Collaborator or Assistant? How AI Coding Agents Partition Work Across Pull Request Lifecycles

Young Jo(seph) Chung
University of Toronto
Faculty of Information
Toronto, Ontario, Canada
jo.chung@utoronto.ca

Safwat Hassan
University of Toronto
Faculty of Information
Toronto, Ontario, Canada
safwat.hassan@utoronto.ca

Abstract

When AI coding agents open branches and submit pull requests (PRs), two questions co-determine oversight design: who *starts* the work (operational agency) and who *authorizes* its completion (merge governance). We characterize tools along a *Collaborator–Assistant spectrum* in how they redistribute initiative, oversight, and endorsement, while merge governance remains predominantly human across five tools (OpenAI, Copilot, Devin, Cursor, Claude Code). We analyze 29,585 PR lifecycles using an Initiator × Approver taxonomy with six interaction scenarios; lifecycle reconstruction supplies the *how* behind those roles. Collaborator tools (Cursor, Devin, Copilot) concentrate operational initiative in agents that open and carry PR work forward, with humans retaining review and endorsement on the path to merge; Assistant tools (OpenAI, Claude) leave task direction primarily with humans and supply bounded support within human-led workflows. Across the spectrum, agency and governance *decouple*: Collaborator workflows are ≥96% agent-initiated, yet terminal merge authority remains almost exclusively human, with agent-classified approvers confined to a small fraction of PRs. Where automation executes a merge, logs record the executor but not the decision-maker, marking a boundary of observation. We contribute the taxonomy, per-tool state machines, and a replication package for research on automation, oversight, and governance in PR workflows.

CCS Concepts

• **Software and its engineering → Collaboration in software development**; *Software development process management*.

Keywords

AI coding agents, human-AI collaboration, merge governance, software engineering, pull request workflows, empirical study

ACM Reference Format:
Young Jo(seph) Chung and Safwat Hassan. 2026. Collaborator or Assistant? How AI Coding Agents Partition Work Across Pull Request Lifecycles. In *Proceedings of the 3rd ACM International Conference on AI-Powered Software (AIware '26)*. ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

AIware '26, Montreal, QC, Canada

ACM ISBN 978-1-4503-XXXX-X/26/06
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

When AI coding agents submit pull requests, two coupled questions co-determine oversight design: do these agents function as *collaborators* or as *assistants* on operational work, and who holds *merge governance* when they do? The first concerns *operational agency*—who initiates and carries the PR forward; the second concerns *governance authority*—who may authorize its completion. These are not the same dimension: agents now create branches, write code, run tests, and submit PRs autonomously [15], yet a tool can concentrate initiation on agents while reserving merges for humans, or the reverse. We use *Collaborator* and *Assistant* as behavioral workflow paradigms, not merely as labels for the actor who submits the first commit. In the Collaborator paradigm, operational initiative is redistributed toward the AI system: the agent proposes or opens the work, structures the PR artifact, and carries the lifecycle far enough that humans primarily exercise oversight, review, and endorsement. In the Assistant paradigm, the human remains the practical director of the work: the agent supplies bounded capability, such as generation, revision, or recommendation, within a workflow whose goals and trajectory remain human-led. We distinguish *endorsement* from *accountability* along this axis. Endorsement refers to clear acceptance of a specific PR outcome, such as review, approval, or merge authorization; accountability refers to who is responsible for the merged code after it enters the codebase. The human team may still be accountable for that code even when no person gives a clear approval and the final step is handled by repository rules such as branch protection or auto-merge. This distinction draws on agency and delegation research that treats initiative, autonomy, and responsibility as separable but accountability-relevant dimensions of human–AI work arrangements [3, 7, 20]. Together they shape tool design, team workflows [20], and trust calibration [17]. Empirically, they also vary by tool: in some ecosystems, nearly every agent-generated PR passes through human review; in others, most PRs reach the main branch without a recorded review step. Without lifecycle data that separates initiation from terminal authorization, review allocation, trust calibration, and tool selection rely on anecdote rather than evidence.

Recent studies report what AI-agent PRs achieve but not how they get there. Gao et al. find reduced review engagement in bot-associated PRs [6]; Rahman et al. evaluate task-level acceptance rates [19]; Pinna et al. compare merge rates stratified by task type [18]; and Agarwal et al. contrast IDE assistants with

autonomous agents on productivity metrics [1]. These studies measure *what happened* (merge rates, comment counts) rather than *who did what, when*: none reconstructs the full PR lifecycle or separates the actor who initiates work from the actor who authorizes its completion.

More broadly, empirical research on human–AI collaboration in software engineering has examined discrete interactions (code suggestions, commit patterns, or review comments [26]) and predominantly studied single phases rather than full lifecycles [25]. Related evidence from outside repository mining also shows that large-scale user feedback on Gen-AI software surfaces recurring concerns (e.g., performance, output quality, and policy/censorship), but this lens captures user-perceived app behavior rather than PR lifecycle governance [2]. Without a lifecycle view, teams lack evidence-based guidance for allocating oversight in AI-assisted development.

To address this gap, we draw on Sheridan and Verplank's levels of automation [24] and process mining analysis [27] to study 29,585 PR lifecycles across five AI coding tools [14]. **Main finding.** Tools fall along a **Collaborator–Assistant spectrum** on initiation and review routing (strong tool–scenario association: Cramér's $V = 0.50$), while merge governance stays concentrated in human hands. Our Initiator × Approver taxonomy separates *operational agency* (who starts work) from *governance authority* (who merges); these dimensions decouple empirically: Collaborator tools show ≥96% agent-initiated PRs, yet <0.1% receive agent-authorized merges. Agents do much of the operational work but hold almost none of the merge authority, a "junior teammate" pattern that directly informs review allocation (Section 5). The dataset concentrates temporally (94% of PRs from 2025 Q2–Q3), making this a cross-sectional snapshot of an emerging practice rather than a longitudinal trend analysis. In the rare cases where agents *do* hold merge authority, event logs record who executed the merge but not who made the governance decision, exposing a measurement boundary at the limits of what lifecycle mining can reveal.

Research Questions.

RQ1: How do humans and AI agents partition work across complete PR lifecycles?

RQ2: How do collaboration workflows and phase transitions differ across AI coding tools?

RQ3: What do automation-authorized merges reveal about the boundaries of the Initiator × Approver taxonomy?

We address these in Section 4.

