From Human-Centric to Agentic Code Review: The Impact of Different Generations of Generative AI Technology on Review Quality

Suzhen Zhong, Shayan Noei, Bram Adams, Ying Zou

***Abstract*—Code review helps maintain software quality before code integration, but it also imposes a substantial workload on human reviewers. As generative artificial intelligence becomes part of software development, code review is shifting from a primarily human review process toward AI-supported review processes in which large language model (LLM) reviewers and AI agent reviewers participate alongside human reviewers. However, we still lack empirical evidence on how this transition affects review efficiency and review quality. In this paper, we study 1.02 million reviewed pull requests from 207 GitHub projects that transition across three code review eras: human-centric review, LLM-assisted review, and agentic code review. We identify three AI reviewer adoption practices: Gradual AI Adoption, Rapid LLM Adoption, and Rapid AI Agent Adoption. We further model pull request review discussions as reviewer interaction sequences to characterize how human, LLM, and AI agent reviewers collaborate during the review process. Our results show that agent-involved collaboration patterns, especially reviews initiated by AI agents or involving multiple AI agents, are associated with faster review decisions under Gradual AI Adoption and Rapid AI Agent Adoption. However, these efficiency gains do not translate into better review quality. We also find that review activity and pull request type remain important across eras, while human-AI collaboration patterns become the strongest explanatory factor for review efficiency once LLM and AI agent reviewers participate. These findings provide empirical guidance for designing AI-supported code review processes that improve efficiency without weakening review quality.**

***Index Terms*—Agentic Code Review, Modern Code Review**

I. Introduction

Generative AI has been rapidly integrated into software development. For example, AI-generated code now accounts for 75% of new code at Google [1]. GitHub reports that developers pushed nearly one billion commits in 2025, while public repositories using an LLM SDK grew by 178% year over year [2]. As AI-generated code increases the volume of code changes, code review and code integration are becoming downstream bottlenecks. Recent blog posts from Google [3] and Amazon [4] warn that faster code generation can overwhelm human reviewers and delivery pipelines.

With this rapid progress in generative AI, developers are increasingly adopting generative AI to write and review code and automate software engineering tasks [5], [6]. Initially (Pre-LLM era), human reviewers manually provided review feedback, potentially assisted by bots or older machine learning (ML) tools. Later (LLM era), early LLM reviewers assisted developers in generating natural-language feedback from code changes [7]. More recently (agent era), AI agent reviewers have extended the capabilities of LLMs by autonomously retrieving project context, running development tools, and verifying findings before producing feedback [8]. Across these three eras, code review shifts from a human-centric process to a human-AI collaboration review process in which human reviewers, LLM reviewers, and AI agent reviewers may need to collaborate to complete the code review.

Prior work finds that frequent human reviews of small code changes support efficient review decisions [9], [10], while reviewer workload and patch characteristics influence review time [11], [12]. Other studies have examined review smells [13], showing that assigning unsuitable reviewers or reviewing pull requests with too many code changes at once can reduce review quality. Recent studies find that LLM-assisted review may increase review time [14]. Thus far, existing studies provide limited evidence on how code review practices and review quality evolve as teams transition from human-centric review (without generative AI participation) to LLM-assisted and agentic review. Therefore, it remains unclear how projects utilize AI reviewers over time, how human, LLM, and AI agent reviewers collaborate throughout the review process, and which practices and factors are associated with review quality.

To answer these questions, we analyze 1.02 million pull requests (PR) from 207 open-source projects whose code review practices have evolved from the human-centric to the LLM-assisted and agentic review eras. We aim to answer the following research questions (RQs):

RQ1. What are the common AI adoption practices and how do they relate to review quality? Although LLM and AI agent reviewers are increasingly used in code review, it is unclear how projects adopt these reviewers and whether different adoption practices affect review quality. Across the 207 studied projects, we identify three AI reviewer adoption practices: (1) Gradual AI Adoption, (2) Rapid LLM Adoption, and (3) Rapid AI Agent Adoption. By comparing review efficiency and review smells across the three review eras, we find that the *Rapid LLM Adoption* is significantly associated with an increase in code review smell prevalence. However, *Gradual AI Adoption* and *Rapid AI Agent Adoption* practices

Suzhen Zhong, Shayan Noei, and Ying Zou are with the Department of Electrical and Computer Engineering, Queen's University, Kingston, ON K7L 3N6, Canada. E-mail: {suzhen.zhong, s.noei, ying.zou}@queensu.ca.

Bram Adams is with the Maintenance, Construction and Intelligence of Software Lab (MCIS), School of Computing, Queen's University, Kingston, ON K7L 3N6, Canada. E-mail: bram.adams@queensu.ca.

show significantly more efficient review in the agent era.

RQ2. What is the impact of human-AI collaboration patterns on code review quality? To identify which human-AI review collaboration patterns are associated with better review quality under different AI adoption practices, we examine the interactions among human, LLM, and AI agent reviewers within individual pull requests. We find that no human-AI collaboration pattern consistently outperforms human-only review in both efficiency and quality. In the agent era, reviews initiated by AI agents or involving multiple AI agents are significantly more efficient than human-only reviews in *Gradual AI Adoption* and *Rapid AI Agent Adoption*. However, most collaboration patterns involving LLM or AI agent reviewers show significantly higher review quality risk than human-only review.

RQ3. How do human-AI collaboration patterns relate to traditional factors impacting review quality? To better understand the factors explaining the code review quality issues identified in RQ2, we build explanatory logistic regression models to assess the important factors related to review quality. We find that once LLM and AI agent reviewers join, human-AI collaboration patterns emerge as a strong explanatory factor alongside traditional review factors, especially for Review Buddies, a smell where repeated reliance on the same reviewers may narrow review perspectives. Meanwhile, agent-involved collaboration patterns under the Gradual AI Adoption and Rapid AI Agent Adoption practices are consistently associated with larger changesets, indicating that these collaboration patterns are more frequently used to review larger code changes.

Our work makes the following contributions:

- We provide a large-scale empirical study of 1.02 million pull requests from 207 open-source GitHub projects across the transitions from human-centric review to LLM-assisted and agentic review. We also release a replication package [15] containing the longitudinal dataset to support future research on AI reviewer adoption.
- We characterize the adoption of LLM and AI agent reviewers in projects and quantify the association between the adoption practices and review quality, giving practitioners evidence to guide their adoption of AI reviewers.
- We model review quality across the three generative AI review eras, jointly considering human-AI collaboration patterns and the traditional factors (e.g., pull request characteristics, review activity). We show that human-AI collaboration patterns and these traditional review factors both remain important for explaining review quality.
- We present the first large-scale quantitative studies of agentic code review, analyzing how AI reviewer adoption evolves across the Pre-LLM, LLM, and Agent eras.

II. Case Study Setup

This section details the case study setup. We describe our data collection and analysis approaches. The detailed approaches for each RQ are presented in the Sec. III-A–III-C.

An overview of our study is shown in Fig. 1. Starting from 2,490 candidate GitHub projects with continuous review activity, we retain 207 with sufficient reviewed pull requests across the pre-LLM, LLM, and agent eras. We cluster the generative AI adoption series for each project into common AI reviewer adoption practices. Within each practice, we examine the pull request types (e.g., bug fixing) that increasingly involve AI reviewers and investigate the changes in review quality across the three eras. To understand the collaboration among human, LLM, and AI agent reviewers, we identify common human-AI collaboration patterns, and compare the patterns by review efficiency and quality. Finally, we build explanatory models to assess the relationship between review quality and the human-AI collaboration patterns, alongside traditional review-process factors (e.g., pull request characteristics).

A. Project Selection

To capture code review across the transition from human-centric review to LLM-assisted and agentic review, we select active GitHub projects with continuous review activity before and after the emergence of generative AI reviewers. Using the GitHub advanced search [16], we identify initial candidate projects with continuous review activity from May 2022 to February 2026. We retain projects that meet the criteria below:

- at least 100 stars [17], to ensure that a project has sufficient community adoption and is actively maintained;
- created before May 2022, providing at least six months of review history before the public release of ChatGPT in November 2022 [18]; and
- at least one reviewed pull request per month from May 2022 to February 2026, ensuring continuous review activity throughout the observation window.

The criteria yield 2,490 candidate projects, with continuous activity from May 2022 through February 2026.

B. Data Collection and Preprocessing

For each of the 2,490 candidate projects, we use the GitHub REST API [19] to collect their reviewed pull requests and associated review conversations, including review events, reviewer identities, timestamps, pull request decisions after review (i.e., accepted if merged or rejected if closed without merge), and review comments from each reviewer.

