Test Coverage of Code Changes in AI-Generated Pull Requests

Tales Alves
tvac@cin.ufpe.br
Informatics Center, Federal University of Pernambuco
Recife, PE, Brazil

Leopoldo Teixeira
lmt@cin.ufpe.br
Informatics Center, Federal University of Pernambuco
Recife, PE, Brazil

Abstract

As autonomous coding agents increasingly integrate into software development workflows, understanding their testing practices is essential. We study how well tests in a pull request (PR) exercise the code changed by that PR, and whether those tests contain meaningful assertions. Using AIDev dataset v2, we compare three authorship types: AI-Only (all commits authored by agents), Coauthor (mixed human-AI commits), and Human (baseline PRs from the same repository). Analyzing 2,314 Python PRs that modify tests, we measure diff-coverage: the percentage of modified non-test source lines executed by the PR's tests. AI-Only PRs achieve higher mean diff-coverage (19.96%) than Coauthor (17.35%) and Human PRs (13.09%). Moreover, 80.4% of AI-Only PRs have non-zero diff-coverage, compared to 62.3% for Coauthor and 66.2% for Human. To assess oracle strength, we manually classify assertions in 255 diffs using a four-level taxonomy (L1–L4) with three independent evaluators. Functional assertions (L3) are dominant, regardless of authorship type: 66.7% AI-Only, 80.0% Coauthor, and 69.9% Human. High-quality assertions (L3+L4) appear in over 90% of test files across all groups, with no statistically significant pairwise differences (Fisher's exact, $p>0.05$). Overall, within PRs that include test changes, AI-authored submissions tend to achieve higher coverage of changed code without evidence of weaker assertions, providing empirical evidence to inform strategies for integrating AI-assisted development into software projects.

CCS Concepts

• **Software and its engineering → Software testing and debugging**; *Software creation and management*.

Keywords

autonomous coding agents, test coverage, code quality, pull requests, software testing, AI-assisted development

ACM Reference Format:
Tales Alves and Leopoldo Teixeira. 2026. Test Coverage of Code Changes in AI-Generated 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/10.1145/3793302.3793618

1 Introduction

The emergence of autonomous coding agents (e.g., Devin and GitHub Copilot) marks a significant shift in contemporary software development [8]. Unlike earlier code completion tools that assist developers at the token or line level, these agents operate as semi-autonomous collaborators: they create branches, implement end-to-end features, write tests, and submit pull requests (PRs) with minimal human intervention. While early studies report productivity benefits from AI-assisted development [15], concerns persist regarding the quality of the resulting code. Previous work documented latent defects in LLM-generated code that pass functional tests [9] and security vulnerabilities in AI-assisted outputs [10].

Testing plays a central role in mitigating such risks. Test coverage, the proportion of code exercised by automated tests, is widely used as a baseline quality indicator [14]. However, coverage alone does not guarantee fault detection. Determining whether test outcomes are meaningfully validated, namely the *test oracle problem*, remains a fundamental challenge [6]. Tests may execute code paths without asserting correct behavior, yielding high coverage with limited effectiveness. Recent works suggest that LLMs can generate structurally adequate tests, yet their fault-revealing power varies considerably [3, 11]. This raises a critical question: when autonomous agents contribute tests in a pull request, do they provide stronger coverage of changed code and meaningful assertions?

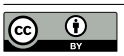
Despite its rapid adoption, empirical evidence on the testing practices of autonomous coding agents remains scarce. Existing work has focused on agent productivity and acceptance rates [8], leaving testing behavior largely unexplored at scale. This gap is particularly relevant for understanding human-AI collaboration in software quality assurance. Therefore, in this work, we investigate testing behavior in agent-authored PRs using the AIDev dataset by addressing the following research questions:

- **RQ1:** What is the test coverage of code changes exercised by tests within the same PR, comparing AI-Only, Coauthor, and Human PRs?
- **RQ2:** What is the assertion quality of tests in AI-Only, Coauthor, and Human PRs?

