

# **Behavioral Analysis of AI Coding Agents in Software Engineering: An Empirical Study Using the AIDev Dataset**

CHONGWEN SUN, University of British Columbia, Canada

INAARA RAJWANI, University of British Columbia, Canada

AI coding agents (e.g., Copilot, Claude Code, Cursor, Devin) are increasingly submitting complete pull requests (PRs) across open-source repositories. Yet their testing behavior, review responsiveness, and merge characteristics remain poorly understood. Using the 2025 snapshot of the AIDev dataset, this study investigates: (RQ1) the extent to which agents write tests, (RQ2) the types of review comments they receive and resolve, and (RQ3) whether early PR features can predict acceptance. Results show that test inclusion is extremely rare (1–5%), correctness dominates reviewer concerns, and PR size is strongly associated with acceptance. These findings provide actionable implications for the design and governance of Agentic Software Engineering (SE 3.0).

Additional Key Words and Phrases: AI Coding Agents, Pull Requests, AIDev Dataset, Empirical Study

## **1 Introduction**

AI coding agents have evolved from completing single lines to autonomously modifying multiple files, running tests, and submitting PRs. This transformation marks a shift toward *Agentic Software Engineering*, where AI systems participate directly in human workflows. Despite rapid adoption, fundamental uncertainties remain: Do AI agents produce reliable tests? How do reviewers respond to their changes? And can maintainers quickly determine whether an AI-generated PR is likely to be merged?

To address these questions, we analyze thousands of AI-generated PRs across diverse open-source ecosystems using the AIDev dataset. This study focuses on three core research questions.

## **2 Dataset Description**

We use the 2025 AIDev dataset [1], a large-scale collection of AI-generated PRs. Four structured components support our analysis:

- **PR metadata** (`pull_request.parquet`): titles, descriptions, timestamps, and acceptance outcomes.
- **Commit-level patches** (`pr_commit_details.parquet`): line-level diffs for computing churn and detecting test files.
- **Reviewer comments** (`pr_review_comments_v2.parquet`): text feedback linked to file regions.
- **Commit sequences** (`pr_commits.parquet`): revision histories used to detect comment resolution.

Human PRs are removed using AIDev's agent-identification rules. PR acceptance is represented as a binary label.

## **3 Methodology**

Our methodological design follows the three research questions and integrates file-level, comment-level, and PR-level signals into a unified analysis pipeline.

---

Authors' Contact Information: Chongwen Sun, chongwen.sun1992@gmail.com, University of British Columbia, Canada; Inaara Rajwani, inaarari11@gmail.com, University of British Columbia, Canada.

### 3.1 RQ1: Detecting Test Files

Test-related behavior is studied by identifying test files introduced or modified by AI agents. We detect tests using a combination of directory-based (e.g., `tests/`, `test/`, `__tests__/`) and filename-based heuristics (e.g., `*_test.py`, `.spec.ts`). For each PR, we aggregate detections across all commits to compute: `contains_test` and `test_file_count`. This enables PR-level behavior comparisons across tasks and agents.

### 3.2 RQ2: Classifying Review Comments and Resolution

Review comments reveal where agents fall short. We classify comments into six categories—correctness, style, documentation, security, testing, other—using keyword matching guided by prior empirical studies. A comment is considered resolved when a subsequent commit touches the referenced region, linking comment metadata with commit sequences. This allows comment-level resolution rates to be computed across categories and agent types.

### 3.3 RQ3: Early PR Feature Extraction

We extract early-accessible PR features that do not require reviewer input: description length, churn (additions+deletions), number of modified files, and presence of tests. These variables allow us to analyze whether structural PR characteristics correlate with acceptance outcomes prior to review.

## 4 Results

### 4.1 RQ1: Test Inclusion Is Rare and Unstable

Test inclusion among AI-generated PRs is extremely low. Across all agents and task types, only 1–5% of PRs contain any test files. When tests are produced, they tend to appear in large batches (typically 8–17 files), suggesting that test-writing is driven by template-triggered generation rather than incremental or behavior-driven additions. This variability indicates that test-writing is not a stable capability and likely depends on contextual cues present in certain repositories.