Labeling reviewers. To distinguish human reviewers from automated reviewers, we label each reviewer account with the reviewer type represented by the tool behind the account at the time of review. We first use the GitHub REST API [19] to distinguish human and bot accounts. For bot accounts, we manually inspect the official web documentation of the corresponding tool to determine whether the account represents a rule-based bot (e.g., GitHub Actions [20]), a traditional machine-learning review tool (e.g., Amazon CodeGuru Reviewer [21]), an LLM reviewer (e.g., LlamaPReview [22]), or an AI agent reviewer (e.g., Claude Code [23]). To validate that the AI agent reviewer labels reflect agentic review behavior, we manually inspected a statistically representative sample of 384 agent-era pull requests involving AI agent reviewers, corresponding to a 95% confidence level and a 5% margin of error [24]. Among these sampled pull requests, 360 contained evidence of agentic behavior, such as planning, retrieval of

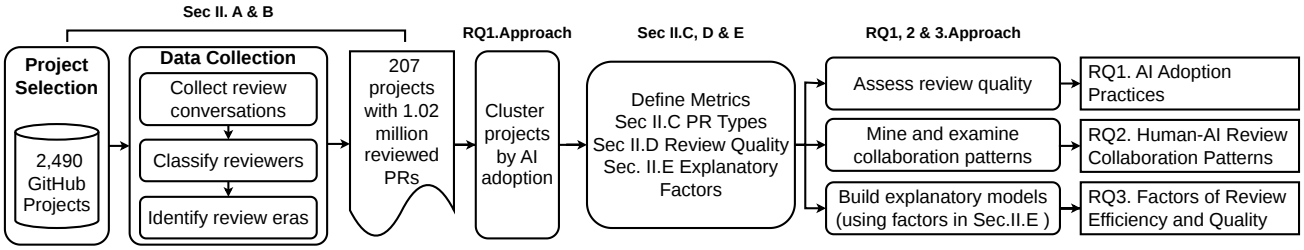

Fig. 1: Overview of our approach.

TABLE I: Pull request (PR) categories adopted from Li et al. [5].

Category	Description
Chore	Routine maintenance or configuration tasks.
Fix	Corrects bugs.
CI	Updates CI/CD pipelines or workflow configurations.
Feature	Adds or implements new functionality.
Performance	Tries to improve speed or efficiency.
Refactor	Restructures existing code without changing behavior.
Documentation	Updates or adds documentation.
Style	Adjusts formatting or naming conventions.
Test	Adds or modifies test cases.
Build	Modifies build process or dependencies.
Other	Unclassified or mixed-purpose changes.

repository context, or commit actions. For the remaining 24 sampled cases with only natural-language comments, we checked project documentation and configuration files and found no evidence that agentic review was disabled. Since official documentation identifies these reviewers as AI agents, we label these cases as involving AI agent reviewers.

After assigning reviewer labels, we define review eras for each project based on the first participation of each LLM or AI agent reviewer. For each project, its **pre-LLM era** includes all reviewed pull requests before any generative AI reviewer participates. Its **LLM era** starts with the first reviewed pull request involving an LLM reviewer and continues until the first reviewed pull request involving an AI agent reviewer. The project's **agent era** starts with the first reviewed pull request involving an AI agent reviewer. Because review eras are defined separately for each project, each era's start time and end time vary across projects. After defining the review eras, we retain only projects with more than 400 reviewed pull requests in each era to reduce the influence of one-off or experimental AI use and to ensure sufficient review activity for within-project statistical comparisons [24]. Finally, we obtain 1.02 million reviewed pull requests from 207 projects.

C. Pull Request Types

As shown in Fig. 1, all three RQs use pull request type as a shared measure. We therefore describe here how each pull request is assigned a type.

Watanabe et al. [25] proposed 11 pull request types for characterizing code-change purpose, with the full taxonomy shown in Table I. Li et al. [5] classify pull requests using this taxonomy by analyzing their titles and descriptions with GPT-4.1-mini. We apply the same PR-level classification process to assign each pull request a type. To validate the reliability of the classifications produced by GPT-4.1-mini, we manually label a statistically representative sample of 384 pull requests with 95% confidence level and 5% margin of error [24]. Comparing manual labels against LLM classifications yields a Cohen's κ [26] of 0.91, indicating almost perfect agreement between the human and GPT-4.1-mini classifications.

TABLE II: Code review smell taxonomy and detection rules adapted from Doğan and Tüzün [13].

Smell	Potential Side Effect	Detection Rule
Sleeping Review	Delayed decisions may cause context loss or block downstream work.	Time from PR creation to final decision exceeds two days.
Review Buddies	Repeated reliance on the same reviewers may narrow review perspectives.	Same reviewer (human, LLM, or AI agent) reviews at least 50% of an author's PRs.
Large Changeset	Large changes may reduce reviewer focus and feedback effectiveness.	Code churn exceeds 500 changed lines of code.
Ping-Pong	Excessive back-and-forth may indicate inefficient communication or rework.	PR has more than three change-request iterations from human or AI reviewers.
Missing Context	Missing context may weaken reviewer understanding and feedback usefulness.	PR description is empty or lacks linked issue/context information.
Lack of Review	Missing review may let defects or maintainability issues enter the codebase.	No human, LLM, or AI agent reviewer other than the author participates.

D. Review Quality Evaluation

As projects shift from human-centric review toward agentic review, we examine whether these transitions are associated with changes in review efficiency and review quality risks. As shown in Fig. 1, review efficiency and review smells are the shared review-quality measures used across all three RQs.

1) Review Efficiency: To assess changes in user-visible review efficiency, we follow prior work [27] and measure the number of days from the creation of a pull request to the final review decision. Since larger changes may require more review effort, we normalize the review duration by the size of the code change in thousands of lines of code (KLOC) [27]:

$$\text{Review Efficiency} = \frac{T_{\text{decision}} - T_{\text{creation}}}{\#\text{ Thousand lines of code}} \quad (1)$$

where $T_{creation}$ is the timestamp of the creation of the pull request and $T_{decision}$ is the timestamp of the final decision.

2) Review Smells: Review smells capture observable anti-patterns in the review process that may reduce review effectiveness and introduce quality risks [28]. For example, low review participation is associated with more post-release

TABLE III: Independent explanatory variables used to model how reviewer collaboration patterns and alternative factors are jointly associated with review efficiency and review smells. The Abbrev. column shows the shortened metric names used in Table VI.

Metric	Abbrev.	Category	Source	Description
Review era	Era	Review era	This study	The three review eras (i.e., pre-LLM, LLM, and agent eras)
PR type	PR type	PR Characteristics	[5], [25]	Purpose of the pull request, such as Feature, Fix, and Chore (Table I)
Initial code churn	Churn	PR Characteristics	[29], [12]	Number of added and deleted lines when the pull request is opened
Initial files changed	Files	PR Characteristics	[29], [12]	Number of files changed when the pull request is opened
Initial commit count	InitCommit	PR Characteristics	[30]	Number of commits present when the pull request is created
Commit count	NumCommit	Review activity	[30]	Total number of commits in the PR when the review process ends
Inline comments	InlineThread	Review activity	[31]	Number of reviewer discussion threads on specific changed lines
Reviewer count	ReviewerCount	Review activity	[12]	Number of unique reviewers of the pull request
Author experience	Author Exp	Experience	[30], [29]	Number of prior commits by the author in the same project
Human reviewer experience	Human Exp	Experience	[12], [11]	Mean prior reviews by the PR's human reviewers in the same project
Bot reviewer experience	Rule Exp	Experience	This study	Mean prior reviews by the PR's rule-based bot in the same project
ML reviewer experience	ML Exp	Experience	This study	Mean prior reviews by the PR's ML reviewers in the same project
LLM reviewer experience	LLM Exp	Experience	This study	Mean prior reviews by the PR's LLM reviewers in the same project
Agent reviewer experience	Agent Exp	Experience	This study	Mean prior reviews by the PR's AI agent reviewers in the same project

*ML stands for traditional Machine Learning, which is not generative AI.

defects [29], and large code changes tend to receive less useful review feedback [30]. Although review smells were originally defined for human code review, they remain relevant in human-AI review because AI participation does not remove core review-process risks, such as missing context, repeated back-and-forth, and reliance on narrow feedback sources. We therefore use review smells as indicators of review-process quality risk in both human-only and human-AI reviews. Following Doğan and Tüzün [13], who define code review smells and detection rules from code review histories, we adapt six of their smells that can be operationalized using our pull request review conversations, then apply these rules to each reviewed pull request to identify the review smells, as shown in Table II.

E. Explanatory Factors for Review Efficiency and Quality

As shown in Fig. 1, we show the explanatory factors used in the RQ3 analysis. Here, we introduces traditional review factors from prior code review studies, while Section III-C shows the modeling procedure. Prior work on code review with human reviewers shows that review quality is associated with review process factors, including pull request characteristics [5], review activity [30], and reviewer experience [29]. Following this work, we adopt pull request characteristics, review activity, and participant experience as our explanatory factors, listed in Table III and grouped as follows:

1) Pull Request Characteristics: Pull request characteristics capture the purpose and initial scope of a code change before review begins, including pull request type, initial code churn, initial files changed, and initial commit count (Table III).

