
Test Coverage Analysis of Agentic Pull Requests

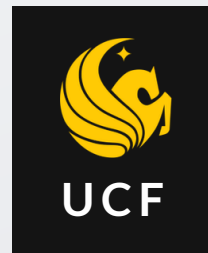
Atish Kumar Dipongkor • Kevin Moran

University of Central Florida

Talank Baral • Wing Lam

George Mason University

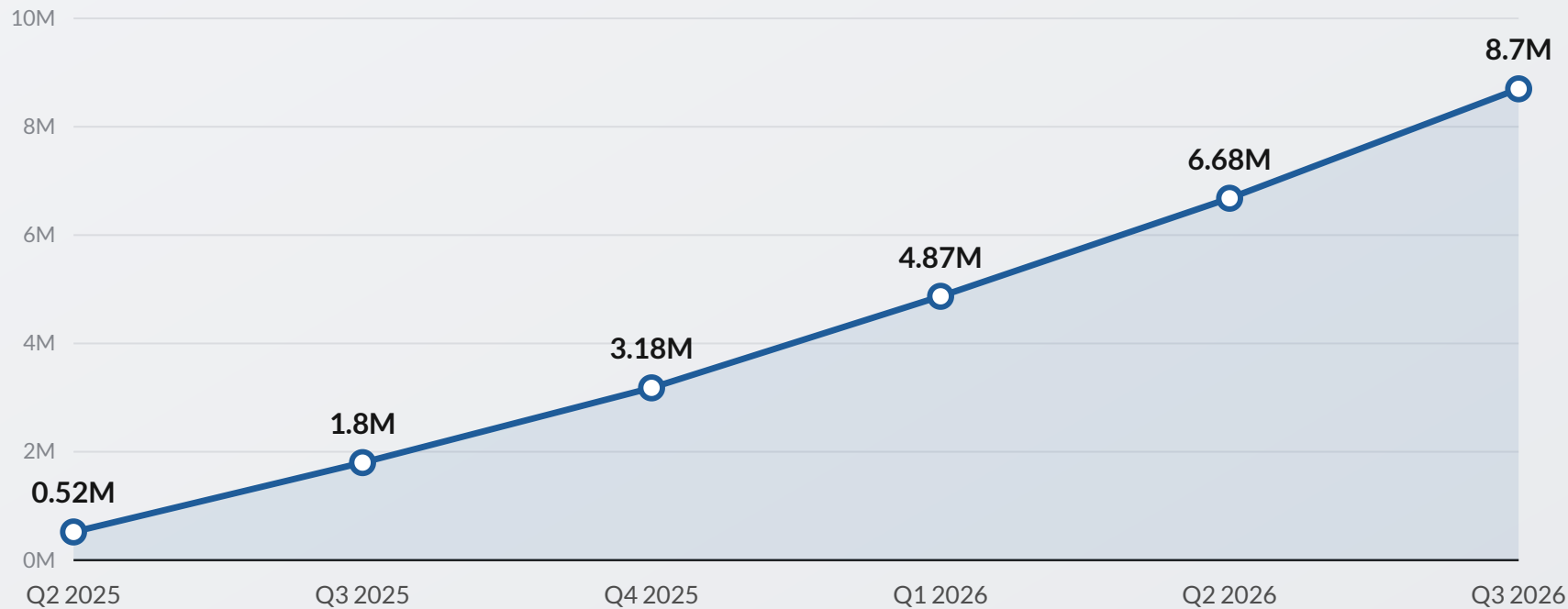
ICSME 2026 • Visions & Emerging Results Track



Merged Agentic PRs Over Time

Cumulative pull requests authored by coding agents on public GitHub and merged into real codebases.

MILLIONS OF MERGED PRS, CUMULATIVE



+2.0M

new merged PRs each quarter

Source: PR Arena (prarena.ai)

Basic Testing Expectations for PRs

Merged #482

feat: apply discount rules at checkout

codex/apply-discount-rules merged into **main**

1 file changed **+6** **-1**

Authored by an autonomous coding agent • no human commits • approved and merged

1 PR should include tests

2 All added lines should be tested

Existing Benchmarks and Novelty

SWE-BENCH • CAN THE AGENT FIX THE ISSUE?



SWT-BENCH • CAN THE AGENT WRITE A TEST FOR IT?



SWE-bench: the patch's six added lines

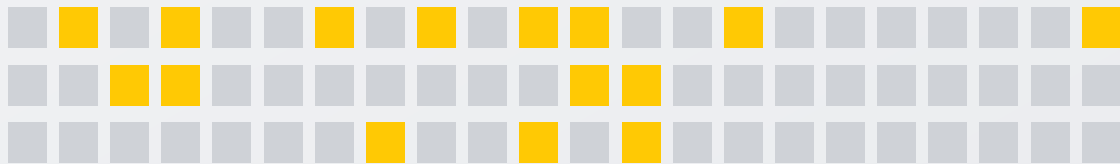


2 of 6 never executed, error path included.

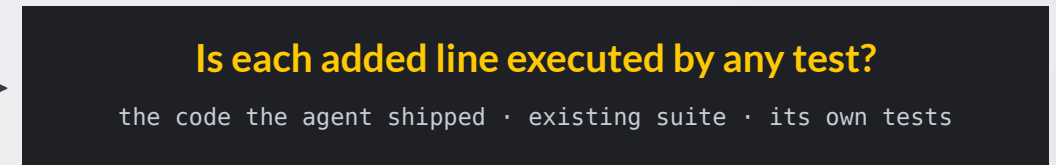
SWT-bench: tests written on request

The agent is told to test, and its tests are judged against a reference patch.

THIS WORK • AGENTS IN THE WILD • HOW DO AGENTS TEST THE CODE THEY SHIP?



4,882 agent-authored PRs merged into real repositories • five agents, two languages



*Our goal is not to score agents on a benchmark. It is to see how agents test **in the wild**, on the pull requests they actually ship.*

Research Questions

RQ1

How often do agentic PRs contain test code changes?

RQ2

What is the test coverage of the code changes in agentic PRs?

RQ2a · Existing tests

RQ2b · Agent-written tests

Different Types of Studied PRs

Code + Tests PR #482

```
diff --git payments/discount.py
+6 -1
diff --git tests/test_discount.py
+ def test_unknown_code():
+     with pytest.raises(UnknownCode):
+         apply_discount(o, "NOPE")
```

The agent shipped code and tests together.

Functional-Code-Only PR #517

```
diff --git payments/tax.py
+9 -2
```

(no test file anywhere in the diff)

1 file changed

The agent shipped behaviour and nothing to check it.

Dataset: 4,882 Agentic PRs from AIDev

AIDev v3: agent-authored PRs from real GitHub repositories, five agents: Codex, Copilot, Cursor, Claude Code, Devin.

