross-product author-reviewer crosstab ($N \geq 500$). OpenAI Codex dominates as author (31,601 cross-product PRs, 69.8% of the 45,269). Its row in Table 1 sums to 32,379 rather than 31,601 because those cells count author-reviewer pairs, not unique PRs: a PR reviewed by agents from two different vendors counts once in Table 2 but contributes a pair to each reviewer column here (and the crosstab shows only cells with $N \geq 500$). Copilot dominates as reviewer (21,022 of the 47,259 author-reviewer pairs; modal pair *Codex authored, Copilot reviewed*, 18,114 pairs), and CodeRabbit, the only dedicated reviewer-only bot with usable volume, reviews PRs from at least six authoring agents.

² <https://github.com/Niruthiha/AI-AI-CodeReviews>

³ <https://huggingface.co/datasets/taher-ghaleb/CodAGE>



■ **Figure 1** Quarterly counts of agent-authored PRs receiving AI review, split into cross-product ($A \times B$ -cross-product) and same-product ($A \times B$ -same-product) groups. Both increased by over two orders of magnitude from 2025-Q1 to 2025-Q3. Hatched 2025-Q4 values are lower bounds because GHArchive attributes lag event data by about one quarter. The y-axis is log-scaled.

■ **Table 1** Closed-loop, cross-product author-reviewer pairs ($A \times B$ -cross-product), cells with $N \geq 500$. Cells report author-reviewer pairs. Because one PR may receive reviews from multiple agents, row totals may exceed the corresponding number of unique PRs.

Author → Reviewer	Copilot	Codex	Gemini	CodeRabbit	Total (≥ 500)
OpenAI Codex	18,114	–	8,995	5,270	32,379
Copilot	–	4,772	956	1,627	7,355
Cursor	1,000	–	593	516	2,109
Claude Code	956	–	501	716	2,173
Devin	656	–	–	–	656

4.3 Same- vs. cross-product composition

Table 2 reports the same- vs. cross-product split per authoring agent (≥ 200 $A \times B$ PRs). The pattern is sharply non-uniform, with three groups: *product-internal* authors reviewed mostly by their own product (Copilot 95.7%, Amazon Q 91.8%, Devin 76.5% same-product); *balanced* (OpenAI Codex at 54.1% self, the only author with substantial population in both datasets); and *cross-product* authors reviewed almost entirely by a different product (Cursor, Google Jules, Claude Code). This asymmetry motivates RQ2, in which we (i) hold the reviewer constant (CodeRabbit) and vary the author, (ii) contrast same- vs. cross-product review volume, and (iii) contrast time-to-first-review across reviewer bots and product pairings.

RQ1 Summary: AI review remains a minority of agent activity but is already substantial: 248,641 of the agent-authored PRs we identified (8.8%) received at least one AI review, including 45,269 (1.6%) across products, spanning 10,345 repositories. The counts are lower bounds; the percentages are conditioned on signature attribution. Activity grew by over two orders of magnitude in 2025. Copilot-authored PRs were reviewed mainly by Copilot, whereas Cursor-, Claude Code-, and Google Jules-authored PRs were reviewed almost entirely across products.

■ **Table 2** Same- vs. cross-product reviewer share, per authoring agent ($A \times B$ -cross-product + $A \times B$ -same-product, ≥ 200 PRs). Sorted by Total $A \times B$ descendingly.

Authoring agent	Cross-product PRs	Same-product PRs	Total $A \times B$	% Same-product
Copilot	7,515	166,442	173,957	95.7 %
OpenAI Codex	31,601	37,247	68,848	54.1 %
Devin	1,171	3,811	4,982	76.5 %
Cursor	2,309	0	2,309	0.0 %
Claude Code	2,042	10	2,052	0.5 %
Amazon Q	48	535	583	91.8 %
Google Jules	478	0	478	0.0 %
Sweep AI	100	100	200	50.0 %

5 RQ2: Reviewer Behavior Across Authors and Vendor Pairings

We characterize reviewer behavior using three measures. First, we examine CodeRabbit’s *comment-category mix* while holding the reviewer fixed and varying the authoring agent. Because these are self-declared labels rather than validated defects, we interpret them as reviewer output composition, not code quality [14, 21, 11]. Second, we compare per-PR comment *volume* between same- and cross-product pairs for Copilot, OpenAI Codex, Devin, and Amazon Q. Third, we measure *latency* from PR creation to the first AI review and examine whether aggregate differences reflect product pairing or reviewer composition.