We address RQ1 through automated diff-coverage analysis of 2,314 PRs from Python repositories that include test modifications, and RQ2 through a manual classification of assertion sophistication in 255 test-files evaluated by three independent annotators. Our contributions are: (1) a large-scale empirical comparison of diff-coverage across three PR authorship types, (2) a qualitative assessment of assertion quality with measured inter-rater reliability (Fleiss' κ=0.586), and (3) empirically grounded insights into testing practices in AI-assisted software development.

2 Dataset and Study Design

We base our study on the AIDev dataset v2 [8], a large-scale collection of PRs authored by autonomous coding agents. The dataset covers five agents (OpenAI Codex, Devin, GitHub Copilot, Cursor, and Claude Code). A curated subset of 33,596 Agentic-PRs from 2,807 repositories with over 100 GitHub stars provides enriched



MSR '26, Rio de Janeiro, Brazil

ACM ISBN 979-8-4007-2474-9/2026/04
https://doi.org/10.1145/3793302.3793618

metadata. Our analysis considers PRs between January and June 2025.

Data Pipeline. We applied a sequential filtering process. (1) *Language filter*: We restrict our analysis to Python repositories, using mature coverage tooling (e.g., `pytest`[1] and `coverage.py`[2]), yielding 7,190 PRs across 597 repositories. (2) *Test-modification filter*: We retain only PRs that modify at least one test file, identified using standard naming and directory conventions (`test_*.py`, `*_test.py`, or files under `tests/` directories). (3) *Coverage analysis*: For each remaining PR, we execute the modified test files in isolated Docker containers to measure diff-coverage, resulting in 4,494 test-file observations from 2,660 PRs across 348 repositories. All PRs where tests failed to execute or coverage could not be collected were excluded. The final dataset comprises 2,314 PRs (Table 1). Execution success rates differ across authorship types (AI-Only 85.9%, Coauthor 82.9%, Human 72.4%), possibly due to more complex environment configuration.

Table 1: Dataset Overview and Sample Sizes

Stage	AI-Only	Coauthor	Human	Total
Initial PRs	537	1,765	358	2,660
Repositories	87	191	70	348
Test file obs.	856	2,913	725	4,494
Final obs.	735	2,416	525	3,676
Final PRs	474 (85.9%)	1,559 (82.9%)	281 (72.4%)	2,314

Authorship Classification. We classify PRs into three authorship categories based on metadata: *AI-Only* PRs contain exclusively bot-authored commits; *Coauthor* PRs contain a mix of agent-authored and human-authored commits; and *Human* PRs serve as a baseline and are drawn from the `human_pull_request` table in AIDev. Human PRs are sampled from the same Python repositories in which Agentic PRs occur. Bot commits are identified using explicit markers (e.g., `[bot]` suffixes) and known agent identifiers (e.g., `devin-ai-integration[bot]`). This captures meaningful workflow differences: Devin commonly authors commits directly under bot identities, whereas OpenAI Codex PRs are predominantly Coauthor PRs because humans typically commit the generated code.

3 Test Coverage of Code Changes

Research Question 1: What is the test coverage of code changes exercised by tests within the same pull request, comparing AI-Only, Coauthor, and Human PRs?

Motivation. Test coverage quantifies the extent to which automated tests exercise program code. Although coverage does not guarantee fault detection, it identifies code paths that remain untested and serves as a baseline proxy for testing effort [14].

Method. For each PR that modifies at least one test file, we compute *diff-coverage*, the percentage of modified non-test source lines that are executed when running the modified tests. Formally,

$$\text{Diff-Coverage} = \frac{|\text{Modified source lines} \cap \text{Executed lines}|}{|\text{Modified source lines}|} \times 100\% \quad (1)$$

Modified source lines correspond to added or changed lines in Python files excluding test files, as identified from the PR diff at the merge commit. Deleted lines and whitespace-only changes are excluded. Executed lines are obtained from coverage reports generated by running the modified test files.

Our measurement process uses isolated Docker containers per PR: (1) clone the repository at the PR's merge commit; (2) parse the PR diff to identify modified source files and line ranges; (3) execute each modified test file using `pytest` with `coverage.py` instrumentation; (4) parse the coverage report to identify executed source lines; and (5) compute diff-coverage as defined above. For PRs that modify multiple test files, diff-coverage is computed per test file and aggregated at the PR level by taking the mean.