Contributions. We make three contributions:

- **Analytical Framework.** A lifecycle model (three phases, two terminal outcomes) paired with an Initiator × Approver taxonomy of six interaction scenarios (Section 3).
- **Empirical Characterization.** Analysis of 29,585 PRs with five per-tool state machines quantifying workflow dynamics, plus path analysis of all automation-authorized merges (Section 4).
- **Replication Package.** The complete dataset, analysis pipeline, and generated artifacts.[1]

[1] https://doi.org/10.6084/m9.figshare.31343038

2 Background and Related Work

2.1 AI Coding Agents and Levels of Automation

AI-assisted programming evolved from syntax autocomplete to context-aware code generation (e.g., GitHub Copilot), and the current generation (Devin, Cursor, Claude Code) marks a further shift toward *agentic* behavior: autonomous task execution rather than suggestion [15]. Sheridan and Verplank's levels of automation [24] frame this trajectory: predictive assistants operate at Levels 2–3 (computer suggests, human chooses), whereas agentic systems advance toward Level 5 (execute after human approval) or Level 6+ (execute autonomously). Li et al. characterize this transition as "AI teammates" in SE 3.0, where agents participate as dialectical partners rather than passive tools [15].

Difference from outcome-centric evaluations. Much prior work on AI-assisted PRs is outcome-centric, comparing tools by merge rate, time-to-merge, or comment volume [14, 18, 19]. These metrics are informative but ambiguous about *who did what* across the lifecycle. Our contribution is process-centric: we reconstruct PR trajectories over phases (Created, Review, Revision, terminal outcome) and classify each PR into one of six Initiator × Approver scenarios (S1–S6). Researchers can then surface distinctions that aggregated outcome metrics conflate; for example, high acceptance may reflect agent-initiated work accepted by others (S1) versus human-initiated work selectively submitted and later self-merged (S4).

Difference from design-centric tool taxonomies. Another strand classifies tools by intended interaction mode (e.g., IDE assistants versus autonomous agents) [1, 15]. Such capability-based taxonomies describe affordances but do not necessarily reflect observed usage, especially when tools support multiple modes. Our Collaborator–Assistant spectrum is a behavioral classification grounded in event traces (initiation and terminal authorization), disentangling operational agency (who initiates work) from governance authority (who merges).

Complement to findings on reduced oversight. Reports of "silent merges" in AI-associated PRs highlight reduced oversight, but outcome- and comment-level analyses were not designed to localize where oversight is bypassed [6, 19]. By modeling lifecycle transitions, we provide a structural mechanism: in the Assistant paradigm, PRs frequently resolve directly from initial development to merge without recorded entry into a review phase. This complements social and behavioral interpretations by showing that the difference is also a workflow-level pattern observable in event sequences.

2.2 Pull-based development and GitHub evidence

Foundational empirical work established pull-based development as a systematic research topic on GitHub-scale data [10] and cataloged methodological pitfalls of mining GitHub [13]. We follow the first line by modeling lifecycle-level PR dynamics with explicit roles; we follow the second by treating RQ3 as a concrete case where event logs underdetermine governance (*who decided* versus *who executed* the merge).

2.3 Human-AI Collaboration and Trust

The *automation-augmentation paradox* [20] frames a central tension: AI can replace human tasks (automation) or enhance human capabilities (augmentation). *Distributed cognition* theory [12] posits that cognitive work distributes across actors and artifacts, but distributed *activity* does not imply distributed *responsibility*, a distinction central to PR workflows where agents initiate work yet humans retain merge authority.

This tension makes trust calibration critical: overtrust produces automation complacency, while undertrust limits productivity gains [9, 17]. Roychoudhury et al. propose "programming with trust," earned through transparency and verification [21]. Evans and Stanovich [5] and Halpern [11] appear in HCI-related work as sources of language about defaults, deliberate override, and monitoring in review-like settings; we cite them only as *optional vocabulary* for readers who want a familiar parallel to the observed split between PRs merged with little review and PRs routed through explicit review. The empirical claims in this paper do not depend on that parallel (or on any cognitive model of developers).

2.4 Process Mining in Software Engineering

Process mining treats event logs as traces through state machines [27], with applications spanning software configuration management [22, 23] and CI/CD workflows [16]. Classical process discovery induces models from data; we instead define the lifecycle model *a priori* from domain knowledge and adopt the process mining *analysis* framework, computing transition probabilities and median sojourn times to quantify human-agent workflow dynamics.

3 Methodology

3.1 Overview

We analyze **29,585 PR workflows with terminal outcomes** from the AIDev dataset across five AI coding tools (Figure 1). After excluding incomplete timelines (Section 3.2.1), we treat each PR as a complete lifecycle and compute phase transition probabilities and median hours by tool, yielding five state machines (Section 4).

Pipeline. For each PR the pipeline:

(1) Parses events chronologically.
(2) Classifies actors as Agent or Human via a two-step heuristic.
   (a) If GitHub's `actor.type` field equals `"Bot"`, the actor is classified as Agent.
   (b) For all remaining actors the lowercased login is checked against the patterns: `bot`, `copilot`, `devin`, `cursor`, `codex`, `openai`, `claude`. A login that contains any listed substring is classified as **Agent**; otherwise the actor is classified as **Human**.

   The pattern check covers missing or non-standard `actor.type` values. Validation against events with available `actor.type` shows that residual errors are rare (0.36%) and conservative for our main contrast: they classify some humans as Agent, but do not classify true Bots as Human. We return to the remaining small-sample risk in Section 6.
(3) Assigns phases (PR created → Review ⇌ Revision → terminal).

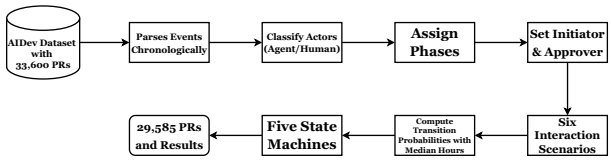

Figure 1: High-level overview of the data pipeline.

(4) Sets initiator (first `committed` actor) and approver (merge actor).
(5) Classifies the PR into one of six scenarios (S1–S6; Table 3).

Aggregation steps:

(6) **Transition probabilities.** Computes P(To|From) and median hours per phase and transition by tool.
(7) **Generate state machines.** Generates workflow diagrams per tool with phase nodes and transition edges.
(8) **Results.** Produces six-scenario distribution, paradigm classification, and state machine figures.

All steps are deterministic and reproducible.