2) Review activity: The review activity in the pull request indicates the amount of revision and discussion during the review process. We measure it using commit count, unique reviewer count, and the number of reviewer discussion threads opened on specific code changes.

3) Participant experience: Participant experience captures the pull request author and reviewers' familiarity with the project before the current review. As projects now adopt LLM and AI agent reviewers, we also include reviewer type (e.g., LLM reviewer) as a factor that could explain review quality.

III. Results

A. RQ1: What are the common AI adoption practices and how do they relate to review quality?

Motivation. With the rapid adoption of AI in software engineering [32], generative AI reviewers now participate alongside reviewers across millions of open-source projects [5]. Yet, it remains unclear how AI reviewers are introduced and used in software projects over time and how their adoption influences the code review quality. In this research question, we aim to identify common AI reviewer adoption practices and measure their effects on review quality. Our findings provide guidance for projects considering the adoption of AI reviewers.

Approach. **Identifying AI Adoption Practices.** To capture the adoption of LLM and AI agent reviewers in each project, we first construct a time series of the proportion of pull requests reviewed by each reviewer type (e.g., human or AI agent reviewer). Furthermore, similar to prior work [33], to reduce noise caused by AI reviewer participation varying from one pull request to the next, we aggregate the monthly proportion of pull requests reviewed by the era-specific generative AI reviewer type using *era-specific AI reviewer rate* ($P_{m,e}$), which is defined as follows:

$$P_{m,e} = \begin{cases} 0 \quad \text{(no GenAI reviewer)}, & e = \text{Pre-LLM}, \\ \left(N_m^{\text{LLM}}/N_m\right) \times 100\%, & e = \text{LLM}, \\ \left(N_m^{\text{Agent}}/N_m\right) \times 100\%, & e = \text{Agent}. \end{cases} \tag{2}$$

where N_m is the total number of pull requests in month m, N_m^{LLM} and N_m^{Agent} are the numbers of pull requests in month m involving LLM and AI agent reviewers, respectively.

To make the AI adoption time series comparable across projects, we use linear interpolation to resample the monthly $P_{m,e}$ values in each era into 10 evenly spaced points, following prior work on time-series standardization [34], [35]. We then concatenate the three eras' normalized segments in chronological order to form the final AI adoption time series. We cluster the resulting time series using soft-DTW clustering [36], [37], [35]. Soft-DTW allows time series with similar overall trajectories to be grouped even when corresponding changes

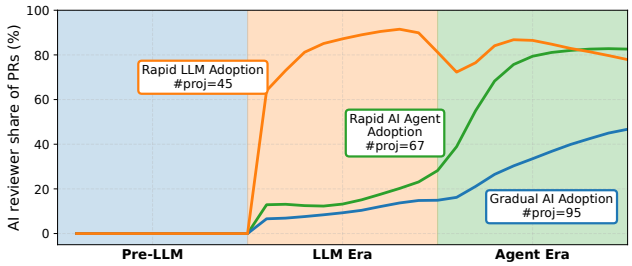

Fig. 2: Three identified AI reviewer adoption practices as code review shifts from human-centric to LLM-assisted and agentic review. Higher trend lines indicate greater AI reviewer participation.

occur at slightly different time points. To select the number of clusters, we calculate the silhouette score using the same soft-DTW dissimilarity [38] for different numbers of clusters. The silhouette score ranges from -1 to 1, with a higher value indicating better-separated clusters. We find three as the optimal number of clusters, with a silhouette score of 0.40, suggesting fair separation [39].

Quantifying Pull Request Types. To examine whether LLM and AI agent reviewers are used differently across pull request types, we use the *relative participation rate* metric. The relative participation rate is computed as follows:

$$\text{Relative participation rate}_t = \frac{N_t^{\text{AI}} / \left(N_t^{\text{AI}} + N_t^{\text{non-AI}}\right)}{N_{\text{other}}^{\text{AI}} / \left(N_{\text{other}}^{\text{AI}} + N_{\text{other}}^{\text{non-AI}}\right)} \quad (3)$$

where N_t^{AI} and $N_t^{non\text{-}AI}$ denote pull requests of type t reviewed with and without an LLM or AI agent reviewer, respectively. N_{other}^{AI} and $N_{other}^{non\text{-}AI}$ denote pull requests of all other types with and without these reviewers.

This metric compares the AI reviewer participation rate for type-t pull requests against that for all other types. We then use a chi-square test [40], which measures the associations between categorical variables, to determine whether AI reviewer participation differs significantly across pull request types.

Comparing Review Quality. To examine differences in review quality across eras within each AI adoption practice, we compare the review efficiency and review smell metrics defined in Section II-D. Specifically, the pre-LLM era captures code review practices before generative AI reviewers are adopted, so we use it as the baseline for comparing review efficiency and the prevalence of each review smell in the LLM and agent eras. Since the metric values do not follow a normal distribution, we use the Wilcoxon signed-rank test with Bonferroni correction [40] for significance testing.

***Results.* We identify three AI adoption practices: Gradual AI Adoption (46% of 207 studied projects), Rapid LLM Adoption (22%), and Rapid AI Agent Adoption (32%).** Fig. 2 illustrates the three identified AI reviewer adoption practices. Below, we describe each practice.

- *Gradual AI Adoption* shows a steady transition from human-centric review to generative AI reviewer participation. LLM reviewers participate in only 8% of pull requests on average in the LLM era, whereas AI agent reviewers participate in 36% of pull requests in the agent era. The projects span diverse domains, from operating systems (e.g., RT-Thread [41]) to educational platforms (e.g., Archive's OpenLibrary [42]), with no single domain dominating.
- *Rapid LLM Adoption* captures projects that adopt generative AI reviewers early, with high LLM reviewer participation (91% of pull requests on average) in the LLM era followed by high AI agent reviewer participation (93% of pull requests on average) in the agent era. In this practice, web applications make up a larger proportion of projects, accounting for 12 of the 45 (27%).
- *Rapid AI Agent Adoption* shows limited LLM reviewer participation in the LLM era, followed by rapid AI agent reviewer uptake in the agent era. In this practice, LLM reviewers participate in 19% of pull requests on average during the LLM era, followed by AI agent reviewer participation in 76% of pull requests in the agent era. This practice contains projects from large companies such as Microsoft [43] (12 projects) and Google [44] (3 projects).

TABLE IV: Review quality and PR types across review eras. We report the smells whose change is statistically significant; smell results are available in the replication package [15]

	Era	W/AI	Eff.	Smell%	RevBud%	PR Types
Gradual AI	PreLLM	—	8.5	83.8	47.8	—
	LLM	8%	-1.6	+3.8	+15.1	chore↑ fix↑
	Agent	36%	-2.5*	+2.0	+15.6	feat↑ refac↑ fix↑
Rapid LLM	PreLLM	—	5.4	83.6	45.9	—
	LLM	91%	+0.6	+8.0*	+26.0*	docs↑ build↑ perf↑ style↑
	Agent	93%	-0.1	+4.4*	+23.2*	perf↑
Rapid Agent	PreLLM	—	10.4	81.0	36.4	—
	LLM	19%	-3.4	+4.4	+16.6*	perf↑ refac↑ fix↑ test↑ feat↑
	Agent	76%	-4.5*	+2.5	+12.4	—

Gradual AI Adoption and Rapid AI Agent Adoption show significantly more efficient reviews in the agent era. Table IV shows the review efficiency and review smell rate of each practice per era. From the pre-LLM to the agent era, review time drops by 2.5 days/KLOC for *Gradual AI Adoption*, and by 4.5 days/KLOC for *Rapid AI Agent Adoption*. However, *Rapid LLM Adoption* shows no significant change across review eras. This suggests that early, heavy reliance on LLM reviewers does not necessarily improve review efficiency, whereas projects that gradually adopt AI reviewers or shift rapidly in the agent era show significant efficiency gains.

Rapid LLM Adoption Adoption have a significantly higher review smell rate than their pre-LLM baseline, for both the LLM and agent eras. As listed in Table IV, review smells significantly increase in *Rapid LLM Adoption*, by 8.0% points in the LLM era and 4.4% points in the agent era. This increased smell rate is especially reflected in Review Buddies (i.e., repeated assignment of the same reviewer), which rises by 26% in the LLM era and 23.2% in the agent era from the pre-LLM baseline. Other review smell types did not show significant increases in our full smell analysis. These results suggest that *Rapid LLM Adoption* may increase reliance on the same LLM reviewer accounts (e.g., LlamaPReview [22]), which narrows review perspectives and reduces feedback diversity.

