AIDev Dataset Challenge: Studying AI Coding Agents on GitHub

[Do Clarification-Seeking AI Agents Ship Better Code? Answering AIDev RQ 5(a), 5(b), and 5(c)]

Chrisogonus Nwaiwu,
Federal University of Technology Owerri, Nigeria, University of Sunderland, UK
{chrisogonusc.nwaiwu@gmail.com}

***Abstract* —** Studied merge outcomes for agentic pull requests (PRs) in the AIDev dataset and address RQ 5: (a) failure patterns and touched paths; (b) predictive value of early textual signals; and (c) security-related risks. Using N = 932,791 agentic PRs, overall acceptance is 84.71%. An early textual cue—`clarify strict` in the PR title/body—correlates with lower acceptance: 84.78% without the cue vs 70.24% with the cue (−14.54 pp raw). A bootstrap places the acceptance-rate gap at [−15.80, −13.27] pp (95% CI). A regularized logistic model controlling for repository popularity and PR size estimates a −20.61 pp change in acceptance probability when toggling the cue; model AUC = 0.503 (near chance), indicating limited predictive power from this single signal. For RQ 5(a), high-prevalence patterns (e.g., `pat_docs` 20.2%, `pat_build_ci` 18.4%, `pat_lint` 16.4%) associate with lower acceptance (e.g., `pat_docs` −7.01 pp; `pat_revert_hotfix` −6.22 pp; `pat_build_ci` −4.00 pp). For RQ 5(c), 5.68% of PRs are security-flagged; acceptance is 76.69% vs 85.19% for non-flagged PRs; same-day median time-to-merge in both groups (0.00 vs 0.00 days at rounding). Results are associational, not causal, and remain directionally stable across top languages and in popular repositories (stars ≥ 100).

1. SCOPE OVERVIEW

Agentic software development tools, powered by advancements in artificial intelligence, are increasingly taking on responsibilities that were once exclusively human-driven. These tools can now autonomously propose code modifications, generate detailed descriptions of their intent, and even submit pull requests (PRs) to shared repositories without direct developer intervention. While this automation offers the promise of accelerated development cycles and reduced manual effort, it also raises critical questions about trust, reliability, and integration into collaborative workflows. Development teams must carefully assess where and why such PRs succeed or fail, since wasted review cycles on low-quality submissions could diminish the net benefits of automation.

A central challenge lies in identifying the conditions under which agentic PRs are more likely to be rejected. These failures may stem from recurring patterns, such as modifications to particularly sensitive or high-risk parts of the codebase, or from issues related to style, correctness, or maintainability. By studying these patterns, teams can build guidelines for safer and more productive use of automated contributors. Additionally, the textual descriptions accompanying AI-generated PRs often contain early signals that may predict eventual outcomes. The clarity, detail, and justification in these descriptions may play a significant role in how human reviewers perceive the value and correctness of the contribution.

Equally important is the role of security in this emerging paradigm. Autonomous tools may inadvertently introduce vulnerabilities if their changes are not thoroughly vetted, but they may also propose patches that mitigate risks overlooked by human developers. Understanding how teams handle such security-related contributions—whether they are prioritized, scrutinized more heavily, or systematically rejected—offers insight into the evolving collaboration between humans and agentic systems.

This work therefore addresses RQ5 by exploring: (a) what failure patterns or touched code paths correlate with PR rejection; (b) how informative early textual signals are for predicting acceptance; and (c) whether agentic PRs introduce or mitigate security risks, and how teams handle these cases in practice.

2. DATASET UTILIZATION

Made use of the AIDev pull-request table (agent-generated PRs). After minimal cleaning, the analysis sample contains N = 932,791 PRs with title/body text and merge outcome indicators. Where present, we use repository popularity proxies (stars/watchers), basic diffs (additions/deletions, changed files), timestamps, and language tags.

3. FEATURES AND OPERATIONABILITY

Outcome. accepted ∈ {0,1} via robust merge detection using merged booleans, merged_at timestamps, and state/status='merged'. Early textual signal (5b). clarify_strict = 1 if the PR title/body (after stripping code blocks/URLs/inline code) contains both a question mark and a question word (what/when/why/how/…). Controls. Repository popularity log_stars = log1p(stars), PR size pr_size_log1p = log1p (additions + deletions), and language (categorical). Failure patterns & touched paths (5a). Textual patterns: pat_docs, pat_build_ci, pat_lint, pat_revert_hotfix, pat_test_fail. Touched paths: touch_docs, touch_tests, touch_src, touch_deps, and docs_only (docs touched but not src/tests).
Security signals (5c). sec_text (security keywords such as 'security', 'CVE', 'auth', 'encrypt', 'token'), sec_dep_bump (dependency-bump phrasing + lock/manifest touched), sec_secret_terms (secrets keywords), security_flag = 1 if any of the above.

4. PROCESS METHODOLOGY

i. **Descriptive Analysis:** Started with descriptive statistics to establish baseline patterns in the dataset. Specifically, computed group means for key variables and estimate acceptance-rate gaps using non-parametric bootstrap resampling with 1,000 iterations. This allows me to quantify variability and provide confidence intervals around observed differences.

ii. **Modeling Approach:** To assess predictors of pull request acceptance, I employed a regularized logistic regression model with L2 penalties to prevent overfitting. The feature set includes a binary signal of textual clarity (clarify_strict), repository popularity log-transformed (log_stars), pull request size as a log-transformed measure (pr_size_log1p), and programming language encoded via one-hot indicators. Reported both the marginal change in acceptance probability when toggling clarify_strict and the overall predictive performance measured by AUC. For research questions 5(a) and 5(c), I additionally run per-signal logistic models with identical controls to assess directional associations.

iii. **Robustness Checks:** Finally, I verified the consistency of results across the most common programming languages and within a high-visibility subset of repositories (stars ≥ 100).