Results. Table 2 summarizes diff-coverage statistics for 2,314 PRs (474 AI-Only, 1,559 Coauthor, and 281 Human), aggregated from 3,676 test-file observations.

Table 2: Test Coverage Statistics by Authorship Type

Metric	AI-Only	Coauthor	Human
N (PRs)	474	1,559	281
Mean (%)	19.96	17.35	13.09
Median (%)	15.97	5.18	2.82
Std (%)	19.37	23.61	18.25
Min (%)	0.0	0.0	0.0
Max (%)	77.27	100.0	100.0
Q25 (%)	1.12	0.0	0.0
Q75 (%)	34.29	28.17	22.14
PRs >0%	381 (80.4%)	971 (62.3%)	186 (66.2%)
PRs 0%	93 (19.6%)	588 (37.7%)	95 (33.8%)

Differences across authorship types. AI-Only PRs exhibit higher mean and median diff-coverage (Table 2) than Coauthor and Human PRs. Median differences indicate that, on PRs that include test changes, agent-authored PRs often exercise a larger fraction of modified code.

Prevalence of non-zero diff-coverage. A larger proportion of AI-Only PRs achieve non-zero diff-coverage (80.4%) compared to Coauthor (62.3%) and Human (66.2%). Coauthor PRs have the lowest rate, suggesting that humans commit AI code without corresponding tests.

Distribution patterns. Figure 1 shows that AI-Only PRs are underrepresented in the 0% diff-coverage range and overrepresented in the $25-49\%$ range relative to the other groups. All groups are underrepresented in high diff-coverage ($> 75\%$).

Agent-level patterns. Within AI-Only PRs, Devin exhibits the highest mean diff-coverage (22.96%), followed by Copilot (11.53%), Cursor (2.25%), and Claude Code (2.50%). Devin AI-Only PRs achieve higher diff-coverage than Devin Coauthor PRs (22.96% vs. 11.01%), suggesting differences from authorship workflows rather than the underlying agent.

[1] https://docs.pytest.org/en/stable/
[2] https://coverage.readthedocs.io/

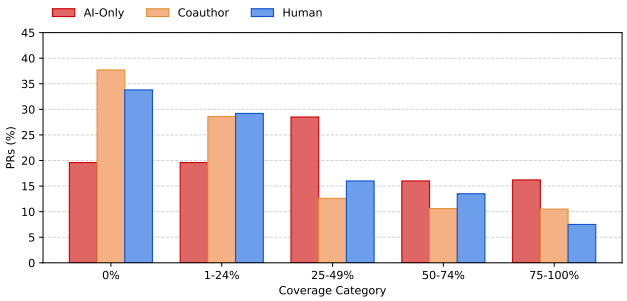


Figure 1: Distribution of PRs by coverage category across three authorship types (N=474 AI-Only, N=1,559 Coauthor, N=281 Human).

Statistical analysis. A Kruskal-Wallis test indicates significant differences across authorship types ($H = 39.05$, $p < 0.0001$). Post-hoc Mann–Whitney U tests with Bonferroni correction show AI-Only PRs have significantly higher diff-coverage than both Coauthor ($p < 0.0001$, Cliff's $\delta = 0.167$) and Human ($p < 0.0001$, $\delta = 0.241$), while the difference between Coauthor and Human is not significant ($p = 0.241$, $\delta = 0.043$). Although effect sizes are small ($\delta < 0.33$), the absolute difference between AI-Only and Human amounts to 6.9 percentage points.

Summary of RQ1: Among pull requests that modify tests, AI-Only PRs achieve higher diff-coverage of changed source code than both Coauthor and Human PRs. In addition, a larger fraction of AI-Only PRs exhibit non-zero diff-coverage (80.4%), indicating that their tests more frequently exercise modified code.

4 Assertion Quality of Generated Tests

Research Question 2: What is the assertion quality of tests in AI-Only, Coauthor, and Human PRs?

Motivation. While coverage reflects how broadly code changes are exercised, it does not capture whether tests meaningfully validate program behavior. Tests may execute code paths without asserting correct outcomes [6]. To complement diff-coverage, we assess the sophistication of assertions in test code, as a proxy for oracle strength and test effectiveness.