In Gradual AI Adoption, LLM reviewers are utilized significantly more in reviewing maintenance (chore) and bug-fixing pull requests, whereas AI agent reviewers participate significantly more often in feature or refactoring

pull requests. For this adoption practice, LLM reviewers participate more often in chore pull requests (2.0 times the rate of other types) and fix pull requests (1.2 times) during the LLM era. In contrast, during the agent era, AI agent reviewers participate more often in feature pull requests (1.4 times) and refactoring pull requests (1.3 times). This pattern suggests that, with the gradual shift toward agentic code review, AI-reviewed tasks move from routine maintenance and bug fixing toward more context-aware work such as feature development and refactoring. In Rapid LLM Adoption, generative AI reviewer participation is spread across several pull request types in the LLM era and is only concentrated in performance pull requests in the agent era. In Rapid AI Agent Adoption, generative AI reviewer participation is concentrated in several pull request types during the LLM era but does not concentrate on any single type in the agent era.

Summary for RQ1

We identify three AI reviewer adoption practices: *Gradual AI Adoption*, *Rapid LLM Adoption*, and *Rapid AI Agent Adoption*. *Rapid LLM Adoption* significantly increases review smells, whereas *Gradual AI Adoption* and *Rapid AI Agent Adoption* significantly improve review efficiency in the agent review era.

B. RQ2: What is the impact of human-AI collaboration patterns on code review quality?

Motivation. The AI adoption practices capture how projects adopt AI reviewers across review eras. However, they do not show how human, LLM, and AI agent reviewers interact when reviewing individual pull requests. For example, an AI reviewer may initiate the review, or an AI reviewer may join the code review only after human reviewers initiate the discussion. Various interaction sequences can frequently reoccur and therefore emerge as collaboration patterns. In this research question, we aim to identify collaboration patterns and examine their impact on review quality. Understanding these collaboration patterns can guide developers in deciding when and how to involve human, LLM, and AI agent reviewers during code review.

Approach. To analyze the collaboration patterns between human and AI reviewers, we represent each pull request as an ordered sequence of review comments labeled by reviewer type (e.g., AI agent reviewer). We then group these sequences into collaboration patterns and compare their associations with review efficiency and quality across different review eras.

Identifying Collaboration Patterns. To capture reviewer participation within each pull request, we record the temporal sequence of reviewer types from pull request-level review comments and the review decision of the pull request (i.e., accepted or rejected). We order the comments by timestamp and record the reviewer type (e.g., LLM reviewer) of each comment, preserving repeated participation by the same reviewer type, and end each sequence with the pull request decision. For example, `Human` → `Agent` → `Human` → `accepted` represents a pull request reviewed first by a human reviewer, then by an AI agent reviewer, then by a human reviewer again before acceptance.

We then abstract reviewer interaction sequences into collaboration patterns using Markov chains with expectation-maximization, following prior software engineering process analyses [45], [46], [47]. A Markov chain represents one candidate collaboration pattern by estimating the probability of the first reviewer category and the transition probabilities between reviewer categories. Expectation-maximization fits multiple Markov chains to the reviewer interaction sequences and assigns each pull request sequence to the chain that best explains its reviewer transitions. To select the number of collaboration patterns, we fit models with different candidate numbers of Markov chains and compare them using the Bayesian Information Criterion (BIC) [48]. BIC is a model-selection criterion that balances goodness of fit against model complexity, with lower BIC values indicating a better-fitting model. The lowest-BIC model yields 10 collaboration patterns, which we use in the subsequent analysis.

Measuring Review Quality. To measure the quality and efficiency of AI participation in the code review process, we compare the collaboration patterns involving generative AI against human-only code review.

(1) *Review Efficiency:* Within each adoption pattern and era, we use the Scott-Knott Effect Size Difference (ESD) test [49], [50] to rank collaboration patterns by review efficiency. The Scott-Knott ESD test ranks collaboration patterns by significant differences in review efficiency, allowing us to identify the most efficient patterns within the same practice and era.

(2) *Review Smells:* We compare the prevalence of review smells between each AI-involved and human-only collaboration pattern during each combination of adoption practice and era, using chi-square tests with Bonferroni correction [40], since smell presence is categorical.

Comparing Pull Request Types. To identify whether collaboration patterns are associated with different pull request types (e.g., bug fixing), we compare the distribution of the pull request types defined in Table I between each AI-involved collaboration pattern and human-only review. Since collaboration patterns and pull request types are categorical, we use chi-square tests with Bonferroni correction [40] within each adoption practice and era.

Results. **In the LLM era, human-bot and multi-LLM collaboration patterns are as efficient as human-only review in Gradual AI Adoption and Rapid AI Agent Adoption. However, for Rapid LLM Adoption, all LLM-involved review patterns are significantly slower than human-only review.** Table V shows the Scott-Knott ESD efficiency rank for collaboration patterns within each adoption practice and review era. As listed in Table V, LLM-Assist reviews are less efficient than human-only reviews across all three adoption practices (R2–R3 vs. R1). Multi-LLM reviews are as efficient as human-only reviews under Gradual AI Adoption and Rapid AI Agent Adoption (both R1), but are less efficient under Rapid LLM Adoption (R2 vs. R1). In Multi-LLM reviews, the LLM reviewer contributes a median of two comments, and in 75% of these pull requests, human reviewers respond with only a single comment before the review ends. By contrast,

TABLE V: Prevalence of each human-AI review collaboration pattern, its review efficiency rank, review smell prevalence, and the pull request types associated with each pattern. Effi: Scott-Knott ESD rank result for review efficiency; R1 = most efficient. Smell (%): percentage of pull requests with at least one review smell. PR Type: for Human-Only rows, the top-1 common pull request types with their percentages (e.g., feat(46) = 46% are feature pull requests). For other rows, pull request types that are significantly more frequent than in Human-Only reviews within the same era.

	Collaboration Pattern	Gradual AI Adoption			Rapid LLM Adoption			Rapid Agent Adoption		
		Effi.	Smell	PR Type	Effi.	Smell	PR Type	Effi.	Smell	PR Type
Pre-LLM	Human-Only	R2	75	feat(46)	R2	72	feat(46)	R2	75	feat(43)
	Human-Bot	R2	84↑	chore↑	R1	86↑		R2	84↑	
	Human-Bot-ML	R1	88↑	chore↑ fix↑	R1	82↑	chore↑ fix↑	R1	87↑	feat↑
LLM	Human-Only	R1	76	feat(41)	R1	69	chore(32)	R1	72	feat(33)
	Human-Bot	R1	82↑	chore↑	R2	78↑	fix↑	R1	88↑	
	LLM-Assist	R2	81↑		R3	87↑	fix↑	R2	83↑	feat↑
	Multi-LLM	R1	81	chore↑ feat↑	R2	86↑	feat↑ fix↑	R1	81↑	fix↑
	LLM-Bot-Assist	R1	94↑	chore↑ refac↑	R2	80↑	docs↑ fix↑	R2	87↑	fix↑ refac↑
Agent	Human-Only	R3	75	feat(42)	R1	74	feat(34)	R2	71	feat(39)
	Human-Bot	R2	83↑	chore↑	R2	73	fix↓ docs↑	R2	84↑	
	Agent-Init	R2	78↑	chore↑	R1	84↑	fix↓ chore↑	R1	80↑	chore↑
	Agent-Assist	R3	84↑		R2	84↑	fix↓ feat↑	R2	83↑	
	Multi-Agent	R1	83↑	chore↑	R1	83↑	fix↓ docs↑	R1	82↑	fix↑
	Agent-ML-Assist	R3	86↑		R2	84↑	feat↑	R2	84↑	

in LLM-Assist reviews, humans comment first and often continue the discussion after the LLM comment, resulting in more human back-and-forth than in human-only reviews (with medians of four and two human comments, respectively; see our replication package [15]). This suggests that a single LLM comment is often insufficient to resolve a pull request by itself. Instead, reviewers still need continued human-AI back-and-forth before reaching a decision.

In the agent era, agent-init and multi-agent reviews are significantly faster than human-only review under Gradual AI and Rapid AI Agent Adoption. As listed in Table V, agent-initiated and multi-agent reviews rank ahead of human-only review (R1–R2 vs. R2–R3) for both Gradual AI and Rapid AI Agent Adoption. However, we do not observe a significant efficiency difference from human-only review under Rapid LLM Adoption. In the agent era of Rapid LLM Adoption, human-only review ranks first in review efficiency, with 61% of its pull requests resolved in a two-turn exchange [15]. This indicates that human review is already fast in this practice. Furthermore, we observe that in agent-init and multi-agent reviews, the agents take over the inspection step in 75% to 95% of these pull requests, posting a summary of the code change, after which the human responds with a single short comment. For example, after the agent summarizes a bug fix, the human replies *"I cannot reproduce the bug, but the change looks very reasonable to me."*, after which the pull request is merged. This suggests that agent reviewers can support the inspection step and provide a useful starting point for the review, potentially reducing the exchanges needed from human reviewers.