STARTING AIDEV PRS

Java **1,278** · Python **7,191**



CHANGES A .JAVA / .PY FILE (NOT COMMENT-ONLY)

Java **532** (68 repos) · Python **4,350** (448 repos)



MERGED PRS FROM BUILDABLE, INSTRUMENTABLE REPOS

Java **213** (10 repos) · Python **1,664** (34 repos)

RQ1 · Do these PRs include tests?

Test-change behaviour classified for every PR that touches executable code.

4,882 PRs

RQ2 · Do changes have coverage?

Each project built and run; coverage measured line by line on the merged diff.

1,877 PRs

RESULTS • RQ1

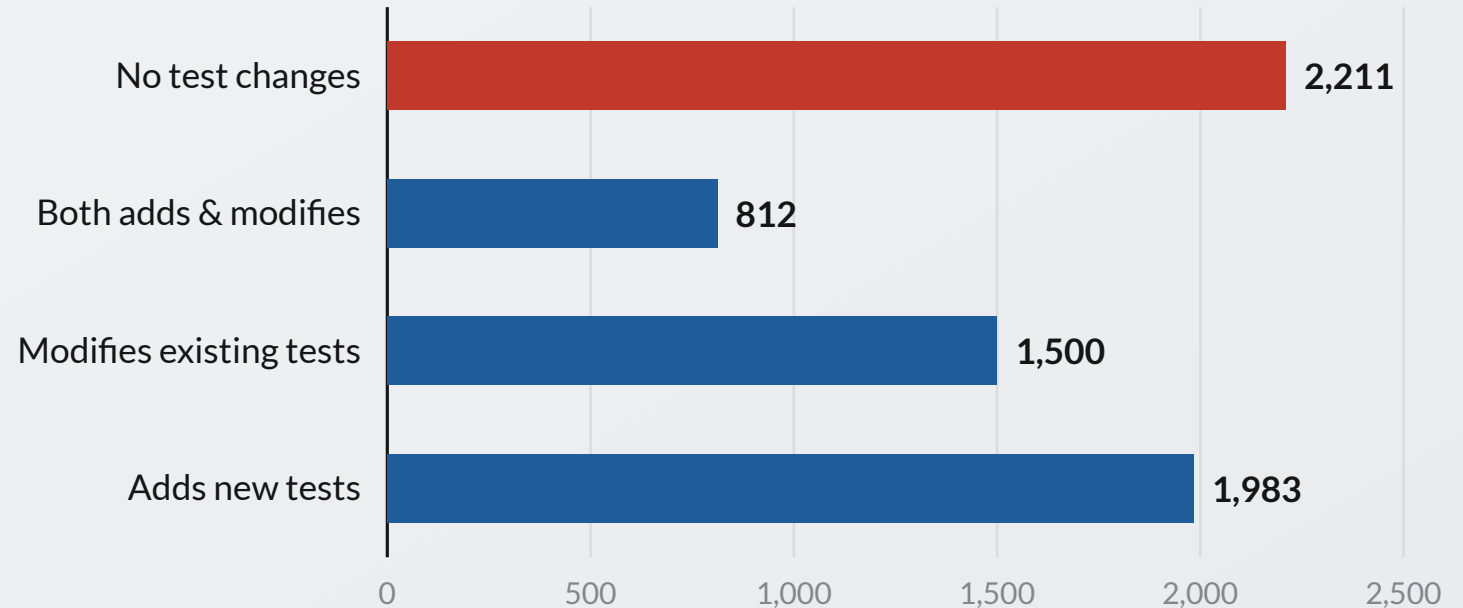
How Often Do Agents Include Tests?

RQ1: How Often Agents Include Tests?

50.4%

of the 4,387 PRs that modify code under test include **no test changes at all** (2,211 PRs).

TEST-CHANGE BEHAVIOUR ACROSS 4,387 CODE-TOUCHING PRS



WHAT ARE THOSE UNTESTED PRS DOING?

Java: fix 34.4% · feat 15.2%

Python: feat 40.5% · fix 30.3% · refactor 11.8%

These are features and bug fixes, exactly the changes you would want tests for.

RESULTS • RQ2

How Well Are Agentic Changes Covered?

RQ2a: Coverage of Agentic Changes by Existing Tests

Functional-Code-Only PR #517

```
diff --git payments/tax.py
```

```
+9 -2
```

(no test file anywhere in the diff)

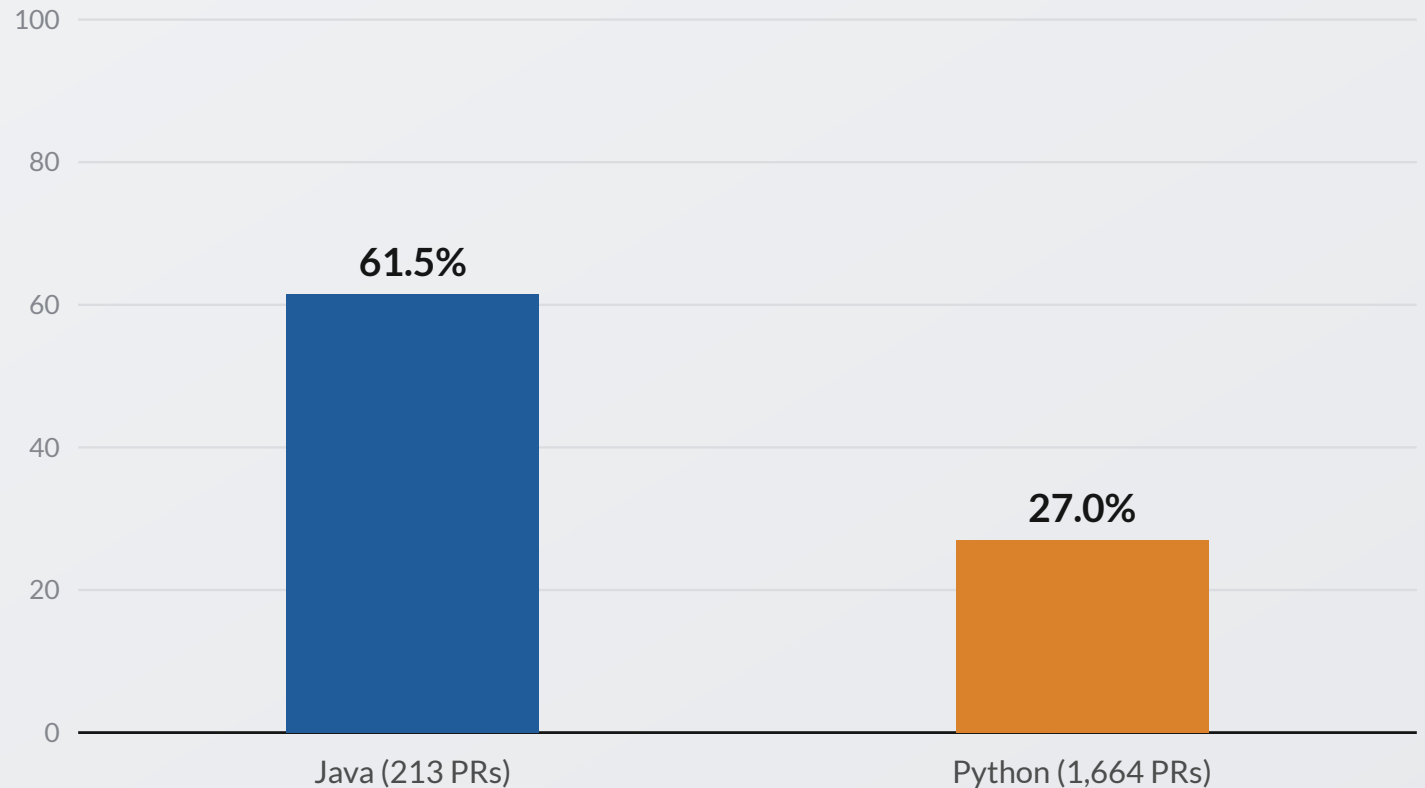
1 file changed

The agent shipped behaviour and nothing to check it.

