A Comparative Analysis of Test Contributions by Agentic AIs

1. Introduction

Autonomous coding agents are actively reshaping software engineering by accelerating contributions, altering workflows, and raising new challenges around trust, quality, and governance.[4] It has already changed the way people used to code and has shown a new path in the software engineering industry. Agentic coding tools like Claude Code are already producing widely accepted contributions in open-source projects.[5] Therefore, it has become extremely important to evaluate the Ai agents and the quality of generated codes. The AIDev dataset provides the empirical foundation for studying this transformation and guiding the next generation of human AI collaboration in software development.[4] By combining planning, memory, reflection, and multi-agent coordination, Ai Agents promise transformative applications across industries.[6]

2. Motivation

While AI agents autonomously generate, modify, and submit code, test generation is shown to be a critical but underexplored area compared to code generation.[3][5] The current testing practices for AI agents are rational but incomplete.[1] Through evaluating GPT-3.5/4 and CodeX for generating Java and Python unit tests it is concluded that LLM tests are syntactically correct but often semantically weak, missing edge cases and deeper logic.[7] However, there are multi-agents like AgentCoder that sets a new benchmark in multi-agent code generation by combining simplicity with rigorous testing and refinement. [2] Therefore, this brings the necessity of this statistical study where we find the percentage of test case coverage by Agentic Ai. Furthermore, we also discuss the contribution of 5 coding agents in test cases. For this data analysis we used AIDev dataset which mainly consists of pull requests from ai agents and humans.[4]
We tried to answer 2 RQ's from the MSR mining challenge 2026. The RQ's mainly hover around the contribution of Ai agents in the test coverage. The two RQ's are given below:
RQ1: How frequently do Coding Agents contribute tests? What types of tests are most common?
RQ2: How do different Agentic AIs vary in their test-related contributions?
While the first question discusses the overall contribution of coding agents and the common type of tests, the second question finds out the contribution in test coverage of 5 available coding agents in the AIDEV dataset.

3. Methodology and Result

3.1.1 Approach of RQ1

To investigate how often Coding Agents contribute tests and what kinds of tests they produce, we analyzed agent-authored pull requests from the *hao-li/AIDev* dataset. We combined basic information about each pull request with details about the files changed in its commits so that we could examine what each agent actually contributed. We then identified test files by looking for common signs that a file is used for testing, such as whether it appears in a test-related directory or follows typical testing naming conventions (for example, filenames ending in *.test.js, _test.py*, or located in *tests/* folders). When filenames were unclear, we also looked at short descriptions in commit messages for clues about the type of test being added. After identifying these test files, we grouped them into broad categories such as unit

tests, integration tests, or end-to-end tests based on their naming patterns and the parts of the system they appeared to target. Finally, we summarized how many pull requests included at least one test file and what types of tests were most commonly contributed. These descriptive summaries allowed us to understand the overall testing behavior of Coding Agents without relying on more complex statistical or machine-learning techniques.

3.1.2 Findings of RQ1

From a total of 33,580 agent-authored pull requests, we found that 8,372 of them (24.93%) included at least one test file. This shows that Coding Agents do generate tests, but they do so in only about one out of every four PRs, indicating that test creation is not yet a consistent part of their development behavior.
When looking at the types of tests contributed, most files were placed into the "unknown" (589,632 files) or "other" (97,331 files) categories. These represent test-related files whose names or paths suggest testing activity but do not clearly indicate a specific type. Among the test files that could be confidently categorized, integration tests were the most common (12,041 files), followed by unit tests (4,349 files). More specialized test types appeared far less frequently, including performance tests (3,164), benchmark tests (2,366), and end-to-end tests (2,036). Very few regression (905) or smoke tests (99) were identified.

TABLE I
TEST TYPE AND COUNT FOR AGENTIC PR

	Test Type	Count
0	unknown	589632
1	other	97331
2	integration	12041
3	unit	4349
4	performance	3164
5	benchmark	2366
6	End to end	2036
7	regression	905
8	smoke	99

Overall, these results show that Coding Agents contribute tests in a meaningful but limited way. Their testing output is dominated by test files that cannot be easily classified, and when specific types are present, they tend to focus on integration- and unit-level testing. More advanced or specialized tests appear relatively rarely. This suggests that while Coding Agents are capable of generating tests, their test contributions are still narrow in scope and uneven across different testing needs.

