Investigating the Impact of Bad Practices in Continuous Integration on Closed-source Projects

Ruben Blenicio Tavares Silva[1], **Carla Ilane Moreira Bezerra**[1]

[1]Campus Quixadá – Universidade Federal do Ceará (UFC)
Quixadá – CE – Brazil

{rubensilva,carlailane}@ufc.br

Abstract. *This research aims to understand how Continuous Integration (CI) bad practices affect the progress of activities in closed-source projects. To guide the research, we sought to answer three key questions: (1) What are the most frequent CI bad practices with the greatest negative impact on closed-source projects?; (2) How does CI assist the code review process in closed-source projects?; and, (3) How are internal quality attributes affected by CI bad practices?. Thus, we present the planning of this research and preliminary results obtained. Our study aims to help organizations address CI bad practices in the context of their software projects.*

Resumo. *Esta pesquisa tem por objetivo entender como as más práticas de Integração Contínua (IC) afetam o andamento das atividades em projetos de código fechado. Para guiar a pesquisa, procuramos responder a três questões-chave: (1) Quais as más práticas de IC mais frequentes e com maior impacto negativo nos projetos de código fechado?; (2) Como a IC auxilia o processo de revisão de código em projetos de código fechado?; e, (3) Como os atributos internos de qualidade de software são afetados pelas más práticas de IC?. Dessa forma, apresentamos neste artigo o planejamento deste pesquisa e resultados preliminares obtidos. O nosso estudo pretende auxiliar as organizações a tratarem o problema das más práticas de IC no contexto dos seus projetos.*

Keywords: Continuous Integration, Bad Practices, Software Quality

Related CBSoft symposia: SBES

Entry year in the Program: 2020

Expected date for final presentation: May 2022

Date of approval master's thesis proposal (qualification): September 2021

1. Introduction

The initial concepts of CI emerged around the 70s, and over time they gained a lot of acceptance both by the academy and the industry [Brooks 1978]. Many organizations today adopt CI concepts in their development processes and, in general, have obtained satisfactory results in their projects, whether open-source or closed-source [Rebouças et al. 2017, Pinto et al. 2018].

Recent studies indicate that CI can bring benefits, such as: (1) faster availability of new features in production [Vasilescu et al. 2015], (2) general increase in the productivity of development teams [Hilton et al. 2016], (3) better efficiency in detecting and correcting problems [Beller et al. 2017], (4) improving team communication [Downs et al. 2010], among others [Elazhary et al. 2021]. These benefits can be achieved in part thanks to a considerable set of tools that have been created to support CI practices within organizations [Felidré et al. 2019].

The CI concept itself basically consists of the frequent integration of source code into a shared code repository so that individual contributions are made available in a production version as soon as possible [Beller et al. 2017]. However, there are a series of procedures that aim to guarantee the code quality and, consequently, the software health. Among these procedures, we can highlight (1) automated tests and (2) code static analysis [Fowler 2006]. Recent studies in CI point to the problem of the existence of CI bad practices that can actually result in damage to the product quality to be developed [Duvall 2018, Zampetti et al. 2020].

In addition, to automated code reviews, another practice employed in conjunction with CI is code review (CR). CR consists of a technique of manual code analysis by the project team before the modifications are integrated into the mainline of development [Rigby and Bird 2013, McIntosh et al. 2014].

Overall, the goal of CI and CR practices is to promote the software quality [McIntosh et al. 2016, Thongtanunam et al. 2016, Beller et al. 2017]. Thus, one of the possible ways to measure the software quality that allows us to perform basic analyzes (only the analysis of quality attributes may not be enough to determine the quality of a system) on the health of a system, are the internal quality attributes, such as: cohesion, coupling, size, complexity and inheritance [Malhotra and Chug 2016, Chávez et al. 2017].

In this context, the proposal of this master's thesis has as main objective to investigate, through empirical studies, the impact of CI bad practices in closed-source software projects. Our work is guided by the following research questions: (1) What are the most frequent CI bad practices with the greatest negative impact on closed-source projects?; (2) How does CI assist the CR process in closed-source projects?; and, (3) How are internal quality attributes affected by CI bad practices?

2. Background and Related Work

In their work [Zampetti et al. 2020] carried out an empirical study where the authors cataloged 79 CI bad practices divided into 7 categories. For this, the authors conducted semi-structured interviews and mined around 2300 posts on Stack Overflow[1]. In the study by

[1] https://stackoverflow.com/

[Felidré et al. 2019] were considered about 1270 projects for analysis of CI bad practices related to: (1) infrequent commits on the main branch, (2) low test coverage, (3) builds broken for a long time, and (4) very long builds. In our study, we also intend to analyze CI bad practices in closed source projects and consider the perceptions of the teams involved in the projects, both developers and project managers.

