Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

How Developers Discuss Generative AI Across the AI-Integration Gradient: A Longitudinal, Multi-Repository Study of GitHub Issues

Panida Rumriankit

panida.rum@ku.th

Kasetsart University

Akito Monden

Okayama University

Peerapas Chinnapatpakdeekul

Kasetsart University

Research Article

Keywords: Generative AI, GitHub issues, Mining software repositories, Inter-rater reliability, AI-generated contributions

Posted Date: November 13th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11214426/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

How Developers Discuss Generative AI Across the AI-Integration Gradient: A Longitudinal, Multi-Repository Study of GitHub Issues

Panida Rumriankit[1*], Akito Monden[2] and Peerapas Chinnapatpakdeekul[1]

[1]Faculty of Engineering, Kasetsart University, Bangkok, Thailand.
[2]Faculty of Environmental, Life, Natural Science and Technology, Okayama University, Okayama, Japan.

*Corresponding author(s). E-mail(s): panida.rum@ku.th;
Contributing authors: monden@okayama-u.ac.jp; peerapas.c@ku.th;

Abstract

How open-source communities discuss generative artificial intelligence (AI) remains unclear. We study generative-AI discussion in GitHub Issues from January 2021 to August 2026 in seven repositories spanning an *AI-integration gradient*: Visual Studio Code (VS Code), where AI features are prominent; Home Assistant, where AI is one feature; and five projects without built-in AI features. Two annotators independently labeled a stratified sample of 2,080 from 52,598 keyword-matched issues with large language model assistance; we add topic modeling, interrupted time series, and survival analysis. AI relevance agreement is high in repositories with AI features (Cohen's $\boldsymbol{\kappa = 0.78}$–$\mathbf{0.85}$) but near zero in CPython and Rust ($\boldsymbol{\kappa \leq 0.08}$), where most disagreements involve issues whose AI evidence appears only in the comment thread, typically a maintainer invoking an AI-contribution policy. We therefore report every estimate under both annotators. Consistently across annotators, AI-feature trackers carry operational discourse; keyword precision falls from 53–61% in VS Code to under 3% in Flutter and Godot; and among issues both annotators deem AI-relevant, AI-generated contributions make up 8 of 20 in the bystander projects but 1 of 293 in VS Code. Their prevalence, however, depends on the annotator (CPython: 0.6–16%). Issue volume increased in early 2025, but estimates associated with agent-mode releases depended on breakpoint choice; the largest discontinuity coincided with consolidation of Copilot Chat issue tracking. AI-related issues close faster in the AI-feature repositories (hazard ratio 1.4) but not consistently elsewhere. The relevance classifier has lower precision outside VS Code.

Keywords: Generative AI, GitHub issues, Mining software repositories, Inter-rater reliability, AI-generated contributions

1 Introduction

Generative artificial intelligence (AI) tools—ChatGPT, GitHub Copilot, and agentic coding systems—have changed how developers write, test, and review code. Open source software (OSS) issue trackers preserve naturally occurring discussion of these tools at scale. Prior work has measured the productivity effect of AI assistants [30, 34], catalogued their risks [11, 13, 25, 40], and mined AI-specialist repositories for discussion topics [2, 24]. In the conference version of this work [31], we characterized generative-AI discussion in the Visual Studio Code (VS Code) issue tracker and found it dominated by *operational* concerns—configuring, operating, and troubleshooting AI assistants—while conceptual risks emphasized in survey studies (hallucination, licensing, code quality) rarely surfaced. Issue trackers also coordinate geographically distributed, collaborative development, and the assistants discussed in them are networked services reached through remote model endpoints, rate-limited APIs, and protocols such as the Model Context Protocol; how distributed communities absorb these services and the contributions they enable concerns collaborative networked computing as much as software engineering.

VS Code ships Copilot as a first-party feature, which leaves open whether that operational profile is a property of issue trackers or of repositories where AI features are prominent. This article therefore arranges seven repositories along an *AI-integration gradient*: VS Code, where AI features are prominent; Home Assistant, which ships large language model (LLM) integrations as one feature among hundreds; and five high-activity projects without AI features (CPython, Rust, Electron, Flutter, Godot), all mined over January 2021–August 2026. Comparing repositories also raises a measurement question that a single-repository study can leave implicit: *when is an issue "about" generative AI?* In a compiler or language runtime, AI often enters sideways—a maintainer asks a contributor to disclose LLM use, or dismisses a reply as machine-written, inside a thread whose subject is an ordinary defect. To make this visible rather than resolve it silently, two LLM-assisted annotators independently labeled a 2,080-issue sample, and we report every label-dependent result under each annotator and their intersection. We first revisit and extend the three research questions examined in the conference study for VS Code (RQ1–RQ3) [31], and then investigate whether these findings generalize across repositories with different levels of AI integration (RQ4) and how issue-closure patterns differ across these repositories (RQ5):

- **RQ1:** What topics related to generative AI emerge in the VS Code community?
- **RQ2:** What concerns and expectations do these discussions reveal?
- **RQ3:** How have the discussions evolved from 2021 to 2026, and which product milestones coincide with changes?
- **RQ4:** Do the findings generalize across repositories with different levels of AI integration?

- **RQ5:** Do AI-related issues close faster or slower than incidental keyword matches?

This article makes three main contributions. First, we extend the conference study from VS Code to seven repositories spanning different levels of AI integration. Second, we provide a two-annotator analysis showing that the nature and prevalence of AI-related discussion depend strongly on both repository context and the definition of AI relevance. Third, we analyze temporal changes and issue-closure outcomes using interrupted time-series and survival analyses.

1.1 Extension of the conference paper

The SNPD 2026 paper [31] studied VS Code alone with 300 single-annotator labels. This article shares with it RQ1–RQ3, the keyword-collection and preprocessing pipeline, and the BERTopic configuration. New are: six further repositories (4,827 keyword-matched issues) arranged in a pre-screened gradient; a window extended to August 2026; a new 2,080-issue sample coded under a 13-theme codebook by two independent annotators; held-out classifier evaluation; RQ4 and RQ5; an interrupted time-series analysis with an internal-control robustness check; and design-weighted and censoring-aware inference.

2 Related Work

Generative AI in software development. Copilot users completed a task 55.8% faster [30], and OSS projects gained 5.9% more contributions alongside more coordination time [34]. The risks include hallucinated code [25], security weaknesses in roughly 30% of Copilot snippets [11], code duplication [13], and unattributed licensed code [40]. Tufano et al. [36] studied self-admitted AI use in OSS; explicit disclosure likely underestimates AI-related discussion, which motivates keyword mining. Agent-authored pull requests have been studied through rejection and review outcomes [9], while OSS contribution guidelines increasingly address AI disclosure and human review [19]. We examine how these concerns surface inside issue threads.

Mining GitHub issues and topic modeling. Yang et al. [41] analyzed 24,953 issues in AI research repositories, Lin et al. [24] applied latent Dirichlet allocation (LDA) to issues of ChatGPT-related projects, and Asgari et al. [2] mined AI-agent frameworks; all target AI-specialist communities, whereas we contrast AI-feature, AI-adjacent, and AI-bystander projects. Han et al. [18] found that GitHub issues capture operational, workflow-oriented concerns. Topic models and trends in software engineering [3, 5, 33] need corpus-specific configuration [35], and BERTopic [16] suits short, jargon-dense issue text.

