Pseudo-tested Methods in an Android Continuous Integration Environment

John Kvarnefalk (johnkv@kth.se)

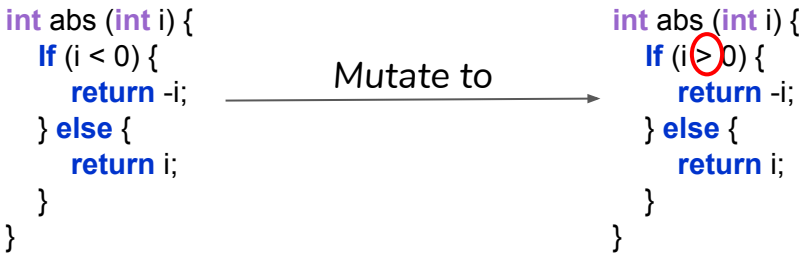
Supervisor: Deepika Tiwari (deepikat@kth.se)
Examinator: Martin Monperrus (martin.monperrus@csc.kth.se)

Introduction

- Mobile is everywhere
- Complex and important applications
- Robust and high quality mobile applications
- Write tests!
    - But do they work as expected?

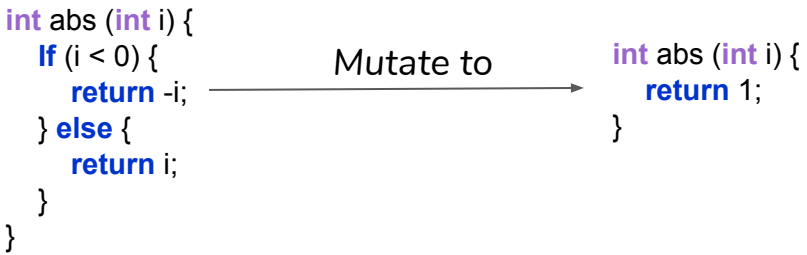
Mutation Testing

- PITest [1]



Extreme Mutation Testing

- Pseudo-tested methods
- Descartés [2]



Mutation Testing Downsides

- Time-consuming
- Produce lot of errors
  - Some irrelevant
- Developers don’t run manual tools [3]

Continuous Integration (CI)

- Practice of merging code changes into a shared master
- Quality checks
- Pull requests
- CI Server

Mutation testing in CI

- Implemented on Large-scale CI system at Google [4]
- Run on changed lines
- Heuristics to select mutation operator
- Useful to developers

What we Explore

- Extend work on regular mutation testing in CI
- Descartés in CI
- Large-scale Android project
- Interviews to explore sentiments

RQs

RQ1: *According to the developer feedback collected, what is the perceived usefulness of presenting pseudo-tested methods within the CI system?*

RQ2: *What can be concluded about developers' opinions on the highlighted pseudo-tested methods, judging by the interviews?*

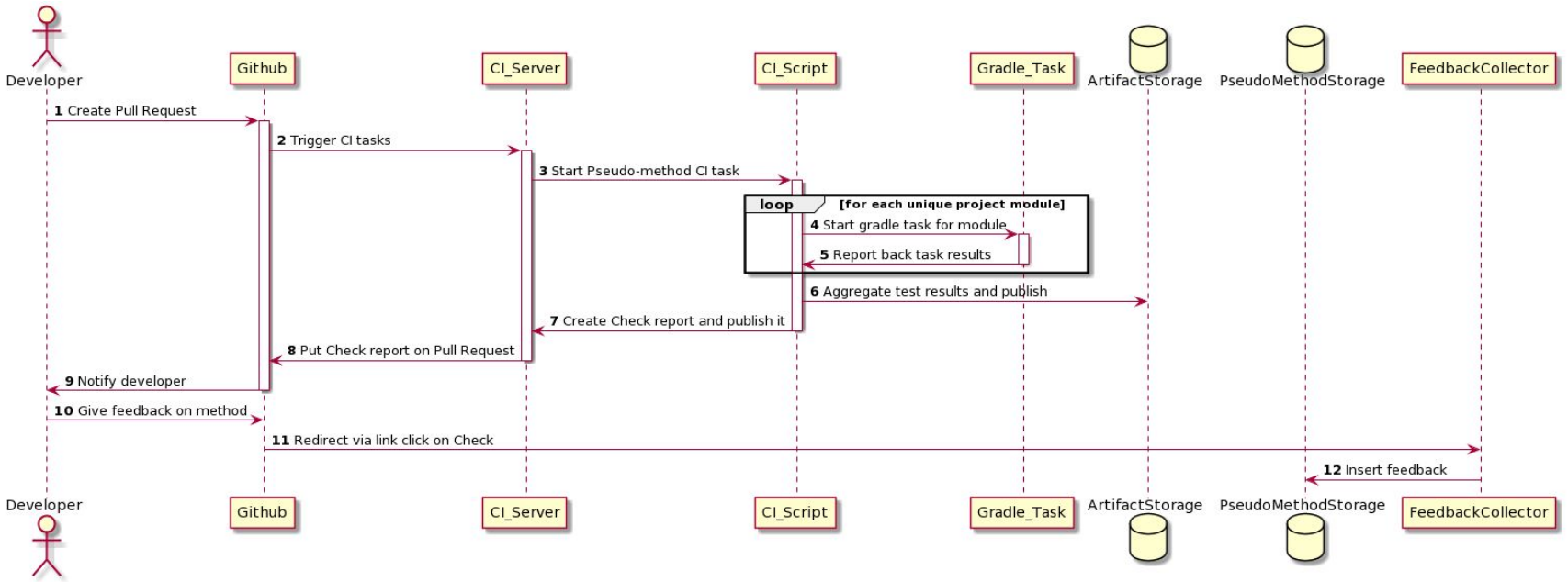
RQ3: *To what extent do developers take actions on improving their tests after being presented with the report on pseudo-tested methods?*