3.2 Dataset

We use the AIDev dataset with 33,600 PRs from GitHub repositories with AI coding agent activity [14]. The dataset includes a curated subset of **qualification PRs** (repositories with >100 GitHub stars) with enriched event data (review comments, commit-level diffs, and full PR timelines; Table 1). We abbreviate the OpenAI tool family (GPT-4o, o1, etc.) as "OpenAI" and Claude Code as "Claude." Throughout, "Copilot" refers to GitHub Copilot's agent and Workspace variants captured in the AIDev dataset, distinct from Copilot's inline-suggestion IDE mode, which is not represented in this corpus and would likely exhibit Assistant-like initiation behavior.

Repository selection in the AIDev dataset is *tool-driven*: repositories enter the dataset when at least one PR involves a recognized AI coding agent actor (e.g., `copilot[bot]`, `devin-ai[bot]`). This strategy captures repositories *where tools are used*, not a random sample of all GitHub repositories; it follows a tool-driven sampling strategy used in empirical SE studies of emerging GitHub phenomena [13, 14], but one that limits generalizability to the population of AI-tool-adopting repositories.

Each timeline contains `committed`, `review_requested`, `reviewed`, `commented`, and `merged` or `closed` events. Not all PRs have complete lifecycle data; we apply exclusion criteria to obtain an analytic sample (Section 3.2.1, Table 1).

3.2.1 Exclusion Criteria. We exclude PRs that cannot be classified into our workflow taxonomy. Two criteria apply: (1) no `committed` event (15 PRs; initiator undefined) and (2) no `merged` or `closed` event (4,000 PRs; 11.9%; no terminal outcome). Table 1 shows the composition by tool before and after applying these criteria. The analytic sample comprises **29,585 PRs** with terminal outcomes. All statistics and figures use this sample.

3.3 Workflow Lifecycle Model

We define a PR lifecycle with three non-terminal phases and two terminal outcomes (Figure 2, Table 2).

Table 1: Dataset composition by tool. Total: raw count. No-Commit: excluded (no `committed` event). Incomplete: excluded (no `merged` or `closed` event). Included: analytic sample with terminal outcomes.

Tool	Total	No-Commit	Incomplete	Included
Cursor	1,541	1	241	1,299
Devin	4,829	5	595	4,229
Copilot	4,971	2	2,107	2,862
Claude Code	459	1	120	338
OpenAI	21,800	6	937	20,857
TOTAL	**33,600**	**15**	**4,000**	**29,585**

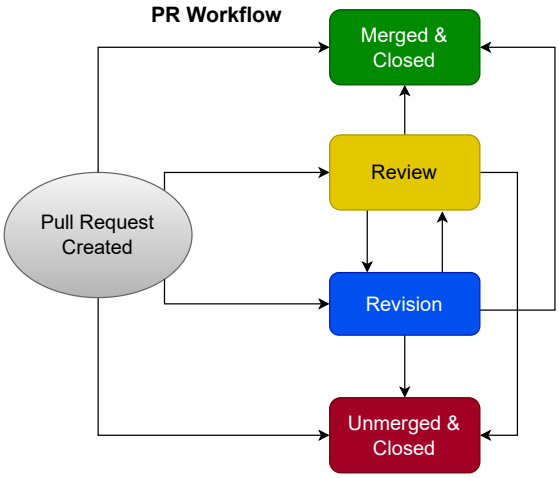

Figure 2: PR workflow lifecycle: phases and terminal outcomes.

We scan each timeline chronologically and map events to phases. Some PRs close and reopen (e.g., 46 Devin, 27 OpenAI); the final terminal event is definitive. State machines exclude intermediate Closed transitions. Revision cycle counts (REVIEW ⇌ REVISION) are computed separately and available in the replication package.

3.4 Interaction Scenario Taxonomy

We operationalize the Collaborator–Assistant distinction through six scenarios defined by initiator (first `committed` actor) and approver (Table 3). The taxonomy separates **operational agency** (who starts and carries work into the PR lifecycle) from **governance authority** (who authorizes completion through merge).

Merged PRs are split by approver (Human or Agent); non-merged PRs collapse into Not-Merged. The approver is a derived classification from the merge actor, not a raw governance field in the dataset. Approver classification uses the same bot-pattern heuristics as initiator classification. GitHub timelines often attribute merges to generic bots (e.g., `github-actions[bot]`), so S2/S5 are *automation-authorized*, not agent-tool-authorized.

Table 2: Phase definitions for the PR lifecycle model. *"PR created" denotes the *phase* (the interval from PR opening to first review); the instantaneous event of opening a PR is referred to as "PR creation." State machine diagrams label this phase "Pull Request Created."

Phase / Outcome	Definition
PR created*	Entry: PR timeline opens at the first recorded event. The PR remains here while work is being prepared and before recorded review begins. Exit: first `review_requested` or `reviewed` after ≥1 `committed`.
Review	Entry: first review-routing event after a commit, either `review_requested` or `reviewed`. The PR remains here while approval or rejection is pending. Exit: `CHANGES_REQUESTED`, merge, or close.
Revision	Entry: reviewer submits a `reviewed` event with state `CHANGES_REQUESTED`. The PR remains here while the author or agent responds through commits and renewed review requests. Exit: return to Review, merge, or close.
Merged & Closed	Terminal outcome entered when the PR is merged.
Unmerged & Closed	Terminal outcome entered when the PR is closed without merge.

Table 3: Six interaction scenarios (Initiator × Approver).

ID	Type	Criteria
	Collaborator	Agent-led operational agency; tools with ≥90% of PRs in S1+S2+S3.
S1	Agent-Init + Human-Approved	initiator=Agent, approver=Human
S2	Agent-Init + Agent-Approved	initiator=Agent, approver=Agent
S3	Agent-Init + Not-Merged	initiator=Agent, closed without merge
	Assistant	Human-led task direction with agent support; tools with ≥90% of PRs in S4+S5+S6.
S4	Human-Init + Human-Approved	initiator=Human, approver=Human
S5	Human-Init + Agent-Approved	initiator=Human, approver=Agent
S6	Human-Init + Not-Merged	initiator=Human, closed without merge

We classify tools as *Collaborator* when S1+S2+S3 exceeds 90%, or *Assistant* when S4+S5+S6 exceeds 90%. This threshold is an empirical operationalization of the broader workflow concepts introduced above. Collaborator tools make the AI system the practical source of initiative while humans retain review, endorsement, and merge governance; Assistant tools keep the human as task director while the agent contributes intermediate activities such as generation, revision, or recommendation within human-led workflows. The taxonomy therefore measures a sharper axis than initiation alone: where operational agency sits, whether endorsement is visible or thin, and how far accountability remains assigned to humans despite automation in the lifecycle.

