Assignment 7: NSF Research Proposal

Xiaoxiao Gan

1 Project Summary

We request support to study the correlation and potential improvement of GUI Testing in Continuous Integration / Continuous Delivery environment. As we mentioned in previous multiple assignments, "CI/CD workflow implementation is essential in nowadays software development process. GUI Testing, as an important part of a lot of software products such as web applications, plays a crucial role in the testing phase of CI/CD workflow. However, little is known about how exactly popular functional testing techniques, such as Selenium-based technologies, are adopted in CI/CD workflow and their impact on software development performance and productivity. The lack of understanding and standard guidance of such usage causes practitioners to hesitate to migrate their projects including critical GUI tests to modern CI/CD practice or adopting GUI testing in their existing CI/CD workflow, which will impact the final product development performance and quality." Therefore, we proposed to conduct an empirical study on the use of Selenium-based technologies automated by CI/CD service in Github, which means we are planning to scrape data from open source repositories, identify patterns, impacts and issues in the projects that both using GUI testing techniques and CI/CD practice, in order to generalize the situation of GUI testing technologies usage in CI/CD environment. After the preliminary study, we are expected to propose appropriate solutions to address the issues found by preliminary study, ultimately improve the GUI testing usage and adoption in modern projects to improve the software development process. We are aiming to collect and analyze data from popular open sources repositories which use CI/CD workflow and Selenium-based technologies. We expect the result indicates the common patterns of Selenium-based technologies used in open source repositories and the potential challenges of such adoption. Based on the significance of GUI testings and the popularity of CI/CD practice in modern projects, we believe our findings and proposed solutions are valuable not only to practitioners who are considering adopting functional GUI testing in their CI/CD workflows, but also insightful for researchers who is interested in CI/CD adoption and testing.

After literature review, we found there is no clear answer for the automated functional testing usage in CI/CD workflow, compared with other integration of functional testing in CI/CD workflow, which a lot of prior studies focus on. This means there is a clear gap in the current situation of functional testing usage in the CI/CD environment that requires researchers to work on it and fill in.

In conclusion, this proposal seeks to fill in the gap of unknown knowledge of GUI functional testing technologies (such as Selenium) adoption in the literature. By scraping the project data from open source repositories, identifying the pattern, impact and issues of GUI testing usage in CI/CD environment, proposed and developed a useful tool to address the issues found by data analysis. We believe not only the preliminary study can provide guidance to the companies and software developers who want to add GUI testing to the project or who want to integrate CI/CD workflow into a GUI-testing required projects, but also they can use the tool to perform a smooth migration or integration, which can significantly improve software developer productivity and product quality

in the long term.

2 Intellectual Merits

The main research objective of this project is investigating the current situation of GUI testing in CI/CD environments, figuring out the latent impact of GUI testing usage, and looking for what kind of strategies can be used to improve the GUI testing practices in CI/CD environment, ultimately improve the product quality and maintenance efficiency after release. In particular, we focus on analyzing projects on open source repositories (GitHub) which also use modern CI/CD tools (Jenkins, Github Actions and Travis CI). After categorization and visualization the projects that use GUI testing techniques in CI/CD framework, we will work on looking for the impact of GUI testing on the project, such as time of release, time of bug fix, to identify the GUI testing element correlation with project development. We also work on identifying the issues that will hinder the GUI testing usage in CI/CD. For example, will a specific GUI testing programming language tend to have more CI/CD build failure in the whole project compared to other open source repositories projects.

After getting enough insight into GUI testing, we wish to propose a promising solution to fill in the gap and address the issues we investigated from preliminary study, and that solution can help developers adopt GUI testing in projects. This kind of solution can be in any form, such as a plugin that helps adding GUI testing in CI/CD scripts, and suggest when is best time and position to add GUI testing in the projects. Finally, we will evaluate the proposed solution to existing projects to see if tools can actually improve the product quality and maintenance effectiveness and efficiency. Additionally, we are also considering collecting software engineers' opinions on the solution evaluation, the potential strategies including using surveys and interviews to the software engineers who are working on our target study repositories in the open source coding platform (such as Github). Since they are definitely more familiar and expertise in the CI/CD environment and GUI testing techniques.

3 Broader Impacts

There will be several important impacts of this project. First, this project can fill in the gap of lack of knowledge of GUI testing usage and related impact in CI/CD practices. As we mentioned before, little is known about how exactly popular functional testing techniques, such as Selenium-based technologies, are adopted in CI/CD workflow and their impact on software development performance and productivity. Therefore, this can provide guidance for the tech companies who considering transfer traditional delivery process to modern CI/CD practice, or who want to apply GUI testing to the project but hesitate to do that due to laborious work. Second, this project findings also can offer directions for researchers who is interested in improve either GUI testing efficiency or CI/CD adoption efficiency.

On the other hand, the tool or plugin developed by the project will be useful for software engineer to adopt the GUI testing easier in CI/CD environment, which can ultimately improve product quality. Finally, this project also can encourage the companies who is considering adopt CI/CD practice in their projects but hesitate to do so due to great deal of work for unstable GUI testings. Since testing flakiness is an crucial in impact software development process and developer productivity, as it requires software engineer times to debug, locate issues and come up with repairing solution. Especially, this will be helpful for small or medium size companies, which

not have enough budget for QA assessment like big tech companies but still looking for improving their product quality and reduce the cost of maintenance after product delivery.

4 Literature survey/Related work

There are different kind of research working on testing automation in continuous integration and continuous delivery (CI/CD). One of them is investigating several strategies implementing in CI/CD environment to optimize testing stage, including test case prioritization, test case selection, execution [1], and automated generated unit test [2]. There also researchers proposed a mixed method incorporating different techniques to improve the regression testing process. For instance, ebastian Elbaum et al. [3], 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. [4] 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. [2] trying to enhance the testing process in CI environment by automating generating test cases continuously. They extend the classical test generation tool EVOSUITE [5], and incrementally test the units based on historical tested data.