Pull requests reviewed only by human reviewers exhibit significantly lower review smell prevalence than most AI-involved collaboration patterns. Table V shows that smell prevalence ranges from 69% to 76% for human-only reviews and from 78% to 94% for patterns involving AI across the LLM and agent eras. This higher smell prevalence is mainly driven by Review Buddies, which rises from 16% for human-only reviews to 60% for LLM-involved patterns and 53% for agent-involved patterns on average [15]. The other smells do not follow this trend, with Sleeping Review declining and Large Changeset staying flat across eras [15]. These results suggest that repeated use of the same AI reviewer or model identity may reduce reviewer diversity and narrow review perspectives. Practitioners could adopt more collaborative multi-agent setups that distribute review across several agents and encourage knowledge transfer, so that the review process does not depend on a single default AI reviewer.

AI-involved collaboration patterns are associated with different pull request types across adoption practices and eras. As shown in Table V, under Gradual AI Adoption, several AI-involved patterns, such as Multi-LLM, LLM-Bot-Assist, and Multi-Agent, are associated with chore pull requests more often than human-only review. Under Rapid LLM Adoption, most AI-involved patterns in the LLM era are associated with bug-fixing pull requests and are less efficient than human-only review (R2–R3 vs. R1). In the agent era of Rapid LLM Adoption, fix pull requests become significantly less frequent for several AI-involved patterns, such as Agent-Init and Multi-Agent, while other patterns are associated with documentation, chore, or feature pull requests. These results suggest that, under Rapid LLM Adoption, agent-era collaboration patterns change the types of pull requests involving AI more than they change review efficiency.

Summary for RQ2

Collaboration patterns with LLM or AI agent reviewers show significantly higher review buddies than human-only review. In the agent era, agent-init and multi-agent reviews are significantly faster than human-only review in Gradual AI and Rapid AI Agent Adoption, but show no significant difference in Rapid LLM Adoption.

C. RQ3: How do human-AI collaboration patterns relate to traditional factors impacting review quality?

Motivation. In RQ1 and RQ2, we find that review quality differs significantly across AI adoption practices and human-AI collaboration patterns. However, this quality difference may not necessarily result from AI collaboration patterns or AI adoption practices, and may be explained by other factors of the review process, such as reviewer experience and review activity (e.g., commit count, inline discussion threads) within a pull request [12], [11]. We therefore examine whether review quality is more strongly associated with collaboration patterns or with alternative factors in the review process (e.g., pull request characteristics) . This analysis helps identify the factors associated with better review quality, guiding what developers could consider when designing agentic review pipelines.

Approach. To examine how traditional review factors and human-AI collaboration factors are associated with review

quality, we build explanatory models for each AI adoption practice and review era, using the following process.

Training Explanatory Model. To examine how our explanatory factors are associated with review quality, we use logistic regression [51], [52], [53]. Logistic regression estimates how each factor is associated with the probability of an efficient review or the presence of a review smell while accounting for the other factors in the model. In each model, the independent variables include the collaboration patterns (identified in Sec. III-B), pull request characteristics (e.g., initial code churn), review activity (e.g., commit counts during the review), and participant experience(e.g., human reviewer experience in the same project), which are listed in Table III. The dependent variable is a binary review outcome that captures whether a review is efficient or whether it contains a given review smell, which is labeled as follows:

(1) Review efficiency analysis: The review efficiency metric is continuous (#days/KLOC), so we discretize it into "efficient" and "inefficient" classes. Within each adoption practice and review era, pull requests below the median #days/KLOC are labeled "efficient", and the rest are labeled "inefficient". This yields nine efficiency models (i.e., 3 practices $\times$ 3 eras) in total, which estimate whether the explanatory variables are associated with efficient review.

(2) Review smells analysis: For each selected review smell, the dependent variable is binary, indicating whether a pull request contains that smell. Because the review smell models are trained separately for each adoption practice and review era, each modeled smell must include enough pull requests with and without that smell in every model. Therefore, we focus on the three most prevalent smells among modeled pull requests: Review Buddies (46.9%), Sleeping Review (38.6%), and Large Changeset (20.6%). This yields 27 smell models in total (3 smells $\times$ 3 practices $\times$ 3 eras), which estimate whether the explanatory variables are associated with each smell.

Processing explanatory variables. Since highly correlated variables can distort model estimates [52], [54], we compute the Variance Inflation Factor (VIF) for each variable in Table III. VIF measures how much a variable's variance is inflated by its correlation with the other variables [55]. All variables yielded a VIF below 5 [51], so none are removed. To reduce the influence of extreme values, we then apply a log transformation to each numeric variable with positive skew [53]. For categorical variables (e.g., collaboration pattern and pull request types), we use dummy encoding. For collaboration patterns, human-only review serves as the reference category. For pull request type, the "feature" type serves as the reference category. Thus, coefficients for dummy-encoded collaboration patterns are interpreted relative to human-only review, and those for pull request types are interpreted relative to feature pull requests [51].

Measuring the goodness of fit. To analyze how well the explanatory models distinguish between the binary review quality classifications, we use the Area Under the ROC Curve (AUC) [56], [57] to assess whether the explanatory models provide sufficient separation between the binary review outcomes used in our analysis. An AUC of 0.5 indicates random separation, while substantially larger (or smaller) values indicate that the explanatory variables better distinguish efficient reviews from inefficient ones or smelly from non-smelly reviews. To avoid evaluation on the same data used to train the model, we train each model (9+27=36 models in total) on 90% of the data within each combination of AI adoption practice and review era, and use the remaining 10% held-out test set to measure AUC [51].

Estimating Factor Importance. We follow prior modeling analyses [51], [12] by adding our explanatory features to the model one at a time and applying a likelihood ratio chi-square test [40]. A p-value below 0.05 from the likelihood ratio chi-square test indicates that the newly added factor significantly improves model fit beyond the factors already included. For example, consider a logistic regression model with efficient review as the dependent variable, where reviewer collaboration pattern and pull request type are already included as independent variables. When review activity is added as another independent variable, the likelihood-ratio chi-square test compares the expanded model with the previous model that includes only reviewer collaboration pattern and pull request type. A p-value below 0.05 shows that review activity significantly improves model fit after those variables have already been considered. This analysis helps determine which variables (e.g., review activity) have an independent explanatory contribution to review efficiency or review smells. To this end, we first measure a baseline prediction (P_{base}) by setting all variables to their baseline values (i.e., median values for numeric variables and reference categories for categorical variables). We then compute a second prediction (P_{changed}) by changing only the target variable by a typical amount (i.e., increasing a numeric variable by one standard deviation or setting a categorical variable to the target category), while keeping all other variables at their baseline values [58], [12]. We then calculate the impact percentage as follows:

$$\text{Impact Score (Imp\%)} = \frac{P_{\text{changed}} - P_{\text{base}}}{P_{\text{base}}} \times 100\% \quad (4)$$

Positive Imp% indicates a higher predicted probability of the modeled outcome, and negative Imp% indicates a lower predicted probability. By dividing each effect by the baseline probability (P_{base}), Imp% places variables with different units and ranges on a common scale. We can therefore directly compare the practical impact of each significant variable (e.g., the number of reviewers) and identify which factors are most strongly associated with review efficiency or review smells.

Results. We organize our findings by review efficiency and three selected review smells. We discuss each outcome below.

(1) Review Efficiency. **Greater reliance on AI reviewers does not necessarily translate into more efficient code reviews, as traditional review factors continue to play an important role.** Table VI presents the impact scores of the significant explanatory factors for each outcome. In the Pre-LLM era, author experience and pull request characteristics (e.g., chore and refactoring pull request types) are the primary factors associated with review efficiency. For example, author experience is associated with lower review delay across the three adoption practices, with impact scores from -17 to -22 in the Pre-LLM era. In the LLM era, collaboration patterns

TABLE VI: Impact of collaboration patterns and traditional factors on review delay (lack of efficiency) and review smells across review eras and AI-adoption practices. Green text indicates a significantly better outcome (e.g., lower review delay), whereas red text indicates a significantly worse outcome. Blank cells indicate non-significant associations ($p >= 0.05$ and $|\text{impact}| < 10\%$.).