Median Java PR: 71.1% of added lines covered. **Median Python PR:** 0% covered — 64.8% of Python PRs have no added line executed by any existing test.

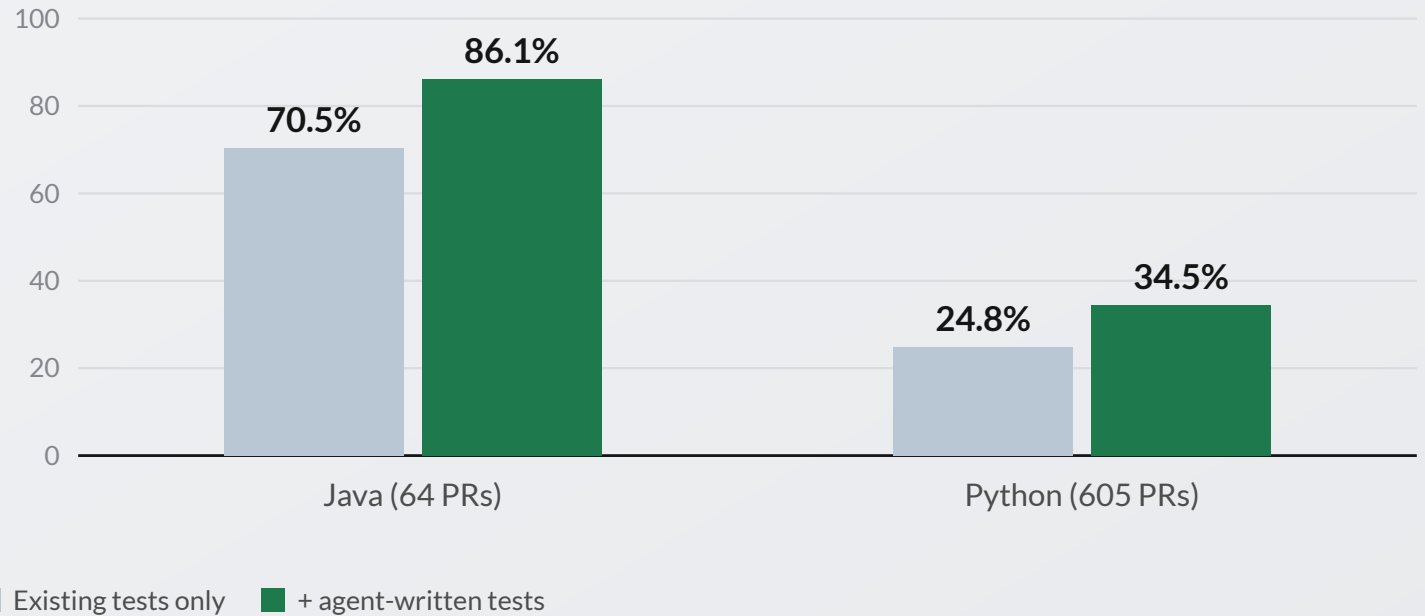
A green CI run may not be a useful signal.

ADDED LINES COVERED BY EXISTING TESTS (% , MEAN OVER PRS)



RQ2b: Coverage of Agentic Changes by Agentic Tests

DIFF COVERAGE OF CODE + TESTS PRS (%)



Code + Tests PR #482

```
diff --git payments/discount.py
+6 -1
diff --git tests/test_discount.py
+ def test_unknown_code_raises():
+     with pytest.raises(UnknownCode):
+         apply_discount(o, "NOPE")
```

The agent shipped code and tests together.

BUT THE AVERAGE HIDES THE DISTRIBUTION

35.9%

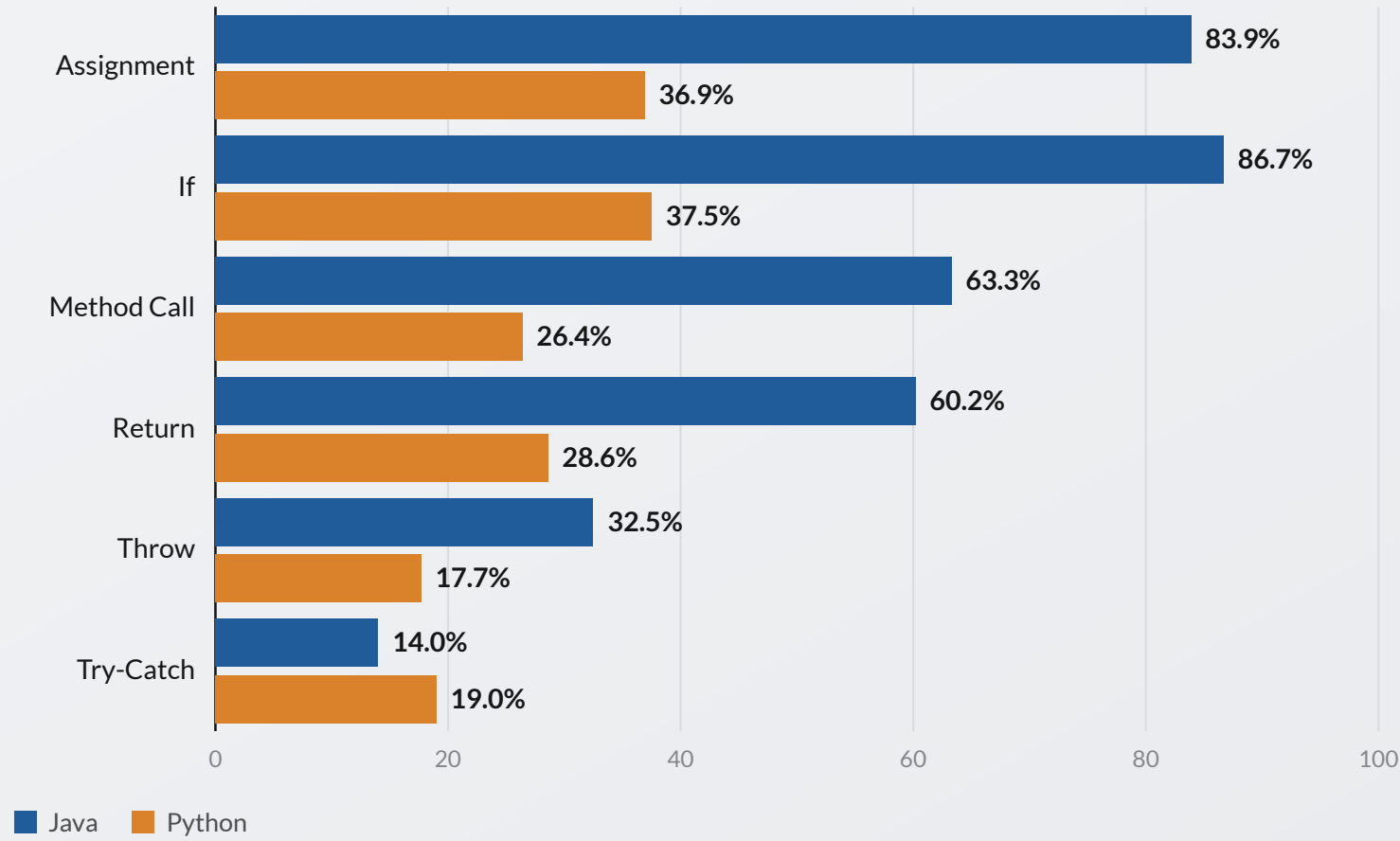
of Java Code + Tests PRs gain any coverage
(23/64)