Method. Because manual analysis does not scale, we perform a qualitative analysis on a sample drawn from the same PRs [4]. We focus on test files from PRs with diff-coverage > 10%, retaining only assertions *added in the diff* (lines prefixed with +) to ensure attributability to the PR author. The resulting populations contain 295 AI-Only, 484 Coauthor, and 103 Human test-file observations (power analysis indicated $N = 82$ required for 95% confidence, ±5% error). Starting from 103 test files, we discard items that cannot be reliably rated and resolve remaining disagreements via majority voting. The final annotated dataset comprises 87 AI-Only, 85 Coauthor, and 83 Human test files ($N = 255$ total), yielding near-balanced groups above the minimum threshold.

Three independent evaluators (two undergraduate and one graduate CS students with at least two years experience) classified assertions with authorship masked. Each test file was assigned the highest assertion level present (the *highest-peak rule*), representing the upper bound of sophistication. Table 3 shows the taxonomy adapted from prior work [11, 12].

Table 3: Assertion Quality Taxonomy (L1–L4)

Level	Type	Example
L1	Existence	`assert result is not None`
L2	Structural	`assert len(items) == 3`
L3	Functional	`assert compute(5) == 25`
L4	Robustness	`with pytest.raises(ValueError)`

To assess inter-rater reliability, all evaluators classified a *golden set* of 30 test files (10 per authorship type), while the remaining files were divided evenly. Agreement on the golden set yields a Fleiss' κ of 0.586 (95% CI: 0.42–0.75), indicating moderate agreement [7]. Evaluators agreed unanimously on 76.7% of items (23/30), with disagreements typically on adjacent levels (e.g., L3 vs. L4).

Results. Table 4 summarizes assertion quality across authorship types. Functional assertions (L3) dominate all groups (67–80%). High-quality assertions (L3+L4) appear in the vast majority of test files: 95.4% for AI-Only, 91.8% for Coauthor, and 96.4% for Human.

Table 4: Assertion Quality Distribution by Authorship Type

Level	AI-Only	Coauthor	Human
L1 (Existence)	2 (2.3%)	6 (7.1%)	1 (1.2%)
L2 (Structural)	2 (2.3%)	1 (1.2%)	2 (2.4%)
L3 (Functional)	58 (66.7%)	68 (80.0%)	58 (69.9%)
L4 (Robustness)	25 (28.7%)	10 (11.8%)	22 (26.5%)
Total	87 (100%)	85 (100%)	83 (100%)
L3+L4	83 (95.4%)	78 (91.8%)	80 (96.4%)

Statistical Analysis. A Chi-square test indicates a statistically significant association between authorship type and assertion-level distribution ($\chi^2 = 12.68$, $df = 6$, $p = 0.048$). However, pairwise Fisher's exact tests with Bonferroni correction reveal no significant differences between any pair of authorship types ($p > 0.05$). The significant omnibus result is primarily driven by differences in robustness-oriented assertions (L4), which are less frequent in Coauthor PRs (11.8%) than in AI-Only (28.7%) and Human PRs (26.5%), suggesting distinct patterns across the four levels without meaningful quality hierarchy

Summary of RQ2: Assertion quality is comparable across AI-Only, Coauthor, and Human pull requests. High-quality assertions appear in over 90% of test files across all groups, with no statistically significant pairwise differences. Differences arise mainly in robustness-oriented assertions, which are less frequent in Coauthor PRs.

5 Discussion

More Tests, Comparable Quality. AI-Only PRs achieve significantly higher diff-coverage and are more likely to have non-zero

coverage than Human PRs, while maintaining comparable assertion quality (95.4% high-quality vs 96.4%) These results suggest that autonomous agents produce tests that exercise more of the changed code without sacrificing quality.

The Coauthor Paradox. Coauthor PRs show intermediate coverage (17.35%) but the lowest proportion of robustness assertions (L4: 11.8%, compared to 28.7% in AI-Only and 26.5% in Human PRs). This pattern may reflect workflow dynamics in human–AI collaboration: developers committing AI-generated code may focus on validating expected functionality while deferring edge-case and error-handling tests, or robustness concerns may be addressed earlier through informal review rather than explicit assertions. While speculative, this divergence suggests that mixed authorship introduces distinct testing practices.