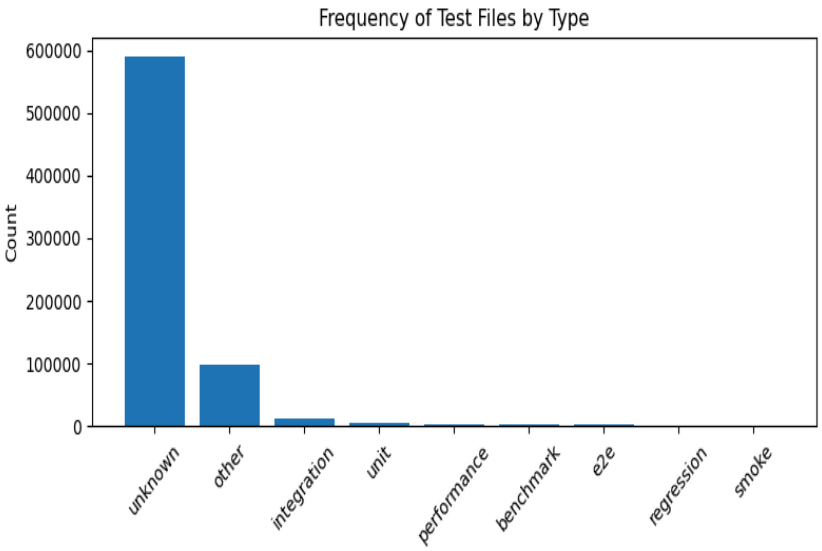

Fig 1: Frequency of test by type.

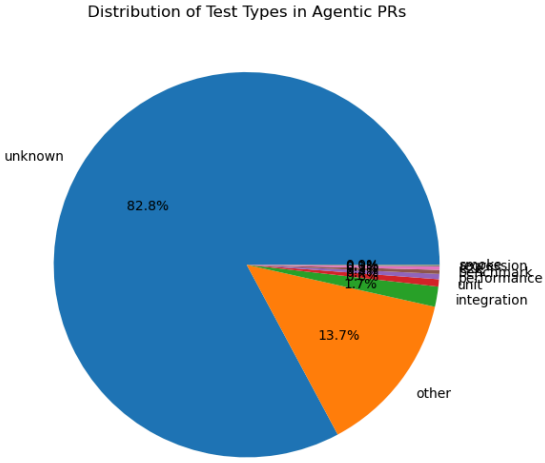

Fig 2: Distribution of test types in agentic pr

3.2.1 Approach of RQ2

To examine how different Agentic AIs vary in their test-related contributions, we analyzed the pull_request split of the hao-li/AIDev dataset, which contains metadata for thousands of PRs submitted by various autonomous AI agents. Because this table does not include file-level diffs, we identified test-related PRs using a keyword-based heuristic commonly used in MSR studies. Specifically, we searched for testing-related terms such as “test”, “unit test”, “integration test”, “pytest”, “junit”, and “assert” within the title and body of each PR. A PR was labeled as test-related if any of these keywords appeared. After labeling, we grouped the PRs by the agent that submitted them and calculated three metrics for each agent: the total number of PRs submitted, the number of test-related PRs, and the percentage of their PRs that involved testing. Finally, we visualized these differences using bar charts to compare both the counts and percentages of test-related PRs across Agentic AIs. The Python code used for this analysis performed dataset loading, test-PR detection, metric calculation, and visualization, ensuring the process is fully reproducible.

3.1.2 Findings of RQ2

Our analysis shows substantial variation in test-related contributions across Agentic AIs. OpenAI Codex demonstrated the strongest emphasis on testing, with 96.24% of its 21,799 pull requests identified as test-related indicating that nearly all of its contributions involve testing activities. Claude Code also showed a high testing focus, with 77.34% of its PRs classified as test-related, followed closely by Copilot at 71.73% and Devin at 69.92%, suggesting that these agents consistently incorporate testing within their

workflows. In contrast, Cursor showed a markedly lower rate, with only 33.03% of its PRs being test-related, making it an outlier among the agents studied.