	Factor	(1) Lack of Efficiency (Review Delay)									Review Smells																										
											(2) Review Buddies									(3) Sleeping Review									(4) Large Changeset								
		Pre-LLM			LLM			Agent			Pre-LLM			LLM			Agent			Pre-LLM			LLM			Agent			Pre-LLM			LLM			Agent		
Collaboration	Multi-Agent							−50	+37	−13							+220	+250	+513							−48		−16							+29	−60	+54
	Agent-Init							−37	+39	−16							+200	+249	+605							−36	+42	−26							+27	−55	+20
	Agent-ML							−11	+50								+246	+269	+678								+94								+48	−44	
	Agent-Assist								+53	+13							+174	+227	+418								+88	+26							+25	−51	+29
	LLM-Bot					+33	+28							+220	+2K	+130						+26	+124	+56							−44	−41					
	LLM-Assist				+28	+45	+30							+65	+2K	+62						+45	+214	+79							−43						
	Human-Bot					+17	+12	−21			+287	+1K	+283	+170	+2K	+108	+197	+96	+437		−14			+46	+32	−24			−21			−18	−35	+38	+14	+26	+20
Traditional	build			−18			−20	+16		−40	−61		−15	−86	−43	−20	−41	−55	−19	−10	−38	−26	−26	−22	−14		−24	−25		−31		−47	+52	+29		+20	+44
	chore	−32	−41	−24	−30	−11	−12			−25	+19	+52	+27		−39	−30	+14	−25		−40	−50	−42	−48	−33	−40	−14	−28	−32	+31			+29					+11
	refactor	−11	−16	−12		−10		−11				+34	+29	+25	−19	+20	+17	−12	+33		−22	−14		−24			−19		+32	+12		+14		+15	+26	+15	
	Author Exp	−17	−22	−18	−12			−15		−14	+115	+238	+151	+72	+516	+67	+93	+228	+116	−19	−24	−22	−19	−23	−24	−17	−24	−18		+12	+10		+10		+12		+20
	NumCommit											−14				+10				+36	+31	+28	+48	+70	+41	+29	+27	+37	+95	+100	+73	+96	+73	+77	+74	+49	+76
	InlineThread	+11	+20	+15	+12		+15	+12		+10					+16	−19			−15	+30	+36	+30	+45	+43	+51	+30	+26	+27	+16	+12		+33	+44		+32	+30	+24
		Gradual AI	Rapid LLM	Rapid Agent	Gradual AI	Rapid LLM	Rapid Agent	Gradual AI	Rapid LLM	Rapid Agent	Gradual AI	Rapid LLM	Rapid Agent	Gradual AI	Rapid LLM	Rapid Agent	Gradual AI	Rapid LLM	Rapid Agent	Gradual AI	Rapid LLM	Rapid Agent	Gradual AI	Rapid LLM	Rapid Agent	Gradual AI	Rapid LLM	Rapid Agent	Gradual AI	Rapid LLM	Rapid Agent	Gradual AI	Rapid LLM	Rapid Agent	Gradual AI	Rapid LLM	Rapid Agent

involving LLMs or bots are generally associated with higher review delay (i.e., lower efficiency), while traditional review factors remain relevant but less consistently, particularly for author experience and refactoring. In the agent era, AI agent collaboration becomes the dominant factor associated with review efficiency. Specifically, multi-agent and agent-init collaboration patterns are negatively associated with review delay (i.e., higher efficiency) under Gradual AI Adoption and Rapid AI Agent Adoption. However, under Rapid LLM Adoption, where traditional review factors are no longer significant, AI collaboration patterns remain consistently associated with higher delay and therefore lower review efficiency, suggesting that projects relying more on AI-assisted reviews, without the influence of traditional review factors, do not necessarily achieve more efficient reviews.

(2) Review Buddies. **Build pull requests are often associated with fewer Review Buddies across review eras, suggesting more diverse reviewer participation for maintenance-oriented tasks.** As shown in Table VI, build, chore, and refactoring pull requests exhibit negative associations with Review Buddies under the Rapid LLM Adoption in the LLM era (-43, -39, and -19) and the Agent era (-55, -25, and -12). This pattern is also observed for build pull requests under Rapid AI Agent Adoption, where build pull requests exhibit negative associations with Review Buddies across the three eras (-15, -20, and -19). This suggests that these maintenance-oriented pull requests are less likely to rely on repeated reviewer pairings and instead involve a more diverse set of reviewers. However, refactoring pull requests show positive associations with Review Buddies under Rapid AI Agent Adoption across the three eras ($+29$, $+20$, and $+33$), indicating more repeated reviewer pairings for this pull request type. Furthermore, author experience is consistently positively associated with Review Buddies across all review eras, indicating that more experienced authors are more likely to review with recurring collaborators, whereas less experienced authors tend to participate in more diverse review interactions.

(3) Sleeping Review. **In the Agent era, multi-agent and agent-init reviews are associated with fewer Sleeping Reviews under Gradual AI Adoption and Rapid AI Agent Adoption.** Across the three review eras, pull requests with more commits and more inline discussion threads are consistently associated with a higher probability of Sleeping Review, with impact scores ranging from $+27$ to $+70$ for commits and from $+26$ to $+51$ for inline discussion threads. By contrast, pull requests submitted by authors with more prior experience in the same project, as well as build and chore pull requests, are associated with a lower probability of Sleeping Review, with impact scores ranging from -10 to -50. In the LLM era, LLM-Assist and LLM-Bot-Assist are associated with a higher probability of Sleeping Review, with impact scores ranging from $+26$ to $+214$. In the agent era, Multi-Agent and Agent-Init patterns are associated with a lower probability of Sleeping Review in Gradual AI Adoption and Rapid AI Agent Adoption, with impact scores ranging from -16 to -48. By contrast, under Rapid LLM Adoption, agent patterns such as Agent-Init and Agent-ML are associated with a much higher probability of Sleeping Review, with impact scores of $+42$ and $+94$.

(4) Large Changeset. **Agent-involved collaboration patterns under Gradual AI Adoption and Rapid AI Agent Adoption are often associated with a higher likelihood of Large Changeset.** As shown in Table VI, among traditional factors, only build pull requests show mixed associations with Large Changeset across practices and eras. Other factors, including commit count, inline discussion threads, author experience, chore and refactoring pull requests, show positive significant associations where they appear. In the agent era, significant agent-involved collaboration patterns show positive associations with Large Changeset under Gradual AI Adoption and Rapid AI Agent Adoption, with impact scores ranging from $+25$ to $+48$ and from $+20$ to $+54$, respectively. These findings suggest that agent-involved collaboration is more frequently associated with the review of larger code changes than in the Pre-LLM and LLM eras.

Summary for RQ3

Traditional review factors remain important after generative AI reviewers are adopted. Agent-era collaboration patterns reduce Sleeping Review under Gradual AI Adoption and Rapid AI Agent Adoption, but they are also associated with more Review Buddies and larger changesets.

IV. Implications

In this section, we discuss the implications of our findings for practitioners, tool builders, and researchers.

Adopting AI reviewers selectively rather than uniformly. Our results show that AI reviewer adoption practices are associated with differences in review quality. Gradual AI Adoption and Rapid AI Agent Adoption are associated with faster reviews, while Rapid LLM Adoption is associated with lower review quality. Therefore, practitioners may consider adopting AI reviewers selectively, based on pull request context and observed review efficiency and quality, rather than applying the same AI reviewer configuration uniformly across pull requests.

Building context-aware support for human-AI review decisions. Our findings identify several signals that can inform AI reviewer assignment, including human-AI collaboration patterns, pull request type, changeset size, discussion activity, and author or reviewer history. Tool builders can use these signals to help practitioners decide whether a pull request should remain human-led, use lightweight LLM summarization, or involve AI agent inspection.

Balancing AI collaboration with traditional review practices. Our findings show that AI collaboration does not uniformly improve review quality. While some agent-era collaboration patterns are associated with fewer Sleeping Reviews, they are also associated with more Review Buddies and larger changesets. Traditional review factors, such as author experience and review activity, remain strongly associated with review efficiency and review smells. Practitioners should therefore integrate AI reviewers alongside established review practices rather than relying on AI collaboration alone.

Studying developer decisions behind human-AI review practices. AI review practices vary across projects, but our results do not explain why developers choose particular human-AI review practices. Future research can build on our results through interviews or surveys to understand why developers assign AI reviewers to some pull requests, how they interpret AI feedback, when they keep humans in the lead, and how these decisions shape review efficiency and quality.

Developing benchmarks that reflect pull request context. Evaluating AI reviewers only by generic issue-finding ability may miss important differences across review contexts. Researchers and tool builders can use signals such as pull request type, changeset size, discussion activity, author or reviewer history, and human-AI collaboration pattern to design benchmarks across contexts such as routine maintenance, bug fixing, refactoring, and large changesets. Such benchmarks would help assess when AI reviewers provide useful support and when human-led review remains necessary.

V. Threats to Validity

Threats to construct validity. In our analysis, the agent era denotes a chronological period in a project's review history rather than guaranteeing that every AI reviewer during that period is an AI agent. A project enters the agent era after its first pull request involving an AI agent reviewer, but may continue to use other LLM reviewers afterward. Therefore, agent-era pull requests can still contain LLM reviewer participation. For example, the identified collaboration patterns include LLM-involved patterns in the agent era, such as LLM-Assist. However, these older patterns do not have sufficient support (less than 5%), so we exclude them from the pairwise comparisons reported in Table V. The complete pattern results and their frequencies are available in our replication package [15].