The authors [Rahman and Roy 2017] conducted a study where they considered the logs of automated builds of open-source projects hosted on GitHub. The authors analyzed the status (success or failure) and frequency of builds and found that integration requests with more builds successfully executed are more likely to receive code reviews. In another work [Zampetti et al. 2019] conducted a qualitative study of pull request discussions in 69 open-source projects on GitHub[2] and found that those with successful builds have a greater chance of being integrated. In our study, we sought to analyze how CI impacts CR and for that, we intend to analyze factors such as time to code integration and correlations between CI and CR metrics.

In the study by [Gallaba and McIntosh 2020] a large-scale investigation was conducted into 9312 open-source projects to detect misuses of CI specifically in the Travis CI environment[3]. Among the results, the authors found that much of the CI code in the projects were related to CI job processing node configurations. Thus, the authors propose a tool to automate the detection of CI anti-patterns in projects. Our study will investigate the impact of CI bad practices on software quality by analyzing internal quality attributes.

3. Research Agenda

The main objective of our work is to analyze the impact of CI bad practices on software projects. Therefore, we chose to apply our research in the context of closed-source projects in the environment of our industrial partners. We've derived our investigation into three main parts that reflect our research questions:

- **RQ1** - What are the most frequent CI bad practices with the greatest negative impact on closed-source projects?
- **RQ2** - How does CI assist the CR process in closed-source projects?
- **RQ3** - How are internal quality attributes affected by CI bad practices?

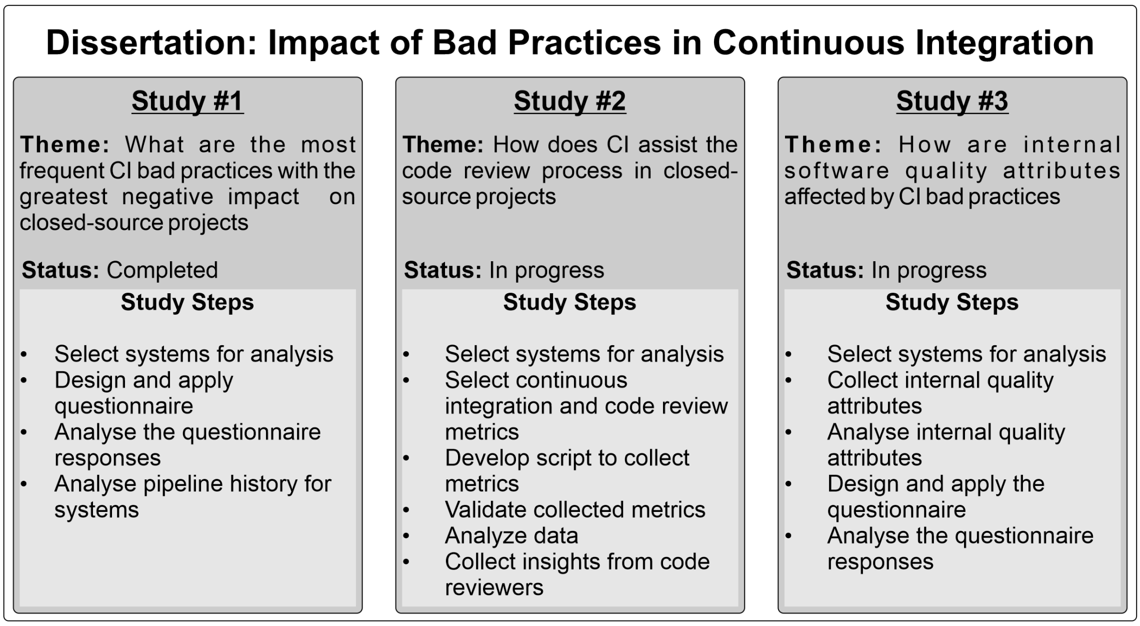
It is noteworthy that although we have three topics to explore, our research is not necessarily sequential. Figure 1 presents the studies that make up the master's thesis. As you can see, we have already completed the first of three studies where we investigated the frequency and harm of CI bad practices and are currently working on the remaining two studies in parallel. Below, we present each of the studies in more detail.