22.5%

of Python Code + Tests PRs gain any coverage
(136/605)

Which Programming Constructs Are Least Tested?

SHARE OF ADDED LINES COVERED BY TESTS (%)



14% / 19%

of added lines inside try-catch blocks are covered (Java / Python)

Agents test the happy path.

What Can We Learn from This?

For practitioners

- A green build is not proof of testing
- Review error paths first
- Gate merges on diff coverage

For agent developers

- Have agents check their own diffs
- Engineer harnesses with a default testing behavior
- Explicitly engineer testing for error handling

For researchers

- Benchmarks should measure diff coverage
- Diagnose why agent tests miss the diff
- Generate tests for error paths

Conclusion: Agentic PRs Are Often Poorly Tested

50.4%

ship with no tests

61.5% / 27.0%

covered by existing tests

36% / 23%

gain from agent tests

19% / 14%

try-catch covered

Java / Python throughout.

Agents need to check their own work.

Code and data

github.com/SageSELab/Agentic-Pull-Request-Test-Coverage

atish.kumardipongkor@ucf.edu



THESE SLIDES
kpmoran.com/talks



REPLICATION PACKAGE
code + data

APPENDIX

Backup Slides

Motivation, methodology, and extended results for Q&A.

From LLM Assistance to Agentic Delivery

Assisted era: the LLM never shipped anything

The developer owns the change. The model completes a line or a function inside it. The developer decides what to test, and signs the commit.

Agentic era: the agent ships the whole change

Branch, commits, PR description, review request, all produced end to end with no human in the loop until review time.

Merged #482

feat: apply discount rules at checkout

codex/apply-discount-rules merged into main

1 file changed +6 -1

Authored by an autonomous coding agent • no human commits • approved and merged

Those 6 lines are now in production.

In Production, but Is It Properly Tested?

The diff inside PR #482. Every added line is either executed by some test, or it is not.

payments/discount.py (agent-authored PR)

```
def apply_discount(order, code):  
-     return order.total  
+     rule = RULES.get(code)  
+     if rule is None:  
+         raise UnknownCode(code)  
+     if order.total < rule.min_total:  
+         return order.total  
+     return order.total - rule.amount
```

6 new executable lines, including one error path.

Neither source has been measured in practice.

ONLY TWO THINGS CAN EXECUTE THEM:

1

The project's existing tests

Written before this change, so no reason to reach a branch that did not exist yesterday.

2

Tests the agent adds itself

Only if it writes any, and only if they hit these lines, not just the code around them.

Related Work: SWE-bench Asks a Different Question



Benchmarks reward a patch that makes tests pass, not a patch whose own lines are tested.

the same six added lines, under the issue's own tests

```
+ rule = RULES.get(code)      ✓ executed
+ if rule is None:            ✓ executed
+     raise UnknownCode(code)  ✗ never executed
+ if order.total < rule.min_total: ✓ executed
+     return order.total       ✗ never executed
+ return order.total - rule.amount ✓ executed
```

FAIL_TO_PASS is green. Two of the six lines were never run, including the error path.

What the verdict does not say

FAIL_TO_PASS

checks that the issue's test now passes; one path through the patch is enough.

PASS_TO_PASS

guards the rest of the suite from breakage; it says nothing about the new lines.

The Question We Ask Instead

RQ1

How often do agentic PRs contain test code changes?

Presence, not quality: classify every PR as Code + Tests, Code-Under-Test-Only, or Tests-Only.

RQ2

What is the test coverage of the code changes in agentic PRs?

RQ2a • Existing tests

How much of the agent's added code does the project's own suite already cover?

RQ2b • Agent-written tests

How much coverage do the agent's own tests add on top of the existing suite?

RQ1 by Example: Does the PR Touch a Test File?

Code + Tests PR #482

```
diff --git payments/discount.py
+6 -1
diff --git tests/test_discount.py
+ def test_unknown_code_raises():
+     with pytest.raises(UnknownCode):
+         apply_discount(o, "NOPE")
```

The agent shipped code and tests together.

Code-Under-Test-Only PR #517

```
diff --git payments/tax.py
+9 -2

(no test file anywhere in the diff)
```

1 file changed

The agent shipped behaviour and nothing to check it.

The left PR counts as tested even if its one test never reaches the error path. **Presence is not adequacy.** That is why we need RQ2.

RQ2 by Example: Is Each Added Line Executed?

The same six added lines, checked line by line against the existing suite and then against the agent's own tests.

ADDED LINE	EXISTING TESTS	+ AGENT'S TESTS
rule = RULES.get(code)	✓	✓
if rule is None:	✗	✓
raise UnknownCode(code)	✗	✗
if order.total < rule.min_total:	✗	✓
return order.total	✗	✗
return order.total - rule.amount	✓	✓

Two lines, both on the error path, are executed by neither source.

RQ2A

2 / 6 = 33%

existing tests alone

RQ2B

4 / 6 = 67%

with the agent's tests

Δ DIFFCOV

+33 pp

the agent's contribution

Dataset: 4,882 Agentic PRs from AIDev

AIDev v3: agent-authored PRs from real GitHub repositories, five agents: Codex, Copilot, Cursor, Claude Code, Devin.