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.

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.

Johannes Gmeiner et al. [6] 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. [7], 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 [8], 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.

Similar to the papers focus on strategies, 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.

5 Proposed Research

The proposed research will be divided into four main research activities, the research activities detail is listed in below:

The first phase activity will be literature review and topic exploring. In this stage, we will conduct a system literature review on all related papers / research on GUI testing, continuous integration and continuous delivery, and their cross-disciplines papers. This literature review will help to have great fundamental understanding on our topic, which is GUI testing in CI/CD environment. Based on what we learned from the literature review, we will refine and finalize our research questions to be more focused on our goal, and use the research questions to guide and design the study. Currently, the proposed research questions can be described as (which we already mentioned in research question assignment:

RQ1: How are SELENIUM-based functional tests technologies used in GitHub Actions?

RQ2: What is the pattern and impact of Selenium-based functional tests in GitHub Actions?

RQ3: What is the issue preventing adopting Selenium-based functional tests in GitHub Actions?

RQ4: What solution can we use to address the issue?

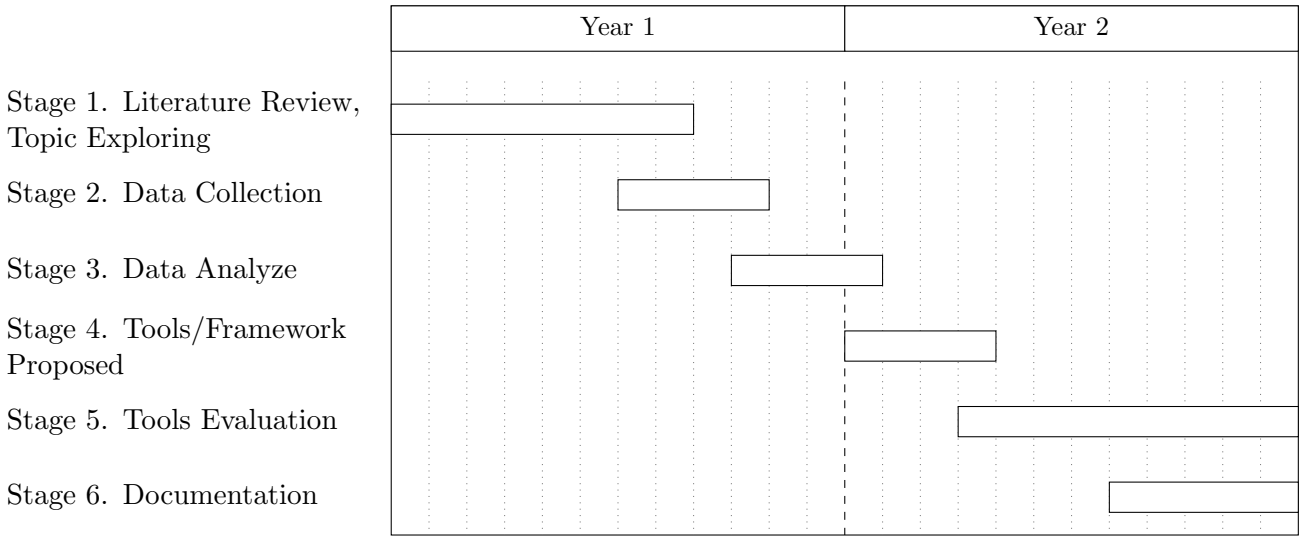
The second phase activity will be data collection and data analysis for basically answering RQ1, RQ2 and RQ3. In this phase, we will start to collect related data. More specifically, We will start to set up a related environment and write a scraper to collect the data based on our research questions. We will learn the necessary skill set on mining data from open source repositories and start to use this skill to collect and filter data. In particular, We will start to look up most popular repositories that not only use GUI testing techniques, in our case, will be selenium-related code, but also use CI/CD framework, such as has Github-Action related yaml file in the repositories. We will use different keywords such as "selenium", "Github-Actions", "Jenkins" and so on to select and filter out our target repositories. After collecting enough data, we will start to categorize and visualize the data.

The third phase activity will be the solution proposed and development, which generally will be trying to answer RQ4. After data analysis in the second phase activity. We will identify the pattern, impacts of the GUI testing in CI/CD practices project, we also will expect to identify different issues that hinder using GUI testing in the projects, such as GUI testing flakiness will

significantly impact CI/CD practice efficiency. Based on the findings, we will work on proposing an applicable and feasible solution to the existing situation as the fourth stage, to address the issue we extracted from the data analysis.

The proposed solution will be evaluated as the final phase. We will also start documenting our results from all previous stages and conclude our findings, discussion and implication. We will design the evaluation based on the nature of our proposed solution. For instance, if we develop a recommended plugin that suggests developers adding any GUI testing code in the CI/CD platform, we will work on evaluating the plugin by collecting developer opinion in different ways, such as surveys or interviews. If we proposed a framework that automatically generate GUI testing corresponding code in CI/CD platform, we will evaluate it on the existing repositories to see if it actually perform well on auto-completed code and indeed reveal any related bugs by automatically adding GUI testing to the CI/CD platform, and if performance of whole CI/CD process gets improve or degrade by using the framework.

6 Gantt Chart



Discussion of Gantt Chart: Similar like the proposed research description, but this Gantt chart one is more specific by dividing second phase and fourth phase into different time period. In general, we are planning finish the preliminary study on year 1, which will include literature review, data collection and analysis for understanding the current situation of GUI testing adaption in CI/CD environment. In the year two we will work on proposing and develop useful tools to improve the situation based on the findings on year 1.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.