Empirical Analysis of the GitHub Actions ecosystem*

Pooya Rostami Mazrae[1][0000−0002−4859−1546]

Software Engineering Lab, University of Mons, Belgium
pooya.rostamimazrae@umons.ac.be
PhD duration: October 2021 – October 2025
Supervisor: Prof. Tom Mens

Abstract. The importance of open-source software (OSS) development has increased significantly throughout the last years, covering almost every application domain. Moreover, the usage of continuous integration and delivery (CI/CD) tools in software development has increased significantly. CI/CD tools aim to automate the build and delivery process with the goal of early error discovery and overall quality improvement. One of the most growing CI/CD tools is GitHub Actions, the integrated solution within GitHub. The aim of my PhD, under the supervision of Prof. Tom Mens is to explore the GitHub Actions ecosystem, by empirically analysing the use, impact, and evolution of GitHub Actions in collaborative OSS development.

Keywords: GitHub Actions · continuous integration · collaborative software development · workflow automation.

1 Research area and context

The importance of open-source software (OSS) development has increased significantly throughout the last years, covering almost every application domain [4]. Today, over 80% of the software in technological products or services is OSS, and this trend is still growing. In addition to this, software ecosystems play an ever-increasing role in collaborative software development practices. A software ecosystem can be defined as "a collection of software projects which are developed and which co-evolve together in the same environment" [3]. As software projects are not usually developed in isolation, it is important to take into account the ecosystem of which they are part to understand the bigger picture.

The popularity of using continuous integration/delivery (CI/CD) tools in software development continues to increase [9]. These tools have been used widely to automate predefined tasks [13]. Introducing these tools in projects help teams to decrease repetitive tasks and increase the productivity and quality of the software development progress. It is only natural that the introduction and extensive

* This work is supported by the ARC-21/25 UMONS3 Action de Recherche Concertée financée par le Ministère de la Communauté française – Direction générale de l'Enseignement non obligatoire et de la Recherche scientifique.

use of any automated tool will have important positive and negative effects on the projects they have been used in.

GitHub Actions is a CI/CD platform that allows developers to automate their build, test, and deployment pipeline. Developers can create workflow which is an automated process that is made up of one or multiple jobs and can be triggered by an event. Jobs are made up of multiple steps and run in an instance of the virtual environment and steps are a set of tasks that can be executed by a job. Steps can run commands or Actions. Information about the Actions can be found in the *.github/workflow* directory which contains at least one YAML file to configure the automated tasks.

The ecosystem of GitHub Actions is built on a series of interconnected Actions. This ecosystem is consist of reusable artifacts that can be used in different workflows. Actions in workflows can depend on each other since workflows can depend on each other. Moreover, each of the Actions used in the workflows has versioning and the problem with versions can affect the workflows that Actions have been used in and other workflows which are dependent on this workflow. Furthermore, The study by Golzadeh et al [8] shows that more than 50K from 91,810 repositories are using GitHub Actions as one of their CI/CD tools which indicates the importance of this ecosystem.

2 Research goal

CI/CD tools are being used in different parts of the software development workflow to increase both productivity and quality of software developments [12]. CI/CD tools aim at automating the build process on dedicated servers, with the goal of early error discovery as well as of an overall quality assessment and improvement. For such reasons, CI/CD is quite frequently adopted in industry and open-source [15]. For example, a recent study revealed that within the npm package management ecosystem for the JavaScript programming language, there are more than 91K active project repositories that have used at least one CI/CD tool for automation [8].

One of the fastest-growing CI/CD tools is *GitHub Actions*. A recent study by Golzadeh et al. [8] shows that more than 50% of 91,810 repositories in GitHub use GitHub Actions as their CI/CD tool. Furthermore, [8] indicates that in the time window of 18 months since its first appearance, GitHub Actions has become the dominant CI/CD tool on GitHub. Also, GitHub Actions reuses by developers shows that it has become a new ecosystem. This ecosystem let the developers reuse the actions and integrate a variety of simple to complex tasks without having the needed code for it. Like software components in any other software ecosystem, Actions can depend on each other, have different versions, and even their input variable is changed by the output of other Actions.