Reliability of manual and LLM-assisted coding. Mining studies conventionally report Cohen's κ [7] on the Landis and Koch [23] scale. When one category dominates, κ can be low despite high raw agreement [10]; Gwet's AC1 [17] and Krippendorff's α [22] behave differently under such skew. LLMs are increasingly used to annotate text [12], including software artifacts, where agreement with human labels varies strongly by task [1].

3 Methodology

3.1 Repositories and data

We define three tiers of AI integration: *AI-prominent* (VS Code, which ships Copilot and agent mode), *AI-adjacent* (Home Assistant, whose OpenAI Conversation integration arrived in release 2023.2), and *AI-bystander* (no AI features). For the bystander tier we pre-screened twelve candidates across six domains by querying the GitHub Search application programming interface (API) for issues matching our core AI keywords (excluding the ambiguous "cursor" and "gemini"), and selected the highest-signal repository per domain: CPython (278 hits), Flutter (170), Rust (86), Godot (70), and Electron (60).

We collected issues with twelve case-insensitive keywords (ChatGPT, GPT, Copilot, Claude, Cursor, Gemini, LLM, generative AI, gen AI, AI-generated, AI-assisted, AI assistant), one Search API query per keyword per monthly window, splitting windows that exceeded the 1,000-result cap. "GPT" was applied only as a client-side filter to avoid substring matches. For each issue we retrieved title, body, labels, state, timestamps, and all comments, keeping issues in which a keyword appears anywhere; pull requests were excluded. All seven trackers cover 1 January 2021 to 31 August 2026. This yielded 52,598 keyword-matched issues: 47,771 from VS Code and 4,827 from the six comparison repositories. Documents (title, body, comments) were cleaned of code, markup, URLs, and environment dumps, lemmatized with spaCy, and stripped of stop words; documents with fewer than five tokens were dropped. The VS Code topic model and trend analyses (RQ1, RQ3) were fit before the August extension, on the July-complete corpus of 46,250 documents.

3.2 Labeling sample and annotation

We drew a new sample from the August-complete corpora (Table 1), replacing an earlier pass (round v1; 1,449 issues) that mixed codebook vintages. Sampling is stratified by repository and creation year: proportional for VS Code, a census for the three smallest repositories, and recency-weighted for Home Assistant, Flutter and Godot, because the contributor themes barely existed before 2025. Population prevalence and annual-count estimates are design-weighted—an issue in repository–year stratum h carries weight $w_h = N_h/n_h$ [20]—with percentile confidence intervals (CIs) from a stratified bootstrap (2,000 resamples). For the three census repositories, whose labeled set is the whole corpus, intervals do not reflect sampling error; like the others, they treat each repository's 2021–2026 issues as one realization of an ongoing process and support comparisons between repositories and years. No interval includes annotation error. Raw sample counts, agreement statistics, Fisher tests, and Cox models are unweighted. The label sheet omits repository and issue number and is shuffled, although the repository remains inferable from the issue URL.

Each issue receives one of thirteen themes: the ten conference themes (code generation, testing, documentation, bug fixing, security, licensing, code review, hallucination, productivity, other) plus *contribution policy* (a project rule about AI use is invoked, debated, or set), *AI-generated spam* (a specific issue or reply is treated as

Table 1 Corpus and labeling design (2021-01-01 to 2026-08-31). *Raw*: keyword-matched issues; *Corpus*: documents after preprocessing; *Fraction*: per-year sampling fraction (%); *recency*: year-stratified with 2025 and 2026 over-sampled

Repository	Tier	Raw	Corpus	Labeled	Allocation	Fraction
VS Code	AI-prominent	47,771	47,721	550	proportional	1.1–1.2
Home Asst.	AI-adjacent	1,051	1,051	250	recency	15.2–31.8
CPython	AI-bystander	486	485	485	census	100
Rust	AI-bystander	244	244	244	census	100
Electron	AI-bystander	251	251	251	census	100
Flutter	AI-bystander	1,605	1,605	150	recency	7.8–15.6
Godot	AI-bystander	1,190	1,190	150	recency	10.2–20.2
Total		52,598	52,547	2,080		

machine-generated low-quality content), and *AI feature request*. "Other" covers keyword matches that are not substantively about generative AI. The *AI-as-contributor* class comprises contribution policy, AI-generated spam, and licensing; the remaining non-"other" themes form the *AI-operational* class. A written field guide makes two ordered decisions—is the issue substantively about AI, and which theme fits its primary subject—with boundary rules (plan and quota limits are productivity; sign-in and tokens are security) and four traps that look like AI but are not (ambiguous words, bot boilerplate, a reporter who merely consulted a chatbot, a commenter's unrelated tangent). It also instructs annotators to read the full thread, because contributor themes are usually established by a maintainer's reply.

Two annotators labeled all 2,080 issues independently: the first author (annotator 1) and the third author (annotator 2). *Use of LLMs.* Both used LLM assistance with different models and review rules, all models at high reasoning effort and given the field guide and the full issue thread. For annotator 1, Claude Opus 5.5 (Anthropic) labeled every issue and GPT-6 Sol (OpenAI) labeled it independently as a check; she kept labels on which the models agreed and decided conflicts after comparing their arguments. Annotator 2 labeled about the first 120 rows of the shuffled sheet unaided; for the rest, Claude Sonnet 5 (Anthropic) proposed a theme with a self-rated confidence of 0–10, and he reviewed proposals rated below 8. Model outputs, confidence scores, and review counts were not logged. The label sets A1 and A2 are thus outputs of two human-in-the-loop LLM pipelines, not of unaided coders. Labels were frozen before any agreement statistic was computed, and all results use these pre-adjudication labels; the 365 disagreements are analyzed by category.

3.3 Reliability and estimation under disagreement

We report raw agreement, Cohen's κ, Krippendorff's α, and Gwet's AC1 for the 13 themes, for three groups (operational, contributor, other), and for binary AI relevance, with bootstrap CIs; per-repository κ is read together with AC1 and prevalence because "other" makes up 71–80% of labels. An exact McNemar test [27] tests whether one annotator systematically codes more issues as AI-relevant. For every issue we record whether its core AI keywords occur in the title or body or only in the comments. As

a stability check, 524 of the issues had also been coded by annotator 1 in round v1, about two months earlier, manually and without the field guide. Because annotator 2's switch to LLM-first labeling falls at about row 120 of a shuffled sheet, we compare agreement on rows 1–120 with the rest. This observational comparison also permits a learning effect.

Rather than adjudicate before reporting, we report every label-dependent estimate three ways: under A1 (reported first; its pipeline checked every label against a second, independent model), under A2, and under their intersection ("both"), which counts an issue as AI-relevant or AI-as-contributor only if both annotators do. The intersection is a conservative coding rule, not a bound on true prevalence: both pipelines can make the same error. Conclusions that hold under all three are called *robust to annotator*, meaning agreement between pipelines, not demonstrated accuracy (Section 6). Keyword location is only a proxy for the distinction the field guide left implicit: AI evidence in the title or body suggests *subject-level* relevance; evidence only in the comments, a *thread-level* AI event.

3.4 Relevance classifier, topics, and trends

As in the conference version, a logistic-regression classifier over MiniLM sentence embeddings (trained on 1,449 round-v1 labels pooled across all seven repositories) filters out incidental matches. In pooled five-fold cross-validation, its precision is 0.80 (recall 0.93) on VS Code but 0.33–0.54 on the AI-bystander repositories. We use it only to filter the VS Code corpus for topic modeling.