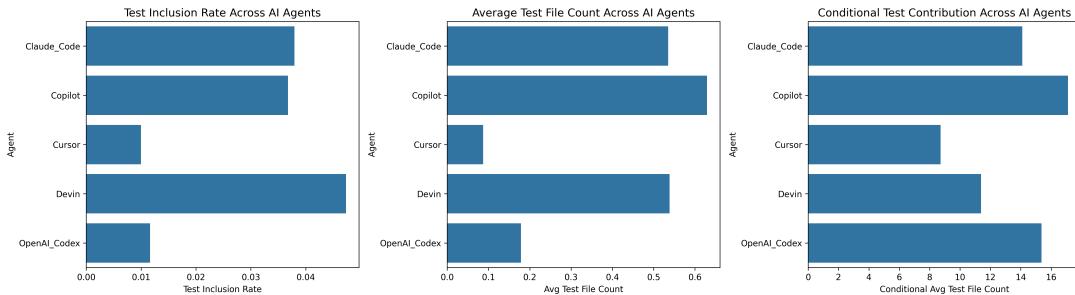


Fig. 1. RQ1: Test inclusion rate, average test count, and conditional test contribution across AI agents.

To further understand the contextual effects behind test-writing, Figure 2 examines test inclusion by task type. Agents are most likely to write tests in chore and build PRs, while documentation- or CI-related PRs almost never contain any tests. This reinforces that test-writing is driven more by PR category than by agent identity.

Behavioral Analysis of AI Coding Agents in Software Engineering:  
An Empirical Study Using the AIDev Dataset

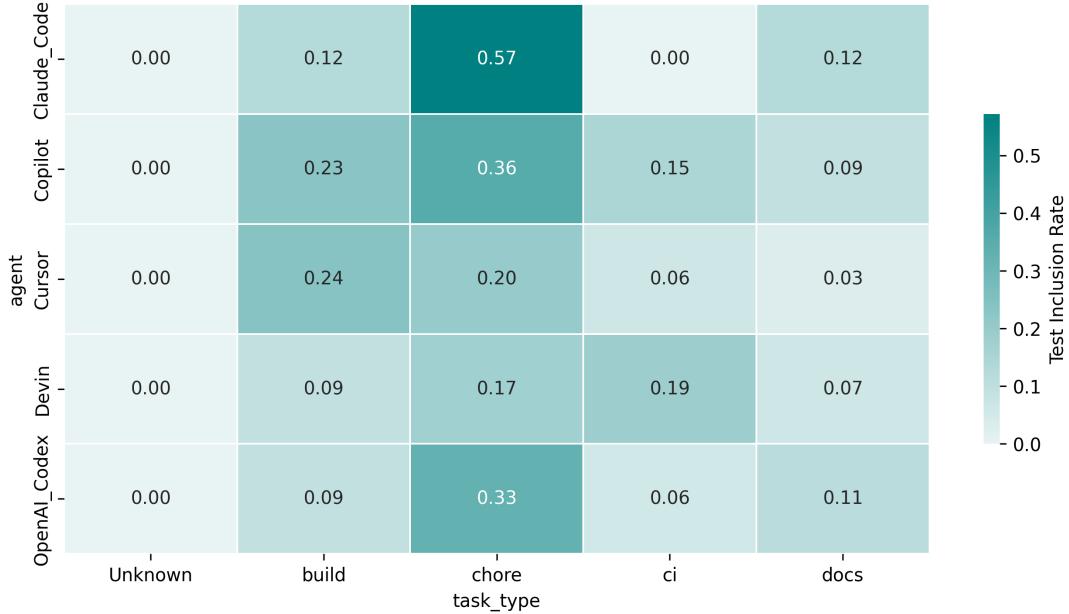


Fig. 2. Test inclusion rate by agent and task type. Agents rarely write tests outside of build or chore tasks.

Lastly, we present a consolidated behavior profile across all AI agents (Figure 3). Devin and Claude\_Code show slightly higher inclusion rates, but conditional test counts remain large for all agents. This confirms that test-writing is irregular and often driven by large template-based insertions.

