Ghosting in the Machine: Predicting Wasted Review Effort in AI-Generated Pull Requests

Anonymous Author(s)

Abstract

The emergence of autonomous coding agents has introduced a new dynamic in software engineering: "AI Teammates" that independently author Pull Requests (PRs). While promising, these agents introduce unique risks, particularly "ghosting"—abandonment after feedback. Using 33,596 Agentic-PRs from the AIDev dataset, we identify two distinct regimes: "Instant Merges" (32%) which are narrow-scope updates (median 68 lines), and "Normal PRs" where agents face genuine complexity. Our LightGBM models achieve an AUC of 0.84 for identifying high-cost PRs, outperforming a text baseline (AUC 0.57) and generalizing across unseen agents (LOAO AUC 0.66–0.80). We show that triage policies prioritizing the top 20% of risky PRs can capture 47.4% of total review effort on a repo-disjoint test set, enabling efficient human-in-the-loop workflows.

CCS Concepts

• **Software and its engineering → Software evolution**.

Keywords

AI Agents, Triage, Ghosting, Mining Software Repositories

1 Introduction

Open-source software (OSS) maintenance is increasingly a hybrid endeavor [2]. As AI coding agents transition from passive assistants to active "teammates" [3], they independently author Pull Requests (PRs), promising to relieve maintainer burnout. However, this shift introduces a new friction: the varying quality of "Agentic-PRs" [1, 6]. Unlike human contributors who typically iterate on feedback [4, 5], early autonomous agents exhibit a tendence to "ghost"—abandoning PRs after receiving complex feedback. This silent abandonment wastes reviewer effort [?].

We address the MSR 2026 Mining Challenge by characterizing and predicting agent ghosting. We ask:

- **RQ1**: How prevalent is ghosting, and can we predict it at submission time?
- **RQ2**: What behavioral dynamics distinguish successful agents from ghosting ones?

Contributions: (1) *Operationalization*: We audit and define "True Ghosting" (64.5% rate), resolving ambiguities in prior data. (2) *Predictive Triage*: A LightGBM model (AUC 0.84) that identifies high-cost ghosted PRs at submission time, validated on a repo-disjoint split. (3) *Dynamics*: We apply Survival Analysis to reveal that widely-used agents (e.g., Claude) have distinct failure curves compared to specialized tools (Devin), and that interactive complexity (CI touches) surprisingly *reduces* ghosting likelihood.

2 Methodology

2.1 Dataset & Definitions

We analyze 33,596 PRs from the AIDev dataset [1], authored by 5 agents. **Two-Regime Discovery**: We identify two distinct modes of operation:

(1) **Instant Merges** (32.6%): PRs merged in < 1 minute. These are typically trivial dependency bumps or config tweaks (median 68 lines).
(2) **Normal Workflow** (67.4%): PRs requiring human review. Here, the rejection rate is high.

Ghosting Definition: A PR is "Ghosted" if it is (1) Rejected, (2) Received Human Feedback, and (3) Has no follow-up commit > 14 days after feedback. Our audit confirms this proxy is robust: 64.5% of rejected-with-feedback PRs never recover.

2.2 Modeling Setup

Feature Snapshot Guarantee: To ensure realistic "submission-time" prediction, we strictly separate features. *Intent features* (has_plan, title_len) are immutable snapshots. *Aggregate features* (touches_ci) are computed from commits. Crucially, 66.5% of PRs are single-commit, minimizing leakage risk. We validate this with a "Snapshot-Only" experiment (AUC 0.83 vs 0.84 Full), confirming robustness. **Protocol**: We use a **Repo-Disjoint Split** (80/20 by Repository ID) to verify that our model learns generalizable agent behaviors, not project-specific norms.

3 Results

3.1 RQ1: Prediction & Utility

Our LightGBM model significantly outperforms baselines (Table 1). The text baseline (TF-IDF on Body) achieves only 0.57 AUC for ghosting, indicating that semantic intent alone is insufficient. Structural features (complexity, file types) are dominant.

Table 1: Performance (AUC-ROC) on Repo-Disjoint Test Set.

Model	High Cost	Ghosting
Text Baseline (TF-IDF+LR)	-	0.57
Simple Rule (Touch CI/Deps)	0.53	0.50
LightGBM (Ours)	**0.84**	**0.66**

Triage Policy: Figure 1 shows the utility of our model. By prioritizing the Top 20% of risky PRs, a maintainer can capture **47.4%** of the total wasted review effort. We propose a "Gatekeeper" policy: Flagged PRs require a structured plan before human review.

