Assignment 5: Literature Review

Xiaoxiao Gan

1 Introduction

This mini-literature review is focused on the topic of testing automation in continuous integration and continuous delivery (CI/CD). In other words the papers we discussed in this assignment are associated with this topic. Continuous practices nowadays are becoming a critical part in software development, testing is an important phase in software development process and CI/CD process as well. However, according to the prior study, testing still is one of the biggest challenges in adopting CI/CD practices [1]. This assignment will review 6 papers regarding this topic. The reviewed papers will be divided into 2 different sets based on their theme. The first one will be different techniques for testing in CI/CD environments to improve software developers productivity. The second one will be the different studies conducted to understand practitioners and researchers' perceptions.

2 Review Topic: Different Strategies and Techniques

There are several strategies implementing in CI/CD environment to optimize testing stage, including test case prioritization, test case selection, execution [2], and automated generated unit test [3]. There also researchers proposed a mixed method incorporating different techniques to improve the regression testing process.

Summary: The first paper we reviewed is from Sebastian Elbaum et al. [4], who presented two new algorithms to select and prioritize based on test suited execution history, to improve regression testing under CI environment. On the other hand, Maral Azizi et al. [5] also focuses on test case prioritization and selection, but takes advantage of information retrieval method to automatically select test cases based on the code similarity. Different from the previous two papers, Campos et al. [3] trying to enhance the testing process in CI environment by automating generating test cases continuously. They extend the classical test generation tool EVOSUITE [6], and incrementally test the units based on historical tested data.

Review: The above papers all have commonalities and differences. The first one and second one are all trying to improve testing process efficiency in CI/CD environment on test case prioritization and selection, but their algorithms are based on different data sources, while one is using commit history and developer feedback data, and another one is using data of code changes. The third one, which is similar to the first one that tries to use historical data, but aims to improve the automated test case generation process instead of test case selection and prioritization. For the evaluation, all three papers conducted empirical studies to evaluate their approach performance, they all used different dataset for the evaluation (Google public dataset, 37 versions of 6 open source applications and open source projects randomly selected from SourceForge and GitHub). Their proposed approaches all presented some degree of improvement. However, their approaches all have different limitations, such as some evaluation only testing in Java projects, the ones using historical data will require projects already have enough information to use, which makes the new project hard to take advantage of these methods. Additionally, evaluation only on open source data without using industrial software projects will also be a threat on generality. In general, it is refreshing to see there are different viewpoints and different methods that can be adopted to improve the test process under CI environment, which is strict on time compared with the traditional testing process.

3 Review Topic: Case Study, Survey and Systematic Review

Besides studies that proposed novel techniques to improve testing process, there are also researchers conducting surveys, case study and empirical study on real world projects under CI/CD environment to offer practical insights.

Summary: Johannes Gmeiner et al. [7] investigated an online business company software development process, to gain the understanding of position and usage of automated testing in continuous delivery process and the related challenges from lessons learned. Yuqing Wang et al. [8], on the other hand, surveyed open source repositories contributors and mined software repositories to identify the correlation of testing automation maturity in CI context with the final product quality. The last paper [1], presented a systematic literature review for peer-reviewed papers to understand the current continuous practices and related methodologies, difficulties analysis and classification, which also includes the testing factor in the paper.

Review: Similar to the papers in the first section, the above papers have commonalities and differences as well. While all three papers focus on understanding the testing process situation in CI/CD environment, they use different sources - the first one just studied from a real world application (the online business web application), which is more relatable but may lack representativeness. The second not only studies the real world open source repositories, but also gains perception of developers who are working on this project, which will give a more comprehensive understanding. Additionally, compared with the case study paper which just simply studies without any target or research questions, the second one already targets test automation maturity, and trying to figure out if this element has any strong impacts on product quality, this will make the paper more focused and have a clear goal. But their dataset may not be representative enough as they only use Java projects that use Travis CI. The third one, instead of studying real world cases, chooses to review and organize the published papers that already gain insight from the experimental study. This literature review provides a more researcher-centered viewpoint to help capture different aspects and impacts of continuous practice in testing, and provides a more thorough analysis. Such as the biggest challenges of CI is lack fully automated testing. However, different from first section papers, this section's papers only analyze existing data or prior research without providing any novel approaches. Nonetheless, these papers are beneficial for new researchers who are trying to gain a basic knowledge of current situations and different guidance or direction to extend the research.

References

[1] Mojtaba Shahin, Muhammad Ali Babar, and Liming Zhu. Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices. *IEEE Access*, 5:3909–3943, 2017.

[2] Yang Yang, Zheng Li, Liuliu He, and Ruilian Zhao. A systematic study of reward for reinforcement learning based continuous integration testing. *Journal of Systems and Software*, 170:110787, 2020.

[3] José Campos, Andrea Arcuri, Gordon Fraser, and Rui Abreu. Continuous test generation: Enhancing continuous integration with automated test generation. ASE '14, New York, NY, USA, 2014. Association for Computing Machinery.

[4] Sebastian Elbaum, Gregg Rothermel, and John Penix. Techniques for improving regression testing in continuous integration development environments. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 235–245, 2014.

[5] Maral Azizi. A tag-based recommender system for regression test case prioritization. In *2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 146–157, 2021.

[6] Gordon Fraser and Andrea Arcuri. Evosuite: Automatic test suite generation for object-oriented software. ESEC/FSE '11, page 416–419, New York, NY, USA, 2011. Association for Computing Machinery.

[7] Johannes Gmeiner, Rudolf Ramler, and Julian Haslinger. Automated testing in the continuous delivery pipeline: A case study of an online company. In *2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 1–6, 2015.

[8] Yuqing Wang, Mika V. Mäntylä, Zihao Liu, and Jouni Markkula. Test automation maturity improves product quality—quantitative study of open source projects using continuous integration. *Journal of Systems and Software*, 188:111259, 2022.