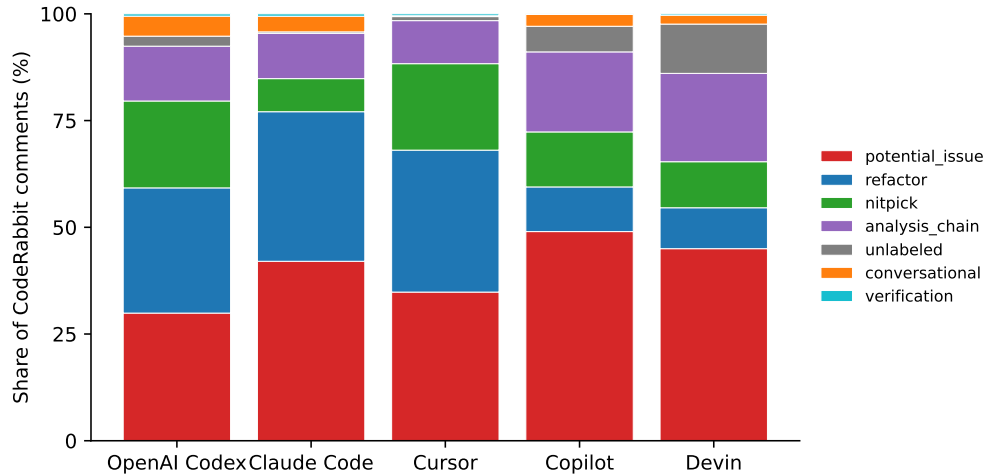
5.1 CodeRabbit comment categories by authoring agent

We join the CodeRabbit classifier output against the union of the $A \times B$ -cross-product and $A \times B$ -same-product datasets on (`repo_name`, `pr_number`), restricting to the five authoring agents with ≥ 200 classified comments (35,248 comments total; Figure 2 and Table 3). The comment-category distribution is visibly non-uniform: Copilot- and Devin-authored PRs receive far more `potential_issue` comments (49.0 %, 43.9 %) and far fewer `refactor` comments (10.5 %, 9.7 %) than Claude Code, Cursor, and OpenAI Codex (29.2 % to 35.0 % `refactor`), and the highest `unlabeled` shares (6.1 %, 12.3 %). Claude Code sits at the opposite end, with the highest `refactor` (35.0 %) and lowest `unlabeled` (0.3 %).

■ **Table 3** Normalized distribution of CodeRabbit comment categories (%) on agent-authored PRs, authors with $N \geq 200$ classified comments. Last column is the total number of entries (N).

Author	p.issue	refactor	nitpick	a.chain	unlab.	conv.	verif.	N
OpenAI Codex	29.9	29.2	20.5	12.8	2.4	4.7	0.5	16,292
Claude Code	42.0	35.0	7.8	10.6	0.3	3.7	0.6	8,757
Cursor	34.9	33.5	19.8	10.2	0.9	0.2	0.5	5,084
Copilot	49.0	10.5	12.9	18.9	6.1	2.5	0.1	4,260
Devin	43.9	9.7	11.6	20.0	12.3	2.2	0.4	855

The association between authoring agent and CodeRabbit comment category is statistically significant but small (Cramér’s $V = 0.150$; Pearson $\chi^2 = 3177.5$, $df = 24$, $p < 10^{-300}$). We therefore focus on the largest category-level contrast. CodeRabbit labels 35.0 % of comments on Claude Code-authored PRs as `refactor`, compared with 10.5 % on Copilot-authored PRs, a difference of 24.5 percentage points (95% CI [23.1, 25.9]; Wald two-proportion test; $N = 8757$ and $N = 4260$ comments, respectively). We make no claim about code quality,



■ **Figure 2** Normalized distribution of CodeRabbit comment categories on agent-authored PRs, grouped by authoring agent (bars sum to 100%). The same reviewer bot produces visibly different mixes downstream of different authoring agents; the largest single difference is **refactor**, contrasting the Claude Code / Cursor / Codex cluster (29–35%) with the Copilot / Devin pair (10–11%).

as change size, language, repository, and configuration may influence the patterns observed. Our result is therefore observational and indicates that CodeRabbit produces different comment-category distributions for Claude Code- and Copilot-authored PRs.

