An Empirical Study of Tests in Agentic Pull Requests

Sabrina Haque
Computer Science & Eng. Dept.
University of Texas at Arlington
Arlington, Texas, USA
sxh3912@mavs.uta.edu

Sarvesh Ingale
Computer Science & Eng. Dept.
University of Texas at Arlington
Arlington, Texas, USA
smi1081@mavs.uta.edu

Christoph Csallner
Computer Science & Eng. Dept.
University of Texas at Arlington
Arlington, Texas, USA
csallner@uta.edu

Abstract

Testing is a critical practice for ensuring software correctness and long-term maintainability. As agentic coding tools increasingly submit pull requests (PRs), it becomes essential to understand how testing appears in these agent-driven workflows. Using the AIDev dataset, we present an empirical study of test inclusion in agentic pull requests. We examine how often tests are included, when they are introduced during the PR lifecycle and how test-containing PRs differ from non-test PRs in terms of size, turnaround time, and merge outcomes. Across agents, test-containing PRs are more common over time and tend to be larger and take longer to complete, while merge rates remain largely similar. We also observe variation across agents in both test adoption and the balance between test and production code within test PRs. Our findings provide a descriptive view of testing behavior in agentic pull requests and offer empirical grounding for future studies of autonomous software development.

CCS Concepts

• **Software and its engineering → Software creation and management**.

Keywords

Software testing, agentic AI, coding agent, agentic coding

ACM Reference Format:
Sabrina Haque, Sarvesh Ingale, and Christoph Csallner. 2026. An Empirical Study of Tests in Agentic Pull Requests. In *23rd International Conference on Mining Software Repositories (MSR '26), April 13–14, 2026, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3793302.3793604

1 Introduction

Autonomous coding agents based on Large Language Models (LLMs) are becoming active software development participants. Beyond code completion, these agents now plan tasks, implement features, and submit pull requests (PRs) to third-party code repositories [20]. Open-source platforms such as GitHub already show a growing volume of agentic PRs, commits, and issue resolutions [14, 23]. As these agents take on responsibilities, it becomes important to understand how they follow established software engineering practices.

Software testing is key to increase confidence in software correctness and reliability [12]. Tests help detect bugs, prevent regressions, and support maintainability [10]. Open-source projects commonly encourage contributors to accompany functional changes with appropriate tests [1]. Prior research has studied the adoption and nature of testing in human-authored projects [12, 15, 21]. As development increasingly incorporates automated and AI-assisted contributions, it is important to understand how they handle testing. Recent studies explore the use of LLMs for automated test generation [2, 4], but in isolation, without examining how tests appear and evolve within agent-human workflows.

When autonomous agents act as teammates and contribute code through PRs, their testing behavior becomes just as important as their ability to write functional code. Despite the growing presence of agentic contributors, we currently lack a clear understanding of how often autonomous agents integrate testing into the PR lifecycle and how their testing behavior differs across agents.

To address this gap, we conduct a mining study using AIDev [14]. Our code and data are on Figshare [8]. The AIDev dataset contains about 933k PRs[1] GitHub PRs, created by five major autonomous coding agents (OpenAI Codex, GitHub Copilot, Devin, Cursor, and Claude Code) across 61k repositories [14]. While the original AIDev study characterizes broad properties of agentic PRs, our work focuses specifically on testing practices within these workflows. We analyze the AIDev-pop subset, consisting of 33.5k (i.e. 7% of the AIDev dataset) PRs from repositories with more than 100 stars. This allows us to focus on popular projects where expectations around testing and review are likely to be more clearly established.

In this work, we analyze testing practices in agentic PRs via the following research questions:

- **RQ1:** How often do the agentic PRs include test code, and how is test adoption changing over time?
- **RQ2:** When do test files first appear in agentic pull requests, and are test files introduced early modified later?
- **RQ3:** What characteristics distinguish test-containing agentic PRs from non-test PRs?

2 Dataset Description

We use the AIDev dataset[2] of agent-generated GitHub pull requests [14]. We focus on the AIDev-pop subset containing pull requests from repositories with more than 100 stars.