RQ4: *What is the feasibility of classifying pseudo-tested methods as interesting or irrelevant?*

Methodology

- Setup architecture to detect pseudo-tested methods in CI
  - Also collect developer feedback
- Perform interviews with developers who’ve given feedback
  - Invites to all developers
  - Nudged them if no response
  - Meaning units into themes

Architecture



Report Format

videoCapabilities(VanillaUriHelper)

Click here to view the method

The entire body of this method has been replaced by:

- `return "A";`
- `return "";`
- `return null;`

Yet, all of the test cases in the test suite passed.
These are the test cases exercising this method:

- `testAddRequestParametersContainsTheExpectedParamsAndValuesWithVideosEnabled` **Click to View**
- `getQueryMap` **Click to View**
- `testAddRequestParametersContainsTheExpectedParamsAndValuesWithVideoDisabled` **Click to View**
- `testAddRequestParametersContainsTheExpectedParamsForNft` **Click to View**

Consider fixing this to make sure that this method is properly tested.

Please press on one of the links below with your decision. We use this feedback for research and improving this tool

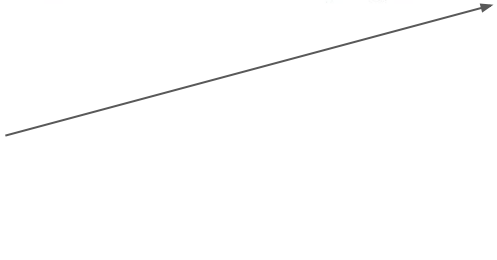
👷 **I will fix this** 👷

🛠️ **This is a HIGH priority fix** 🛠️

🔧 **This is a LOW priority fix** 🔧

🤷 **This is not worth fixing** 🤷

❌ **This report is not correct** ❌



Thanks a lot for giving feedback

You voted that this was a low priority fix.

If you can provide an additional comment, it would be awesome!

Enter comment here...

Add Comment

If you have any questions. Feel free to DM me on Slack or send me an email.

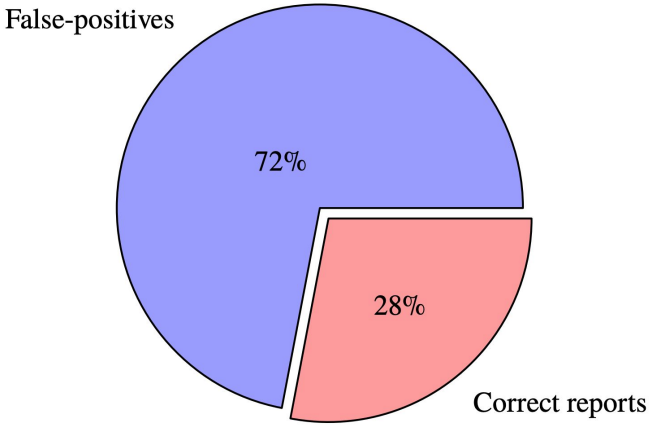
Experiment

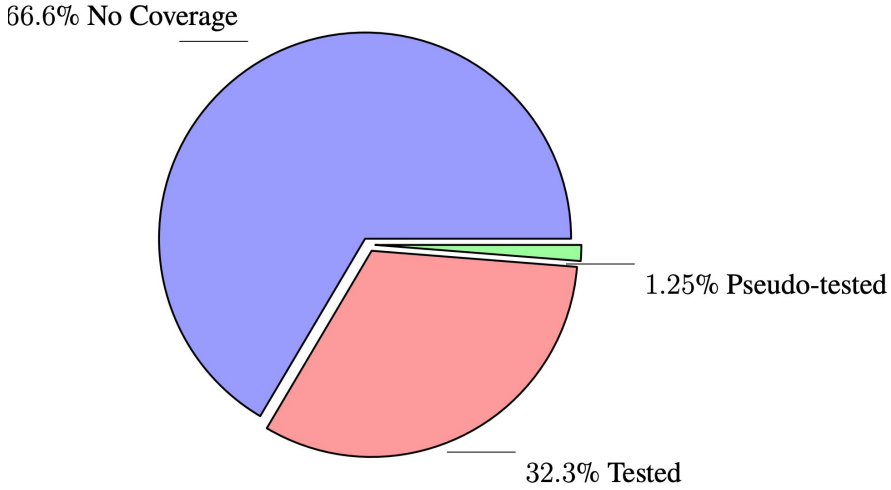
- 2020-04-22 until 2020-04-28 (24 working days)
- Manually nudged developers