STARTING AIDEV PRS

Java **1,278** · Python **7,191**



FILTER 1

Changes a .java / .py file



FILTER 2

Change is not comment- or docstring-only (tree-sitter)



RQ1 SET

Java **532** (68 repos) · Python **4,350** (448 repos)

Coverage subset (RQ2)

Merged PRs only, from repos with ≥ 10 agentic PRs
→ 14 Java · 55 Python repos

Of those, buildable + instrumentable:
→ 10 Java · 34 Python repos

213

Java PRs
with coverage

1,664

Python PRs
with coverage

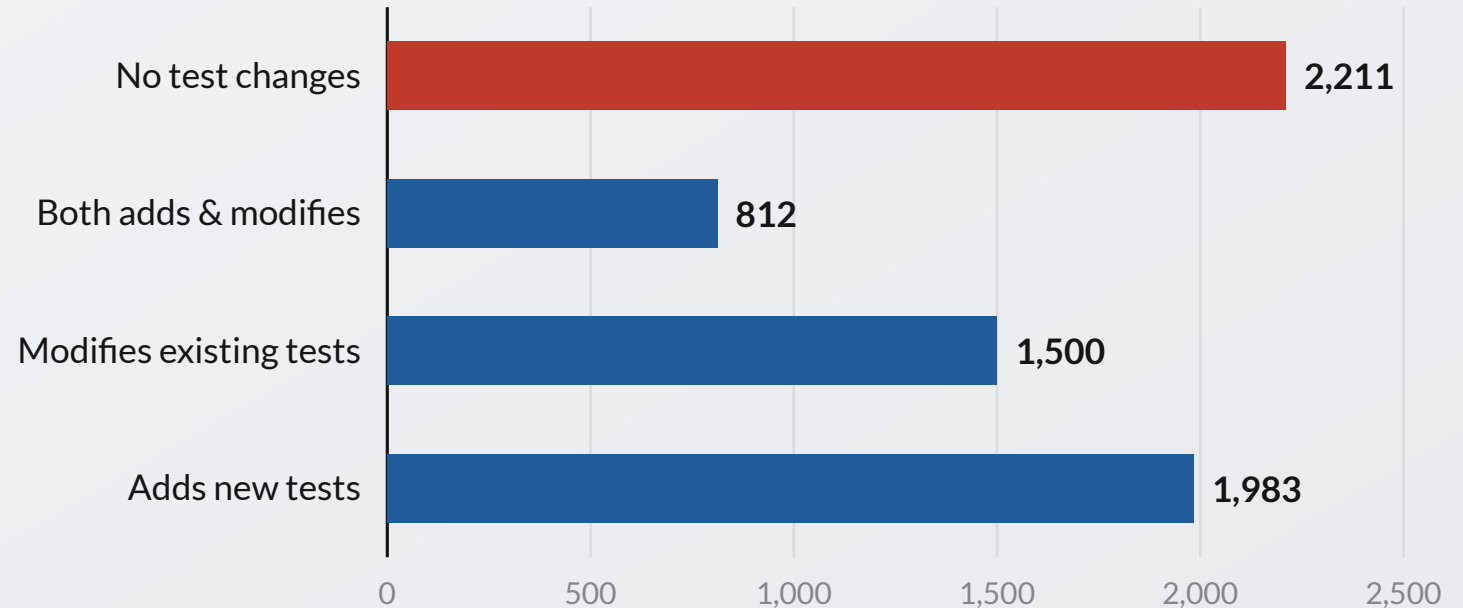
Codex is the most common agent in both languages; Java skews docs-heavy, Python feature-heavy.

RQ1: Half of Agentic PRs Ship Code with No Tests

50.4%

of the 4,387 PRs that modify code under test include **no test changes at all** (2,211 PRs).

TEST-CHANGE BEHAVIOUR ACROSS 4,387 CODE-TOUCHING PRS



WHAT ARE THOSE UNTESTED PRS DOING?

Java: fix 34.4% · feat 15.2%

Python: feat 40.5% · fix 30.3% · refactor 11.8%

These are features and bug fixes, exactly the changes you would want tests for.

RQ2a: The Existing Suite Is an Incomplete Safety Net

Code-Under-Test-Only PR #517

```
diff --git payments/tax.py
```

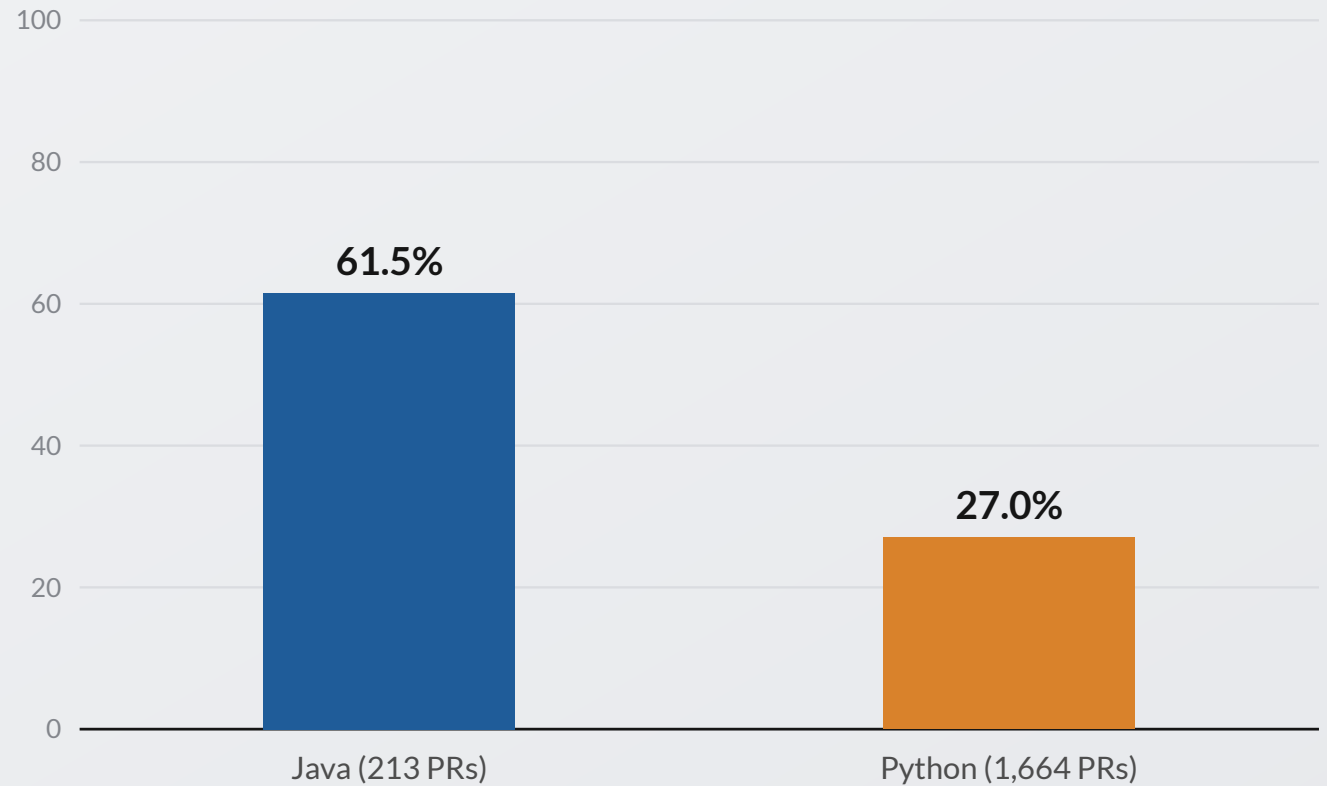
```
+9 -2
```

(no test file anywhere in the diff)

1 file changed

The agent shipped behaviour and nothing to check it.