TABLE II
TEST RELATED PR METRIC BY AGENTIC AI

agent	total_prs	test_prs	test_pr_percentage
Claude_Code	459	355	77.342048
Copilot	4970	3565	71.730382
Cursor	1541	509	33.030500
Devin	4827	3375	69.919204
OpenAI_Codex	21799	20980	96.242947

These results indicate that some AI agents naturally adopt more test-oriented development behavior, while others contribute tests far less frequently. The differences likely reflect variations in model training data, prompting strategies, or system design goals. Overall, the findings highlight that test contribution behavior is not uniform across AI models and that certain agents play a more significant role in supporting software quality assurance than others.

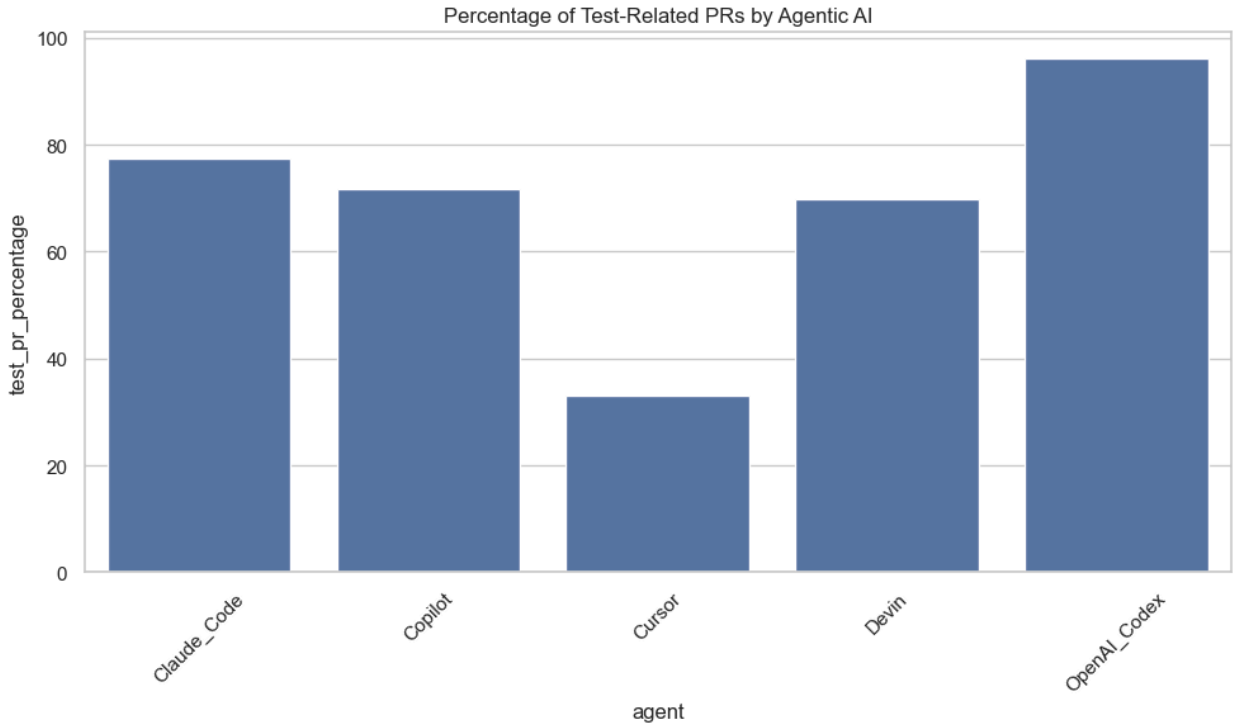

Fig 3: Percentage of Test-Related PRs by Agentic AI.