BERTopic (all-MiniLM-L6-v2 embeddings, UMAP dimensionality reduction, HDBSCAN density-based clustering with minimum topic size 50, and a unigram–bigram class-based TF-IDF representation) is fit on the classifier-filtered July-complete VS Code corpus (32,096 of 46,250 documents, 69.4%). The conference version fit its primary model on the unfiltered keyword corpus to guard against classifier-induced theme loss [31]; we filter here because the classifier retains most labeled AI-relevant VS Code issues (recall 0.93), although recall within individual themes was not established. The conference paper validated BERTopic against LDA [31]. Trends are tested per topic with Mann–Kendall tests [26] and Sen's slope [32] on monthly shares, with Benjamini–Hochberg (BH) correction [4].

3.5 Interrupted time series

To assess whether issue volume changed around three milestones, we fit a segmented regression to VS Code's monthly count of keyword-matched issues ($\log(n+1)$, 67 months) with level- and slope-change terms at the ChatGPT release (November 2022), Copilot Chat public beta expansion (September 2023), and the agent-mode preview in VS Code Insiders (February 2025) [14, 15, 29], using heteroskedasticity- and autocorrelation-consistent (HAC) standard errors [28]. Agent mode reached VS Code Stable with release 1.98 on 5 March 2025 [37] and general availability in April 2025; we test both dates. Because Copilot Chat issues have been tracked in the VS Code repository since 30 June 2025 [38], every specification adds level and slope terms for this consolidation (July 2025), and a month-by-month scan locates the best-fitting break.

A logit model of the AI-keyword share of all monthly issues addresses tracker-wide growth; a difference-in-differences (DiD) model against VS Code's non-AI-keyword issues is, up to a count offset, the same model.

3.6 Keyword ambiguity and closure outcomes

Because "cursor" (the text caret) and "gemini" are lexically ambiguous, we recompute the gradient on labeled issues supported by at least one of the ten unambiguous *core* keywords. For outcomes, we re-fetched every labeled issue's state from the GitHub API on 28 September 2026 (2,079 of 2,080 retrieved) and compared AI-relevant with incidental issues using Kaplan–Meier estimation with log-rank tests [21] and Cox models [8] with creation year as a covariate, per repository (BH-corrected) and pooled with repository fixed effects. A hazard ratio (HR) above one means AI-relevant issues close faster. Robustness checks stratify the baseline hazard by repository and year, restrict to issues created from 2025, censor at first collection instead of the common date, and, because AI evidence can enter a thread after an issue opens, treat AI relevance as a time-varying covariate starting at the first AI-keyword comment (or keep only AI-relevant issues whose AI evidence is in the title or body). *Closed* means GitHub's closed state, including duplicates, spam, and issues not planned, not a verified fix; the comparison group is incidental keyword matches. We also report a closed-only Mann–Whitney test with Cliff's δ [6]—which is confounded by calendar time and discards still-open issues.

4 Results

4.1 RQ1: Topics discussed in VS Code

The classifier retains 32,096 of 46,250 documents, on which BERTopic identifies 107 topics—fewer than the conference version's 125, which were fit on the unfiltered keyword corpus. Table 2 lists the ten largest in this filtered corpus. As in the conference version, the largest topics concern operating AI-assisted development environments: agent file-reading and editing tools, agent terminal commands, Model Context Protocol (MCP) server configuration, inline completion feedback, rate limiting, and bring-your-own-key model endpoints. Ordinary editor topics (Windows Subsystem for Linux, Remote-SSH) enter through incidental keyword mentions. Two caveats apply: 41.8% of documents are HDBSCAN outliers, and some large topics (footnote a of Table 2) are organized around VS Code's triage-bot boilerplate rather than issue content. Word sequences repeated verbatim in at least 1% of documents (bot replies, template fields, stock notices) make up 16% of the text. Removing them and refitting with five seeds (Online Resource 1, Tables S6–S7) dissolves both boilerplate clusters and splits inline completion feedback; the other seven top topics persist, rate limits and quotas become the largest topic, and rising topics still outnumber falling ones (61 vs. 8 of 88). Top-ten-term coherence is $c_v = 0.65$ and $c_{\mathrm{NPMI}} = 0.23$ (bigrams split into tokens); these corpus-level scores do not validate single labels.

Table 2 The ten largest of the 107 topics in the classifier-filtered VS Code corpus. Labels are the authors', derived from the top terms. *Share*: % of all 32,096 filtered documents. *Trend*: Mann–Kendall on monthly shares over the 41 volume-adequate months from March 2023 (↑ increasing, – none; $p < 0.05$)

ID	Label	Docs	Share	Trend	Top terms
0	Agent file tools and skills	917	2.9	–	`user user, file read`
1	Agent terminal commands	844	2.6	–	`shell, run terminal`
2	Windows Subsystem for Linux	745	2.3	–	`standard wsl, linux standard`
3	MCP server configuration	625	1.9	↑	`mcp server, parser`
4	Inline completion feedback	550	1.7	–	`copilot completion, completion feedback`
5	Non-English reports	547	1.7	↑	`bye, que`
6	Chat response errors[a]	456	1.4	–	`response return, response type`
7	Rate limiting	444	1.4	↑	`rate, rate limit`
8	Custom model endpoints	415	1.3	↑	`ollama, byok`
9	Chat request failures[a]	410	1.3	–	`chat type, day chat`

[a]Cluster organized around VS Code's triage-bot boilerplate rather than issue content

4.2 Reliability of the LLM-assisted labels

Table 3 summarizes agreement. Over all 2,080 issues, the annotators agree on 82.5% of the 13-theme labels ($\kappa = 0.57$ [0.54, 0.60], moderate [23]) and on 87.8% of relevance decisions ($\kappa = 0.67$ [0.64, 0.71], substantial). The pooled figures hide a sharp split (Fig. 1a): relevance agreement is high in the AI-feature tiers (VS Code $\kappa = 0.78$, Home Assistant $\kappa = 0.85$), but in CPython and Rust it is no better than chance ($\kappa = 0.04$ and 0.08), although raw agreement is 77–82%. In Flutter and Godot, AI-relevant prevalence is 1–3%, so κ is uninformative, while AC1 (0.97–0.98) reflects agreement that almost every match is incidental.

The disagreement is directional. A1 codes 28.6% of issues as AI-relevant and A2 19.8%; of the 253 binary discordant pairs, 218 are relevant for A1 only (McNemar $p < 10^{-30}$). Table 4 classifies all 365 disagreements. The largest class (T1, 119 issues: 72 in CPython and 30 in Rust) consists of issues A1 coded as AI-as-contributor and A2 as other; a related class (T2) consists of defect reports produced with LLM-assisted review tools, coded as code review by A1 and as other by A2.

