Using GitHub Copilot for Test Generation in Python: An Empirical Study

Khalid El Haji, **Carolin Brandt**, Andy Zaidman

AST 2024

TU Delft

How usable are tests generated by generic code LLMs for developers?

TU Delft

Invoking Test Generations

Original human-written test

```
def testProperties(self):
    '''Test all of the twitter.Status properties'''
    status = twitter.Status()
    status.id = 1
    self.assertEqual(1, status.id)
    created_at = calendar.timegm((2007, 1, 26, 23, 17, 14, -1, -1, -1))
    status.created_at = 'Fri Jan 26 23:17:14 +0000 2007'
    self.assertEqual('Fri Jan 26 23:17:14 +0000 2007', status.created_at)
    …
```

Stripped

```
def testProperties(self):
    '''Test all of the twitter.Status properties'''
    [INSERT]
```

Test generated with Copilot

```
def testProperties(self):
    '''Test all of the twitter.Status properties'''
    status = self._GetSampleStatus()
    self.assertEqual('Fri Jan 26 23:17:14 +0000 2007', status.created_at)
    self.assertEqual(4391023, status.id)
    self.assertEqual(u'A légpárnás hajóm tele van angolnákkal.', status.text)
    self.assertEqual(self._GetSampleUser(), status.user)
```

TU Delft

Project Selection + Study Execution

7 Open-source python projects

GitHub (Less popular ones) + GitLab

Generate batch of tests

Manually analyze label problems in generation

Repeat until theoretical saturation → 53 test pairs

Aspects of Usability

Syntactic Correctness

Runtime Correctness

Passing

Coverage

TU Delft

Variation 1: Invoking Generations Without a Test Suite

Original test code file (With-Context) [RQ1]

```
import pytest
from gcip import Cache, CacheKey, CachePolicy

def test_cache_policies():
    expected_members = ["PULL", "PULL_PUSH"]
    for member in CachePolicy.__members__:
        assert member in expected_members

def
test_default_cache_key_matches_ci_commit_ref_slug():


def test_cache_key_with_custom_value():
    cache_key = CacheKey(key="mykey")
    expected_render = "mykey"
    assert expected_render == cache_key.render()
    assert cache_key.key == "mykey"
    assert cache_key.files is None
    assert cache_key.prefix is None
```