Our study focuses on PRs that involve test-related changes. As our research questions concern when test files are added, we need the PRs' commit level timestamps. AIDev does not include commit timestamps. To reconstruct commit timestamps for test PRs, we retrieve the `author.date` and `committer.date` fields for each commit using the GitHub REST API [7]. The author date records when

MSR '26, Rio de Janeiro, Brazil

ACM ISBN 979-8-4007-2474-9/2026/04
https://doi.org/10.1145/3793302.3793604

[1] https://github.com/SAILResearch/AI_Teammates_in_SE3/blob/main/AIDev_Challenge.pdf
[2] As downloaded from the official Hugging Face repository [13] on October 29, 2025

a commit was originally authored, while committer date records when the commit was applied to the branch (which may differ due to rebasing, cherry-picking, etc.) [3, 18]. We use the committer date, as it shows when the commit appears in the PR timeline.

For each identified test PR, we collect all associated commits from the `pr_commit_details` table. Each commit is identified by its commit hash (SHA), a unique identifier assigned by Git to each commit. Using the unique `SHA` with repository metadata, we query the GitHub REST API to retrieve missing commit timestamps, yielding each test PR's commit timeline.

3 Methodology

We classify a pull request as a *test PR* if it touches (which we define as adding or modifying—but not deleting or renaming—a file) at least one test file. We exclude merge commits (identified by commit messages starting with "merge", case-insensitive) [17] to avoid generated merge metadata.

The AIDev dataset links each PR to the agent that created it. To identify test files, we use regex-based pattern matching heuristics on file paths and filenames. Previous large-scale studies similarly identify test files using language-agnostic checks, such as the presence of the word "test" in the file path [10, 12] and avoid common false positives (e.g., `contest`). Specifically, to be considered a test file, a file first has to satisfy any of the following criteria.

Name: Path contains a file or directory named `test`, `tests`, `testing`, or `cypress`:
`(^|[\\/])(tests?|testing|cypress)([\\/]|\$)`
E.g.: `/src/tests/`, `a/test/b.c`, `/cypress/foo.js`

Token: File or directory name includes `test` or `spec` as a token delimited by `\\/_.-`
preventing matches such as `contest`:
`(^|[\\/_.-])(test|spec)([\\/_.-]|\$)`
E.g.: `test_math.py`, `user.spec.ts`

Suffix: Directory or file name contains `Test` or `Spec` as a suffix or followed by a dot. The latter captures many test files' basenames:
`(Test|Spec)(\$|\.)`
E.g.: `UserTest.java`, `LoginSpec.go`

We apply case-insensitive matching for name and token, case-sensitive matching for suffix, and non-capturing groups `(?:)` instead of `()` for faster matching. We also exclude several types of non-code files commonly used for documentation or configuration (i.e., `.csv`, `.doc`, `.json`, `.md`, `.mk`, `.rtf`, `.txt`, `.yaml`, and `.yml`).

3.1 Initial vs Updated PR

To distinguish tests introduced with the agentic PR's initial commit(s) from those added via later commits during subsequent PR modifications (e.g., in response to comments), we define when the initial PR is complete. A study on Claude-generated PRs treats the first commit as the initial contribution and any subsequent commits as follow-up modifications [23]. But agentic PRs often contain multiple commits created before the PR is opened, making a single commit insufficient for capturing the agent's initial solution.

Prior work also commonly uses the PR's `created_at` timestamp [11]. But we observe that GitHub Copilot's commits at the time of PR creation do not modify any files (99% of test-containing Copilot PRs in our dataset). As a result, using PR creation time as the cutoff systematically misclassifies Copilot PRs as having empty initial contributions. In contrast, we found 96% of these PRs include a `review_requested` or `ready_for_review` event, indicating a clear workflow boundary before review begins.

We thus define the *initial submission* for Copilot as "all commits up to the first `review_requested` or `ready_for_review` event" and for other agents as "all commits up to the PR's `created_at` timestamp". For the few PRs without a commit before the cutoff or Copilot PRs without relevant events, we treat the earliest non-merge commit with at least one file change as the initial submission.

3.2 Pull Request (PR) Metrics

When calculating the following metrics we only use the 93% of pull requests (31,284/33,596) that are marked closed (as open PRs could either be ignored by developers or be in-progress).

