A Replication Study on the Usability of Code Vocabulary in Predicting Flaky Tests

Mining Software Repositories, 17-19 May, 2021


Guillaume Haben


Sarra Habchi


Mike Papadakis


Maxime Cordy


Yves Le Traon

What is a flaky test?

*“ A test that **passes** and **fails** for the same version of a program ”*

An example of a flaky test

```python
# https://github.com/python-telegram-bot/python-telegram-bot/blob/master/tests/test_updater.py
def test_idle(self, updater, caplog):
        updater.start_polling(0.01)
        Thread(target=partial(self.signal_sender, updater=updater)).start()
        with caplog.at_level(logging.INFO):
            updater.idle()
        rec = caplog.records[-2]
        assert rec.getMessage().startswith('Received signal {signal.SIGTERM}')
        assert rec.levelname == 'INFO'
        rec = caplog.records[-1]
        assert rec.getMessage().startswith('Scheduler has been shut down')
        assert rec.levelname == 'INFO'
        # If we get this far, idle() ran through
        sleep(0.5)
        assert updater.running is False
```

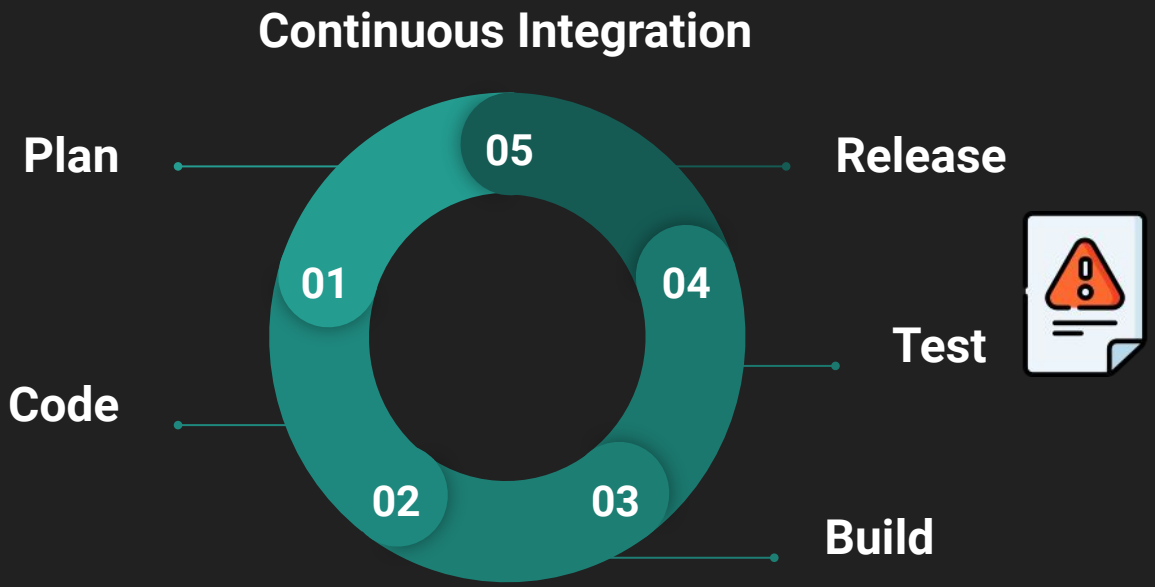
Thread usage

Hard-coded wait

Common causes for flakiness

- Concurrency issues
- Usage of Date / Time
- Test order dependencies
- I/O (File, database)
- Network calls
- ...

Why does it matter?



Flakiness impact many companies including Google, Microsoft, Spotify, Mozilla...

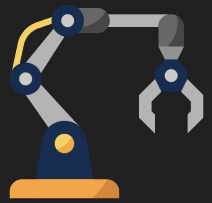
Hypothesis

It is possible to train a model to predict if a test will be flaky or not

Goal: Help developers find flaky tests without executing them.