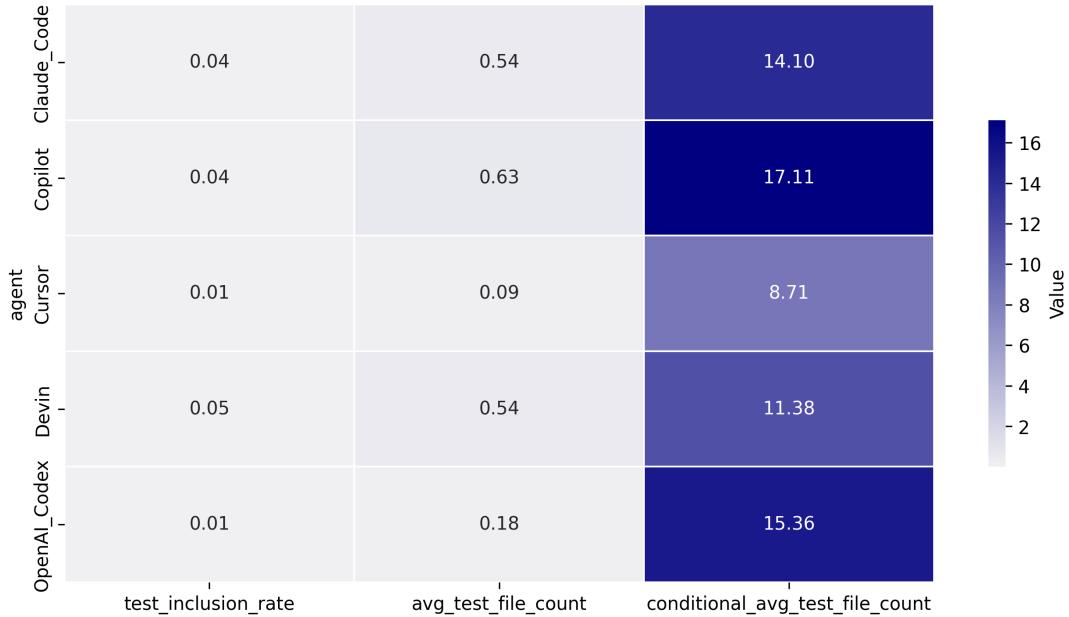


Fig. 3. Overall testing behavior profile across agents, including inclusion rate, average test count, and conditional test count.

Overall, RQ1 shows that AI-generated PRs contain tests only rarely, and when they do, test contributions are unstable, context-dependent, and typically generated in large batches. These findings highlight the need for explicit prompting or built-in agentic self-testing mechanisms to ensure reliable validation behavior.

#### 4.2 RQ2: Review Comment Types and Resolution Difficulty

To understand how maintainers react to AI-generated changes, we examine two aspects of review interactions: (1) the *types* of comments reviewers provide, and (2) how effectively AI agents resolve these comments in subsequent commits.

Reviewer comments are classified into six categories—correctness, style, documentation, security, testing, and other—using a keyword-based scheme informed by prior work. This enables us to identify which issues attract the most reviewer attention.

Figure 4 shows that correctness-related feedback overwhelmingly dominates across agents, indicating frequent logic errors or incomplete implementations. Style and documentation comments also occur but at much lower frequencies, suggesting that reviewers prioritize semantic correctness over surface-level concerns.