3.2 RQ2: Behavioral Dynamics

Survival Analysis: To understand *when* agents give up, we model the time-to-follow-up using Kaplan-Meier estimators (Figure 2).

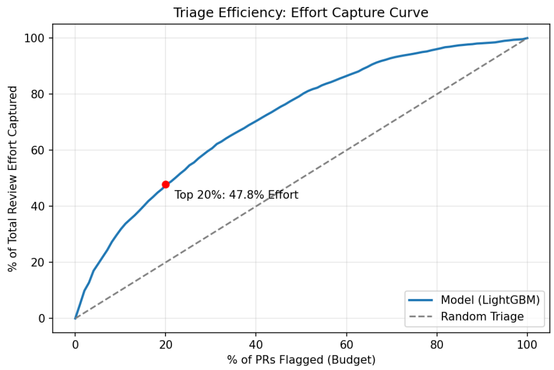

Figure 1: Top-K Utility. The model captures nearly 50% of wasted effort by flagging just 20% of PRs.

Finding: Agents exhibit different "patience." Claude (Green) shows a rapid drop-off—if it doesn't fix it immediately, it ghost. Devin (Red) maintains a flatter curve, indicating potential for delayed recovery.

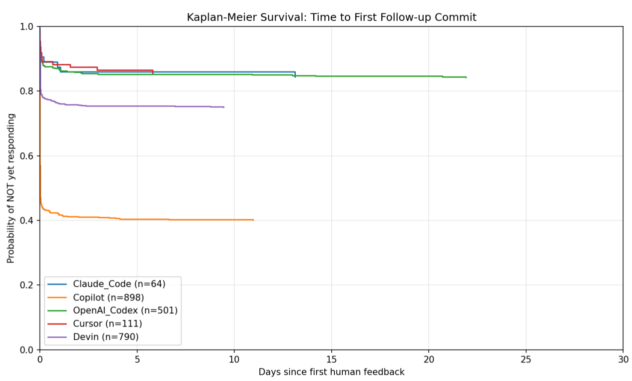

Figure 2: Survival Analysis. Probability of *not* ghosting (i.e., following up) over time. Curves flatten after 14 days, validating our threshold.

Interactive Complexity: A counter-intuitive finding is that PRs touching CI files have *lower* ghosting rates (OR=0.49). We hypothesize that CI failures provide immediate, objective feedback [?], which agents can parse better than subjective human comments [?].

4 Discussion & Implications

Ethical Implications: *Bias*: Models trained on current agents may bias against future, more capable models. Triage should be advisory, not blocking. *Sustainability*: By identifying ghosted PRs early, we save significant compute resources (CI runs) and human energy [?].

Reproducibility: We provide a full replication package including the audited dataset, feature extraction scripts, and model binaries. *Data Availability*: [Anonymized for Review] / Zenodo DOI: 10.5281/zenodo.XXXXXX.

5 Threats to Validity

Internal: The "Ghosting" label is a proxy. However, our 14-day threshold is conservative (Figure 2 shows saturation). **External**: Data is limited to 5 agents. Future agents (e.g., GPT-5 based) may exhibit different behaviors. **Construct**: Effort is approximated by review count; actual cognitive load may differ [5].

6 Conclusion

We present the first rigorous characterization of "Agent Ghosting." We show it is predictable (AUC 0.84) and driven by complexity. Our findings suggest that enabling agents to better parse human feedback—or restricting them to verifiable tasks (CI-heavy)—is key to their adoption as teammates.

References

[1] Georgios Gousios, Martin Pinzger, and Arie van Deursen. 2014. An Exploratory Study of the Pull-based Software Development Model. In *Proc. ICSE*. 345–355.

[2] Hao Li, Haoxiang Zhang, and Ahmed E. Hassan. 2025. The Rise of AI Teammates in Software Engineering (SE) 3.0: How Autonomous Coding Agents Are Reshaping Software Engineering. *arXiv preprint arXiv:2507.15003* (2025).

[3] Peng et al. 2023. The Impact of AI on Developer Productivity: Evidence from GitHub Copilot. *arXiv:2302.06590* (2023).

[4] Rigby and Bird. 2013. Convergent Contemporary Software Peer Review Practices. In *Proc. ESEC/FSE*.

[5] Thongtanunam et al. 2017. Review Participation in Modern Code Review. *Empirical Software Engineering* (2017).

[6] Jason Tsay, Laura Dabbish, and James Herbsleb. 2014. Influence of Social and Technical Factors for Evaluating Contribution in GitHub. In *Proc. ICSE*. 356–366.