5.2 Same- vs. cross-product review volume

We restrict to reviewer bots with ≥ 200 PRs in *both* datasets (Copilot, OpenAI Codex, Devin, Amazon Q) and compare per-PR comment counts on same- vs. cross-product authored PRs (Table 4; means with Mann–Whitney U tests, as counts are heavily right-skewed).

■ **Table 4** Mean review comments per PR by reviewer, split by same- and cross-product authors. Counts use the reviewer perspective: “Self PRs” match the same-product totals in Table 2, whereas “Cross PRs” do not. p values are from two-sided Mann–Whitney U tests; $|\delta|$ is Cliff’s delta (self vs. cross; < 0.147 negligible, 0.147 – 0.33 small).

Reviewer bot	Cross PRs	Mean cross	Self PRs	Mean self	p	$ \delta $
Copilot	21,022	1.49	166,442	2.35	$< 10^{-292}$	0.15
OpenAI Codex	5,485	0.91	37,247	0.89	2.0×10^{-3}	0.02
Devin	392	1.09	3,811	1.80	3.0×10^{-6}	0.14
Amazon Q	522	4.94	535	8.08	2.2×10^{-14}	0.27

The per-PR distributions overlap heavily and Cliff’s δ is small-to-negligible across all four bots (Devin and Copilot sit within 0.01 of the 0.147 boundary, so labels are indicative only). The mean-level gap is therefore not a population-wide shift but a long right tail of high-comment same-product PRs: three of four bots show same-product PRs receiving 58–65% more comments on average (Copilot 58 %, 1.49 to 2.35; Devin 65 %, 1.09 to 1.80; Amazon Q 64 %, 4.94 to 8.08), while OpenAI Codex is the lone exception (0.91 vs. 0.89, too small for a directional claim despite a sample-size-driven p). This gap is consistent across three otherwise dissimilar products, even though it is concentrated in the upper tail of the

distribution. Because same- and cross-product reviews differ in several key characteristics, including change size, language, repository, and reviewer composition, the observed differences should not be interpreted causally.

5.3 Latency from PR open to first AI review

We measure, for every qualifying PR-bot pair in both datasets, the elapsed time between PR creation and the bot’s first recorded review event. After dropping pairs missing either timestamp and the small share with negative differences (clock-skew or backfill), 103,920 pairs remain; distributions are heavily right-skewed, so we report medians with IQRs (Table 5).

■ **Table 5** Median latency from PR open to first AI review, in minutes. Top: same- vs. cross-product pairs aggregated. Bottom: by reviewer bot (datasets pooled), reviewer bots with ≥ 100 pairs.

Split	N	p25	median	p75
Cross-product pairs	37,420	0.4	1.2	4.4
Same-product pairs	66,500	2.4	4.7	38.2
Gemini Code Assist (reviewer)	9,873	0.3	0.5	1.8
Amazon Q (reviewer)	959	0.9	1.2	1.6
OpenAI Codex (reviewer)	34,765	1.9	2.6	3.7
CodeRabbit (reviewer)	6,709	3.7	7.0	31.6
Copilot (reviewer)	51,437	1.0	17.7	73.9

Two observations. First, median first-review latency in the aggregate is 1–5 minutes; AI review on these PRs commonly begins within minutes of PR creation. We do not compare this against a matched human baseline, which we do not have. Second, and counter-intuitively, cross-product pairs have a lower median latency than same-product pairs (1.2 vs. 4.7 minutes), the opposite of a “product rubber-stamps its own” reading. The reviewer-level results suggest that this aggregate difference is compositional: reviewers with shorter median latencies occur more frequently in the cross-product dataset, whereas Copilot, which has a longer median latency (17.7 minutes, p75 74 minutes), dominates the same-product dataset. Because the reviewer populations differ, the observed difference should be interpreted as reflecting reviewer composition rather than any inherent preference for same- or cross-product review.