Churn: Via the `pr_commit_details` table, the sum of code line additions and deletions across the PR's non-merge commits (c): $\text{Churn}(PR) = \sum_{c \in PR} (\texttt{additions}_c + \texttt{deletions}_c)$

Turnaround time: Via the `pull_request` table, the elapsed time between pull request creation and closure:
$\text{Turnaround}(PR) = t_{\texttt{closed\_at}}(PR) - t_{\texttt{created\_at}}(PR)$

Merge rate: The proportion of closed pull requests that are merged: $\text{Merged} = \frac{\text{merged PRs}}{\text{closed PRs}}$

Test-to-code churn: The ratio of code line changes in test files vs. non-test files: $R_{tc} = \frac{\text{test churn}}{\text{non-test churn}}$

4 Results

For context, we classify a pull request (PR) as a test PR if it touches (adds or modifies) at least one test file.

4.1 RQ1: Agentic PRs Became More Common & More Likely to Contain Tests

Table 1 shows that over the observed months PR volume mostly grew each month across agents. The main exception is Devin, which was overall flat March to July, peaking in May.

Table 1: 2025 monthly pull requests (PR) and test inclusion rate (T = test PRs / total PRs). Bold = biggest in time series.

		Jan	Feb	Mar	Apr	May	Jun	Jul
Claude	PR	–	8	29	15	23	140	**244**
	T	–	37	24	7	43	49	**55**
Codex	PR	–	–	–	–	3,864	8,846	**9,089**
	T	–	–	–	–	31	39	**58**
Copilot	PR	–	–	1	–	919	1,952	**2,098**
	T	–	–	100	–	35	42	**44**
Cursor	PR	–	1	–	1	14	496	**1,029**
	T	–	0	–	0	14	**29**	23
Devin	PR	412	530	714	803	**951**	673	679
	T	31	**36**	26	31	31	29	34
Total	T	31	36	26	30	32	38	**52**

Within the growing number of PRs, the test inclusion rate (the portion of PRs that touches a test file) grew in most months across agents, overall from 31% to 52%, making test PRs grow faster than non-test PRs. The main exception is again Devin, where the test inclusion rate was largely flat at around 1/3 form February to July.

Table 2: Task's PR share (W) vs test inclusion rate (T) of 4 most-common task types plus other: W = PRs for task type / agent's total PRs; T = test PRs / agent task type's total PRs.

	feat		fix		test		docs		other	
(%)	W	T	W	T	W	T	W	T	W	T
Claude	55	58	25	45	1	67	7	13	12	34
Codex	46	55	20	36	9	94	12	8	13	27
Copilot	33	48	40	46	3	74	9	6	14	29
Cursor	40	30	27	27	2	84	13	1	18	21
Devin	39	33	26	40	2	74	13	4	19	27

Agents' different test adoption correlates with agents' different task distributions. We perform a Chi-Square test [16], which confirms a statistically significant association between task type and test inclusion for all agents ($p < 0.05$; with moderate effect size). AIDev labels each PR with its task category via GPT 4.1-mini following the Conventional Commits Specification [5, 14]. Table 2 shows that agents differ substantially in their PRs' task types. Agents that mainly work on feature development or bug-fixes tend to have a higher test inclusion rate, whereas documentation-heavy or mixed-task workloads are associated with lower test inclusion.

Table 2 also shows the limits of our test file labeling and AIDev's PR task labeling heuristics, with several AIDev test PRs not touching test files. Manual sampling indicates that about a third of these PRs "only" update configuration or documentation files, another third does not update any files, and a third is due to test file heuristics.

Finding (RQ1). Testing is increasingly common in agentic pull requests but varies across agents, correlating with different agent task distributions.

4.2 RQ2: Tests Are Usually Touched Early But Often Revised Later

We first examine when an agentic PR's lifecycle first touches test files. Using Section 3.1's commit-level timeline cutoff, we classify test touch timing into (I) initial PR only, (L) later only (after the initial PR); and (I+L) initial and later (where a PR first touches some test files in the initial PR and others later).

Table 3 shows that, across agents, test files most often are first touched only in the initial PR (I), most pronounced for Codex with 96%, followed by some two-thirds for Claude, Copilot, and Cursor. On the flip side, the sum of L and I+L are cases that later touch a test file not touched by the initial PR, indicating that the initial PR's touch of test files was not sufficient. For most agents this sum is about a third of closed test PRs.

We next analyze if test files first touched in the initial agentic PR receive additional updates (via commits) after the initial PR. We treat a file as modified only if its contents change (and thus ignore renamed or removed files). While we only examine initial PRs that are agentic, we cannot confidently distinguish between agentic and human commits after the initial PR. Any such update is a sign that the initial agentic PR's test file touches needed revision.