Where the AI evidence sits suggests why (Fig. 1b). In 85% of T1 issues the core AI keywords appear only in the comments, against 7% of the issues both annotators code as AI-relevant. These are threads whose subject is an ordinary defect or API question in which a maintainer invokes the project's generative-AI policy, asks a participant to stop posting LLM output, or dismisses a reply as machine-written. The field guide pulls both ways here: its instruction to read the full thread points to contribution policy; its primary-subject rule and fourth trap point to other. A1's labels fit the thread-level event, A2's the issue's subject. The remaining classes are mostly mechanical: in T5 and T6, A2 departed from the boundary rules for plan limits and sign-in failures. A1's

Table 3 Inter-rater reliability (A1, A2 = annotators 1 and 2), with 95% bootstrap CIs below each coefficient. Krippendorff's α equals κ to two decimals at every level. *Prev.*: % of issues coded AI-relevant by A1 / A2. *Disc.*: issues coded AI-relevant by A1 only / A2 only

Level / repository	n	Agree. (%)	Cohen's κ	Gwet's AC1	Prev.	Disc.
All 2,080 issues						
13 themes	2,080	82.5	0.57 [0.54, 0.60]	0.82 [0.80, 0.84]		
3 groups[a]	2,080	86.1	0.64 [0.61, 0.67]	0.83 [0.81, 0.85]		
Relevance	2,080	87.8	0.67 [0.64, 0.71]	0.81 [0.78, 0.83]		
Relevance, by repository						
VS Code	550	89.3	0.78 [0.72, 0.83]	0.79 [0.74, 0.84]	60.7 / 56.5	41 / 18
Home Asst.	250	94.0	0.85 [0.78, 0.92]	0.90 [0.84, 0.95]	30.0 / 26.4	12 / 3
CPython	485	76.7	0.04 [−0.02, 0.10]	0.70 [0.64, 0.76]	22.9 / 2.5	106 / 7
Rust	244	82.4	0.08 [−0.02, 0.21]	0.78 [0.71, 0.85]	17.6 / 2.5	40 / 3
Electron	251	93.6	0.47 [0.22, 0.68]	0.93 [0.89, 0.96]	8.8 / 4.0	14 / 2
Flutter	150	98.0	0.66 [−0.01, 1.00]	0.98 [0.95, 1.00]	3.3 / 2.7	2 / 1
Godot	150	97.3	0.32 [−0.02, 0.80]	0.97 [0.94, 0.99]	2.7 / 1.3	3 / 1

[a] AI-operational, AI-as-contributor, other

Table 4 Taxonomy of the 365 disagreements. *Comm.*: % of issues whose core AI keywords occur only in the comment thread (7% for issues both annotators code AI-relevant). *Concentrated in*: repositories holding at least 15% of the class

	Disagreement	n	Comm.	Concentrated in
T1	Contributor-class (A1) vs. other (A2): AI event in a technical thread	119	85	CPython 72, Rust 30
T2	LLM review report: code review (A1) vs. other (A2)	35	20	CPython 26, Rust 6
T3	AI-operational (A1) vs. other (A2)	64	25	VS Code 41
T4	Other (A1) vs. AI-relevant (A2)	35	17	VS Code 18, CPython 7
T5	Plan and quota: productivity (A1) vs. licensing (A2)	27	7	VS Code 27
T6	Sign-in and tokens: security (A1) vs. bug fixing (A2)	15	0	VS Code 11, Home Asst. 4
T7	Other swaps between AI-relevant themes	70	10	VS Code 55
	Total	365		

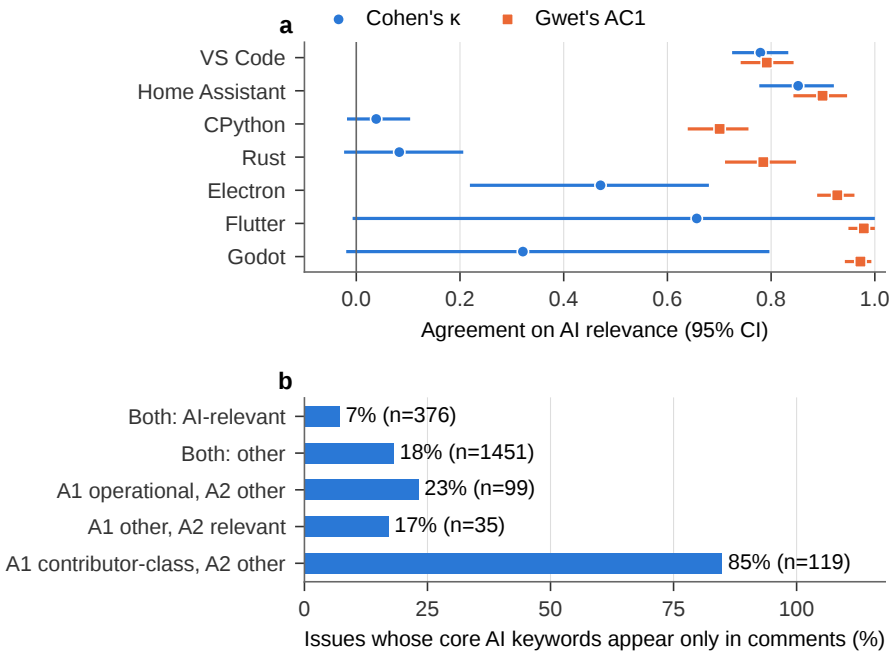


Fig. 1 Inter-rater agreement and where the AI evidence sits. **a** Agreement on AI relevance by repository, Cohen's κ (circles) and Gwet's AC1 (squares) with bootstrap 95% CIs. **b** Share of issues whose core AI keywords appear only in the comment thread, by agreement pattern

own round-v1 and round-v2 labels agree on 81.4% ($\kappa = 0.46$; binary $\kappa = 0.61$), below the conference study's washout-separated $\kappa = 0.73$.

Agreement is higher after A2's workflow change. On the 120 rows A2 labeled unaided, binary κ is 0.31 (73.3% agreement; 13-theme $\kappa = 0.28$), against 0.69 (88.7%; 0.59) on the 1,960 LLM-first rows; the difference, -0.39 $[-0.58, -0.20]$, persists when rows 111–130 are dropped. AI-relevant shares barely change (A1 28.3% vs. 28.6%; A2 23.3% vs. 19.5%, Fisher $p = 0.34$), so the change concerns *which* issues A2 coded as AI-relevant more than how many. A learning effect in A2's first rows cannot be excluded. Yet T1 occurs in 8/120 unaided rows (6.7%) and 111/1,960 Sonnet-first rows (5.7%); all eight unaided T1 cases were comment-only AI threads that A2 coded as "other". Sonnet therefore cannot alone explain the split, although A1's labels remain LLM-assisted. Either way, the coefficients in Table 3 describe agreement between two LLM-assisted pipelines, two of whose three models come from one vendor, and may overstate what unaided coders would reach. Results for the AI-feature tiers therefore rest on labels both pipelines largely agree on; bystander-tier results also depend on how comment-only AI evidence is coded. Keyword location only approximates the subject/thread distinction: 27 of the 376 issues both annotators code as AI-relevant have core keywords only in the comments, and 107 of the 253 binary disagreements have them in the title or body.