Threats to internal validity. Our logistic regression models in RQ3 are explanatory, not causal. The associations we report reflect statistical relationships between factors and review efficiency and quality within each practice and era, but do not imply that changing one factor will directly cause a change in review efficiency or review smells.

Threats to external validity. Given the rapid evolution of AI reviewer capabilities, the collaboration patterns and review quality associations that we observe may shift as tools become more sophisticated. Our dataset covers activity from May 2022 through February 2026, providing a current longitudinal view of AI reviewer adoption. While findings specific to individual tools may evolve, our longitudinal analysis across the pre-LLM, LLM, and agent eras provides a baseline and reference point for future studies to track how AI reviewer adoption and review quality continue to change.

VI. Related work

Code review practices. Code review is a fundamental quality assurance practice where developers examine code changes before integration. Prior work has shown how review intervals and participation converge across diverse organizations [9], characterized modern review practices at scale [10], introduced code review smell taxonomies [13], and shown that reviewer experience and changeset characteristics influence review comment quality [30]. As AI reviewers enter code review, recent studies have evaluated LLM-assisted review tools in specific settings. Cihan et al. [14] find that LLM-assisted review may increase pull request review time despite high comment resolution rates. Aðalsteinsson et al. [59] find that LLM-assisted review reduces reviewer cognitive load and improves comprehension. Sun et al. [60] find that LLM reviewers produce concise comments that are more likely to lead to code changes. In this study, we move beyond evaluating individual AI review tools in isolation and analyze how review practices evolve as projects transition from human-centric review to agentic review. We examine this transition at the project level and pull-request level by identifying AI reviewer adoption practices and human-AI collaboration patterns.

Review efficiency and quality factors. Prior work has studied technical and social factors associated with code review efficiency and quality in human-centric review processes. The factors include organizational structure and developer relationships [11], past review participation and patch description length [12], and reviewer experience and changeset characteristics [30]. Such factors help explain why some reviews complete faster or produce more useful feedback, but they were developed mainly for human-only review settings. In this study, we extend these review outcome factors to the transition from human-centric to agentic review. We model review outcomes together with human-AI collaboration patterns, pull

request characteristics, review activity, and participant experience to examine whether efficiency and quality differences are associated mainly with AI reviewer participation or also with the alternative factors.

VII. Conclusion

In this study, we analyze 1.02 million pull requests from 207 GitHub projects to understand how code review quality evolves across three review eras. We identify three AI reviewer adoption practices and find that Gradual AI Adoption and Rapid AI Agent Adoption are associated with faster reviews but no improvement in review smells, while Rapid LLM Adoption is associated with higher review-smell prevalence and no efficiency gain. Within individual pull requests, agent-init and multi-agent reviews reach a decision faster under Gradual AI and Rapid AI Agent Adoption, while they carry review smells more often than human-only reviews. We further show that, beyond traditional factors, human-AI collaboration patterns remain strongly associated with review efficiency and quality.

Together, these findings suggest that developers should adopt AI reviewers selectively, taking into account their adoption history, who reviews each pull request and when, and the characteristics of each pull request. In future work, we aim to study how agentic review systems can improve review efficiency while preserving the quality standards established in human code review.

References

[1] Hugh Langley, "Google says 75% of the new code is AI-generated," https://www.businessinsider.com/google-ai-generated-code-75-gemini-agents-software-2026-4, April 2026, accessed: 2026-06-30.

[2] GitHub, "A new AI developer joins GitHub," https://github.blog/news-insights/octoverse/octoverse-a-new-developer-joins-github-every-second-as-ai-leads-typescript-to-1/, October 2025, updated February 28, 2026. Accessed: 2026-06-30.

[3] Lee Boonstra, "Google review bottleneck," https://cloud.google.com/transform/when-ai-writes-the-code-who-reviews-it-cto-google-cloud, 2026, accessed: 2026-06-16.

[4] Matthias Patzak, "Amazon review bottleneck," https://aws.amazon.com/blogs/enterprise-strategy/your-ai-coding-assistants-will-overwhelm-your-delivery-pipeline-heres-how-to-prepare, Jan. 2026, accessed: 2026-06-16.

[5] H. Li, H. Zhang, and A. E. Hassan, "The rise of ai teammates in software engineering (se) 3.0: How autonomous coding agents are reshaping software engineering," 2025. [Online]. Available: https://arxiv.org/abs/2507.15003

[6] S. Zhong, Y. Zou, and B. Adams, "Developer-llm conversations: An empirical study of interactions and generated code quality," 2025. [Online]. Available: https://arxiv.org/abs/2509.10402

[7] Z. Rasheed, M. A. Sami, M. Waseem, K.-K. Kemell, X. Wang, A. Nguyen, K. Systä, and P. Abrahamsson, "Ai-powered code review with llms: Early results," 2025. [Online]. Available: https://arxiv.org/abs/2404.18496

[8] Thomas Dohmke, "Coding agent for Copilot," https://github.blog/news-insights/product-news/github-copilot-meet-the-new-coding-agent/, May 2025, accessed: 2026-01-01.

[9] P. C. Rigby and C. Bird, "Convergent contemporary software peer review practices," in *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, ser. ESEC/FSE 2013. New York, NY, USA: Association for Computing Machinery, 2013, p. 202–212. [Online]. Available: https://doi.org/10.1145/2491411.2491444

[10] C. Sadowski, E. Söderberg, L. Church, M. Sipko, and A. Bacchelli, "Modern code review: a case study at google," in *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice*, ser. ICSE-SEIP '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 181–190. [Online]. Available: https://doi.org/10.1145/3183519.3183525

[11] O. Baysal, O. Kononenko, R. Holmes, and M. W. Godfrey, "Investigating technical and non-technical factors influencing modern code review," *Empirical Software Engineering*, vol. 21, no. 3, pp. 932–959, Jun 2016. [Online]. Available: https://doi.org/10.1007/s10664-015-9366-8

[12] P. Thongtanunam, S. McIntosh, A. E. Hassan, and H. Iida, "Review participation in modern code review," *Empirical Software Engineering*, vol. 22, no. 2, pp. 768–817, Apr 2017. [Online]. Available: https://doi.org/10.1007/s10664-016-9452-6

[13] E. Doğan and E. Tüzün, "Towards a taxonomy of code review smells," *Information and Software Technology*, vol. 142, p. 106737, 2022. [Online]. Available: https://doi.org/10.1016/j.infsof.2021.106737

[14] U. Cihan, V. Haratian, A. İçöz, M. K. Gül, Ö. Devran, E. F. Bayendur, B. M. Uçar, and E. Tüzün, "Automated code review in practice," in *2025 IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 2025, pp. 425–436. [Online]. Available: https://doi.org/10.1109/ICSE-SEIP66354.2025.00043

[15] Anonymous, "AI review evolution," https://anonymous.4open.science/r/CodeReviewEvolve-7917, 2026, accessed: 2026-02-26.

[16] GitHub, "GitHub advanced search," https://github.com/search/advanced, Feb. 2026, accessed: 2026-06-30.

[17] J. Coelho, M. T. Valente, L. L. Silva, and E. Shihab, "Identifying unmaintained projects in github," in *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3239235.3240501

[18] Wikipedia contributors, "ChatGPT," https://en.wikipedia.org/wiki/ChatGPT, 2025, accessed: 2025-12-30.

[19] GitHub, "GitHub REST API," https://docs.github.com/en/rest?apiVersion=2026-03-10, Feb. 2026, accessed: 2026-2-24.

[20] GitHub, Inc., "GitHub actions bot details," https://github.com/marketplace/actions/bot-details, Jan. 2026, accessed: 2026-01-01.

[21] Amazon Web Services, "Amazon CodeGuru Reviewer," https://docs.aws.amazon.com/codeguru/latest/reviewer-ug/welcome.html, 2025, accessed: 2026-04-09.

[22] JetXu-LLM, "LlamaPReview: Evidence-based, low-noise AI code reviewer," https://github.com/marketplace/llamapreview, gitHub Marketplace app page. Accessed: 2026-06-23.

[23] Anthropic, "Claude 3.7 Sonnet and Claude Code," https://www.anthropic.com/news/claude-3-7-sonnet, Feb. 2025, accessed: 2026-05-25.