4. Implications

The findings of RQ1 highlight several important points for both practitioners and researchers. For developers working with Coding Agents, the results suggest that agent-generated pull requests often require additional testing before being merged, especially when changes affect multiple components. Since agents mainly produce unit tests, human developers may still need to write integration and end-to-end tests to ensure full system reliability. For researchers, the imbalance between simple and more advanced test types points to opportunities to improve test generation capabilities in future agent models. Better support for higher-level testing could make autonomous coding tools more trustworthy and reduce manual testing effort. For organizations considering the use of Coding Agents, these results emphasize the need for oversight and complementary testing practices to ensure that agent-produced changes meet quality expectations.

In RQ2 the differences we observe in test-related contributions across Agentic AIs carry several important implications for both software engineering practice and research. Most notably, the exceptionally high proportion of test-focused PRs generated by OpenAI Codex (over 96%) indicates that certain agents are inherently more aligned with quality assurance workflows. Such agents may be especially useful in development environments that depend on reliable test generation or require rigorous validation steps before integration. Conversely, the substantially lower testing engagement demonstrated by Cursor suggests that some agents may not prioritize tests unless explicitly prompted or reinforced. These agents may therefore require additional supervision, more detailed user instructions, or external test-generation mechanisms to ensure adequate coverage. For developers and engineering teams, these findings underscore the importance of selecting AI agents not solely based on general model performance but also on how well they support testing, safety, and long-term code maintainability.

For researchers, the clear behavioral differences across agents reveal that test-related activity is highly model-dependent and influenced by the internal design and training strategies of each AI system. This opens new opportunities to study why certain models naturally incorporate testing, how agent prompting or configuration affects testing behavior, and how the quality of AI-generated tests compares to those written by humans. Tool builders can also leverage these insights to design hybrid AI workflows—for example, deploying one agent to generate or modify functional code while delegating testing responsibilities to an agent that has shown stronger aptitude for producing tests. Overall, the variability in test contributions highlights the need to understand the unique behavioral profiles of Agentic AIs when integrating them into real-world development pipelines, especially in contexts where software reliability is critical.

5. Conclusion

This work shows the basic comparison and analysis of test coverage in agentic AI. There are many spaces for future work. For example: Deep research in finding patterns in different agentic AIs regarding their focus on the test type, compare Agentic AI behavior with human developers to determine whether testing patterns produced by agents meaningfully diverge from real-world development practices. To conclude, this study can be used as a guideline for developers or testers to help choose their coding agent depending on test type.

References:

[1] Rahman, M., Li, H., & others. (2025). An Empirical Study of Testing Practices in Open Source AI Agent Frameworks and Agentic Applications. arXiv preprint arXiv:2509.19185. https://arxiv.org/abs/2509.19185

[2]Huang, D., Bu, Q., Zhang, J. M., Qing, Y., Luck, M., & Cui, H. (2024). AgentCoder: Multi-Agent Code Generation with Effective Testing and Self-optimisation. arXiv preprint arXiv:2312.13010. https://arxiv.org/abs/2312.13010

[3]Mündler, N., Mueller, M. N., He, J., & Vechev, M. (2024). *Code Agents are State of The Art Software Testers*. In *ICML 2024 Workshop on LLMs and Cognition*. OpenReview. https://openreview.net/forum?id=2P3LLI5ofh

[4]Li, H., Zhang, H., & Hassan, A. E. (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. https://arxiv.org/abs/2507.15003

[5]Mündler, N., Mueller, M. N., He, J., & Vechev, M. (2025). *On the Use of Agentic Coding: An Empirical Study of Pull Requests on GitHub*. arXiv preprint arXiv:2509.14745. https://arxiv.org/abs/2509.14745

[6]Bandi, A., Kongari, B., Naguru, R., Pasnoor, S., & Vilipala, S. V. (2025). *The Rise of Agentic AI: A Review of Definitions, Frameworks, Architectures, Applications, Evaluation Metrics, and Challenges*. *Future Internet, 17*(9), 404. https://doi.org/10.3390/fi17090404

[7]1. Pan, X., Yang, Z., Chen, J., Liu, Y., & Xu, B. (2024). Do large language models generate good tests? Proceedings of the IEEE/ACM International Conference on Software Engineering (ICSE).
PDF: https://arxiv.org/pdf/2402.04115