Table 5 Theme distribution: counts of labeled issues under annotator 1, with annotator 2 in parentheses (unweighted). The middle block is the AI-as-contributor class (contribution policy, AI-generated spam, licensing); the last row gives unweighted AI-relevant sample shares

Theme	VS Code	Home Assistant	CPython	Rust	Electron	Flutter	Godot
Bug fixing	192 (195)	52 (57)	0 (4)	0 (2)	1 (2)	1 (2)	0 (1)
Feature request	28 (44)	6 (7)	0 (0)	0 (0)	5 (4)	1 (0)	0 (0)
Productivity	53 (3)	2 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
Hallucination	13 (18)	2 (2)	1 (1)	0 (1)	0 (0)	0 (0)	0 (0)
Code generation	21 (2)	0 (0)	6 (0)	4 (0)	0 (1)	0 (0)	0 (0)
Documentation	1 (0)	2 (0)	0 (1)	0 (0)	2 (0)	0 (0)	0 (0)
Code review	0 (0)	2 (0)	26 (1)	6 (0)	0 (0)	2 (1)	0 (0)
Security	15 (3)	5 (0)	0 (0)	0 (0)	2 (2)	0 (0)	0 (0)
Testing	10 (7)	0 (0)	1 (1)	0 (0)	1 (0)	0 (0)	0 (0)
Contrib. policy	0 (0)	4 (0)	52 (3)	24 (3)	9 (0)	0 (0)	4 (1)
AI spam	1 (10)	0 (0)	25 (1)	8 (0)	2 (1)	1 (1)	0 (0)
Licensing	0 (29)	0 (0)	0 (0)	1 (0)	0 (0)	0 (0)	0 (0)
Other	216 (239)	175 (184)	374 (473)	201 (238)	229 (241)	145 (146)	146 (148)
AI-rel. %	**60.7** (56.5)	**30.0** (26.4)	**22.9** (2.5)	**17.6** (2.5)	**8.8** (4.0)	**3.3** (2.7)	**2.7** (1.3)

4.3 RQ2: Developer concerns and expectations

Both annotators find VS Code's AI discourse overwhelmingly operational (Table 5). Of the issues each codes as AI-relevant, bug fixing accounts for 57.5% (A1) and 62.7% (A2), and AI feature requests for 8.4% and 14.1%. A1 codes 53 plan and quota issues as productivity (15.9% of its AI-relevant issues); A2 codes 27 of them as licensing, contrary to the boundary rule. Read under the codebook, both annotations show plan economics—quotas, premium requests, subscriptions—as the second-largest concern after defects. Conceptual risks remain rare: hallucination is 3.9% (A1) and 5.8% (A2) of AI-relevant issues, security 4.5% and 1.0%, and licensing in the codebook's sense accounts for no issue under A1 and at most two under A2. As in the conference profile, bug fixing was 31.7% of its 300 labeled issues and is 34.9–35.5% of the 550 here.

4.4 RQ3: Evolution 2021–2026 and its drivers

Because early months are nearly empty, we restrict Mann–Kendall tests to months with $\geq$30 documents (41 months from March 2023), 79 topics increase and 7 decrease (of 105 tested; unchanged after BH correction). MCP configuration, rate limiting, and custom model endpoints rise, but the two largest agent-tooling topics show no trend. Seven topics decline. Declining topics held about 25% of documents in the first 12 tested months and 2% in the last 12; some change after mid-2025 may reflect the tracker consolidation below.

The interrupted time series (Online Resource 1, Tables S1–S3) also controls for a change in the tracker itself: monthly AI-keyword issues tripled in July 2025 (1,006 to 3,232), when Copilot Chat issues began to be tracked in the VS Code repository, and a month-by-month scan places the best-fitting single break there (Fig. 2). With

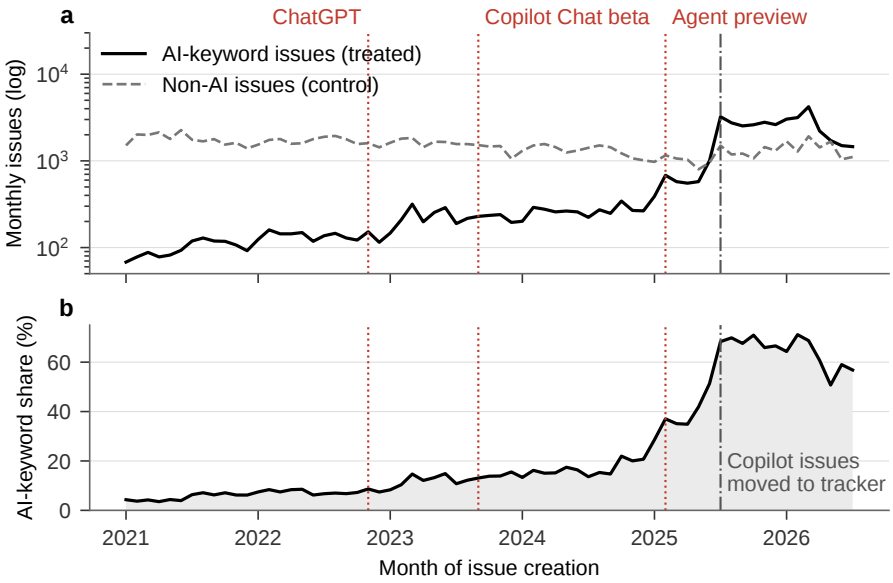


Fig. 2 VS Code monthly issue volume with the three candidate breakpoints (dotted lines) and the July 2025 move of Copilot Chat issue tracking into the repository (dash-dotted). **a** AI-keyword issues (treated, solid) and non-AI issues (control, dashed), log scale. **b** AI-keyword share of all issues

this consolidation term (level +1.38, $p < 0.001$, ≈4.0×), neither the ChatGPT release (−0.04, $p = 0.72$) nor the Copilot Chat public beta (−0.25, $p = 0.12$) shows a significant level shift, whereas the February 2025 agent-mode preview aligns with a +0.59 log-point shift ($p < 0.001$, 1.8×). Total monthly volume grew 2.1× between 2021 and 2026 while AI-keyword volume grew 25×; the preview shift persists in the share model and its DiD form (+0.57, $p < 0.001$, with null ChatGPT and Copilot Chat terms) and on classifier-retained counts (+0.67, $p < 0.001$), so keyword-matching boilerplate alone does not explain it. It depends on the release date, however: +0.28 ($p = 0.026$) at the Stable release (March 2025) and +0.16 ($p = 0.31$) at general availability (April 2025), and omitting the consolidation term inflates it to +1.08 (≈3.0×). Discussion volume thus rose in early 2025, but not at one identifiable release, and because agent mode coincided with an industry-wide agentic turn, no coefficient isolates a causal effect.

4.5 RQ4: Generalization across the AI-integration gradient

Keyword precision falls along the gradient. Table 6 reports the design-weighted share of keyword-matched issues that are about generative AI. Under all three label sets the tier ordering holds, with non-overlapping intervals: VS Code (53.2–60.7%) above Home Assistant (25.5–29.3%) above the AI-bystander tier (1.2–6.2%). Within the bystander tier, the ordering depends on the annotator: under A1 it is monotonic (CPython 22.9%, Rust 17.6%, Electron 8.8%, Flutter 2.4%, Godot 1.7%), whereas under A2 and the intersection CPython and Rust drop to 1–3%, level with the editor-like projects. We therefore claim a tier-level decline, not a rank ordering of the five bystanders.