ADDED LINES COVERED BY EXISTING TESTS (%)



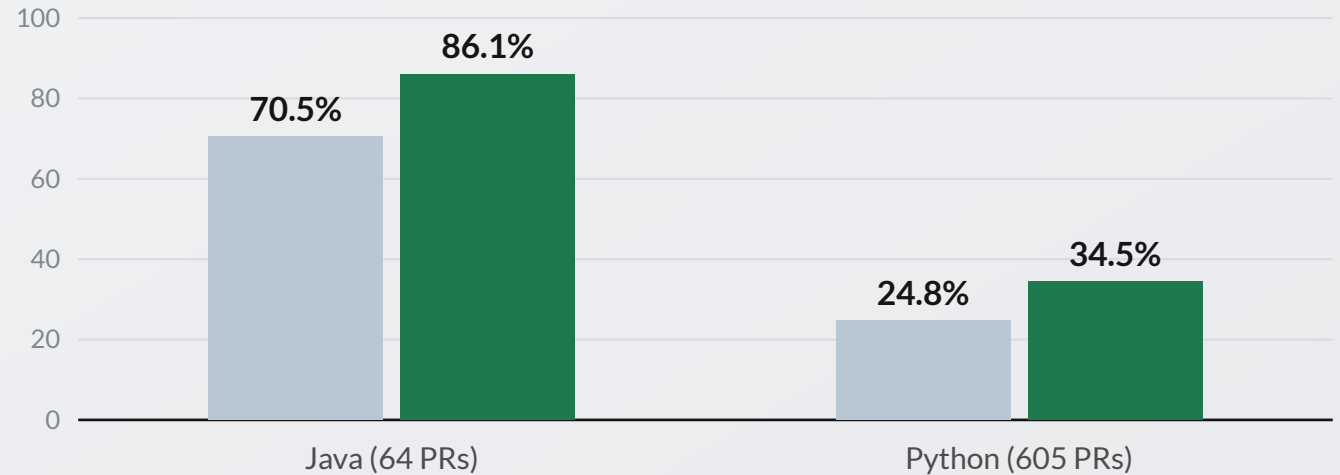
RQ2b: Agent Tests Help, but Only in a Minority of PRs

Code + Tests PR #482

```
diff --git payments/discount.py
+6 -1
diff --git tests/test_discount.py
+ def test_unknown_code_raises():
+     with pytest.raises(UnknownCode):
+         apply_discount(o, "NOPE")
```

The agent shipped code and tests together.

DIFF COVERAGE OF CODE + TESTS PRS (%)



Existing tests only + agent-written tests

BUT THE AVERAGE HIDES THE DISTRIBUTION

35.9%

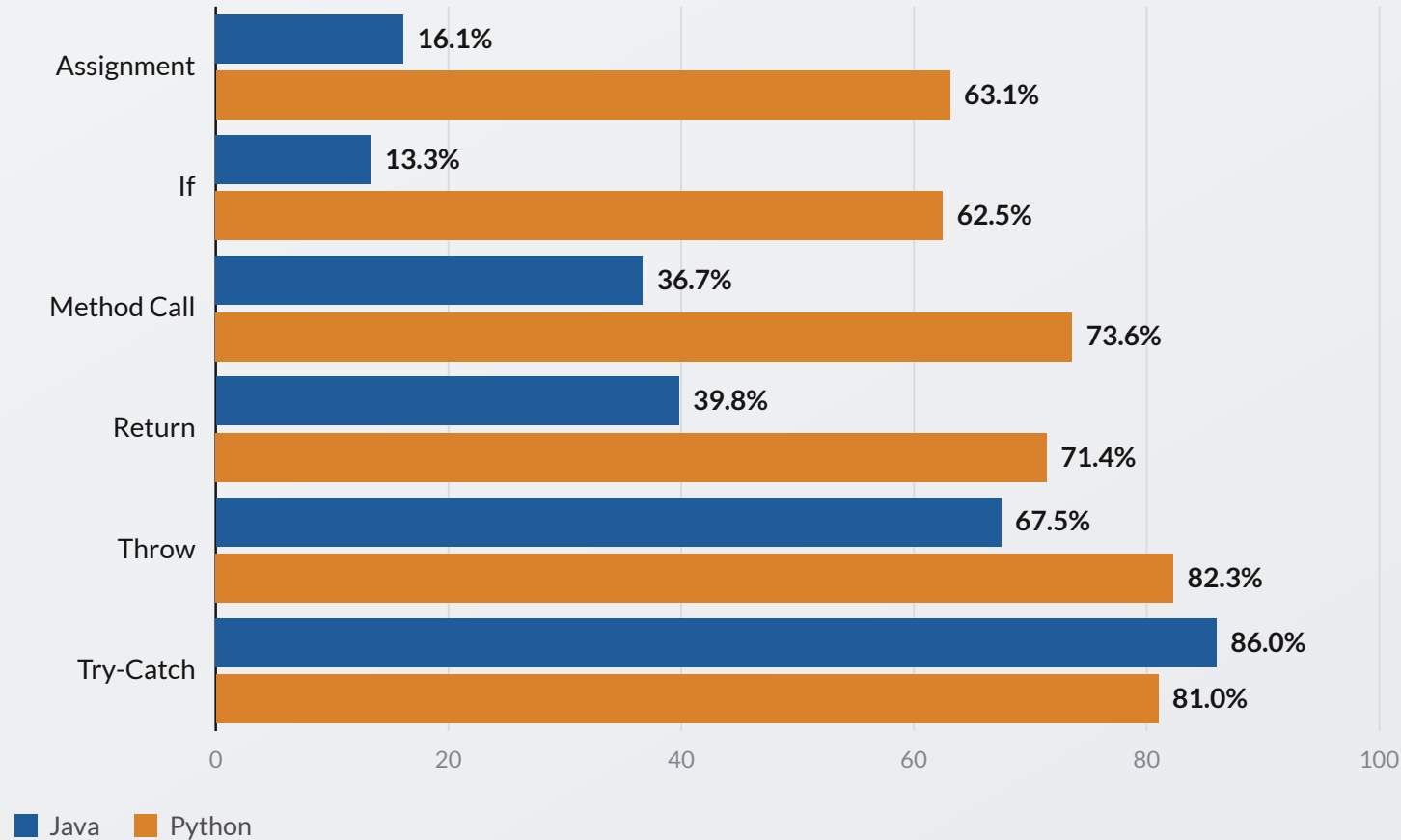
of Java Code + Tests PRs gain any coverage
(23/64)

22.5%

of Python Code + Tests PRs gain any coverage
(136/605)

Error Handling Is the Blind Spot

SHARE OF ADDED LINES NO TEST COVERS (%)



TRY-CATCH BODIES

86.0% Java • 81.0% Python

The only category where Java is worse than Python, despite far higher overall coverage.