[24] T. Dybå, V. B. Kampenes, and D. I. Sjøberg, "A systematic review of statistical power in software engineering experiments," pp. 745–755, 2006. [Online]. Available: https://doi.org/10.1016/j.infsof.2005.08.009

[25] M. Watanabe, H. Li, Y. Kashiwa, B. Reid, H. Iida, and A. E. Hassan, "On the use of agentic coding: An empirical study of pull requests on github," *ACM Transactions on Software Engineering and Methodology*, Mar. 2026, just Accepted. [Online]. Available: https://doi.org/10.1145/3798166

[26] K. E. Emam, "Benchmarking kappa: Interrater agreement in software process assessments," *Empirical Software Engineering*, vol. 4, no. 2, pp. 113–133, Jun 1999. [Online]. Available: https://doi.org/10.1023/A:1009820201126

[27] D. Izquierdo-Cortazar, N. Sekitoleko, J. M. Gonzalez-Barahona, and L. Kurth, "Using metrics to track code review performance," in *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, ser. EASE '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 214–223. [Online]. Available: https://doi.org/10.1145/3084226.3084247

[28] M. Chouchen, A. Ouni, R. G. Kula, D. Wang, P. Thongtanunam, M. W. Mkaouer, and K. Matsumoto, "Anti-patterns in modern code review: Symptoms and prevalence," in *2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2021, pp. 531–535. [Online]. Available: https://doi.org/10.1109/SANER50967.2021.00060

[29] S. McIntosh, Y. Kamei, B. Adams, and A. E. Hassan, "An empirical study of the impact of modern code review practices on software quality," *Empirical Software Engineering*, vol. 21, no. 5, pp. 2146–2189, Oct 2016. [Online]. Available: https://doi.org/10.1007/s10664-015-9381-9

[30] A. Bosu, M. Greiler, and C. Bird, "Characteristics of useful code reviews: An empirical study at microsoft," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, 2015, pp. 146–156. [Online]. Available: https://doi.org/10.1109/MSR.2015.21

[31] S. Zhong, S. Noei, Y. Zou, and B. Adams, "Human-ai synergy in agentic code review," 2026. [Online]. Available: https://arxiv.org/abs/2603.15911

[32] A. E. Hassan, H. Li, D. Lin, B. Adams, T.-H. Chen, Y. Kashiwa, and D. Qiu, "Agentic software engineering: Foundational pillars and a research roadmap," 2026. [Online]. Available: https://arxiv.org/abs/2509.06216

[33] M. Wessel, A. Serebrenik, I. Wiese, I. Steinmacher, and M. A. Gerosa, "Effects of adopting code review bots on pull requests to oss projects," in *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2020, pp. 1–11. [Online]. Available: https://doi.org/10.1109/ICSME46990.2020.00011

[34] M. Lepot, J.-B. Aubin, and F. H. Clemens, "Interpolation in time series: An introductive overview of existing methods, their performance criteria and uncertainty assessment," *Water*, vol. 9, no. 10, 2017. [Online]. Available: https://doi.org/10.3390/w9100796

[35] S. Noei, H. Li, and Y. Zou, "An empirical study on release-wise refactoring patterns," *Proceedings of the ACM on Software Engineering*, vol. 2, no. FSE, Jun. 2025. [Online]. Available: https://doi.org/10.1145/3715734

[36] F. Petitjean, A. Ketterlin, and P. Gançarski, "A global averaging method for dynamic time warping, with applications to clustering," *Pattern Recognition*, vol. 44, no. 3, pp. 678–693, 2011. [Online]. Available: https://doi.org/10.1016/j.patcog.2010.09.013

[37] M. Cuturi and M. Blondel, "Soft-dtw: a differentiable loss function for time-series," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17. JMLR.org, 2017, p. 894–903. [Online]. Available: https://dl.acm.org/doi/10.5555/3305381.3305474

[38] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987. [Online]. Available: https://doi.org/10.1016/0377-0427(87)90125-7

[39] M. Kanwal, M. A. Khan, N. Ismat, N. A. Khan, and A. A. Khan, "Machine learning approach to classification of online users by exploiting information seeking behavior," *IEEE Access*, vol. 12, pp. 53 234–53 249, 2024. [Online]. Available: https://doi.org/10.1109/ACCESS.2024.3383444

[40] S. Boslaugh and D. P. A. Watters, *Statistics in a nutshell*, 1st ed. USA: O'Reilly & Associates, Inc., 2008. [Online]. Available: https://dl.acm.org/doi/book/10.5555/1461408

[41] "RT-Thread, an open source IoT Real-Time Operating System (RTOS)," https://github.com/RT-Thread/rt-thread, accessed: 2026-06-30.

[42] "Open Library, an open, editable library catalog, building towards a web page for every book ever published." https://github.com/internetarchive/openlibrary, accessed: 2026-06-30.

[43] "Azure Command-Line Interface," https://github.com/Azure/azure-cli, accessed: 2026-06-30.

[44] "Flutter Packages, a collection of useful packages maintained by the Flutter team," https://github.com/flutter/packages, accessed: 2026-06-30.

[45] T. Moon, "The expectation-maximization algorithm," *IEEE Signal Processing Magazine*, vol. 13, no. 6, pp. 47–60, 1996. [Online]. Available: https://doi.org/10.1109/79.543975

[46] M. Song, C. W. Günther, and W. M. P. van der Aalst, "Trace clustering in process mining," in *Business Process Management Workshops*, D. Ardagna, M. Mecella, and J. Yang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 109–120. [Online]. Available: https://doi.org/10.1007/978-3-642-00328-8_11

[47] R. P. J. C. Bose and W. M. van der Aalst, "Context aware trace clustering: Towards improving process mining results," in *Proceedings of the 2009 SIAM International Conference on Data Mining (SDM)*, pp. 401–412. [Online]. Available: https://doi.org/10.1137/1.9781611972795.35

[48] S. Watanabe, "A widely applicable bayesian information criterion," *The Journal of Machine Learning Research*, vol. 14, no. 1, p. 867–897, Mar. 2013. [Online]. Available: https://dl.acm.org/doi/10.5555/2567709.2502609

[49] E. Jelihovschi, J. C. Faria, and I. B. Allaman, "Scottknott: A package for performing the scott-knott clustering algorithm in r," *Trends in Computational and Applied Mathematics*, vol. 15, no. 1, p. 003–017, Mar. 2014. [Online]. Available: https://doi.org/10.5540/tema.2014.015.01.0003

[50] M. Ouf, S. Noei, Z. Van Iterson, M. Guizani, and Y. Zou, "An empirical analysis of community and coding patterns in oss4sg vs. conventional oss," *arXiv preprint arXiv:2601.03430*, 2026.

[51] P. Tourani and B. Adams, "The impact of human discussions on just-in-time quality assurance: An empirical study on openstack and eclipse," in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, 2016, pp. 189–200. [Online]. Available: https://doi.org/10.1109/SANER.2016.113

[52] T. A. Ghaleb, D. A. da Costa, Y. Zou, and A. E. Hassan, "Studying the impact of noises in build breakage data," *IEEE Transactions on Software Engineering*, vol. 47, no. 9, pp. 1998–2011, 2021. [Online]. Available: https://doi.org/10.1109/TSE.2019.2941880

[53] Y. Kamei, E. Shihab, B. Adams, A. E. Hassan, A. Mockus, A. Sinha, and N. Ubayashi, "A large-scale empirical study of just-in-time quality assurance," *IEEE Transactions on Software Engineering*, vol. 39, no. 6, pp. 757–773, 2013. [Online]. Available: https://www.doi.org/10.1109/TSE.2012.70

[54] S. Noei, H. Li, and Y. Zou, "Detecting refactoring commits in machine learning python projects: A machine learning-based approach," *ACM Transactions on Software Engineering and Methodology*, vol. 34, no. 3, Feb. 2025. [Online]. Available: https://doi.org/10.1145/3705309

[55] R. M. O'brien, "A caution regarding rules of thumb for variance inflation factors," *Quality & Quantity*, vol. 41, no. 5, pp. 673–690, Oct 2007. [Online]. Available: https://doi.org/10.1007/s11135-006-9018-6

[56] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, no. 7, pp. 1145–1159, 1997. [Online]. Available: https://doi.org/10.1016/S0031-3203(96)00142-2

[57] C. Yu, S. Noei, H. Zhang, and Y. Zou, "An empirical study on the characteristics of reusable code clones," *ACM Transactions on Software Engineering and Methodology*, Jan. 2026, just Accepted. [Online]. Available: https://doi.org/10.1145/3793251

[58] E. Shihab, Y. Kamei, B. Adams, and A. E. Hassan, "Is lines of code a good measure of effort in effort-aware models?" *Information and Software Technology*, vol. 55, no. 11, pp. 1981–1993, 2013. [Online]. Available: https://doi.org/10.1016/j.infsof.2013.06.002

[59] F. S. Aðalsteinsson, B. B. Magnússon, M. Milicevic, A. N. Davidsson, and C.-H. Cheng, "Rethinking code review workflows with llm assistance: An empirical study," in *2025 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2025, pp. 488–497. [Online]. Available: https://doi.org/10.1109/ESEM64174.2025.00013

[60] K. Sun, H. Kuang, S. Baltes, X. Zhou, H. Zhang, X. Ma, G. Rong, D. Shao, and C. Treude, "Does ai code review lead to code changes? a case study of github actions," *IEEE Transactions on Software Engineering*, pp. 1–17, 2026. [Online]. Available: https://doi.org/10.1109/TSE.2026.3688237