Other similar kinds of CI/CD ecosystems already existed even before the emergence of GitHub Actions. For example, CircleCI is another CI/CD tool that has been around since 2011, and it introduced reusable *orbs* a year before the creation of GitHub Actions. Jenkins is another popular CI/CD tool that

has had a large ecosystem of plugins [1] since four years ago. Just like any other software ecosystem, these CI/CD ecosystems can suffer from issues like technical lag [16], dependency issues [5], etc.

An important requirement for conducting empirical studies in this domain is having access to a data source containing recent, reliable, and sufficiently complete information. Due to this reason, I will concentrate on the GitHub Actions ecosystem which has a sufficient amount of data and is accessible through GitHub APIs and other third-party solutions like GHTorrent. Furthermore, the GitHub Actions ecosystem has been growing faster than many similar ecosystems. For instance, in April 2021 it contained more than 12K reusable Actions on its marketplace, about 4 times more than CircleCI orbs and 6 times more than Jenkins plugins.

Lastly, the high usage frequency of Actions in software development and potential interconnection between different Actions in workflows increase the importance of studying them. It should be borne in mind that study about the Actions ecosystem plays an important role since they have been used frequently in many projects and their problems can have a deep impact on all those projects.

My PhD thesis goal is to empirically analyze the impact of GitHub Actions on the open-source software development projects using statistical techniques, providing better insight into the current state of the ecosystem and its usages.

The empirical analysis will help answer a series of questions that provides a better understanding of the GitHub Actions ecosystem. These questions are:

- RQ1: Why are software developers integrating GitHub Actions into their projects?
- RQ2: How are software developers reusing GitHub Actions in their projects?
- RQ3: How does the usage of GitHub Actions evolve over time?
- RQ4: What are the short-term and long-term impacts of adding GitHub Actions to OSS from a statistical analysis point of view on the issue, commits, and pull requests?
- RQ5: What are the dependency network characteristics of GitHub Actions?

3 Related work

For my ongoing work on RQ1 and RQ2, I needed some knowledge about the related work and state-of-the-art. The root of CI/CD goes back to agile software development [1]. One of the earliest studies which lead to increased knowledge about how to use CI/CD tools is [7]. Through the years, there have been studies that aim to better understand the impact of CI/CD tools [12,10,6,8]. Some works concentrated on one specific CI/CD tool to study their changes during some periods, mostly Travis because of its prior dominance as one of the most popular CI/CD tools to integrate with GitHub [2,14]. A recent systematic literature review has surveyed 101 empirical studies that evaluate CI in the context of software development [11].

[1] https://plugins.jenkins.io/

4 Results to date

As the first step in my PhD research, I started conducting a qualitative study about CI/CD tool usage, through online interviews with software practitioners. The main purpose of this study is to analyze and answer RQ1. Working on this research question and writing a paper about it as a lead writer has given me a chance to increase my knowledge about the CI/CD tools which is a type of automation tool in software development by directly writing our paper and also studying other related works in this area. For this qualitative study, 23 software practitioners have been interviewed. As a result of this qualitative analysis, we observed that developers are gradually moving to use GitHub Actions in their software projects for a diverse series of reasons, including *better integration to GitHub*, *better support of different platforms*, and *more reliability*. This paper is currently in the writing phase. I am planning to finish this work by mid-July and submit it to a top software engineering journal.

As the second step in my PhD research, I am currently focusing on RQ2. Together with my team members, I have co-authored a conference submission (currently under review) about CI/CD usage which specifically concentrates on the use of GitHub Actions in software development repositories. In this work, we studied 67,870 highly starred GitHub repositories using GitHub Actions, and we quantified the most frequent jobs used in workflows, the most reused actions and how they have been reused, and the categories to which these actions belong.