What to Do with This

For practitioners

- A green build is not proof of testing
- Review error paths first
- Gate merges on diff coverage

For agent developers

- Check your own diff before opening a PR
- Java: stop deleting tests
- Python: target the diff, not the module

For researchers

- Benchmarks should measure diff coverage
- Diagnose why agent tests miss the diff
- Generate tests for error paths

Conclusion: Agentic PRs Are Often Poorly Tested

50.4%

ship with no tests

61.5 / 27.0%

covered by existing tests

36% / 23%

gain from agent tests

86% / 81%

try-catch uncovered

Java / Python throughout.

Agents need to check their own work.

Code and data github.com/SageSELab/Agentic-Pull-Request-Test-Coverage

atish.kumardipongkor@ucf.edu

Threats to Validity

Sampling bias: Java

The 10 analyzed Java repos are smaller than the full set: 29K vs 102K LOC, 437 vs 745 stars, 116 vs 254 forks.

Comparable on commits (2,271 vs 3,983) and age (4,178 vs 3,785 days).

Java results may reflect testing practice in smaller projects.

Sampling bias: Python

The 34 analyzed Python repos show no substantial difference from the full set on any metric:

31K vs 25K LOC · 1,202 vs 872 commits · 1,065 vs 1,268 days · 676 vs 734 stars · 120 vs 141 forks.

Python results generalize to the wider Python sample.

Other limitations: coverage measured only on merged PRs from buildable, instrumentable repos; line coverage (not branch); test-file identification relies on path and naming conventions; AIDev's own agent-attribution limits are inherited.

Methodology for RQ1: Classifying Test Behaviour

AIDev ships no PR-level diffs, so every measurement starts from a patch we rebuild ourselves.

1

Rebuild the patch

Clone each repository and reconstruct the unified diff from the PR's base and head SHAs, using `git diff <base> . . <head>`, giving one self-contained patch per PR.

2

Identify test files

By directory (`test/`, `tests/`) and by naming convention: `*Test`, `*Tests`, `*TestCase`, `Test*`, `*IT`, `*ITCase` in Java; `test_*.py` and `*_test.py` in Python.

3

Detect newly added tests

Test-method declarations at the head commit minus those at the base. Java: `@Test`, `@ParameterizedTest`, `@RepeatedTest`, `@TestFactory`, `@TestTemplate`, or JUnit 3 naming. Python: `def / async def` starting with `test`.

EVERY PR LANDS IN EXACTLY ONE BUCKET

Code + Tests

changes code under test and test files

Code-Under-Test-Only

changes code under test, no test file

Tests-only

changes test files alone

A second criterion splits Code + Tests further: a PR **adds** tests if it introduces a new test method, and **modifies** tests if it only edits existing ones. A PR can do both. Body-only edits and annotation changes leave the set of declared test names unchanged, so they never count as an added test.

Methodology for RQ2: Measuring Diff Coverage

Same patches, now executed. Run the suite twice per PR, and the difference isolates the agent's own tests.

1

Run with the agent's tests

Check out the PR's head commit and run the repository's whole suite under JaCoCo (Java) or pytest-cov (Python).

→ `DiffCov_with`

2

Run without them

Reverse-apply the test-only slice of the patch with `git apply -R`, then re-run the same suite on the same commit.

→ `DiffCov_without`

3

Label every added line

Snapshot each changed file at head and parse with `srcML --position`, tagging each added line with its construct: If, Return, Try-Catch, Throw ...

→ `miss rate by construct`

$$\text{Diff Coverage} = \frac{\text{covered added lines}}{\text{covered} + \text{missed added lines}}$$

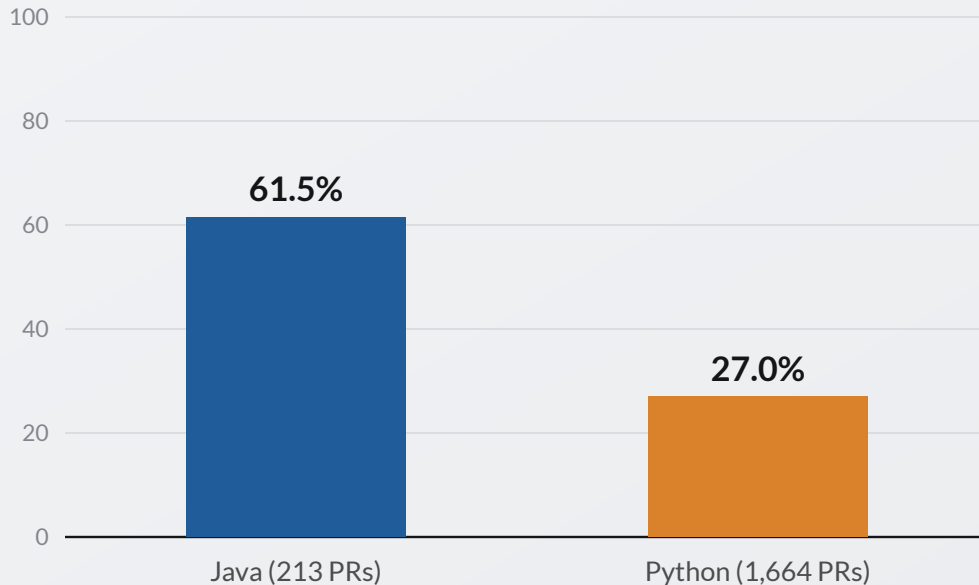
Summed over every changed code-under-test file, giving one value per PR.

$$\Delta \text{DiffCov} = \text{DiffCov_with} - \text{DiffCov_without}$$

Scope: coverage obtained for 213 Java and 1,664 Python merged PRs; the paired with/without comparison applies to the 64 Java and 605 Python PRs that contain test changes. Gains tested with a one-sided Wilcoxon signed-rank test.