Table 3: Closed agentic test PR first touches test files: Initial PR only (I), only later (L), or in both stages (I+L); some PRs omitted due to missing committer date. I-type PRs with a test file re-touched after the initial PR (IM) and same test file re-touched more than once (IM_{2+}).

	1st touch			Test evo	
(%)	I	L	I+L	IM	IM_{2+}
Claude	65	14	19	34	16
Codex	96	2	2	4	1
Copilot	70	11	18	45	24
Cursor	64	15	19	39	16
Devin	58	24	15	59	32

Table 3 shows that the likelihood of revising test files first touched during initial PRs varies across agents. Codex PRs rarely receive follow-up test modifications (4%). Devin PRs receive frequent test modifications (59%) and often a test file is modified more than once (32%). Claude, Cursor, and Copilot fall between these extremes, with 33–49% of test PRs modifying initial test file touches at least once and 15–24% modifying them multiple times.

Finding (RQ2). Agentic initial PRs' tests are often not sufficient. PRs often first touch some test files only after the initial PR. Test files first touched in the initial PR often later receive updates.

4.3 RQ3: Test-containing PRs are larger and have longer turnaround time

Table 4 summarizes code churn, turnaround time, and merge outcome for closed PRs. Across all agents, test-containing PRs have higher code churn, indicating that tests are typically introduced alongside larger changes. Test-containing PRs also tend to have longer turnaround times, indicating more human intervention.

Table 4: Agents' lifecycle traits for *all* closed PRs: C_m = median churn (LOC); T_m = median turnaround time (hours); Merged = merge rate; NT & T = (non-) test PRs; R_{tc} = test PRs' median test-to-code churn ratio.

	C_m (LOC)		T_m (h)		Merged (%)		R_{tc}
	NT	T	NT	T	NT	T	T
Claude	183	1,736	1.03	4.15	70.6	72.1	0.42
Codex	39	133	0.03	0.01	86.2	85.2	0.61
Copilot	49	323	5.51	24.09	53.8	56.8	0.87
Cursor	139	852	0.56	7.04	75.6	71.6	0.42
Devin	78	335	3.43	38.72	60.6	44.1	0.56

Codex is different, with extremely short turnaround across PRs, consistent with prior observations that Codex-generated PRs are often small and integrated rapidly [14]. For Codex, test PRs have comparable or slightly shorter turnaround times, even though test PRs' median churn is higher. Merge rates are largely similar across

PRs and agents. Devin is an exception, merging test-containing PRs less frequently than non-test PRs. The test-to-code churn also varies by agent, with Codex being the highest.

To gauge human PR acceptance, we analyze the subset of PRs that are explicitly linked to GitHub issues. GitHub issues are commonly used to request bug fixes, feature additions, or other changes. Closing a GitHub issue in addition to the linked PR may serve as a stronger signal that a human has accepted the intended change.

Table 5: Agents' lifecycle traits for *GitHub issue-linked* closed PRs: C_m = median churn; T_m = median turnaround time; Merged = merge rate; IC = issue closure rate (fraction of merged PRs whose linked issue was closed); NT & T = (non-) test PRs; R_{tc} = test PRs' median test-to-code churn ratio.

	C_m (LOC)		T_m (h)		Merged (%)		IC (%)		R_{tc}
	NT	T	NT	T	NT	T	NT	T	T
Claude	377	1,414	2.31	5.14	68.3	74.3	100	93	0.89
Codex	28	103	0.81	17.87	93.4	81.5	98	98	0.53
Copilot	55	295	14.81	36.67	64.4	60.5	99	99	0.95
Cursor	54	726	9.9	49.23	77.8	86.2	100	96	0.75
Devin	44	210	13.86	193.02	61.2	26.4	100	100	1.24

AIDev-pop provides a list of GitHub issues that are connected to at least one PR. While issues and PRs are in a m:n relation, we simplify the analysis by only keeping the 4,325 issue-PR tuples forming the 1:1 relation subset (or 88% of the total m:n issue-PR links). This 1:1 subset contains 1,955 test PRs. Table 5 shows that, consistent with overall PRs, GitHub issue-linked test PRs have higher churn and longer turnaround times. Across PRs, merged PRs usually correspond to GitHub issue closure.