RQ2 Summary: AI review typically starts within minutes, with cross-product reviews being faster than same-product ones (median 1.2 vs. 4.7 minutes). CodeRabbit’s **refactor** share was 10 % to 35 % across all agents. Three reviewer bots produced 58 % to 65 % more comments on same-product PRs, though distributions were long-tailed and unmatched.

6 Discussion

Closed-loop AI review is a fast-growing part of the development environment. AI-authored and AI-reviewed pull requests, though still a minority of public GitHub activity, are already frequent enough in absolute terms that empirical software engineering can no longer assume a purely human-authored population. This complicates sampling assumptions: studies must explicitly account for AI-generated artifacts or risk mixing fundamentally different populations.

Reviewer output differs across authoring-agent groups. With CodeRabbit held constant, the observed comment-category distributions differ across authoring agents. These differences may reflect the types of PRs produced by each agent, repository context, or reviewer

behavior, which the present analysis cannot separate. Downstream tooling that summarizes or aggregates AI review output may therefore produce systematically different interpretations for different AI coders, even under the same reviewer.

Same- and cross-product patterns may reflect product design. The two groups differ in reviewer composition and may also differ in product integration and triggering, so the observed gaps need not reflect review strictness or reviewer preference. Product boundaries also need not match model boundaries, since different products may share a foundation model and one product may run several.

7 Threats to Validity

We organize the threats to validity into construct, internal, and external dimensions.

Construct validity. Our attribution uses signature-based identification, which can undercount agents whose signatures are missing, uncatalogued, or removed by vendors. The framework is designed to prioritize precision, but residual misclassification remains possible when signatures are copied, quoted, modified, or shared across product variants. Missing or removed signatures additionally produce under-attribution. The absence of a signature is not evidence that a PR was not AI-authored or AI-reviewed: such PRs are not quarantined but simply never enter our population. Under-attribution is also asymmetric, being far lower on the reviewer side, where the major reviewers post under vendor-controlled bot logins, than on the author side, which depends on body trailers that agents may not emit and that squash merges may strip. Absolute counts are therefore lower bounds, whereas rates expressed per agent-authored PR inherit the coverage of both sides and should be read as describing the attributable population rather than all AI-authored PRs. In addition, our RQ2 content measures rely on CodeRabbit’s self-declared and observed comment categories. These labels describe the tool’s own output and are not independently validated measures of issue type, severity, correctness, or code quality. External validation with human annotation or LLM-based judges is deferred to future work.

Internal validity. Our study reports observational associations rather than causal effects. Confounding factors such as pull request size, language composition, repository-level differences, and product-specific integration behavior may influence observed patterns. While we report several robustness checks in the main analysis, we do not claim causal identification.

External validity. Our dataset covers public GitHub events in CODAGE (CodAGE) [7] from 2024-01-01 to 2026-04-15, excluding private repositories, GitHub Enterprise deployments, and other version control platforms. AI reviewer activity is therefore conditioned on repositories that adopt specific integrations, and results may not generalize to non-adopting projects. Our “closed-loop” definition requires an AI coding agent on each side of the PR and does not require the absence of human review: the review stream we mine is restricted to AI-attributed events, so we cannot determine whether a human also reviewed a given PR, and some closed-loop PRs likely received human review as well. Finally, “cross-product” reflects product-level attribution rather than model-level identity.

8 Conclusion and Future Work

This paper presented a large-scale dataset and empirical study of AI reviews on AI-authored GitHub pull requests, defined as PRs authored by one identified AI coding agent and reviewed by one or more identified AI reviewers. Although still a minority of agent activity, such review grew rapidly over 2025 and already spans tens of thousands of cross-product interactions. Reviewer output differed across authoring-agent groups and author-reviewer configurations, in comment-category mix, per-PR comment volume, and latency. These findings describe observable reviewer behavior, not review correctness or software quality.