Table 6 AI-relevant share of keyword-matched issues (%), 95% CIs below. Left: all keywords, design-weighted estimates with stratified-bootstrap CIs (census repositories: see Section 3.2). Right: issues supported by at least one unambiguous core keyword; n and the share of labeled issues dropped, then sample shares with Wilson CIs [39]

	All keywords (design-weighted)			Core keywords only		
Repository	A1	A2	Both	n (% dropped)	A1	A2
VS Code	60.7 [57.1, 64.2]	56.5 [52.9, 60.1]	53.2 [49.2, 56.9]	509 (7.5)	65.4 [61.2, 69.4]	60.9 [56.6, 65.0]
Home Asst.	29.3 [23.9, 34.7]	26.7 [21.4, 32.1]	25.5 [20.3, 31.0]	192 (23.2)	35.9 [29.5, 42.9]	31.2 [25.1, 38.1]
CPython	22.9 [19.4, 26.4]	2.5 [1.2, 3.9]	1.0 [0.2, 2.1]	277 (42.9)	39.0 [33.4, 44.9]	2.5 [1.2, 5.1]
Rust	17.6 [13.1, 22.1]	2.5 [0.8, 4.5]	1.2 [0.0, 2.9]	123 (49.6)	32.5 [24.9, 41.2]	4.1 [1.8, 9.2]
Electron	8.8 [5.6, 12.3]	4.0 [1.6, 6.4]	3.2 [1.2, 5.6]	65 (74.1)	32.3 [22.2, 44.4]	10.8 [5.3, 20.6]
Flutter	2.4 [0.5, 4.6]	2.0 [0.5, 4.2]	1.5 [0.0, 3.4]	23 (84.7)	4.3 [0.8, 21.0]	4.3 [0.8, 21.0]
Godot	1.7 [0.4, 3.3]	1.0 [0.0, 2.5]	0.4 [0.0, 1.2]	12 (92.0)	33.3 [13.8, 60.9]	8.3 [1.5, 35.4]
Bystander tier	6.2 [5.1, 7.4]	1.9 [1.0, 2.8]	1.2 [0.5, 2.0]			

How much of the decline is lexical? Restricting to issues supported by a core keyword removes 7.5% of VS Code's labeled issues but 74.1% of Electron's, 84.7% of Flutter's and 92.0% of Godot's. Under A1 the ordering then disappears: CPython's core-keyword share (39.0%) exceeds Home Assistant's (35.9%). Under A2, core-keyword shares stay between 2.5% and 10.8% in all five bystanders, against 31.2% in Home Assistant and 60.9% in VS Code. Prevalence in editor-like projects is thus largely a keyword artifact; the broader gradient survives under A2.

Bystander trackers carry relatively more governance discussion. Home Assistant's profile resembles VS Code's under both annotators (Table 5). In the AI-bystander tier, contributor-class issues form a larger share of AI-relevant issues than in VS Code under every label set, by varying margins. In the labeled sample, the AI-as-contributor class makes up 126 of 185 AI-relevant issues in the bystander tier versus 1 of 334 in VS Code under A1; 10 of 34 versus 39 of 311 under A2 (10 of 311 if A2's plan-and-billing "licensing" labels are set aside); and 8 of 20 versus 1 of 293 under the intersection (Fisher's exact test $p \leq 0.016$ in every case). Prevalence, however, diverges: under A1, contributor-class issues make up 15.9% of CPython's keyword-matched issues and 13.5% of Rust's, and 3.9% of the pooled bystander corpus; under A2 and the intersection, 0.6–1.2% of CPython and Rust and 0.5% of the pooled bystander corpus. Read as subject-level coding, the A2 and intersection estimates say AI is rarely what a bystander issue is about; read as thread-level coding, A1's say AI-generated contributions intrude on a project's ordinary work—in CPython, about one keyword-matched issue in six. Either way, these are the provenance, quality, and

Table 7 Closure outcomes under annotator 1 (states refreshed 2026-09-28). *Med.*: median days to close among closed issues; δ: Cliff's delta on closed issues (negative = AI faster), with q_{MW} the BH-corrected Mann–Whitney test on the same closed-only data; Cox HR > 1 means AI-relevant issues close faster, adjusted for creation year, q BH-corrected

	n		Closed		Med. days				Cox (year-adjusted)	
Repository	AI	inc.	AI	inc.	AI	inc.	δ	q_{MW}	HR [95% CI]	q
VS Code	333	216	297	182	6.3	7.4	−0.13	0.047	**1.44 [1.17, 1.78]**	**0.004**
Home Asst.	75	175	67	150	14.9	22.8	−0.12	0.172	**1.55 [1.15, 2.08]**	**0.012**
CPython	111	374	68	275	1.5	7.7	−0.26	0.007	1.00 [0.76, 1.33]	0.990
Rust	43	201	24	126	2.4	11.9	−0.26	0.075	1.06 [0.67, 1.67]	0.990
Electron	22	229	16	198	1.4	23.2	−0.37	0.047	0.84 [0.50, 1.43]	0.990
Flutter	5	145	3	94	3.9	14.5	−0.11	0.776	1.05 [0.33, 3.39]	0.990
Godot	4	146	3	89	1.4	19.3	−0.52	0.172	—	—
Pooled, all seven[a]									**1.18 [1.04, 1.34]**	**0.010**[b]
Pooled, AI-feature tiers[a]									**1.44 [1.22, 1.70]**	**<0.001**[b]
Pooled, AI-bystander tier[a]									0.99 [0.80, 1.22]	0.892[b]

[a]Repository fixed effects + creation year

[b]Uncorrected p (single test)

governance concerns that the conference version found missing from VS Code: *issue trackers carry conceptual AI concerns where AI arrives as an uninvited contributor.*

Classifier precision is lower outside VS Code. On 1,343 held-out round-v2 issues, precision is 0.83 (recall 0.93) on VS Code against A1 but 0.41–0.45 on Home Assistant and 0.39 (A1) or 0.01 (A2) on CPython. These issues were unseen in training, though their repositories were represented. BERTopic found no coherent AI topics in the filtered bystander corpus.

4.6 RQ5: Closure outcomes

In the AI-feature tiers, AI-relevant issues close faster, robustly across annotators and specifications (Table 7). Under A1 the HR is 1.44 [1.17, 1.78] in VS Code ($q = 0.004$) and 1.55 [1.15, 2.08] in Home Assistant ($q = 0.012$); the two-tier model gives 1.37–1.44 under all three label sets ($p < 0.001$), and the positive association persists across the robustness checks. Among closed VS Code issues, median closure times are 6.3 days for AI-relevant issues and 7.4 for incidental matches.

In the AI-bystander tier the answer depends on the label set, and the closed-only comparison misleads. Under A1, the closed-only rank test suggests AI-relevant issues close much faster in CPython (median 1.5 vs. 7.7 days, $q = 0.007$) and Electron (1.4 vs. 23.2, $q = 0.047$), yet none of these contrasts survives censoring-aware analysis (Cox HR 0.84–1.06, $q = 0.99$), and the bystander-tier model is null (HR = 0.99 [0.80, 1.22]). Under A2 and the intersection, by contrast, the bystander tier shows a speed-up (HR = 1.73 and 2.19). The extra issues A1 counts are mostly technical threads with an AI-governance event inside, which close at nearly the pace of the technical problem. Dating does not explain it (Online Resource 1, Table S4): 121 of A1's 185 bystander