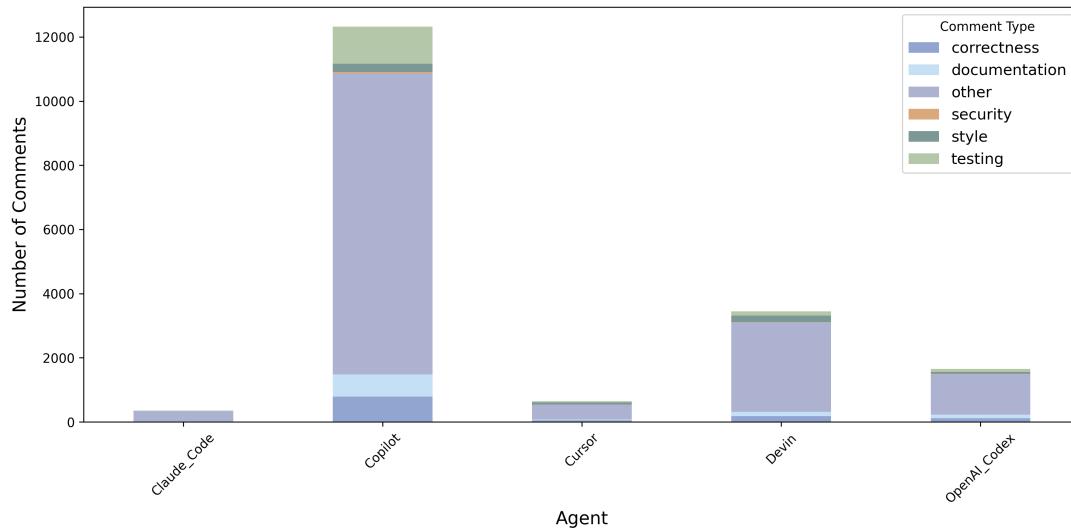


Fig. 4. RQ2: Distribution of review comment types across agents. Correctness issues dominate reviewer feedback.

Next, we analyze how often agents successfully resolve reviewer feedback. A comment is considered resolved if a subsequent commit modifies the referenced code region. This metric captures the agents’ capacity to understand feedback, adjust code behavior, and converge toward reviewer expectations.

As illustrated in Figure 5, correctness and style comments have the lowest resolution rates across agents. These categories often require multi-file reasoning, semantic understanding, or adherence to nuanced project conventions—competencies that current AI agents struggle with. In contrast, documentation- and testing-related comments are resolved more reliably, likely because they involve localized edits rather than conceptual revisions.

Behavioral Analysis of AI Coding Agents in Software Engineering:  
An Empirical Study Using the AIDev Dataset

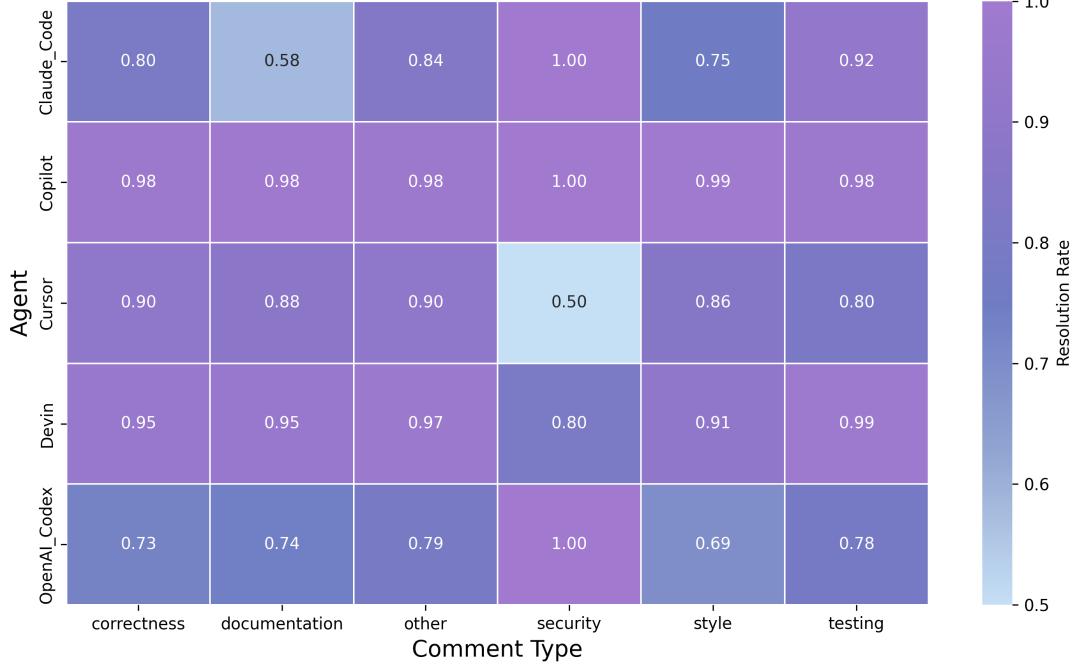


Fig. 5. RQ2: Resolution rates by comment type and agent. Correctness and style issues are the most difficult for agents to fix.

To further investigate cross-agent variation, we compare per-agent resolution rates in Figure 6. Some agents (e.g., Copilot, Claude\_Code) show consistently higher resolution rates for simpler comment types, while others struggle more uniformly. These differences indicate that agent architectures or training data may influence revision behavior, especially when addressing complex feedback.