False Positives

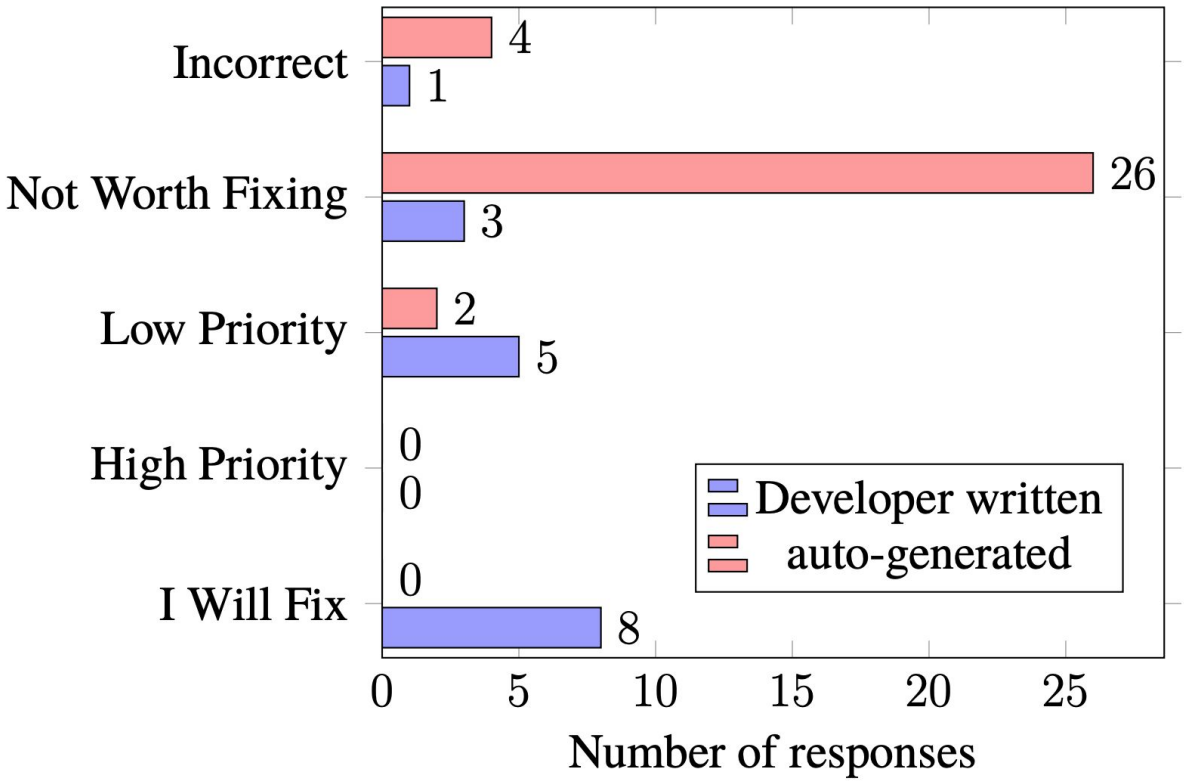
- Discovered twice
  - During development
  - During experiment
- Robolectric Framework
- Unidentified root cause

Context of Results





Votes on Pseudo-tested Methods



Additional Comments

“This is an auto-generated class. The stated method in the report is generated. It is implemented by an external Google tool and I think we generally assume it to be correct. We never test the validity of these methods”

“I did not change any implementation. I only renamed a field.”

“This particular PR is updating a feature my particular team does not own.”

Action on Errors

- 8 Methods marked as “I will fix”
- Nobody fixed

Interview Key Takeaways

- Understanding increase usefulness
- Guide the developer through the issues
- Auto-generated methods are useful to test

Discussion

- Auto-generated methods
- Report context
  - Structure
  - Status
  - Bug tickets
- Small sample size
- Type of change

Conclusion

- Indications that tool is useful
- Developers took no action
- Hard to build classifier
- Future work
  - Fix false positives
  - Investigate structure of report
  - Based on or type, change behavior

References

[1] - Coles, Henry, Thomas Laurent, Christopher Henard, Mike Papadakis, and Anthony Ventresque. "Pit: a practical mutation testing tool for java." In Proceedings of the 25th International Symposium on Software Testing and Analysis, pp. 449-452. 2016.

[2] - Vera-Pérez, Oscar Luis, Martin Monperrus, and Benoit Baudry. "Descartes: a PITest engine to detect pseudo-tested methods: tool demonstration." In 2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 908-911. IEEE, 2018

[3] - Johnson, B., Song, Y., Murphy-Hill, E. and Bowdidge, R., 2013, May. Why don't software developers use static analysis tools to find bugs?. In 2013 35th International Conference on Software Engineering (ICSE) (pp. 672-681). IEEE.

[4] - Petrović, Goran, and Marko Ivanković. "State of mutation testing at google." In Proceedings of the 40th international conference on software engineering: Software engineering in practice, pp. 163-171. 2018.