5. PATTERN PRESENTATION

i. RQ5(b): **Early textual signal → acceptance** Overall acceptance: 84.71%. Group rates: 84.78% (no clarify_strict) vs 70.24% (with clarify_strict). Raw difference: −14.54 pp. Bootstrap 95% CI: [−15.80, −13.27] pp. Regularized logit (controls: popularity, size, language): toggling clarify_strict is associated with −20.61 pp change in acceptance probability; AUC = 0.503. Interpretation: the clarification cue correlates with lower merge likelihood; AUC ≈ 0.50 shows the cue alone has weak predictive power and likely captures confounded scenarios.

ii. RQ 5(a): **Failure patterns and touched paths** Prevalence (textual patterns): pat_docs 20.2%, pat_build_ci 18.4%, pat_lint 16.4%. Acceptance deltas (pp): pat_docs −7.01, pat_revert_hotfix −6.22, pat_build_ci −4.00 (clarify-agnostic raw differences). Per-signal logits (with controls) preserve negative signs. Interpretation: Documentation-heavy, build/CI-oriented, and revert/hotfix-tagged PRs are less likely to merge.

iii. RQ5(c): **Security-related signals** Prevalence: 5.68% of PRs are security-flagged. Acceptance: 76.69% (flagged) vs 85.19% (not flagged). Time-to-merge (merged only): 0.00 vs 0.00 days (same-day at rounding). Focused logit (controls) indicates a negative association between the security flag and acceptance. Interpretation: Security-adjacent PRs are accepted less often, though merges—when they occur—are not detectably slower at available precision.

6. RESOLUTION AND CONCLUSION

i. **Robustness & Sensitivity:** Top languages. The direction of the clarify_strict association remains negative when restricting to the most frequent languages. Popular repositories (stars ≥ 100). The negative association persists.
Model stability: - Regularization prevents

separation; AUC ~ 0.50 indicates early cues alone are insufficient predictors.

ii. **Threats to Validity:** Measurement error: - Heuristics (clarification, security, patterns, paths) are lexical proxies and can misclassify rhetorical or indirect cues; touched-path inferences are coarse.
Confounding: - Unobserved factors (change complexity, agent identity/config, reviewer load, test coverage) may drive both clarifying language and rejections.
Selection:- Missing features (e.g., file lists, labels) may bias prevalence estimates. **External validity: -** Results reflect agentic PRs in AIDev and may not be transferred to all orgs or time periods.

iii. **Practical Implications**

Triage dashboards should treat early textual cues (clarification, security terms) as risk hints, not decision rules. Agent prompting could discourage unnecessary question-tone in PR descriptions for mature changes and encourage crisp, spec-style summaries with reproduction steps and test evidence. Security PRs may benefit from templates that surface CVE links, patch minimality, and test diffs to increase reviewer confidence.

iv. **Reproducibility**

Data: AIDev PR table. Key artifacts produced: acceptance-by-clarification table, pattern/path prevalence and acceptance deltas, security acceptance comparison, and logistic summaries. Modeling details: regularized logistic regression (L2), features = clarify_strict, log_stars, pr_size_log1p, one-hot language; marginal Δprob reported by toggling the target feature; bootstrap with 1,000 resamples for acceptance-rate gaps. Code: Colab notebook cells; results exported to .md and .docx.

v. **Conclusion**
Across N = 932,791 agentic PRs, we find: (5b) An early clarification cue in PR text correlates with lower acceptance (−14.54 pp raw; −20.61 pp in a controlled model), but has weak standalone predictive power (AUC ≈ 0.50). (5a) Common failure/path signals (docs/build/CI/revert) are negatively associated with acceptance. (5c) Security-flagged PRs are less likely to be accepted, with no detectable difference in median time-to-merge at available precision. These findings suggest that how agentic changes are framed and what they touch can signal review friction, while emphasizing that single early cues should inform, not decide, triage.

REFERENCES:

Bird, C., Nagappan, N., Murphy, B., Gall, H., & Devanbu, P. (2011). Don't touch my code! Examining the effects of ownership on software quality. Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering (ESEC/FSE 2011), 4–14. https://doi.org/10.1145/2025113.2025119

Bishop, C. M. (2006). Pattern recognition and machine learning. Springer.

Efron, B., & Tibshirani, R. J. (1993). An introduction to the bootstrap. Chapman & Hall/CRC.

Fan, R. E., Chang, K. W., Hsieh, C. J., Wang, X. R., & Lin, C. J. (2008). LIBLINEAR: A library for large linear classification. Journal of Machine Learning Research, 9, 1871–1874.

Jiang, Y., Li, L., Kim, S., & Cheung, S. C. (2013). Recommending commits via code repository mining. Proceedings of the 10th Working Conference on Mining Software Repositories (MSR 2013), 221–230. https://doi.org/10.1109/MSR.2013.6624030

Mockus, A., Fielding, R. T., & Herbsleb, J. (2002). Two case studies of open-source software development: Apache and Mozilla. ACM Transactions on Software Engineering and Methodology (TOSEM), 11(3), 309–346. https://doi.org/10.1145/567793.567795

Vasilescu, B., Filkov, V., & Serebrenik, A. (2015). Perceptions of diversity on GitHub: A user survey. Proceedings of the 8th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE 2015), 50–56. https://doi.org/10.1109/CHASE.2015.12

Xu, W., Zhang, Y., & Yin, H. (2018). Toward trustworthy AI development: Mechanisms for improving reliability and security of machine learning systems. arXiv preprint arXiv:1810.07416.