Mining Software Repositories (MSR) — Mining Challenge 2026

Comparing AI Coding Agents: A Task-Stratified Analysis of Pull Request Acceptance

Giovanni Pinna[1], Jingzhi Gong[2], David Williams[3], Federica Sarro[3]

[1]University of Trieste, Italy
[2]King's College London, UK
[3]University College London, UK

Problem

Many AI tools exist, but systematic comparisons that account for two crucial factors are lacking: task type (feature, bugfix, documentation, etc.) and evolution over time.

The methodological risk is that an agent primarily handling "easy" tasks like documentation (which naturally has higher acceptance rates) may appear superior to one tackling complex tasks like new feature implementation, even if the latter is technically more capable.

Why It Matters

Practitioners, tool developers, and researchers make decisions based on these evaluations. Biased metrics lead to wrong adoption choices, misguided development priorities, and flawed evaluation methodologies.

Research Questions

RQ1: Does AI coding agent performance evolve over time?
We investigate whether agents demonstrate measurable changes in acceptance rates over extended observation periods.

RQ2: What factors are associated with performance?
We examine how task type, review frequency, and other factors correlate with acceptance rates, and whether differences in task distribution across agents confound global comparisons.

RQ3: How do different agents compare?
We perform task-stratified statistical comparisons of five major AI coding agents to identify differences in performance.

Methodology

Data filtering: we analyze 7,156 pull requests from the AIDev dataset, filtered from an initial 33,596 by retaining only closed PRs from permissively licensed (MIT/Apache-2.0) repositories that received at least one external review, ensuring each contribution underwent meaningful human evaluation.

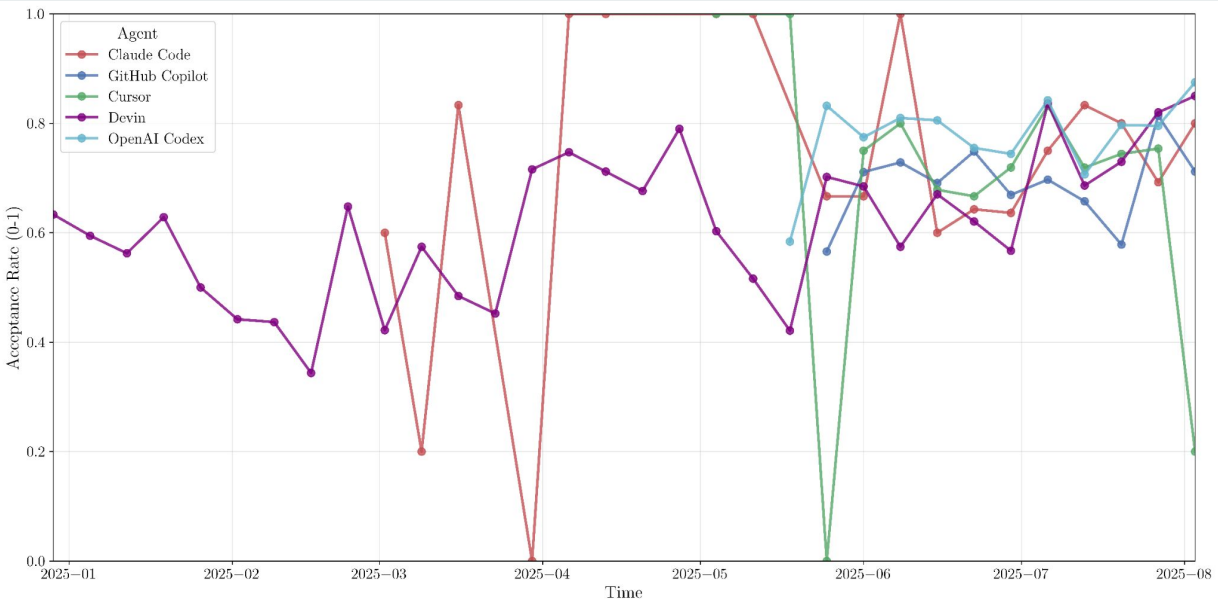
RQ1 — Temporal analysis: linear regression measures weekly change rate in acceptance rate per agent (R^2 for trend strength), complemented by LOESS smoothing to capture non-linear patterns.

RQ2 — Performance factors: acceptance rates computed across 12 task categories, with 552 task-stratified observations (unique agent-task-week combinations). Review frequency also examined.

RQ3 — Stratified agent comparison: pairwise Chi-square tests within each task category (Fisher's exact test for sparse cells), with Bonferroni correction ($\alpha \approx 0.00078$) for 64 stratified comparisons and phi coefficient (φ) for effect size.

Sensitivity analysis: results validated on the 11-week window common to all agents, with examination of within-agent task distribution shifts over time.

Key Finding 1 (RQ1)



Only Devin shows consistent improvement (+0.77%/week, R^2=0.34), rising from approximately 60% to 80%. The LOESS curve suggests three phases: initial, growth, and stabilization. The other agents maintain stable rates from the start, with Codex and Copilot showing a high-performance plateau from their first observed week.

Key Finding 2 (RQ2)

Table 2: RQ2. Acceptance rate by task type. MAR: Mean acceptance rate; SD: standard deviation; Obs: # weekly agent-task observations.

	chore	docs	style	ci	build	refactor	feat	fix	test	revert	perf	other
MAR	**84.0%**	82.1%	78.1%	75.0%	72.5%	71.2%	66.1%	66.0%	61.5%	60.0%	55.4%	0.0%
SD	36.7%	38.4%	42.0%	43.5%	44.8%	45.3%	47.3%	47.4%	48.8%	54.8%	50.2%	0.0%
Obs	58	71	25	41	40	69	81	77	53	4	29	4

Task type is the dominant factor: 29pp gap between chore (84%) and perf (55.4%), and 16pp between docs (82.1%) and feat (66.1%) among high-volume tasks. This gap exceeds inter-agent variance within the same category. Moreover, task distributions vary substantially across agents — Copilot handles 41.6% fix tasks, Claude Code 52.5% features — making global comparisons misleading. A correlation (not causal) between review frequency and acceptance is also observed: Copilot receives 4.94 reviews/PR with 68% acceptance, Codex 1.39 reviews/PR with 77.9%.

Key Finding 3 (RQ3)

No single agent dominates all categories. OpenAI Codex is the most consistent (59.6%–88.6%) and leads in fix (83%) and refactor (74.3%). Claude Code leads in docs (92.3%) and feat (72.6%), but with a limited sample (139 PRs). Cursor excels in fix (80.4%) and test (77.8%). Of 64 stratified tests, only 6 are significant after Bonferroni correction, and 5 involve fix tasks — where Devin is systematically weaker (φ=0.39 vs Codex, medium effect).

Conclusion

In conclusion the task type dominates (29pp gap), no single agent is best overall (Codex most consistent, Claude Code and Cursor superior in specific categories), and only Devin improves over time.

For Practitioners: agent choice matters most for bugfix and testing (large gaps); for documentation differences are minimal. Combined use of different agents for different workflows is recommended.

For Researchers: task stratification should become standard practice, always reporting task distributions. Acceptance rate alone is insufficient — complementary metrics are needed (static analysis, complexity, technical debt) along with mixed-effects models to account for repository-level clustering.

For Tool Developers: results highlight agent-specific weaknesses (e.g., Devin in bugfix), suggesting where to focus improvements. Fix and testing are the areas with the greatest innovation potential.

Future Work: integrate code quality metrics, extend to more diverse repositories, and adopt more sophisticated statistical models.