Collaborator tools (S1–S3) combine agent-led generation and PR lifecycle movement with human oversight at review and merge; Assistant tools (S4–S6) reverse this, with humans initiating and directing work and agents supporting within human-led workflows. (Section 2).

4 Results

4.1 RQ1. How do humans and AI agents partition work across complete PR lifecycles?

Approach. We classify each PR by initiator and approver into six scenarios (S1–S6; Table 3), yielding a contingency table of 5 tools × 6 scenarios. We apply two tests. First, a χ^2 test of independence: under the null hypothesis, tool and scenario are unrelated (each tool would show the same scenario distribution). We reject the null if observed counts deviate significantly from this expectation. Second, Cramér's V measures effect size: how strongly knowing the tool predicts the scenario (0 = no association; 1 = perfect). For a 5×6 table (df = 4), Cohen's "large" threshold is $V \geq 0.25$ [4].

Results. Figure 3 shows the scenario distribution. The χ^2 test rejects independence ($\chi^2 = 29{,}817$, df = 20, $p < 0.001$). Cramér's V = 0.50 (twice the large-effect threshold for df = 4) indicates tool identity strongly predicts which scenario pattern a PR follows.

Tools cluster along a spectrum. Collaborator tools (Cursor, Devin, Copilot) show ≥96% agent-initiated PRs (S1+S2+S3). Assistant tools (OpenAI, Claude) show ≥95.6% human-initiated PRs (S4+S5+S6). OpenAI dominates the Assistant cluster ($n > 20{,}800$ vs. Claude $n = 338$); the pattern is best supported as an OpenAI–Collaborator contrast.

Direct resolution rates vary within paradigms. OpenAI resolves 76.5% of PRs directly from initial development; Claude resolves 37.6%, closer to Cursor (34.6%) than to OpenAI. Claude's small sample ($n = 338$) yields wide confidence intervals (e.g., S4: [45.4, 54.6]%). Claude's patterns are therefore presented as tentative; the main contributions (spectrum, $V = 0.50$, state machines, governance gap) hold across the four larger-sample tools.

Operational agency vs. governance authority. Initiation and approval *decouple*. Agents start ≥96% of Collaborator PRs, yet S2 (Agent-Init + Agent-Approved) accounts for <0.1% across all tools. Agents initiate heavily but hold near-zero merge authority.

> **Summary of RQ 1**
> Tools fall into Collaborator (≥96% agent-initiated) and Assistant (≥95.6% human-initiated) paradigms; tool identity strongly predicts scenario pattern (Cramér's V = 0.50). S1 and S4 (human-approved) dominate merged outcomes; S2 and S5 (agent-approved) remain rare. Governance stays human-centric regardless of who starts the work.

Semi-autonomous agent pattern (S2). S2 totals 14 PRs (≤0.1% per tool). The merge actor is typically a generic automation bot, not the coding agent itself (Section 3).

4.2 RQ2. How do collaboration workflows and phase transitions differ across AI coding tools?

Approach. We compute transition probabilities $P(\text{To} \mid \text{From})$ and median hours per state by tool, producing five state machines (Figures 4–8). Nodes show median hours; edges show transition probability and median hours.

Results. Collaborator tools route most PRs through Review: Copilot 90.3%, Cursor 51.3%, Devin 52.2%. Assistant tools resolve directly: OpenAI 76.5%, Claude 37.6%. Revision → Review return rates are higher for Collaborators (Copilot 95.5%, Devin 72.3%), indicating substantive revision loops. Median Review time: Copilot 3.0 h, Devin 2.0 h vs. OpenAI 0.7 h. Time to Unmerged & Closed varies sharply: Devin 67.8 h vs. Copilot 0.2 h and OpenAI 1.4 h. High rejection latency in Devin indicates extended deliberation; rapid closure in Copilot and OpenAI indicates automated triage.

> **Summary of RQ 2**
> Collaborators route PRs through Review (Copilot 90.3%, Cursor 51.3%, Devin 52.2%) with substantive revision loops; Assistants resolve directly (OpenAI 76.5%, Claude 37.6%). Median hours reveal the contrast: Collaborators 2–3 h in Review vs. OpenAI 0.7 h; time to Unmerged varies sharply (Devin 67.8 h vs. Copilot 0.2 h, OpenAI 1.4 h). Claude has a split profile (37.6% direct, closer to Cursor than OpenAI).

To control for initiation-driven routing differences, we restricted the sample to human-initiated PRs and compared review-routing rates. When initiation is held constant (human-initiated PRs only), the pattern persists: Collaborator tools still route most through Review (Copilot 87.8%, Devin 61.0%, Cursor 57.1%) while OpenAI resolves 76.6% directly to merge. Claude shows an intermediate profile (43.2% to Review, 37.8% to Merged), reinforcing its split profile. Wilson score 95% confidence intervals confirm non-overlapping ranges between paradigms: Copilot [75.8, 94.3]%, Devin [49.9, 71.2]%, Cursor [44.1, 69.2]% versus OpenAI [10.7, 11.6]%. Even at the lower bounds, Collaborator tools exceed OpenAI's upper bound by 3.8–6.5×. Remaining circularity risks are discussed in Section 6.

4.3 RQ3. What do automation-authorized merges reveal about the boundaries of the taxonomy?

Approach. S2 and S5 are the only scenarios with Agent-classified approvers: 54 PRs total (S2: 14; S5: 40), 0.24% of merged PRs. We perform exhaustive path analysis of all 54, classifying each by phase sequence.

Results. We partitioned the 54 PRs by path. In 33 cases (61%), a human completed review before a bot account executed the merge. There the derived approver classification records the merge *mechanism* (automation performed the click), not the governance *decision* (the human reviewer had already accepted the change). These PRs function as S1 or S4: the governance decision was human, and the bot merely executed the merge.

