Artificial Intelligence Based Software Engineering

Project Report

Continuous Test Generation on Pull Request

Group 5
Mathias Sixten
Yixuan
David Budischek, 201863
Gaspard
Gustav Janér, 20186416

December 3, 2018

Abstract

-

1 Introduction

Over the past decades software release cycles have shortened repeatedly. Two decades ago it was common place to release software and then maybe follow up with a patch (in the form of a physical hard disk) a few years down the line. From that we went to pushing out monthly, or even weekly, security updates. In recent years though, it became common place to not bundle releases anymore. Instead every single feature/bug/typo fix is published instantaneously and pushed to every user. With Continuous Delivery come a plethora of challenges. One of which is testing. In a traditional release cycle there is enough time to hand over the release candidate to QA and give them time to verify functionality. But this won't do nowadays. Instead we rely on automated tests to quickly verify functionality. In practice, tests can never guarantee a bugfree product, instead they can only try and cover as much of the code as possible.

In addition to manually writing tests there are advanced unit test generation tools available (such as EvoSuite or Pex). While these tools require no manual input, they do require a lot of computation time. In their 2014 paper [1] the authors of EvoSuite propose a solution to this issue, Continuous Test Generation. Instead of a greenfield test generation for each new iteration, they leverage previous results to speed up the test generation process. In their research this is done by smartly allocating resources to the classes that changed or are not yet fully covered.

When it comes to integrating CTG in the CI/CD workflow commonly used in modern Software projects there is little literature available. In a case study done as part of the SSBSE'15 challenge [2] we can clearly see that CTG performs well enough to warrant implementation. Due to this we implemented an out of the box, platform agnostic tool to easily add CTG to an existing project without interfering with the infrastructure that has previously been set up.

2 Body

3 Results

4 Conclusion

References

[1] J. Campos, A. Arcuri, G. Fraser, and R. Abreu, "Continuous test generation: Enhancing continuous integration with automated test generation," in *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*, ser. ASE '14, New York, NY, USA: ACM, 2014, pp. 55–66, ISBN: 978-1-4503-3013-8. DOI: 10.1145/2642937.2643002. [Online]. Available: http://doi.acm.org/10.1145/2642937.2643002 (visited on 10/29/2018).

[2] J. Campos, G. Fraser, A. Arcuri, and R. Abreu, "Continuous test generation on guava," in *Search-Based Software Engineering*, M. Barros and Y. Labiche, Eds., vol. 9275, Cham: Springer International Publishing, 2015, pp. 228–234, ISBN: 978-3-319-22182-3 978-3-319-22183-0. DOI: 10.1007/978-3-319-22183-0_16. [Online]. Available: http://link.springer.com/10.1007/978-3-319-22183-0_16 (visited on 12/03/2018).