3.1. Study #1

In our first study [Silva and Bezerra 2020], we investigated CI bad practices in closed-source projects to two aspects: frequency of occurrence of bad practices and the level of negative impact caused by them. For this, we designed and applied a questionnaire to developers and project managers who participated in the development of selected systems. In this way, we capture their perceptions about CI bad practices. After obtaining the

[2] https://github.com/

[3] https://travis-ci.com/



Dissertation: Impact of Bad Practices in Continuous Integration

Study #1

Theme: What are the most frequent CI bad practices with the greatest negative impact on closed-source projects

Status: Completed

Study Steps

- Select systems for analysis
- Design and apply questionnaire
- Analyse the questionnaire responses
- Analyse pipeline history for systems

Study #2

Theme: How does CI assist the code review process in closed-source projects

Status: In progress

Study Steps

- Select systems for analysis
- Select continuous integration and code review metrics
- Develop script to collect metrics
- Validate collected metrics
- Analyze data
- Collect insights from code reviewers

Study #3

Theme: How are internal software quality attributes affected by CI bad practices

Status: In progress

Study Steps

- Select systems for analysis
- Collect internal quality attributes
- Analyse internal quality attributes
- Design and apply the questionnaire
- Analyse the questionnaire responses

Figure 1. Study steps

answers, we proceed with the data analysis and identified the most frequent/harmful bad practices. Finally, we analyze the systems' build history for evidence of CI bad practices to corroborate team members' perceptions. The need characterizes the importance of this work we feel to verify (1) whether the bad CI practices presented in the literature occur in the same way in the industrial environment and (2) whether the impacts of these bad practices are (and how much are) perceived in a software development environment.

3.2. Study #2

In our second study, we set out to examine how CR and CI processes relate in practice. This is a work that is still in progress and emerged due to the literature presenting CR as a procedure that aims to reduce the number of bugs that go to the production environment. Furthermore, these two practices have been widely used in several software development organizations, suggesting that these organizations have achieved good results. Therefore, we propose assessing whether there is any correlation between CR and CI and code reviewers' perceptions of (1) the impact of using CI along with CR and (2) whether poor CI practices can harm the CR.

3.3. Study #3

In our latest study, still in progress, we want to investigate how CI bad practices affect the internal attributes of software quality. The motivation for this study is the fact that one of the benefits preached by CI is the improvement in software quality through early detection and correction of errors. As such, we plan to mine the CI repository used by our industrial partners to verify that implementing CI on projects has improved the quality of the projects' code. Furthermore, we collected the perceptions of development team members about the impact of poor CI practices concerning the internal quality attributes of the systems.

4. Preliminary Results

We published an article in which we investigate which bad CI practices are more frequent and which ones cause more damage to software projects[4] [Silva and Bezerra 2020]. This work concludes that the most frequent CI bad practices are related to repository management and development environment culture. Bad practices related to repository management as well as those aimed at quality assurance were rated as most harmful. Furthermore, we found that despite bad practices occurring in the analyzed projects, the team has the perception that there are points to improve and correct to obtain better use of the CI process. As an additional discovery, we found that most CI builds failures are not prioritized and in extreme cases, a build failure can take months to fix.

As mentioned earlier, the other two studies are still ongoing. However, we already have preliminary results based on the steps we have taken so far. In our second work[5], we found some strong correlations (CC is the correlation coefficient) between CI and CR metrics, for example:

1. CI Latency and Review Time $(CC = 0.64)$.
2. CI Jobs successfully executed and Number of reviews $(CC = 0.71)$.
3. Total CI jobs and Number of reviews $(CC = 0.8)$.
4. Total CI builds and Number of reviews $(CC = 0.83)$.

In addition to the correlation between CI and CR metrics, it was possible to discover through the insights of code reviewers that they experienced some benefits of using CI along with CR in their projects, such as:

- More efficient and reliable delivery of software in production.
- Greater security for carrying out the CR.
- Error reduction.
- More flexibility for the review process.
- Better understanding of the changes made.
- Faster identification of improvement points through testing and static analysis.

Already in our third study[6], we found that sometime after CI was implemented in projects, there was an increase in code cohesion across all analyzed systems. Although right after CI deployment, this won't be noticeable. Furthermore, we were able to verify that, according to the perception of the team participants, CI bad practices can lead to an increase in the complexity of the systems.

5. Conclusion

Through the preliminary results obtained, we were able to generate some contributions: (1) we classified CI bad practices as to the frequency with which they occur and to the level of damage caused by them, (2) we found that even with the use of CI they exist errors that are ignored, (3) we establish strong correlations between some aspects of CR

[4] The artifacts of the study #1 are available via the following address: `https://ruben-silva-dev.github.io/PAPER_1/`

[5] The artifacts of the study #2 are available via the following address: `https://ruben-silva-dev.github.io/PAPER_2/`

[6] The artifacts of the study #3 are available via the following address: `https://ruben-silva-dev.github.io/PAPER_3/`

and CI, and (4) we see improvements in the quality level of software projects after CI implementation.