RQ2a: The Existing Suite Is an Incomplete Safety Net

ADDED LINES COVERED BY EXISTING TESTS (%)



MEDIAN JAVA PR

71.1% covered

34.3% of PRs
fully covered

MEDIAN PYTHON PR

0% covered

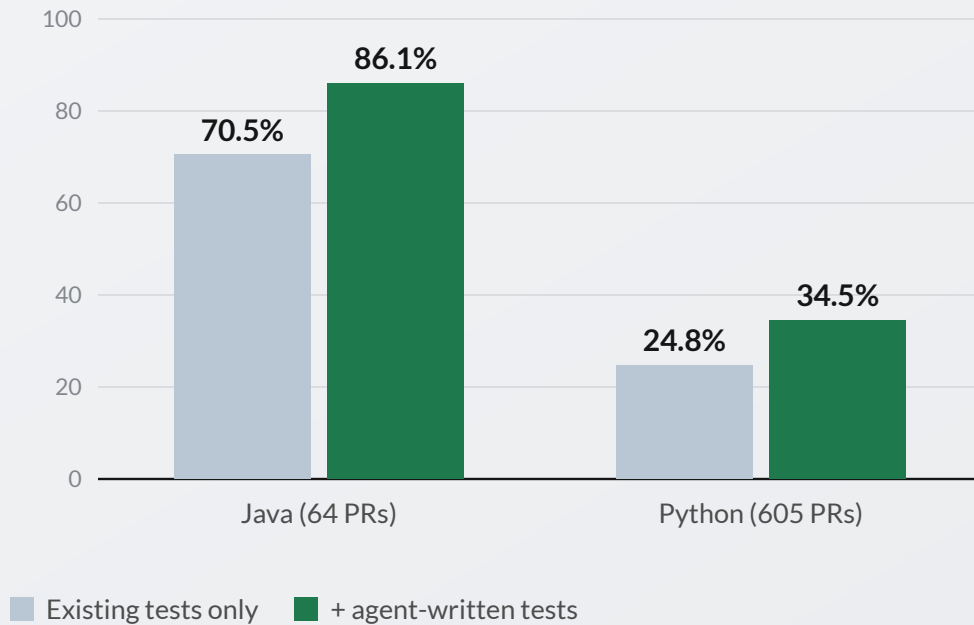
64.8% of PRs
have zero coverage

Distribution is bimodal in both languages, with mass at 0% and at 100%, but Python piles up at the zero end.

File level: Java: 50.2% of changed files fully covered, 18.5% uncovered. Python inverts it: 24.5% fully covered, 54.2% uncovered.

RQ2b: Agent Tests Help, but Only in a Minority of PRs

DIFF COVERAGE OF CODE + TESTS PRS (%)



Significant in both languages ($p < 0.001$), driven by feature PRs: +47.3 pp Java, +13.0 pp Python.

BUT THE AVERAGE HIDES THE DISTRIBUTION

35.9%

of Java Code + Tests PRs gain any coverage (23/64)

22.5%

of Python Code + Tests PRs gain any coverage (136/605)

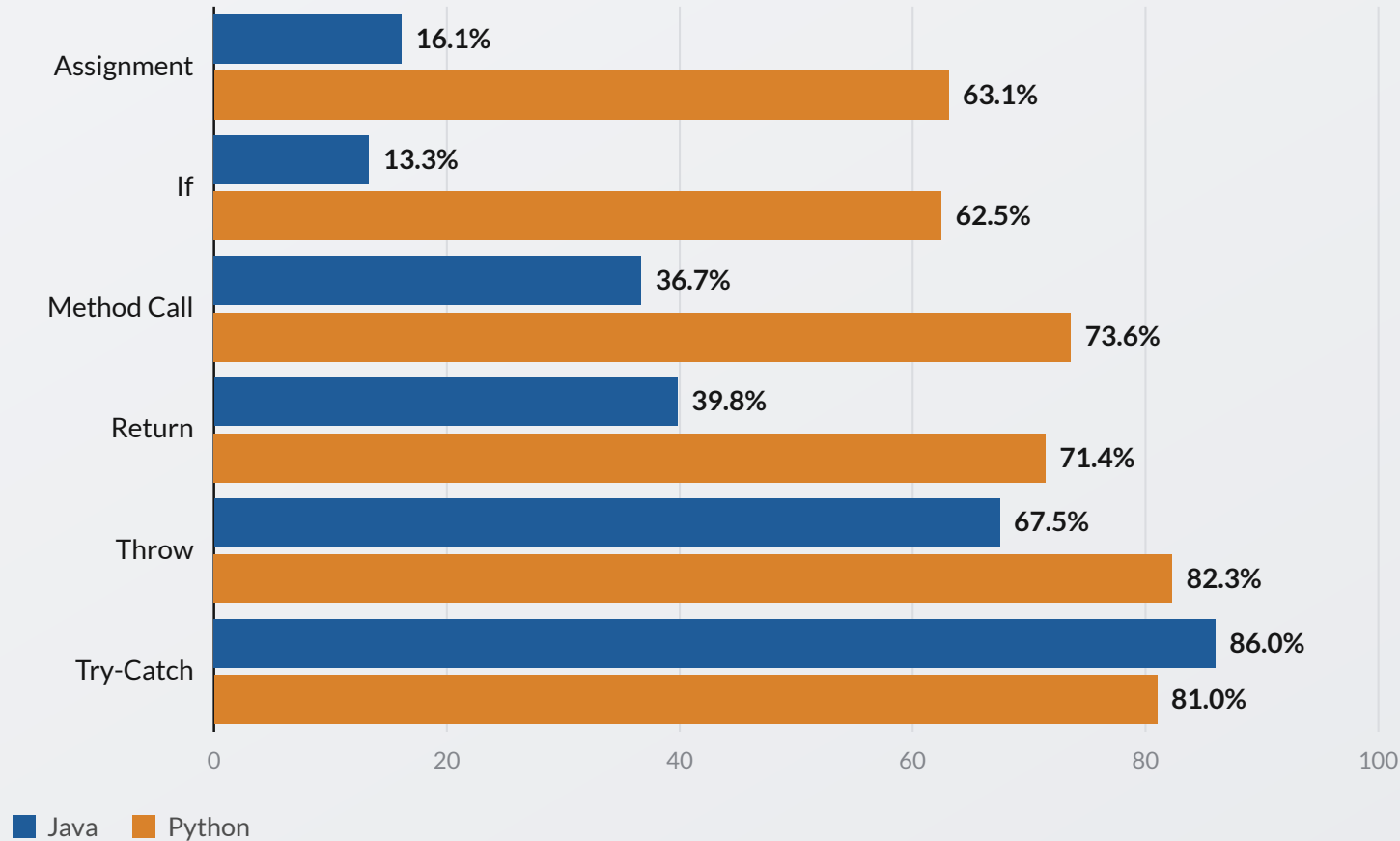
Why the rest gain nothing

Java: 42.2% already at 100%; among the rest agents delete more tests than they add (82 vs 31, 2.6×), and 51.2% only edit existing test bodies.

Python: only 8% hit the ceiling; 74.8% add new tests that never execute the lines the agent introduced.

Error Handling Is the Blind Spot

SHARE OF ADDED LINES NO TEST COVERS (%)



TRY-CATCH BODIES

86.0% Java • 81.0% Python

The only category where Java is worse than Python, despite far higher overall coverage.

Two patterns

Python misses run 2–6× higher than Java across almost every construct.

Even routine lines (assignments, calls, returns) go untested more than half the time: **agents exercise the happy path.**