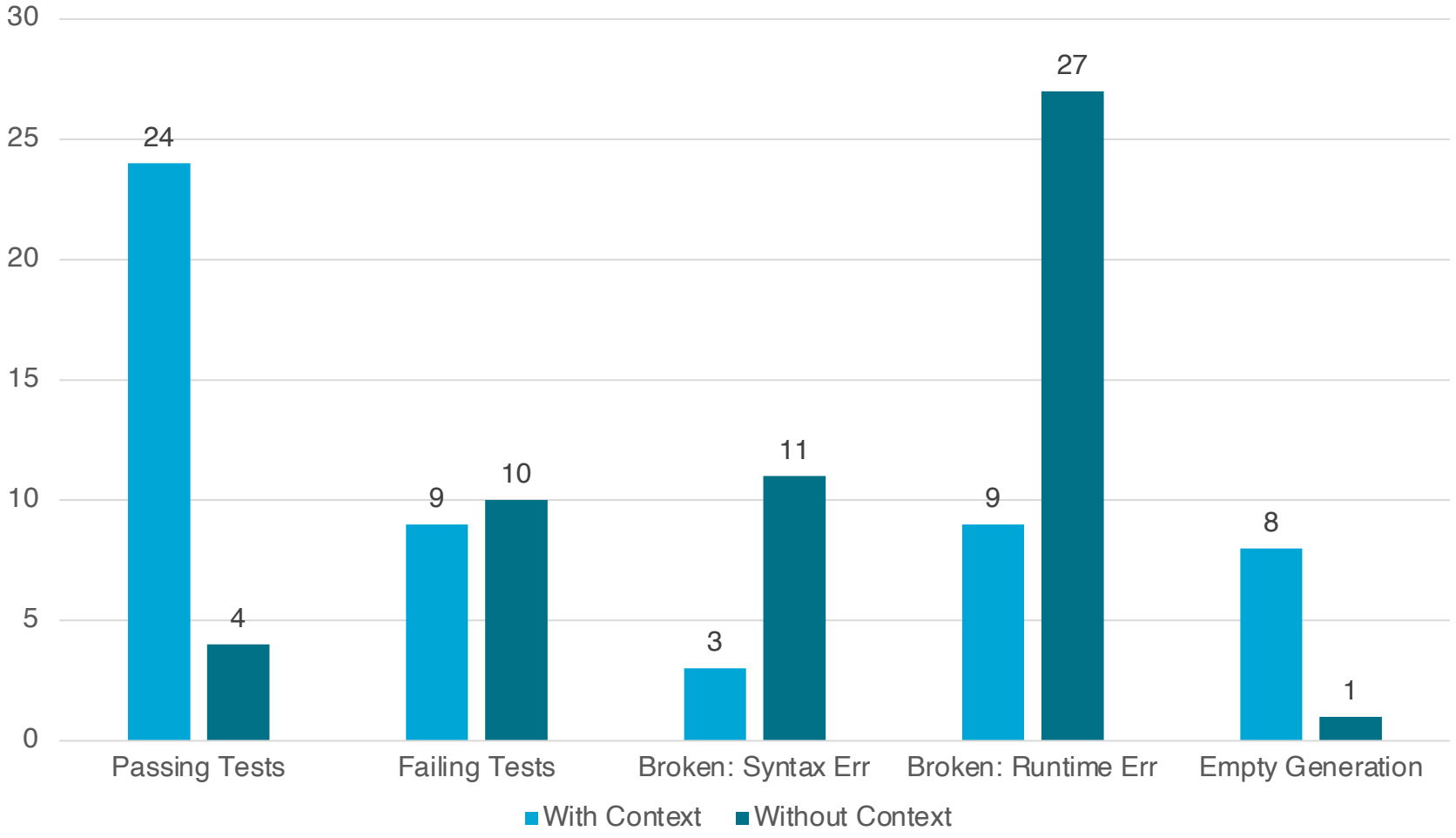
Stripped test code file (Without-Context) [RQ2]

```
import pytest
from gcip import Cache, CacheKey, CachePolicy

def
test_default_cache_key_matches_ci_commit_ref_slug():
```