Finally, with the conclusion of this master's thesis, we intend to contribute to the Applied Software Engineering community and the Industry as follows:

- Providing a ranking of CI bad practices in terms of probability of occurrence and degree of negative impact.
- Mapping the main points of influence of CI in CR.
- Identifying the internal quality attributes most affected by CI.
- Calculating an estimated time until CI changes to internal quality attributes are noticeable by development teams after CI deployment.
- Providing industry insights into the overall impact of CI bad practices on projects.

References

Beller, M., Gousios, G., and Zaidman, A. (2017). Oops, my tests broke the build: An explorative analysis of travis ci with github. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. IEEE.

Brooks, F. (1978). The mythical man-month: Essays on softw. 1st.

Chávez, A., Ferreira, I., Fernandes, E., Cedrim, D., and Garcia, A. (2017). How does refactoring affect internal quality attributes? a multi-project study. In *Proceedings of the 31st Brazilian Symposium on Software Engineering*, SBES'17, page 74–83, New York, NY, USA. Association for Computing Machinery.

Downs, J., Hosking, J., and Plimmer, B. (2010). Status communication in agile software teams: A case study. In *2010 Fifth International Conference on Software Engineering Advances*. IEEE.

Duvall, P. (2018). Continuous delivery patterns and antipatterns in the software lifecycle. *WWW], Available (accessed on 25.7. 2021): https://dzone.com/refcardz/continuous-delivery-patterns*.

Elazhary, O., Werner, C., Li, Z. S., Lowlind, D., Ernst, N. A., and Storey, M.-A. (2021). Uncovering the benefits and challenges of continuous integration practices. *IEEE Transactions on Software Engineering*, pages 1–1.

Felidré, W., Furtado, L., da Costa, D. A., Cartaxo, B., and Pinto, G. (2019). Continuous integration theater. In *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*.

Fowler, M. (2006). Continuous integration. *WWW], Available (accessed on 25.7. 2021): https://martinfowler.com/articles/continuousIntegration.html*.

Gallaba, K. and McIntosh, S. (2020). Use and misuse of continuous integration features: An empirical study of projects that (mis)use travis ci. *IEEE Transactions on Software Engineering*, 46(1):33–50.

Hilton, M., Tunnell, T., Huang, K., Marinov, D., and Dig, D. (2016). Usage, costs, and benefits of continuous integration in open-source projects. In *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 426–437.

Malhotra, R. and Chug, A. (2016). An empirical study to assess the effects of refactoring on software maintainability. In *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 110–117.

McIntosh, S., Kamei, Y., Adams, B., and Hassan, A. E. (2014). The impact of code review coverage and code review participation on software quality: A case study of the qt, vtk, and itk projects. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR 2014, page 192–201, New York, NY, USA. Association for Computing Machinery.

McIntosh, S., Kamei, Y., Adams, B., and Hassan, A. E. (2016). An empirical study of the impact of modern code review practices on software quality. *Empirical Software Engineering*, 21(5):2146–2189.

Pinto, G., Castor, F., Bonifacio, R., and Rebouças, M. (2018). Work practices and challenges in continuous integration: A survey with travis ci users. *Software: Practice and Experience*.

Rahman, M. M. and Roy, C. K. (2017). Impact of continuous integration on code reviews. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, pages 499–502.

Rebouças, M., Santos, R. O., Pinto, G., and Castor, F. (2017). How does contributors' involvement influence the build status of an open-source software project? In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*.

Rigby, P. C. and Bird, C. (2013). Convergent contemporary software peer review practices. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2013, page 202–212, New York, NY, USA. Association for Computing Machinery.

Silva, R. B. T. and Bezerra, C. I. M. (2020). Analyzing continuous integration bad practices in closed-source projects: An initial study. In *Proceedings of the 34th Brazilian Symposium on Software Engineering*, SBES '20, page 642–647, New York, NY, USA. Association for Computing Machinery.

Thongtanunam, P., McIntosh, S., Hassan, A. E., and Iida, H. (2016). Revisiting code ownership and its relationship with software quality in the scope of modern code review. In *Proceedings of the 38th International Conference on Software Engineering*, ICSE '16, page 1039–1050, New York, NY, USA. Association for Computing Machinery.

Vasilescu, B., Yu, Y., Wang, H., Devanbu, P., and Filkov, V. (2015). Quality and productivity outcomes relating to continuous integration in github. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*.

Zampetti, F., Bavota, G., Canfora, G., and Penta, M. D. (2019). A study on the interplay between pull request review and continuous integration builds. In *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 38–48.

Zampetti, F., Vassallo, C., Panichella, S., Canfora, G., Gall, H., and Di Penta, M. (2020). An empirical characterization of bad practices in continuous integration. *Empirical Software Engineering*.