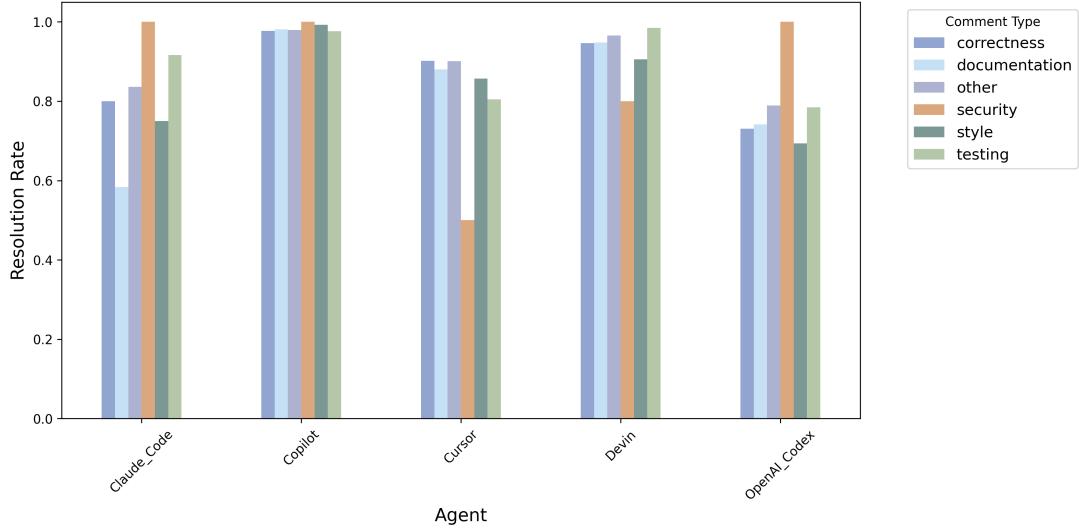


Fig. 6. Comparison of comment resolution rates across agents.

Finally, Figure 7 aggregates multiple performance dimensions—comment distribution, resolution rates, and difficulty profiles—into a unified radar chart. This holistic perspective highlights relative strengths, such as certain agents handling documentation edits well, and weaknesses, such as consistently poor correction of semantic issues.

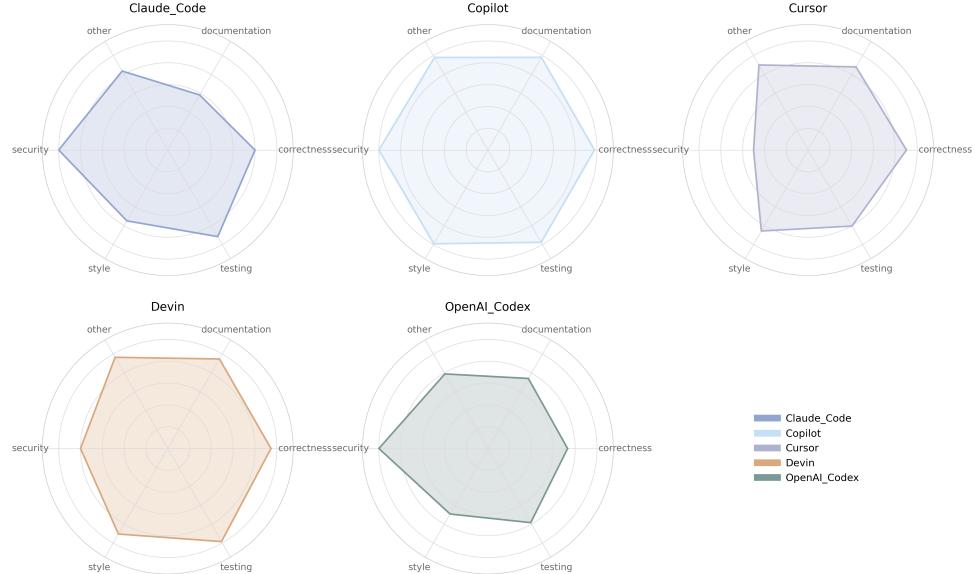


Fig. 7. Radar visualization showing cross-agent performance differences across comment types.