The remaining 21 PRs (39%) merged directly with no recorded review phase. Many repository setups can yield the same GitHub

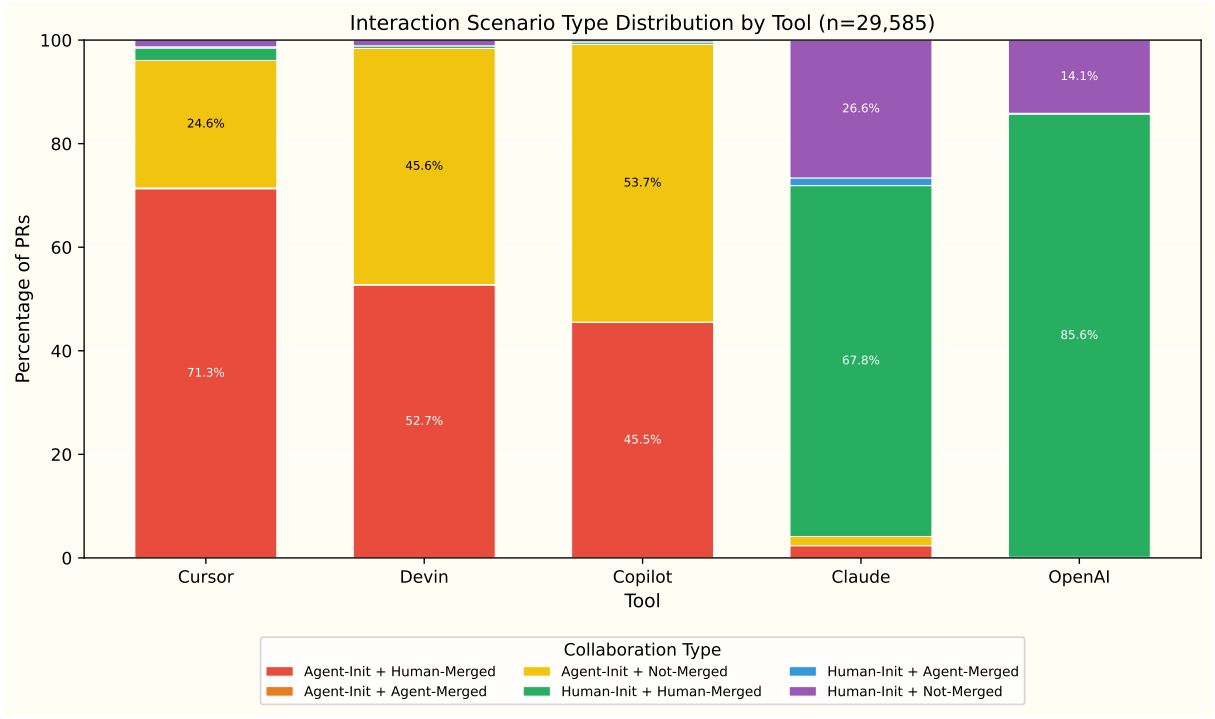

Figure 3: Interaction scenario distribution by tool (29,585 included PRs). Stacked bars show percentage per tool in S1–S6. Copilot has 0.8% Human-Init (S4+S6; 23 PRs).

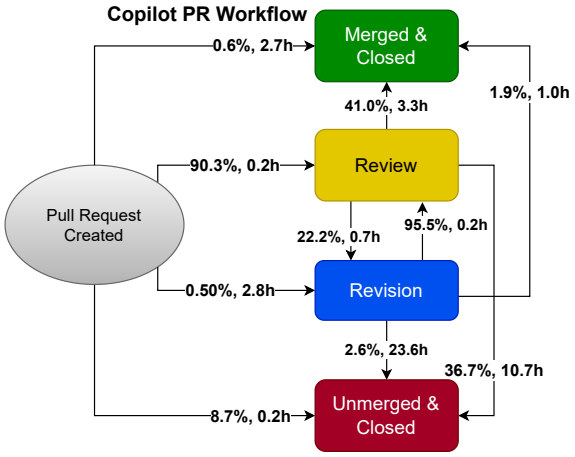

Figure 4: Copilot PR workflow: phase transition probabilities and median hours per state.

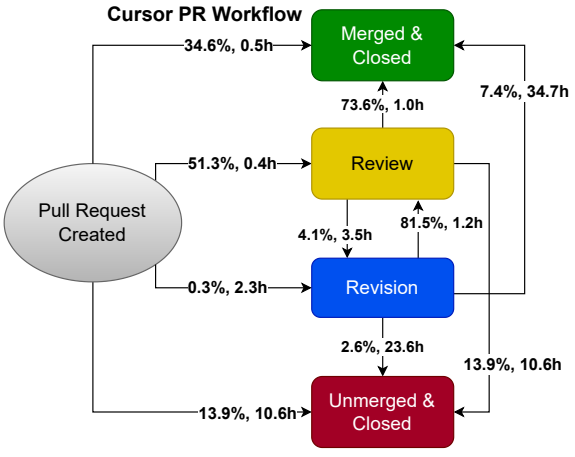

Figure 5: Cursor PR workflow: phase transition probabilities and median hours per state.

event pattern for those merges. Human-configured auto-merge after green checks, one-off waivers of branch protection, and CI/CD merge automation can each produce a bot-attributed merge in the log. Timeline data alone do not distinguish those benign explanations from a merge driven by genuinely autonomous agent governance. We therefore treat 21 of 29,585 PRs (0.07%) only as an *upper* bound on autonomous authority; without branch-protection and merge-policy metadata, that bound is plausibly high.

Tool concentration. S5 concentrates in OpenAI repositories (32/40; 80%), consistent with CI/CD auto-merge infrastructure. S2 distributes evenly (Devin 5, OpenAI 5, Copilot 2, Cursor 2).

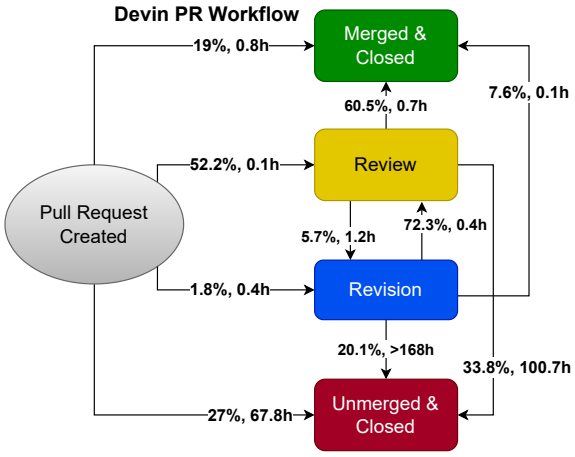

Figure 6: Devin PR workflow: phase transition probabilities and median hours per state.