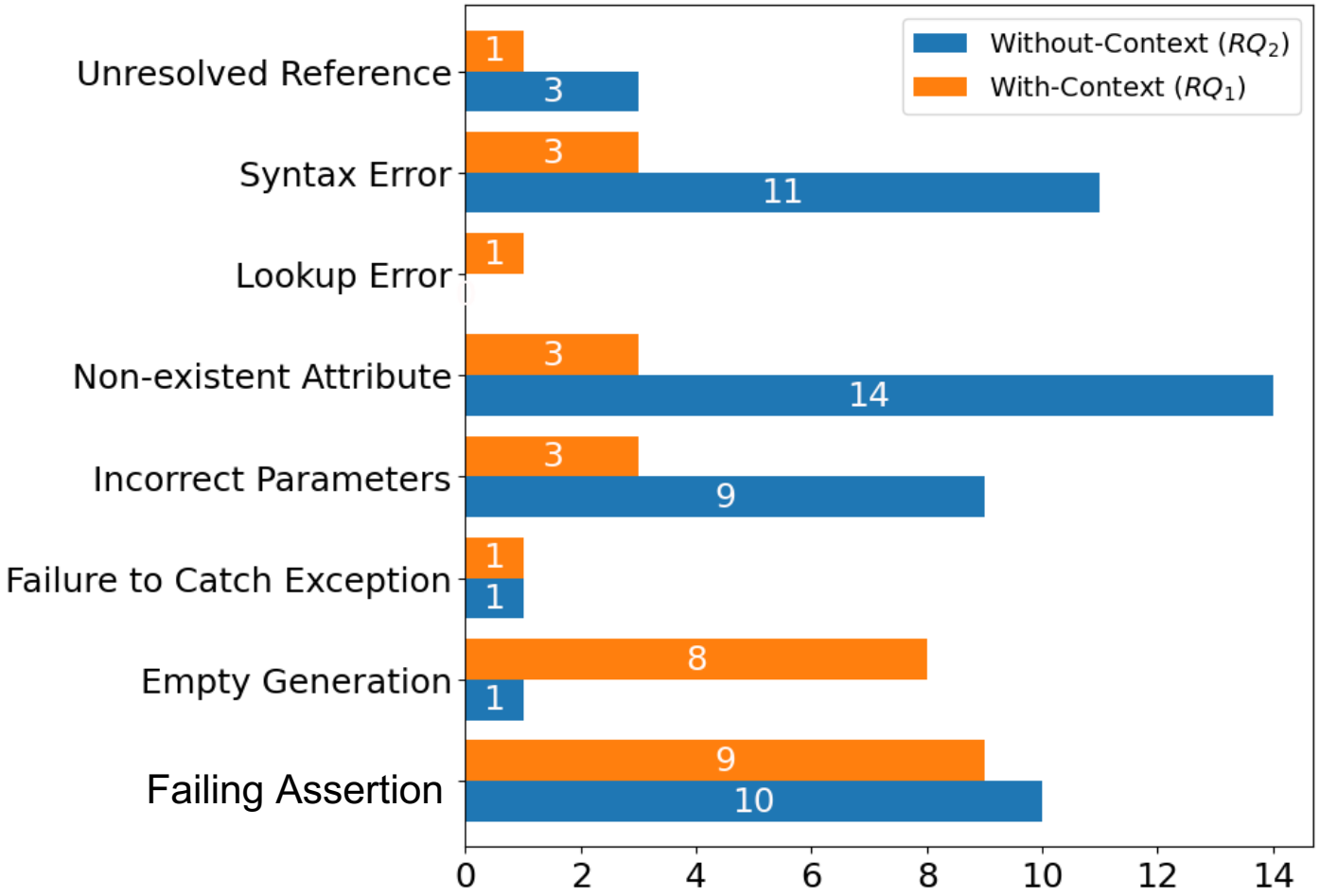
TU Delft

Copilot Generations With + Without Testsuite Context



TU Delft

What were the problems?