Taken together, these analyses show that **correctness issues are both the most frequent and the most difficult for AI agents to resolve**. This finding reveals a fundamental limitation: current agents excel at producing syntactically valid code but struggle with deeper semantic reasoning, multi-step corrective edits, and understanding project-specific behavioral constraints.

### 4.3 RQ3: Early Acceptance Signals

We examine whether acceptance outcomes can be inferred from early PR features available before review begins. Using PR metadata and commit-level diffs, we extract four structural indicators: description length, churn (additions+deletions), number of modified files, and presence of tests.

Figure 8 compares these features across accepted and rejected PRs. Rejected PRs consistently show higher churn, touch more files, and contain longer descriptions, suggesting that large or wide-scope changes are perceived as riskier and more difficult to review. In contrast, accepted PRs tend to be smaller and more self-contained. Test presence shows no clear predictive value, largely because test-writing is extremely rare (as established in RQ1).

## Behavioral Analysis of AI Coding Agents in Software Engineering: An Empirical Study Using the AIDev Dataset

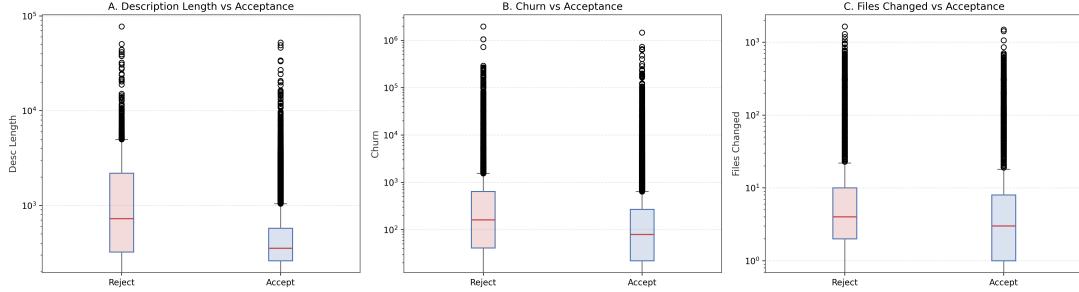


Fig. 8. **RQ3:** Early PR feature distributions for accepted vs. rejected PRs. Larger and higher-churn PRs show substantially lower acceptance rates.

Overall, early PR structure provides strong acceptance signals: **smaller, focused PRs** are more likely to be merged, whereas **large, high-churn PRs** face substantially higher rejection rates.

## 5 Implications for Agentic SE

Our findings provide actionable insights for the emerging paradigm of **Agentic Software Engineering** (SE 3.0):

**1. Reinforcing Automated Self-Testing for AI Agents.** The extremely low test-writing rate suggests that autonomous agents require built-in self-testing mechanisms. Integrating automated self-evaluation pipelines or mandating unit-test templates could mitigate risk before PR submission.

**2. Improving Semantic Reasoning and Error Correction.** Correctness issues dominate reviewer concerns and are least likely to be resolved. This highlights a key weakness in agentic reasoning. Future models must emphasize program semantics, multi-file reasoning, and project context understanding.

**3. Early PR Structure as a Governance Signal.** Early PR features—especially churn and file scope—strongly predict acceptance. Maintainers can use these features for automated triage, prioritization, and gating policies for AI-generated PRs.

**4. Designing Mixed-Initiative Review Workflows.** Since agents struggle with complex revisions, hybrid workflows where agents propose changes and humans provide semantic oversight can improve reliability.

**5. Toward Standards for Agent-Generated Contributions.** Given variability across agent types, repositories may need explicit policies for AI-generated PRs, including size limits, test requirements, and verification protocols.

## 6 Ethical Considerations

Only public GitHub data is used; no personal or private information is included. Analyses focus on aggregate agent behavior rather than profiling individual developers.

## 7 Conclusion

This study presents a multi-dimensional behavioral analysis of AI coding agents across testing, review dynamics, and early acceptance signals. Findings offer concrete guidance for designing and governing Agentic Software Engineering workflows.

## References

- [1] Hao Li et al. 2025. *AIDev: A Large-Scale Dataset of AI-Generated Pull Requests*. <https://huggingface.co/datasets/hao-li/AIDev>