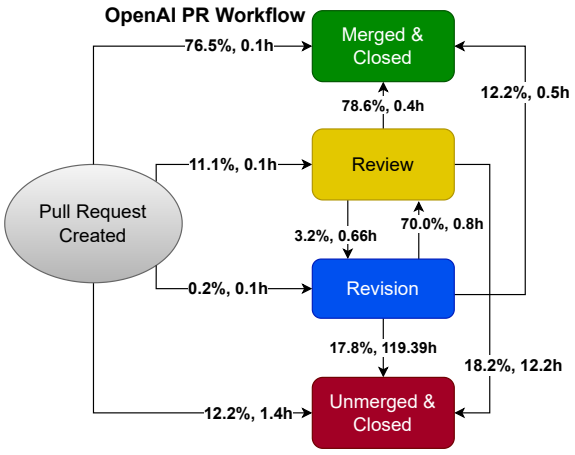

Figure 7: OpenAI PR workflow: phase transition probabilities and median hours per state.

Implication. S2/S5 rarity has at least three separable sources: *technical capability* (GitHub Actions can execute merges; the infrastructure exists), *tool default exposure* (most coding agents do not by default surface merge authority to the agent), and *organizational trust* (where exposure exists, teams may still withhold the affordance). Our event logs localize the joint outcome but cannot separate these layers (Section 5). Analysts using the taxonomy can identify these scenarios and track governance delegation as it evolves.

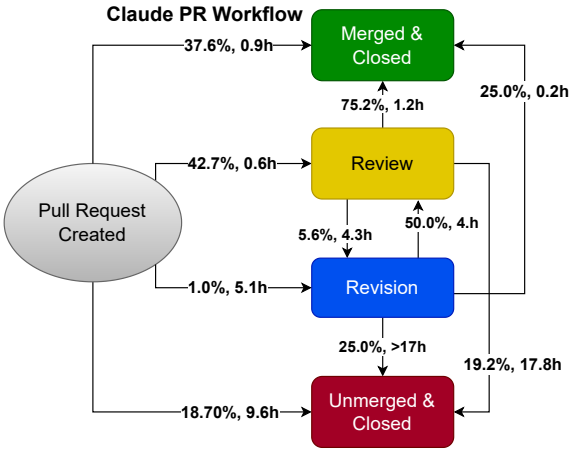

Figure 8: Claude PR workflow: phase transition probabilities and median hours per state.

Summary of RQ 3

Of 54 PRs with Agent-classified approvers (S2+S5, 0.24% of merged), 61% passed through human Review before a bot executed the merge; "Agent-Approved" thus reflects merge *mechanism*, not governance *decision*. The remaining 39% merged directly; the same event pattern can reflect human-configured automation (auto-merge, protection waivers, CI/CD) or genuinely autonomous agent governance, and timelines alone do not separate those cases. The taxonomy captures who *executes* the merge but not who *decides*; resolving this requires repository-level metadata unavailable in the dataset.

5 Discussion

5.1 Two Paradigms of Human-Agent Collaboration

The Collaborator–Assistant spectrum reflects two distinct modes of human-agent integration.

Collaborator Paradigm. Cursor, Devin, and Copilot show agent-initiated PRs routed through review, aligning with Raisch and Krakowski's "augmentation" model [20]: agents generate output autonomously, and humans validate through formal review. Review routing rates (Section 4) confirm that deliberate oversight is structurally embedded in these workflows.

Assistant Paradigm. OpenAI and Claude invert this pattern: human-initiated PRs with higher direct resolution. Phase transition analysis reinforces the contrast: Collaborators show higher review routing and substantive revision loops, whereas Assistants resolve directly.

Repository governance as confound. These patterns, however, do not necessarily reflect developer choice alone. Direct resolution can reflect repository governance as much as behavioral preference. GitHub's *branch protection rules* [8] are repository-level policies that require pull request reviews, mandate passing status

checks, and restrict who can merge; they shape whether a PR receives review before closure. Protected main or release branches often require approving reviews, while feature branches may permit direct merges, and administrators can bypass restrictions unless explicitly included. Thus the same observed lifecycle can reflect tool behavior, repository policy, or their interaction.

Our checks separate these interpretations where the data allow. Among human-initiated PRs only, Wilson score 95% confidence intervals for review routing remain non-overlapping between paradigms, with Collaborator tools exceeding OpenAI's upper bound by 3.8–6.5× (Section 4). The within-repository control (Section 6) confirms that the initiation gap persists (96.8 percentage points), but review-routing contrasts partly reverse. The safest interpretation is therefore asymmetric: the Collaborator–Assistant distinction is strongest for redistributed initiative, while review routing also reflects repository adoption and governance context. Automation complacency [9, 17] is consistent with the data but not confirmed. RQ3 supports the same caution: 61% of automation-authorized merges involved human review, so "Agent-Approved" often reflects infrastructure, not autonomous endorsement. OpenAI's 76.5% direct resolution is consistent with accepting agent output without deliberate review; team-level review norms may matter as much as tool design.

The Collaborator–Assistant spectrum. Claude has a *split profile*: it meets the Assistant threshold by initiation pattern (≥95.6% human-initiated), yet its review behavior resembles Collaborator tools: 37.6% direct resolution, closer to Cursor (34.6%) than to OpenAI (76.5%). This underscores that initiation mode and evaluation intensity are separable dimensions, and that the Collaborator–Assistant classification captures *dominant tendencies* along a spectrum rather than rigid categories. Claude's classification as Assistant rests on its initiation pattern (≥95.6% human-initiated) and should be interpreted with caution given $n = 338$; replication with a larger Claude corpus is needed before drawing firm conclusions about its position on the spectrum.

5.2 The Governance Boundary

The "junior teammate" pattern describes a *current empirical state*, not an architectural constraint. GitHub Actions can already submit reviews, approve changes, and execute merges, so the infrastructure exists; branch protection rules, tool defaults, and organizational trust determine whether such authority is exposed and used. RQ3's 54 automation-authorized merges are among the earliest observed instances of end-to-end lifecycle automation in production repositories, and their rarity reflects where organizations currently draw the line between human and automated merge authority. As tools advance from Level 5 (execute after approval) toward Level 6+ (execute autonomously) [24], the question shifts from *how is work partitioned?* to *how should authority be delegated?* Mined event logs do not answer this question. That gap is the limitation that RQ3's measurement boundary makes explicit.

5.3 Practical Implications

The Collaborator–Assistant spectrum informs three operational decisions.