Validating review content and usefulness. Our content analysis relies on CodeRabbit’s own categories, which describe its output but are not validated measures of correctness, severity, or usefulness, and prior work finds that AI review comments may be rejected, redundant, or poorly grounded in the code [14, 21, 28]. Future work should have human annotators rate a stratified sample of comments for correctness, localization, severity, and actionability, reporting inter-rater agreement, with LLM-based evaluation used only as a secondary check. Comment volume, latency, and category mix do not show whether a review improves the code, so annotation should be paired with outcome linkage to accepted suggestions, later commits, continuous-integration results, merges, and reverts, treating acceptance as a response signal rather than a measure of precision. Matched human- and AI-authored pull requests – aligned on repository, language, task type, and change size, would isolate AI authorship patterns from those of the review system or context.

Separating product effects from model effects. Attribution here operates at the product level because public records rarely reveal the underlying model, so same- and cross-product configurations are not same- and different-model configurations. Because LLM evaluators may favor outputs that resemble their own or appear more familiar [15, 25, 3], controlled experiments could review functionally equivalent pull requests from known models under fixed configurations, using source blinding and function-preserving transformations of naming, formatting, or structure. Outcomes should include defect detection, false-positive rates, severity calibration, comment volume, and review correctness.

Testing correlated failures and security risks. AI authors and reviewers may share failure modes when they rely on related models, training sources, prompts, or tools, which the current dataset cannot assess because model identity and ground-truth defects are unavailable. A controlled benchmark could inject known functional defects and security vulnerabilities into realistic pull requests and compare same-model, related-model, and independent-model reviewers on defect recall, false positives, localization, severity, and repair correctness, showing whether reviewer diversity reduces correlated blind spots.

Provenance and longitudinal measurement. Public records reveal little about how a contribution was produced and reviewed, so future tools should expose structured metadata on authoring and reviewing products, model versions, review triggers, configurations, and human involvement, distinguishing independent review from automatic same-product review. Because the ecosystem changes rapidly, the pairings and activity levels reported here describe one period; the broader contribution is a reproducible method for measuring AI participation in authorship and review, which, repeated over time with model-level provenance and outcome-based evaluation, will help assess whether these workflows improve software correctness, security, and maintainability.

Data Availability Statement

The replication package, including the agent signature registry, data-processing scripts, classifier, and analysis code, is available at <https://github.com/Niruthiha/AI-AI-CodeReviews>. The source dataset, CODAGE(Coding Agent-generated GitHub Events), is available on Hugging Face at <https://huggingface.co/datasets/taher-ghaleb/CodAGE> under CC BY 4.0. These artifacts reproduce the counts, tables, and figures reported in this paper.