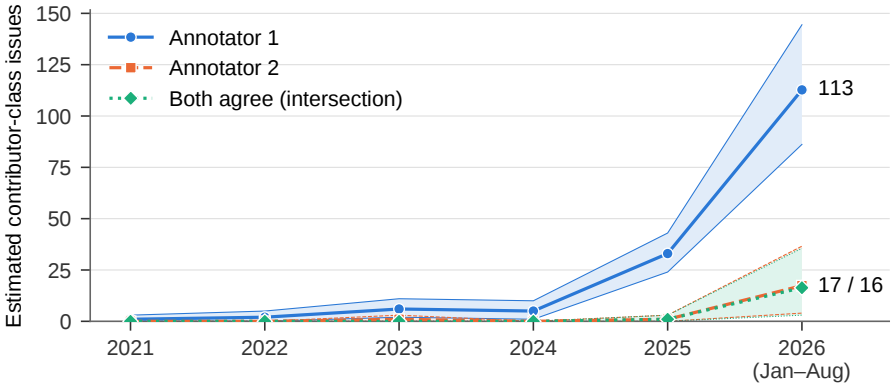


Fig. 3 AI-as-contributor issues in the six comparison repositories: design-weighted population estimates with stratified-bootstrap 95% CIs (shaded) under annotator 1 (circles, solid), annotator 2 (squares, dashed), and their intersection (diamonds, dotted); 2026 covers January–August

AI-relevant issues show AI evidence only in comments (10 only after closure), but dating exposure from the first AI-keyword comment gives an A1 bystander HR of 1.20 [0.96, 1.50], and keeping the 64 with AI evidence in the title or body gives 0.92 [0.65, 1.30]; the A2 and intersection speed-ups remain (2.04–2.77), and the AI-feature association persists under all three label sets (HRs 1.42–1.51).

4.7 The rise of AI-as-contributor discussion

Contributor-class issues in the six comparison repositories grow sharply in 2026 under every label set (Fig. 3). Under A1 the design-weighted estimate rises from 5.0 [1.0, 10.0] in 2024 to 33.0 [24.0, 43.0] in 2025 and 112.7 [86.2, 144.4] in January–August 2026; under A2 and the intersection, at most one per year before 2026 and 17.4 [4.0, 36.6] and 16.4 [3.0, 35.6] in 2026. The 2025 and 2026 intervals are disjoint under A1 and A2 and meet at their endpoints under the intersection, so the 2026 acceleration is robust to annotator; its magnitude—roughly 17 or 113 issues in eight months—is not. Dating issues by first AI evidence changes little (Table S5).

5 Discussion

Issue-tracker discourse differs across repository contexts. In these seven repositories, GitHub Issues capture operational AI concerns where AI is a feature and relatively more governance discourse where it is a bystander. Mining one class of repository, AI-specialist [2, 24] or AI-feature [31], samples one register of the conversation.

What counts as "discussing AI" should be decided explicitly. Many contributor-class disagreements involved AI evidence appearing only in comments, consistent with different subject-level and thread-level interpretations; the resulting prevalence estimates differ by an order of magnitude. Mining studies of AI discussion should state which unit they code, report reliability per stratum (our pooled

$\kappa = 0.67$ conceals per-repository values from 0.04 to 0.85), and report a prevalence-robust coefficient alongside κ. Agreement statistics for LLM-assisted coding partly measure agreement between models, so studies should report the models, the review rule, and ideally agreement with blinded, unaided coders.

Implications. Prevalence should be compared with and without ambiguous keywords, classifiers should be validated separately in each target repository, closure times of classes that emerged at different times need survival analysis, and issue-count time series should model changes in where issues are filed.

6 Threats to validity

Construct validity. Keyword retrieval misses discussion that never names an AI tool, and our list predates terms such as "AI slop" and tools such as Codex and Devin, so contributor-class counts are likely undercounts. Deleted AI-spam issues may be missing disproportionately from earlier years, exaggerating the apparent 2026 growth. The construct "issue about generative AI" admits a subject-level and a thread-level reading that our field guide did not separate; the label sets may partly reflect these readings, which keyword location only approximates. **Internal validity.** Agreement is moderate on themes and near zero in two bystander repositories, and A1's labels shifted between rounds. Both label sets come from LLM pipelines with selective human review whose model outputs were not logged, so human and model judgments cannot be separated, and many final labels are likely unreviewed model proposals. Two of the three models share a vendor, which may inflate agreement. No blinded human audit validates either pipeline, so their agreement, including the intersection, does not establish accuracy. Bot text could also inflate the keyword series; the classifier-filtered shift reduces this concern. Before July 2025 many Copilot Chat issues were filed outside the VS Code repository, so the corpus understates earlier discussion of its AI features and changes composition over time. **External validity.** Seven large, English-dominant repositories cannot represent all of OSS. VS Code and Home Assistant are the sole representatives of their tiers, so integration level cannot be separated from repository-specific characteristics. **Conclusion validity.** Sixty-seven months limit time-series power, and A2 and intersection bystander statistics rest on small counts.

7 Conclusion

Where AI features are prominent, developers discuss operating them, and both annotation pipelines agree. Where AI is a bystander, GitHub Issues host proportionally more discussion of AI-generated contributions, which grew sharply in 2026, though its prevalence depends strongly on the annotation pipeline. Issue volume rose in early 2025, but no single agent-mode release accounts for it, and the largest jump coincided with the consolidation of Copilot Chat issue tracking. AI-related issues closed faster where AI is a feature. Future work includes a blinded human audit, stratified by repository and agreement pattern, and a codebook separating subject- from thread-level events.

Supplementary information. Online Resource 1 contains Tables S1–S7.

Acknowledgements. This article extends our paper presented at SNPD 2026 [31].

Statements and Declarations

- **Funding:** The authors received no funding for the research reported in this article. Article processing charges, if incurred, will be paid by the authors' laboratory.
- **Competing interests:** The authors declare no competing interests.
- **Ethics approval and consent to participate:** Not applicable. The study analyzes public GitHub issues and reports no usernames or other personal data.
- **Consent for publication:** Not applicable.
- **Data availability:** The label sheets, the field guide, the sampling manifest, the disagreement classification, and derived tables are at https://doi.org/10.5281/zenodo.23014222; the sampled issues are public on GitHub.
- **Materials availability:** Not applicable.
- **Code availability:** The collection and analysis scripts are in the same archive.
- **Author contributions:** Panida Rumriankit: conceptualization, methodology, data curation, formal analysis, investigation, visualization, writing–original draft, and writing–review and editing. Akito Monden: critical review and revision of the manuscript. Peerapas Chinnapatpakdeekul: investigation (second annotation). All three authors reviewed and approved the final manuscript.
- **Use of large language models:** LLM-assisted labeling is documented in Section 3.2. Claude also helped draft and restructure the manuscript and write robustness-check scripts. The authors reviewed and revised the text, verified the analyses, claims, and references, and take full responsibility for the article.