Review capacity planning. Collaborator tools generate agent-initiated PRs that route through human review at high rates (Copilot 90.3%, Cursor 51.3%, Devin 52.2%). Teams deploying these tools can use sojourn times (e.g., Copilot median 3.0 h in Review) and revision return rates (e.g., $P(\text{Revision} \mid \text{Review}) = 0.222$ for Copilot) as sprint-planning inputs for review workload.

Risk-based review strategies. In Assistant tool workflows, direct resolution dominates (OpenAI 76.5%). Rather than mandating review for all PRs, teams may consider risk-based strategies, such as reviewing PRs that exceed complexity thresholds (e.g., file count, lines changed) while allowing low-risk PRs to resolve directly.

Graduated trust models. The governance gap (0.07% autonomous merges) marks where organizations currently draw the line on automated merge authority. Tool designers and platform administrators can use this baseline to calibrate graduated trust: auto-merge for low-risk, well-tested changes and human approval for high-risk modifications, rather than blanket autonomy or blanket restriction.

5.4 Future Research

Three directions follow from the measurement boundary and the Collaborator–Assistant spectrum.

From mined logs to live API data. Resolving RQ3's measurement boundary requires data not present in mined event logs: branch protection rules, required-reviewer policies, merge-queue configurations, and bot permission scopes, available through live platform APIs. The AIDev dataset lacks branch protection metadata, and GitHub's API requires admin access to query protection rules (`403` without `administration=read` scope), making retrospective estimation infeasible at scale. Our proxy reasoning (if branch protection were the sole driver of review routing, all tools in protected repositories would show similar rates) is suggestive but incomplete, as tool adoption and governance structures may co-vary. Acquiring live API data (protection rules, merge-queue configurations) is the most critical methodological priority.

Quality and outcome analysis. Correlating S1–S6 with commit complexity, test coverage, and post-merge defect rates would test whether direct resolution trades off quality.

Review content analysis. Examining reviewer requests in Collaborator revision loops would distinguish substantive quality concerns from mandatory process steps that add delay without improving the change. Mixed-methods work (interviews, surveys) would address *why* teams choose direct resolution over formal review [9, 17].

6 Threats to Validity

Internal validity. Bot identification uses naming-pattern heuristics (0.36% error rate vs. `actor.type` metadata). Timestamps reflect GitHub server time, not offline activity. The principal threat is repository governance confounding (Section 5): direct resolution rates may partly reflect repository permission structures.

External validity. The dataset spans 2018–2025 but concentrates temporally: Q2 2025 accounts for 55.4% of PRs, Q3 for 36.5%, Q1 for 4.9%, and Q4 2024 for 0.2%. This 94% concentration in Q2–Q3 2025 reflects tools reaching production maturity in early

2025; findings represent a cross-sectional snapshot, not longitudinal trends. Tool-specific repository selection introduces potential bias. Copilot's exclusion rate (2,107/4,971; 42.4%) warrants scrutiny. Median creation dates for included vs. excluded PRs differ by one day (2025-06-25 vs. 2025-06-24), mitigating temporal bias; however, 63.3% of excluded PRs (1,334/2,107) hit the 30-event API pagination cap, truncating terminal events. Exclusion reflects a *data collection artifact*: excluded PRs are more active PRs whose longer timelines exceeded the pagination window. The included sample may underrepresent complex, multi-event lifecycles; this is a conservative bias, since those PRs likely had richer review-revision cycling. OpenAI comprises 70.5% of the analytic sample. To verify the spectrum is not a sample-size artifact, we downsampled OpenAI to $n = 4{,}885$ across 10 random seeds: Cramér's V remained stable at 0.510 ± 0.001 (range 0.5096–0.5108), and the Collaborator–Assistant bifurcation held in all seeds. As a within-repository control, we restricted analysis to 34 repositories where each Collaborator and Assistant tool had at least five PRs ($n = 2{,}197$ PRs). Agent-initiated shares remained bifurcated (Collaborator 97.1%, Assistant 0.3%), a 96.8 percentage-point gap matching the full dataset—consistent with a tool-design interpretation of initiation. The share of merges with a human approver reversed relative to the cross-repository pattern (Assistant 59.6% vs. Collaborator 49.7%), suggesting that RQ2 review-routing contrasts partly reflect repository adoption and governance, not only tool-intrinsic routing.

Construct validity. The taxonomy captures *who*, not *why*. A circularity risk exists: tools clustered by initiation rate are then analyzed for review frequency, a potential downstream consequence. To mitigate this, we compared human-initiated PRs exclusively (Section 4): Collaborators still route most through review while OpenAI resolves directly, with non-overlapping Wilson 95% confidence intervals. Within-repository analysis partially addresses repository-level confounding for RQ2; initiation contrasts remain stable across specifications. Revision counts do not capture revision quality. RQ3 shows the Approver dimension conflates merge *mechanism* with governance *decision*; resolving this requires governance metadata unavailable in the dataset.

Endorsement measurement. Our event-log measure of *endorsement* records the occurrence of a `reviewed` event, not its semantic depth; a 30-second Looks Good To Me (LGTM) and a thorough code review are observationally identical in the timeline data. Distinguishing substantive review from cursory approval requires content-level analysis (comment text, lines reviewed, and change-request substance) that we leave to future work. The endorsement axis introduced in Section 1 is therefore a theoretical distinction that present logs can *localize* but not *measure*.

Content-level controls. We do not control for PR size, complexity, or task type. If tool affordances systematically interact with PR characteristics—for example, Collaborator tools generating larger or more structurally complex changes—the review-routing contrast in RQ2 could partly reflect content rather than tool design. Pinna et al.'s [18] task-type stratification represents the natural next step for disentangling these effects.

7 Conclusion

29,585 PR lifecycles across five tools reveal a Collaborator–Assistant spectrum (Cramér's V = 0.50): agents initiate work, yet terminal merge authority remains human. These findings represent a snapshot of AI-assisted development during its early production maturity (2025 Q2–Q3); longitudinal replication will be necessary to track how these patterns evolve. Teams and researchers can use the taxonomy and five state machines as a reusable framework for review allocation and tool selection.

The 54 automation-authorized merges (0.24% of merged PRs) expose a measurement boundary between merge *mechanism* and governance *decision*; resolving it requires live platform API data (Section 5).

Acknowledgments

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), RGPIN-2021-03969.

References

[1] S. Agarwal, H. He, and B. Vasilescu. AI IDEs or Autonomous Agents? Measuring the Impact of Coding Agents on Software Development. In *Proc. 23rd Int. Conf. Mining Software Repositories (MSR '26)*, 2026. Mining Challenge Track. arXiv:2601.13597.