References

- 1 Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint*, 2022. [arXiv:2212.08073](https://arxiv.org/abs/2212.08073).
- 2 Chris Brown and Chris Parnin. Sorry to bother you: Designing bots for effective recommendations. In *Proceedings of the IEEE/ACM 1st International Workshop on Bots in Software Engineering*, pages 54–58. IEEE, 2019. [doi:10.1109/BOTSE.2019.00021](https://doi.org/10.1109/BOTSE.2019.00021).
- 3 Zhi-Yuan Chen, Hao Wang, Xinyu Zhang, Enrui Hu, and Yankai Lin. Beyond the surface: Measuring self-preference in LLM judgments. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 1653–1672, 2025. [doi:10.18653/V1/2025.EMNLP-MAIN.86](https://doi.org/10.18653/V1/2025.EMNLP-MAIN.86).
- 4 Umut Cihan, Vahid Haratian, Arda İçöz, Mert Kaan Gül, Ömercan Devran, Emircan Furkan Bayendur, Baykal Mehmet Uçar, and Eray Tüzün. Automated code review in practice. In *2025 IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Practice*, pages 425–436. IEEE, 2025. [doi:10.1109/ICSE-SEIP66354.2025.00043](https://doi.org/10.1109/ICSE-SEIP66354.2025.00043).
- 5 Tapajit Dey, Sara Mousavi, Eduardo Ponce, Tanner Fry, Bogdan Vasilescu, Anna Filippova, and Audris Mockus. Detecting and characterizing bots that commit code. In *Proceedings of the 17th international conference on mining software repositories*, pages 209–219, 2020. [doi:10.1145/3379597.3387478](https://doi.org/10.1145/3379597.3387478).
- 6 Ramtin Ehsani, Sakshi Pathak, Shriya Rawal, Abdullah Al Mujahid, Mia Mohammad Imran, and Preetha Chatterjee. Where do AI coding agents fail? an empirical study of failed agentic pull requests in GitHub. In *Proceedings of the 23rd International Conference on Mining Software Repositories*, pages 807–811, 2026. [doi:10.1145/3793302.3793579](https://doi.org/10.1145/3793302.3793579).
- 7 Taher A. Ghaleb. CodAGE: A dataset of coding agent-generated GitHub events, 2026. URL: <https://huggingface.co/datasets/taher-ghaleb/CodAGE>.
- 8 Taher A Ghaleb. Fingerprinting ai coding agents on github. In *Proceedings of the 23rd International Conference on Mining Software Repositories*, pages 822–826, 2026. [doi:10.1145/3793302.3793582](https://doi.org/10.1145/3793302.3793582).
- 9 Mehdi Golzadeh, Alexandre Decan, Damien Legay, and Tom Mens. A ground-truth dataset and classification model for detecting bots in GitHub issue and PR comments. *Journal of Systems and Software*, 175:110911, 2021. [doi:10.1016/J.JSS.2021.110911](https://doi.org/10.1016/J.JSS.2021.110911).
- 10 Saki Imai. Is GitHub Copilot a substitute for human pair-programming? an empirical study. In *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings*, pages 319–321, 2022. [doi:10.1145/3510454.3522684](https://doi.org/10.1145/3510454.3522684).
- 11 Haolin Jin and Huaming Chen. Are LLMs reliable code reviewers? systematic overcorrection in requirement conformance judgement. *Automated Software Engineering*, 33(3):90, 2026. [doi:10.1007/S10515-026-00638-5](https://doi.org/10.1007/S10515-026-00638-5).
- 12 Hao Li, Haoxiang Zhang, and Ahmed E Hassan. AIDev: studying AI coding agents on GitHub. In *Proceedings of the 23rd International Conference on Mining Software Repositories*, pages 1029–1033, 2026. [doi:10.1145/3793302.3797249](https://doi.org/10.1145/3793302.3797249).
- 13 Zhiyu Li, Shuai Lu, Daya Guo, Nan Duan, Shailesh Jannu, Grant Jenks, Deep Majumder, Jared Green, Alexey Svyatkovskiy, Shengyu Fu, et al. Automating code review activities by large-scale pre-training. In *Proceedings of the 30th ACM joint European software engineering conference and symposium on the foundations of software engineering*, pages 1035–1047, 2022.