References

[1] Ahmed T, Devanbu P, Treude C, et al (2025) Can LLMs replace manual annotation of software engineering artifacts? In: Proceedings of the 22nd International Conference on Mining Software Repositories (MSR), pp 526–538, https://doi.org/10.1109/MSR66628.2025.00086

[2] Asgari A, Panichella A, Derakhshanfar P, et al (2025) What challenges do developers face in AI agent systems? An empirical study on Stack Overflow and GitHub issues. arXiv:2510.25423, https://doi.org/10.48550/arXiv.2510.25423

[3] Barua A, Thomas SW, Hassan AE (2014) What are developers talking about? An analysis of topics and trends in Stack Overflow. Empirical Software Engineering 19(3):619–654. https://doi.org/10.1007/s10664-012-9231-y

[4] Benjamini Y, Hochberg Y (1995) Controlling the false discovery rate: A practical and powerful approach to multiple testing. Journal of the Royal Statistical Society: Series B 57(1):289–300. https://doi.org/10.1111/j.2517-6161.1995.tb02031.x

[5] Blei DM, Ng AY, Jordan MI (2003) Latent Dirichlet allocation. Journal of Machine Learning Research 3:993–1022

[6] Cliff N (1993) Dominance statistics: Ordinal analyses to answer ordinal questions. Psychological Bulletin 114(3):494–509. https://doi.org/10.1037/0033-2909.114.3.494

[7] Cohen J (1960) A coefficient of agreement for nominal scales. Educational and Psychological Measurement 20(1):37–46. https://doi.org/10.1177/001316446002000104

[8] Cox DR (1972) Regression models and life-tables. Journal of the Royal Statistical Society: Series B 34(2):187–220. https://doi.org/10.1111/j.2517-6161.1972.tb00899.x

[9] Ehsani R, Pathak S, Rawal S, et al (2026) 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 (MSR). ACM, pp 807–811, https://doi.org/10.1145/3793302.3793579

[10] Feinstein AR, Cicchetti DV (1990) High agreement but low kappa: I. The problems of two paradoxes. Journal of Clinical Epidemiology 43(6):543–549. https://doi.org/10.1016/0895-4356(90)90158-L

[11] Fu Y, Liang P, Tahir A, et al (2025) Security weaknesses of Copilot-generated code in GitHub projects: An empirical study. ACM Transactions on Software Engineering and Methodology https://doi.org/10.1145/3716848

[12] Gilardi F, Alizadeh M, Kubli M (2023) ChatGPT outperforms crowd workers for text-annotation tasks. Proceedings of the National Academy of Sciences 120(30):e2305016120. https://doi.org/10.1073/pnas.2305016120

[13] GitClear (2025) AI copilot code quality: 2025 data. Industry report, GitClear

[14] GitHub (2023) GitHub Copilot Chat beta now available for all individuals. URL https://github.blog/news-insights/product-news/github-copilot-chat-beta-now-available-for-all-individuals/

[15] GitHub (2025) GitHub Copilot: The agent awakens. URL https://github.blog/news-insights/product-news/github-copilot-the-agent-awakens/

[16] Grootendorst M (2022) BERTopic: Neural topic modeling with a class-based TF-IDF procedure. arXiv:2203.05794, https://doi.org/10.48550/arXiv.2203.05794

[17] Gwet KL (2008) Computing inter-rater reliability and its variance in the presence of high agreement. British Journal of Mathematical and Statistical Psychology 61(1):29–48. https://doi.org/10.1348/000711006X126600

[18] Han J, Shihab E, Wan Z, et al (2020) What do programmers discuss about deep learning frameworks. Empirical Software Engineering 25(4):2694–2747. https://doi.org/10.1007/s10664-020-09819-6

[19] Hora A, Robbes R (2026) AI policy, disclosure, and human in the loop: How are contribution guidelines adapting to GenAI? ICSME 2026, Visions and Emerging Results Track; arXiv:2605.16706, https://doi.org/10.48550/arXiv.2605.16706

[20] Horvitz DG, Thompson DJ (1952) A generalization of sampling without replacement from a finite universe. Journal of the American Statistical Association 47(260):663–685. https://doi.org/10.1080/01621459.1952.10483446

[21] Kaplan EL, Meier P (1958) Nonparametric estimation from incomplete observations. Journal of the American Statistical Association 53(282):457–481. https://doi.org/10.1080/01621459.1958.10501452

[22] Krippendorff K (2004) Content Analysis: An Introduction to Its Methodology, 2nd edn. Sage, Thousand Oaks, CA

[23] Landis JR, Koch GG (1977) The measurement of observer agreement for categorical data. Biometrics 33(1):159–174. https://doi.org/10.2307/2529310

[24] Lin Z, Zhang N, Liu C, et al (2024) An empirical study of ChatGPT-related projects and their issues on GitHub. arXiv:2403.17437, https://doi.org/10.48550/arXiv.2403.17437

[25] Liu F, et al (2024) Beyond functional correctness: Exploring hallucinations in LLM-generated code. arXiv:2404.00971, https://doi.org/10.48550/arXiv.2404.00971

[26] Mann HB (1945) Nonparametric tests against trend. Econometrica 13(3):245–259. https://doi.org/10.2307/1907187

[27] McNemar Q (1947) Note on the sampling error of the difference between correlated proportions or percentages. Psychometrika 12(2):153–157. https://doi.org/10.1007/BF02295996

[28] Newey WK, West KD (1987) A simple, positive semi-definite, heteroskedasticity and autocorrelation consistent covariance matrix. Econometrica 55(3):703–708. https://doi.org/10.2307/1913610

[29] OpenAI (2022) Introducing ChatGPT. URL https://openai.com/index/chatgpt/

[30] Peng S, Kalliamvakou E, Cihon P, et al (2023) The impact of AI on developer productivity: Evidence from GitHub Copilot. arXiv:2302.06590, https://doi.org/10.48550/arXiv.2302.06590

[31] Rumriankit P, Monden A, Inayoshi H, et al (2026) How developers discuss generative AI: A longitudinal study of the Visual Studio Code community. In: Proceedings of the IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), in press

[32] Sen PK (1968) Estimates of the regression coefficient based on Kendall's tau. Journal of the American Statistical Association 63(324):1379–1389. https://doi.org/10.1080/01621459.1968.10480934

[33] Silva CC, Galster M, Gilson F (2021) Topic modeling in software engineering research. Empirical Software Engineering 26:120. https://doi.org/10.1007/s10664-021-10026-0

[34] Song F, Agarwal A, Wen W (2024) The impact of generative AI on collaborative open-source software development: Evidence from GitHub Copilot. arXiv:2410.02091, https://doi.org/10.48550/arXiv.2410.02091

[35] Treude C, Wagner M (2019) Predicting good configurations for GitHub and Stack Overflow topic models. In: Proceedings of the 16th International Conference on Mining Software Repositories (MSR), pp 84–95, https://doi.org/10.1109/MSR.2019.00022

[36] Tufano R, Pepe F, Zampetti F, et al (2026) Developers and generative AI: A study of self-admitted usage in open source projects. Empirical Software Engineering 31(4). https://doi.org/10.1007/s10664-026-10848-w

[37] Visual Studio Code (2025) February 2025 (version 1.98). URL https://code.visualstudio.com/updates/v1_98

[38] Visual Studio Code (2025) Open source AI editor: First milestone. URL https://code.visualstudio.com/blogs/2025/06/30/openSourceAIEditorFirstMilestone

[39] Wilson EB (1927) Probable inference, the law of succession, and statistical inference. Journal of the American Statistical Association 22(158):209–212. https://doi.org/10.1080/01621459.1927.10502953

[40] Xu W, Gao K, He H, et al (2025) LiCoEval: Evaluating LLMs on license compliance in code generation. In: Proceedings of the 47th International Conference on Software Engineering (ICSE), pp 1665–1677, https://doi.org/10.1109/ICSE55347.2025.00052

[41] Yang Z, Wang C, Shi J, et al (2023) What do users ask in open-source AI repositories? An empirical study of GitHub issues. In: Proceedings of the 20th International Conference on Mining Software Repositories (MSR), pp 79–91, https://doi.org/10.1109/MSR59073.2023.00024

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- ESM1.pdf