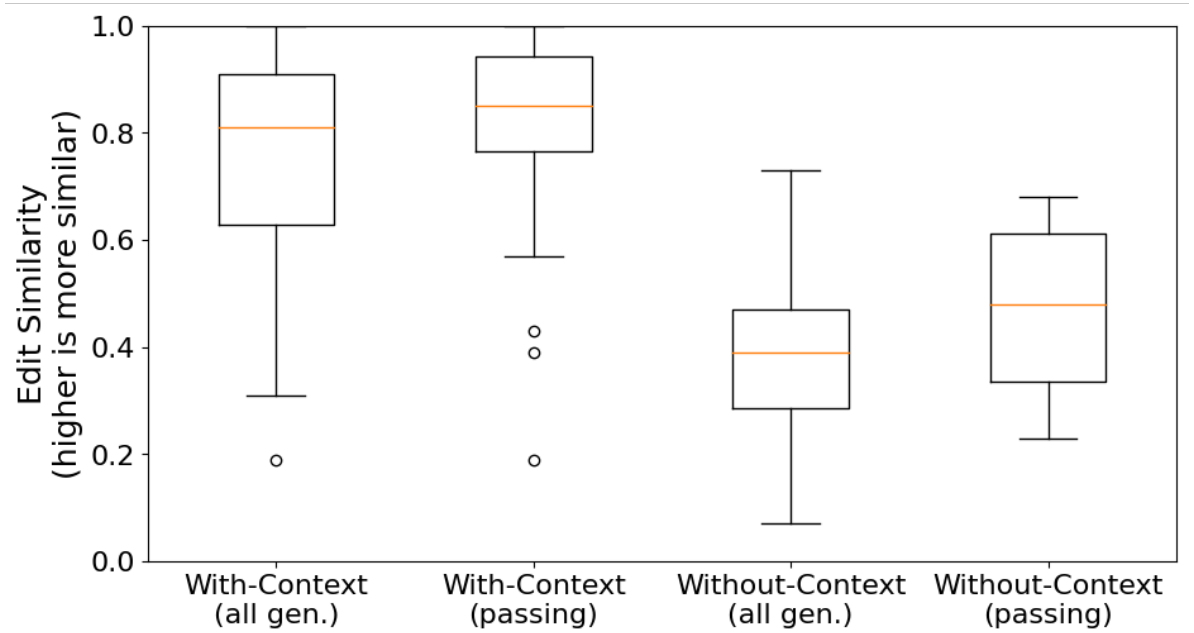
Observation: Mimicking Behavior

Test generated by Copilot

```
def test_option_optional():
    cli = Command("cli", params=[Option(["-c"], optional=True)])
    assert _get_words(cli, ["-c"], "") == []
    assert _get_words(cli, ["-c"], "-") == ["--help"]
```

Similar test in the same test file

```
def test_option_count():
    cli = Command("cli", params=[Option(["-c"], count=True)])
    assert _get_words(cli, ["-c"], "") == []
    assert _get_words(cli, ["-c"], "-") == ["--help"]
```



TU Delft

Variation 2: Different Test Method Comments

Minimal Method Comment (e.g, `"""Test the x function"""`)

Behavior-Driven Development Comment (e.g, `"""Given x when y then z"""`)

Usage Example Comment (e.g, `"""example: <code snippet> gives: <output>"""`)

Combined Comment

- For 23 tests where generation did not work
- Formulate comments based on original test
- Manually analyze problems again

TU Delft

Variation 2: Different Test Method Comments

With-Context	Minimal Method Comment	Behavior-Driven Development Comment	Usage Example Comment	Combined Comment
Passing	21.74%	26.09%	**34.78%**	26.09%
Failing	34.78%	30.43%	34.78%	34.78%
Broken	30.43%	30.43%	**17.39%**	26.09%
Empty	13.04%	13.04%	13.04%	13.04%

Without-Context	Minimal Method Comment	Behavior-Driven Development Comment	Usage Example Comment	Combined Comment
Passing	17.39%	13.04%	**21.74%**	**21.74%**
Failing	30.43%	26.09%	30.43%	47.83%
Broken	52.17%	60.87%	47.83%	**30.43%**
Empty	0.00%	0.00%	0.00%	0.00%

TU Delft

Takeaways

Generating tests **within an existing test suite**:

Poor usability, most generations will need to be edited

A code example in test method comments improves usability

Generations will likely mimic existing tests, can be useful for writing repetitive tests

Generating tests **without an existing test suite**:

Extremely poor usability, almost all generations will need to be edited

Instructive natural language combined with a code example in test method comments improves usability

Using GitHub Copilot for Test Generation in Python: An Empirical Study

Khalid El Haji, Carolin Brandt, Andy Zaidman

AST 2024

TU Delft

How usable are tests generated by generic code LLMs for developers?

Mon 15 April --- 14:00 --- Test Generation Session at Amália Rodrigues

Using GitHub Copilot for Test Generation in Python: An Empirical Study

Khalid El Haji, **Carolin Brandt**, Andy Zaidman

AST 2024

TU Delft