- 14 Hong Yi Lin, Mingzhao Liang, Kla Tantithamthavorn, and Patanamon Thongtanunam. Is agentic code review helpful? mining developers' feedback to CodeRabbit reviews in the wild. *arXiv preprint*, 2026. [arXiv:2607.03316](#).
- 15 Arjun Panickssery, Samuel R Bowman, and Shi Feng. LLM evaluators recognize and favor their own generations. In *Proceedings of the 38th International Conference on Neural Information Processing System*, 2024.
- 16 Sida Peng, Eirini Kalliamvakou, Peter Cihon, and Mert Demirel. The impact of AI on developer productivity: Evidence from Github Copilot. *arXiv preprint*, 2023. [doi:10.48550/arXiv.2302.06590](#).
- 17 Xiaoxue Ren, Chaoqun Dai, Qiao Huang, Ye Wang, Chao Liu, and Bo Jiang. Hydra-Reviewer: A holistic multi-agent system for automatic code review comment generation. *IEEE Transactions on Software Engineering*, 2025.
- 18 Niruthiha Selvanayagam. Does the agent matter? predicting merge outcomes of AI-authored code. In *Proceedings of the 34th ACM International Conference on the Foundations of Software Engineering*, pages 82–84, 2026. [doi:10.1145/3803437.3804439](#).
- 19 Tao Sun, Jian Xu, Yuanpeng Li, Zhao Yan, Ge Zhang, Lintao Xie, Lu Geng, Zheng Wang, Yueyan Chen, Qin Lin, et al. BitsAI-CR: Automated code review via LLM in practice. In *Proceedings of the 33rd ACM international conference on the foundations of software engineering*, pages 274–285, 2025.
- 20 Xunzhu Tang, Kisub Kim, Yewei Song, Cedric Lothritz, Bei Li, Saad Ezzini, Haoye Tian, Jacques Klein, and Tegawendé F Bissyandé. CodeAgent: Autonomous communicative agents for code review. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11279–11313, 2024. [doi:10.18653/V1/2024.EMNLP-MAIN.632](#).
- 21 Kla Tantithamthavorn, Hong Yi Lin, Patanamon Thongtanunam, Wachiraphan Charoenwet, Minwoo Jeong, and Ming Wu. HalluJudge: A reference-free hallucination detection for context misalignment in code review automation. In *Proceedings of the 34th ACM International Conference on the Foundations of Software Engineering*, pages 632–643, 2026.
- 22 Kla Tantithamthavorn, Yaotian Zou, Andy Wong, Michael Gupta, Zhe Wang, Mike Buller, Ryan Jiang, Matthew Watson, Minwoo Jeong, Kun Chen, et al. RovoDev code reviewer: A large-scale online evaluation of llm-based code review automation at Atlassian. In *Proceedings of the IEEE/ACM 48th International Conference on Software Engineering: Software Engineering in Practice*, pages 34–45, 2026.
- 23 Priyan Vaithilingam, Tianyi Zhang, and Elena L Glassman. Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In *CHI Conference on Human Factors in Computing*, pages 1–7, 2022. [doi:10.1145/3491101.3519665](#).
- 24 Miku Watanabe, Hao Li, Yutaro Kashiwa, Brittany Reid, Hajimu Iida, and Ahmed E Hassan. On the use of agentic coding: An empirical study of pull requests on GitHub. *ACM Transactions on Software Engineering and Methodology*, 2025.
- 25 Koki Wataoka, Tsubasa Takahashi, and Ryokan Ri. Self-preference bias in LLM-as-a-judge. *arXiv preprint*, 2024. [doi:10.48550/arXiv.2410.21819](#).
- 26 Mairieli Wessel, Alexander Serebrenik, Igor Wiese, Igor Steinmacher, and Marco A Gerosa. Effects of adopting code review bots on pull requests to OSS projects. In *Proceedings of the 36th International Conference on Software Maintenance and Evolution*, pages 1–11. IEEE, 2020.
- 27 Zezhou Yang, Cuiyun Gao, Zhaoqiang Guo, Zhenhao Li, Kui Liu, Xin Xia, and Yuming Zhou. A roadmap for modern code review: Challenges and opportunities. *ACM Transactions on Software Engineering and Methodology*, 2026.
- 28 Zhengran Zeng, Ruikai Shi, Keke Han, Yixin Li, Kaicheng Sun, Yidong Wang, Zhuohao Yu, Rui Xie, Wei Ye, and Shikun Zhang. SWR-Bench: Assessing LLM performance in real-world code review comment generation. *Proceedings of the ACM on Software Engineering*, 3(FSE):3093–3115, 2026.

- 29 Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.
- 30 Suzhen Zhong, Shayan Noei, Bram Adams, and Ying Zou. From human-centric to agentic code review: The impact of different generations of generative AI technology on review quality. *arXiv preprint*, 2026. doi:10.48550/arXiv.2607.13196.
- 31 Suzhen Zhong, Shayan Noei, Ying Zou, and Bram Adams. Human-AI synergy in agentic code review. *arXiv preprint*, 2026. doi:10.48550/arXiv.2603.15911.
- 32 Xin Zhou, Kisub Kim, Bowen Xu, DongGyun Han, Junda He, and David Lo. Generation-based code review automation: How far are we? In *Proceedings of the IEEE/ACM 31st International Conference on Program Comprehension*, pages 215–226. IEEE, 2023. doi:10.1109/ICPC58990.2023.00036.