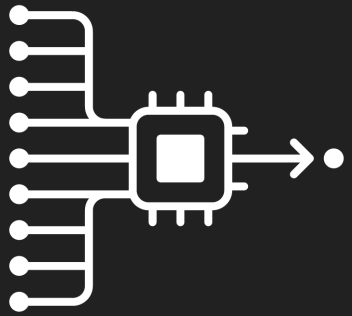
Feature engineering

Dataset of flaky and non flaky tests



Tokenize and vectorize tests (Bag of words)

Supervised classifier



0: Non flaky
1: Flaky

The original study

MSR ‘20, *What is the Vocabulary of Flaky Tests*[3]

Algorithm	Precision	Recall	F1	MCC	AUC
Random Forest	**0.99**	0.91	**0.95**	**0.90**	**0.98**
Decision Tree	0.89	0.88	0.89	0.77	0.91
Naive Bayes	0.93	0.80	0.86	0.74	0.93
Support Vector	0.93	**0.92**	0.93	0.85	0.93
Nearest Neighbour	0.97	0.88	0.92	0.85	0.93

[3] http://gustavopinto.org/lost+found/msr2020.pdf

Replication study

We replicate the original study with 3 goals in mind. We want to:

1. Validate the approach by applying a different evaluation methodology
2. Consolidate the generalizability of the approach
3. Extend the approach by using new features

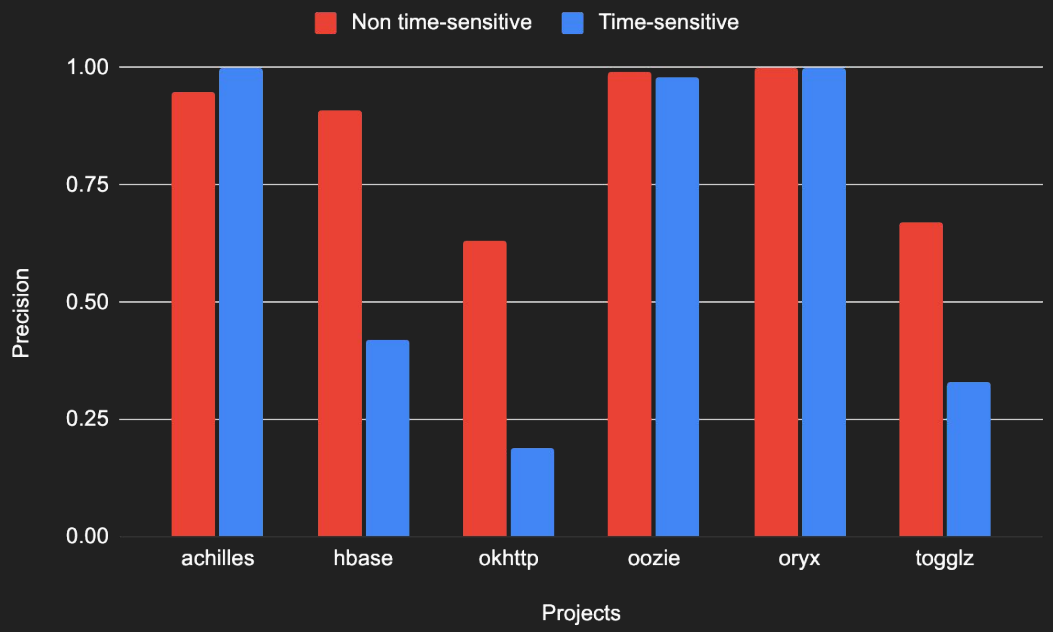
Validation using realistic settings

Dataset: DeFlaker[2]
1,348 flaky tests from 6 Java projects

- Intra-project analysis
- Time-sensitive analysis

[2] https://www.deflaker.org/

Validation using realistic settings



Key findings:

Performance decrease under a time-sensitive validation.

but...

Models are still able to decently predict flaky tests.

Conclusion

A replication study on the usability of code vocabulary in predicting flaky tests

- Flaky tests prediction is possible
- A new dataset of flaky tests built from Python projects.

Replication package: https://github.com/GuillaumeHaben/MSR2021-ReplicationPackage