[2] B. AlMulla, M. Assi, and S. Hassan. Understanding the Challenges and Opportunities of Generative AI Apps: An Empirical Study. arXiv preprint arXiv:2506.16453, 2025.

[3] A. Baird and L. M. Maruping. The Next Generation of Research on IS Use: A Theoretical Framework of Delegation to and from Agentic IS Artifacts. *MIS Quarterly*, 45(1), pp. 315–341, 2021. https://doi.org/10.25300/MISQ/2021/15882

[4] J. Cohen. *Statistical Power Analysis for the Behavioral Sciences.* 2nd ed., Lawrence Erlbaum Associates, 1988.

[5] J. St. B. T. Evans and K. E. Stanovich. Dual-Process Theories of Higher Cognition: Advancing the Debate. *Perspectives on Psychological Science*, 8(3), pp. 223–241, 2013. https://doi.org/10.1177/1745691612460685

[6] H. Gao, P. Banyongrakkul, H. Guan, M. Zahedi, and C. Treude. On Autopilot? An Empirical Study of Human–AI Teaming and Review Practices in Open Source. In *Proc. 23rd Int. Conf. Mining Software Repositories (MSR '26)*, 2026. Mining Challenge Track. arXiv:2601.13754.

[7] A. Fügener, J. Grahl, A. Gupta, and W. Ketter. Will Humans-in-the-Loop Become Borgs? Merits and Pitfalls of Working with AI. *MIS Quarterly*, 45(3b), pp. 1527–1556, 2021. https://doi.org/10.25300/misq/2021/16553

[8] GitHub. About Protected Branches. GitHub Docs, 2024. https://docs.github.com/en/repositories/configuring-branches-and-merges-in-your-repository/managing-protected-branches/about-protected-branches

[9] E. Glikson and A. W. Woolley. Human Trust in Artificial Intelligence: Review of Empirical Research. *Academy of Management Annals*, 14(2), pp. 627–660, 2020. https://doi.org/10.5465/annals.2018.0057

[10] G. Gousios, M. Pinzger, and A. van Deursen. An Exploratory Study of the Pull-based Software Development Model. In *Proc. 36th Int. Conf. on Software Engineering (ICSE)*, pp. 345–355, 2014. https://doi.org/10.1145/2568225.2568260

[11] D. F. Halpern. *Thought and Knowledge: An Introduction to Critical Thinking*, 5th ed. Psychology Press, 2014.

[12] J. Hollan, E. Hutchins, and D. Kirsh. Distributed Cognition: Toward a New Foundation for Human-Computer Interaction Research. *ACM Trans. Comput.-Hum. Interact.*, 7(2), pp. 174–196, 2000. https://doi.org/10.1145/353485.353487

[13] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian. The Promises and Perils of Mining GitHub. In *Proc. 11th Working Conf. Mining Software Repositories (MSR)*, pp. 92–101, 2014. https://doi.org/10.1145/2597073.2597074

[14] H. Li, H. Zhang, and A. E. Hassan. AIDev Challenge: Can You Predict Merge Decisions of AI-Coding-Agent Generated Pull Requests? In *Proc. IEEE/ACM Int. Conf. Mining Software Repositories (MSR), Mining Challenge*, pp. 1–5, 2025. arXiv:2504.20423.

[15] 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. In *Proc. ACM Joint European Software Engineering Conf. and Symp. on the Foundations of Software Engineering (ESEC/FSE)*, pp. 1–22, 2025. arXiv:2502.11387.

[16] A. F. Nogueira and M. Zenha-Rela. Monitoring a CI/CD Workflow Using Process Mining. *SN Comput. Sci.*, 2(448), 2021. https://doi.org/10.1007/s42979-021-00830-2

[17] K. Okamura and S. Yamada. Adaptive Trust Calibration for Human-AI Collaboration. *PLOS ONE*, 15(2), e0229132, 2020. https://doi.org/10.1371/journal.pone.0229132

[18] G. Pinna, J. Gong, D. Williams, and F. Sarro. Comparing AI Coding Agents: A Task-Stratified Analysis of Pull Request Acceptance. In *Proc. 23rd Int. Conf. Mining Software Repositories (MSR '26)*, 2026. Mining Challenge Track. arXiv:2602.08915.

[19] S. Rahman, M. F. Rabbi, and M. F. Zibran. A Task-Level Evaluation of AI Agents in Open-Source Projects. In *Proc. 23rd Int. Conf. Mining Software Repositories (MSR '26)*, 2026. Mining Challenge Track. arXiv:2602.02345.

[20] S. Raisch and S. Krakowski. Artificial Intelligence and Management: The Automation-Augmentation Paradox. *Academy of Management Review*, 46(1), pp. 192–210, 2021. https://doi.org/10.5465/amr.2018.0072

[21] A. Roychoudhury et al. Agentic AI Software Engineers: Programming with Trust. arXiv preprint arXiv:2502.13767, 2025. https://doi.org/10.48550/arXiv.2502.13767

[22] V. A. Rubin, A. A. Mitsyuk, I. A. Lomazova, and W. M. van der Aalst. Process Mining Can Be Applied to Software Too! In *Proc. 8th ACM/IEEE Int. Symp. on Empirical Software Engineering and Measurement (ESEM)*, pp. 1–4, 2014. https://doi.org/10.1145/2652524.2652583

[23] V. A. Rubin and S. A. Shershakov. System Runs Analysis with Process Mining. In *Proc. 30th IEEE/ACM Int. Conf. on Automated Software Engineering Workshops (ASEW)*, pp. 48–51, 2015.

[24] T. B. Sheridan and W. L. Verplank. Human and Computer Control of Undersea Teleoperators. Tech. Rep., MIT Man-Machine Systems Laboratory, 1978.

[25] C. Treude, M.-A. Storey, and J. Weber. Empirical Studies on Collaboration in Software Development: A Systematic Literature Review. Tech. Rep. DCS-352-IR, University of Victoria, 2012.

[26] C. Treude and M. A. Gerosa. How Developers Interact with AI: A Taxonomy of Human-AI Collaboration in Software Engineering. In *Proc. 2nd IEEE/ACM Int. Conf. on AI Foundation Models and Software Engineering (Forge 2025)*, 2025. https://doi.org/10.1109/Forge66646.2025.00033

[27] W. M. P. van der Aalst. *Process Mining: Data Science in Action*, 2nd ed. Springer, 2016.