Implications. Our findings suggest that autonomous agents can be effectively leveraged to increase testing breadth without degrading test quality. However, the prevalence of zero-coverage PRs across all groups (19.6–37.7%) highlights persistent gaps in testing practices that are not resolved by automation alone. Organizations integrating AI-assisted development should (1) leverage agents for test scaffolding, (2) enforce coverage thresholds in CI/CD pipelines independently of authorship, and (3) pay particular attention to assertion robustness in coauthored workflows.

6 Threats to Validity

Dataset scope. Our analysis focuses exclusively on Python repositories, which may limit generalizability to other programming languages with different testing cultures or tooling.

Selection and survivorship bias. We restrict the analysis to PRs that modify test files and whose tests could be successfully executed. This may bias the sample toward PRs where testing is already emphasized, particularly for AI-generated contributions that are explicitly prompted to include tests. In addition, Human PRs exhibit a lower execution success rate (72.4% vs. 85.9% for AI-Only), primarily due to test failures. This may disproportionately exclude complex human tasks, potentially confounding authorship effects with task difficulty.

Coverage measurement. Diff-coverage captures testing breadth but not fault-detection capability [2, 14]. We mitigate this by conducting a qualitative assessment of assertion quality (RQ2).

Assertion analysis. Manual classification involves subjectivity, despite the use of a structured taxonomy and blind annotation. Inter-rater reliability is moderate ($\kappa = 0.586$), and the use of the highest-peak rule may overestimate test quality.

Temporal validity. Our results reflect a snapshot of development practices between 2024 and 2025. As coding agents, tools, and developer workflows evolve rapidly, the observed patterns may change over time.

7 Related Work

Test coverage has long been studied as an indicator of testing effort and adequacy [5, 14]. Barani et al. [2] demonstrate that the relationship between coverage and fault detection is confounded by factors such as test-suite size and structure. Consequently, the test oracle problem—specifically generation and construction—remains a persistent research challenge [6, 12]."

The rise of LLM-based code generation motivated empirical studies on AI-generated code quality. Studies have documented latent defects in generated code [9], security vulnerabilities [10], and variable correctness [13]. Schäfer et al. [11] find that LLMs generate tests achieving reasonable structural coverage. Dakhel et al. [3] combines LLM-based test generation with mutation testing to improve effectiveness.

Li et al. [8] introduce the AIDev dataset, comprising 932,791 PRs authored by autonomous agents. Their analysis shows that agent-authored PRs are accepted slightly less frequently than human PRs despite comparable development speed. In contrast to prior work that focuses on individual tools, or small scale experiments, our study leverages AIDev's scale to provide an empirical comparison of test coverage and assertion quality across AI-Only, Coauthor, and Human pull requests in real-world development settings.

8 Ethical Considerations

Data and Privacy. This study uses publicly available GitHub data from permissive-licensed repositories via the AIDev dataset [8]. We report only aggregate patterns across authorship categories, not individual developer metrics. Our focus on Python repositories with 100+ stars may not represent AI behavior in less visible projects.

Responsible Interpretation. Findings should inform, not replace, human judgment. Higher AI coverage does not imply unconditional trust. Human expertise remains essential for contextual validation and architectural decisions. Our scripts, pre-computed coverage data, and manual evaluation results are available [1].

9 Conclusion

This study presents an empirical comparison of test coverage and assertion quality across AI-Only, Coauthor, and Human pull requests, based on 2,314 PRs from the AIDev dataset. We find that AI-Only PRs achieve significantly higher diff-coverage than Human PRs ($p < 0.0001$), with Coauthor PRs exhibiting intermediate results. Among agents, Devin shows the highest average coverage. A manual analysis of 255 test files reveals that assertion quality is comparable across all authorship types, with over 90% of test files containing high-quality assertions and no statistically significant pairwise differences.

We plan on extending the analysis to additional programming languages, incorporating mutation testing to assess fault-detection capability, and studying how agent testing practices evolve longitudinally as tools and workflows mature.