Finding (RQ3). Test PRs are consistently larger and have longer lifespan. In GitHub issue-linked PRs, merged pull requests are typically followed by issue closure regardless of test inclusion.

5 Related Work

Software testing has long been studied as a core practice in software development. Open source projects, in particular, have been widely used to examine how tests are adopted and maintained at scale. Prior work shows that the presence of test code varies across projects and is associated with factors such as project size, team size, programming language etc. [12]. Beyond adoption rates, studies highlight varying testing practices across languages and domains, including limited unit testing in Python projects and deep-learning repositories, where tests often cover only a narrow set of components [19, 21]. They also suggest that the presence of tests is associated with higher pull request acceptance rates, underscoring the role of testing in code quality and maintainability.

Recent research has also examined how developers write and structure tests [15], and how testing expectations are communicated in open-source projects [6]. Though in contribution guidelines testing is encouraged, but project specific guidance often focus more on running existing tests rather than writing or extending them, leaving expectations around test contributions unclear.

Large language models are now driving the interest in AI-assisted software testing [22]. Research shows that LLMs can generate or extend test cases by utilizing learned code patterns and natural-language reasoning [9]. LLMs generated tests are often accepted by developers in practice, demonstrating their potential to support testing activities in production-level codebases [2]. These studies show the potential of LLMs to support testing in isolation, but do not examine how tests appear in collaborative development workflows where AI agents act as teammates.

Earlier studies of testing practices in open-source projects largely reflect developer-written tests, conducted before the widespread adoption of autonomous coding agents. Our study focuses on when and how tests appear in agentic PRs, and how test inclusion relates to observable PR-level characteristics. By mining large-scale repository data from the AIDev-pop dataset, we provide an empirical view of testing practices within agentic development workflows.

6 Threats to Validity

We identify test PRs via file path and filename heuristics. While these heuristics follows prior large-scale studies, they are not perfect. Some test files may not follow common naming conventions and non-test file names may include "test". We also do not distinguish test types, such as unit, integration, or system tests. We also do not assess test coverage or test effectiveness, so the presence of test files does not necessarily indicate test quality or adequacy.

Our analysis relies on observable PR-level and commit-level signals. We do not assess reviewer intent, discussion context, or other factors that may influence the metrics used in this study. Our analysis therefore provides an exploratory view of testing-related activity in agentic pull requests and highlights observable patterns that can inform future, more detailed investigations.

7 Ethical Considerations

This study uses the publicly available AIDev dataset [14], which is derived from GitHub repositories. We augment the dataset with limited commit-level information for a PR subset. We do not collect any private developer information, interact with contributors, or evaluate individual projects. Our analysis focuses on the overall pattern in agentic PRs and is reported descriptively.

8 Conclusions

This paper presented a large-scale mining study of testing practices in agentic PRs via AIDev-pop. We examined how often agents include test code, when tests were introduced during the PR lifecycle, and how test-containing PRs differed from other PRs in terms of size, turnaround time, and review-related characteristics. Our analysis showed that test inclusion increased over time and varied across agents. When tests were present, they were often modified across multiple commits rather than added once and left unchanged. Test-containing PRs were consistently larger and took longer to complete, while merge rates were similar.

Acknowledgments

Christoph Csallner has a potential research conflict of interest due to a financial interest with Microsoft and The Trade Desk. A management plan has been created to preserve objectivity in research in accordance with UTA policy.

References

[1] 2025. How to Contribute to Open Source. https://opensource.guide/how-to-contribute/. Accessed: 2025-12-11.

[2] Nadia Alshahwan, Jubin Chheda, Anastasia Finogenova, Beliz Gokkaya, Mark Harman, Inna Harper, Alexandru Marginean, Shubho Sengupta, and Eddy Wang. 2024. Automated unit test improvement using large language models at meta. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*. 185–196.

[3] Microsoft Azure. 2025. How dates work in Git. https://learn.microsoft.com/en-us/azure/devops/repos/git/git-dates?view=azure-devops. Accessed: 2025-12-11.

[4] Shreya Bhatia, Tarushi Gandhi, Dhruv Kumar, and Pankaj Jalote. 2024. Unit test generation using generative AI: A comparative performance analysis of autogeneration tools. In *Proceedings of the 1st International Workshop on Large Language Models for Code*. 54–61.