Acknowledgments

We thank the reviewers for their constructive feedback. This work was partially supported by INES.IA (National Institute of Science and Technology for Software Engineering Based on and for Artificial Intelligence) www.ines.org.br, CNPq grant 408817/2024-0. The second author is partially supported by CNPq grant 310405/2025-4.

References

[1] Tales Alves and Leopoldo Teixeira. 2026. Replication Package: Test Coverage of Code Changes in AI-Generated Pull Requests. https://github.com/Tales-Cunha/Test-Coverage-of-Code-Changes-in-AI-Generated-Pull-Requests

[2] Maryam Barani, Yvan Labiche, and Antoine Rollet. 2023. On the Relationship Between Code Coverage and Test Suite Effectiveness. In *Proceedings of the 2023*

IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW). IEEE, 16–20. ISSN: 2159-4848.

[3] Arghavan Moradi Dakhel, Amin Nikanjam, Vahid Majdinasab, Foutse Khomh, and Michel C. Desmarais. 2024. Effective Test Generation Using Pre-trained Large Language Models and Mutation Testing. In *Information and Software Technology*, Vol. 171. Elsevier, 107468. doi:10.1016/j.infsof.2024.107468

[4] Sergio Díaz and Daniel L. Moody. 2022. Applying Inter-Rater Reliability and Agreement in Collaborative Grounded Theory Studies in Software Engineering. *Journal of Systems and Software* 195 (2022), 111520. doi:10.1016/j.jss.2022.111520

[5] Phyllis G. Frankl and Stewart N. Weiss. 1993. An Experimental Comparison of the Effectiveness of Branch Testing and Data Flow Testing. In *IEEE Transactions on Software Engineering*, Vol. 19. IEEE, 774–787.

[6] Ali Reza Ibrahimzada, Reyhaneh Jabbarvand Iyer, Mohid Almaghout, Corina Pasareanu, and Hridesh Rajan. 2022. Perfect is the Enemy of Test Oracle. In *Proceedings of the ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*. ACM, 1–13. doi:10.1145/3540250.3549086

[7] J. Richard Landis and Gary G. Koch. 1977. The Measurement of Observer Agreement for Categorical Data. *Biometrics* 33, 1 (1977), 159–174. doi:10.2307/2529310

[8] 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). doi:10.48550/arXiv.2507.15003 AIDev Dataset v2 (August 2025). DOI: 10.5281/zenodo.16919051.

[9] Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. Is Your Code Generated by ChatGPT Really Correct? Rigorous Evaluation of Large Language Models for Code Generation. In *arXiv preprint arXiv:2305.01210*. doi:10.48550/arXiv.2305.01210

[10] Hammond Pearce, Baleegh Ahmad, Benjamin Tan, Brendan Duan, and Ramesh Karri. 2022. Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions. In *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE, 754–774.

[11] Max Schäfer, Sarah Nadi, Aryaz Eghbali, and Frank Tip. 2024. An Empirical Evaluation of Using Large Language Models for Automated Unit Test Generation. In *IEEE Transactions on Software Engineering*. IEEE.

[12] Masoumeh Taromirad and Per Runeson. 2025. Assertions in Software Testing: Survey, Landscape, and Trends. *International Journal on Software Tools for Technology Transfer* 27, 1 (April 2025), 117–135. doi:10.1007/s10009-025-00794-1

[13] Burak Yetistiren, Isik Özsoy, Miray Ayerdem, and Eray Tüzün. 2023. Evaluating the Code Quality of AI-Assisted Code Generation Tools: An Empirical Study on GitHub Copilot, Amazon CodeWhisperer, and ChatGPT. In *arXiv preprint arXiv:2304.10778*.

[14] Hong Zhu, Patrick A. V. Hall, and John H. R. May. 1997. Software Unit Test Coverage and Adequacy. *Comput. Surveys* 29, 4 (1997), 366–427.

[15] Albert Ziegler, Eirini Kalliamvakou, Shawn Simister, Ganesh Sittampalam, X. Alice Li, Andrew Rice, Devon Rifkin, and Edward Aftandilian. 2022. Productivity Assessment of Neural Code Completion. In *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming (MAPS '22)*. ACM. doi:10.48550/arXiv.2205.06537 arXiv:2205.06537.