[5] Conventional Commits. 2025. A specification for adding human and machine readable meaning to commit messages. https://www.conventionalcommits.org/en/v1.0.0/. Accessed: 2025-11-29.

[6] Bruna Falcucci, Felipe Gomide, and Andre Hora. 2025. What Do Contribution Guidelines Say About Software Testing?. In *2025 IEEE/ACM 22nd International Conference on Mining Software Repositories (MSR)*. IEEE, 434–438.

[7] GitHub. 2025. REST API endpoints for Git commits. https://docs.github.com/en/rest/git/commits. Accessed: 2025-11-21.

[8] Sabrina Haque, Sarvesh Ingale, and Christoph Csallner. 2026. MSR Mining Challenge 2026: An Empirical Study of Tests in Agentic Pull Requests. https://doi.org/10.6084/m9.figshare.30944168

[9] Navid Bin Hasan, Md Ashraful Islam, Junaed Younus Khan, Sanjida Senjik, and Anindya Iqbal. 2025. Automatic High-Level Test Case Generation using Large Language Models. In *2025 IEEE/ACM 22nd International Conference on Mining Software Repositories (MSR)*. IEEE, 674–685.

[10] Anisha Islam, Nipuni Tharushika Hewage, Abdul Ali Bangash, and Abram Hindle. 2023. Evolution of the practice of software testing in java projects. In *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*. IEEE, 367–371.

[11] Jing Jiang, Jiangfeng Lv, Jiateng Zheng, and Li Zhang. 2021. How developers modify pull requests in code review. *IEEE Transactions on Reliability* 71, 3 (2021), 1325–1339.

[12] Pavneet Singh Kochhar, Tegawendé F Bissyandé, David Lo, and Lingxiao Jiang. 2013. An empirical study of adoption of software testing in open source projects. In *2013 13th International Conference on Quality Software*. IEEE, 103–112.

[13] Hao Li, Haoxiang Zhang, and Ahmed E. Hassan. 2025. AIDev: Studying AI Coding Agents on GitHub (The Rise of AI Teammates in Software Engineering 3.0)- dataset. https://huggingface.co/datasets/hao-li/AIDev. Accessed: 2025-10-29.

[14] 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).

[15] Rangeet Pan, Tyler Stennett, Raju Pavuluri, Nate Levin, Alessandro Orso, and Saurabh Sinha. 2025. Hamster: A Large-Scale Study and Characterization of Developer-Written Tests. *arXiv preprint arXiv:2509.26204* (2025).

[16] Daniel Powers and Yu Xie. 2008. *Statistical methods for categorical data analysis*. Emerald Group Publishing.

[17] Eddie Antonio Santos and Abram Hindle. 2016. Judging a commit by its cover: Correlating commit message entropy with build status on travis-ci. (2016).

[18] stackoverflow. 2025. Why is git AuthorDate different from CommitDate? https://stackoverflow.com/questions/11856983/why-is-git-authordate-different-from-commitdate. Accessed: 2025-12-11.

[19] Fabian Trautsch and Jens Grabowski. 2017. Are there any unit tests? an empirical study on unit testing in open source python projects. In *2017 IEEE International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 207–218.

[20] Rosalia Tufano, Antonio Mastropaolo, Federica Pepe, Ozren Dabic, Massimiliano Di Penta, and Gabriele Bavota. 2024. Unveiling chatgpt's usage in open source projects: A mining-based study. In *Proceedings of the 21st International Conference on Mining Software Repositories*. 571–583.

[21] Han Wang, Sijia Yu, Chunyang Chen, Burak Turhan, and Xiaodong Zhu. 2024. Beyond accuracy: an empirical study on unit testing in open-source deep learning projects. *ACM Transactions on Software Engineering and Methodology* 33, 4 (2024), 1–22.

[22] Yuchen Wang, Shangxin Guo, and Chee Wei Tan. 2025. From code generation to software testing: AI Copilot with context-based RAG. *IEEE Software* (2025).

[23] Miku Watanabe, Hao Li, Yutaro Kashiwa, Brittany Reid, Hajimu Iida, and Ahmed E Hassan. 2025. On the use of agentic coding: An empirical study of pull requests on github. *arXiv preprint arXiv:2509.14745* (2025).