5 Research methodology

For needed information, I rely on different data sources. I will use the GitHub API or third-party tools to select the repositories based on certain criteria (e.g., time of last activity, creation date, language).

This step would be paired with developing automated scripts to extract the needed software artifacts like issues, commits, and Actions which can be found in *.github/workflows* directory, recommended directory for saving all the workflows, in GitHub repositories. Issues and commits are needed to study the short-term and long-term effect of adding GitHub Actions since every change manifest itself in them. It is worth keeping in mind that for some repositories, it is needed to gather the issues from other issue tracking systems like *Jira*.

6 Research challenges

There are some research challenges in this area of study which needed to be addressed. Most of these challenges are data-driven. Below I only mention a few of them that I have already been able to identify.

The first challenge is related to data gathering. Data about GitHub Actions is not readily available. Right now, there is no simple, straightforward way to gather the information related to GitHub Actions. To gather such data one needs to crawl the GitHub Marketplace and extract the related information about them.

Moreover, the GitHub REST API and GraphQL API will be used to gather information related to data like issues and commits of the repositories. Creating a different snapshot of Actions based on the date is necessary for further analysis of their evolution.

The second challenge is related to the fact that Actions can be created in any repository. It is possible to find Actions whose creation date differs from their repository date. This is possible since people can create Actions in repositories that they have already created before. This is just one example of possible data inconsistency in this ecosystem which can create threats to validity in studying it.

The third challenge relates to the fact that not all Actions can be found in the GitHub Marketplace since developers can choose to have their Actions without publishing them on the marketplace.

7 Future work

To fully cover my studies related to RQ1 and RQ2 and other state-of-the-art works in this matter, I will be involved in writing a chapter of a book about the GitHub development workflow automation ecosystem. I aim to show the impact of introducing GitHub Actions as a new workflow automation tool in GitHub. To fully cover this topic, I will cover the latest works on the reasons why developers are moving from other CI/CD tools to GitHub Actions and also how they are using Actions in their projects.

Next, I aim to have a study on the evolution of GitHub Actions usage to answer the RQ3. To do this, I will study the monthly snapshot of GitHub workflows to study how they have changed through time (e.g. adding and removing Actions, version changes of existing Actions, etc.). I'm planning to submit this work for MSR 2023 which will be January 2023.

Later, to answer the RQ4, I am planning to statically analyse the short-term and long-term impact of adding GitHub Actions to the repositories. The short-term and long-term changes related to adding GitHub Actions can be found by studying the change in the pattern of issues, commits, and pull requests (e.g. frequency change in opening or closing them, decrease/increase in time windows of opening an issue to closing it, etc.). The aim is to study the changes that occur due to adding GitHub Actions first during adding the CI/CD to the repository and then after using the CI/CD for some period of time to fully cover the short-term and long-term impact. The difference in behavior changes between projects whose already had other CI/CDs before and those adding it to their projects for the first is also interesting for me.

References

1. Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., et al.: Manifesto for agile software development (2001)
2. Beller, M., Gousios, G., Zaidman, A.: Oops, my tests broke the build: An explorative analysis of Travis CI with GitHub. In: International Conference on Mining Software Repositories (MSR). pp. 356–367 (2017). https://doi.org/10.1109/MSR.2017.62
3. Blincoe, K., Harrison, F., Damian, D.: Ecosystems in GitHub and a method for ecosystem identification using reference coupling. In: Working Conference on Mining Software Repositories. pp. 202–211. IEEE (2015). https://doi.org/10.1109/MSR.2015.26
4. Coelho, J., Valente, M.T., Milen, L., Silva, L.L.: Is this GitHub project maintained? Measuring the level of maintenance activity of open-source projects. Information and Software Technology **122** (2020). https://doi.org/10.1016/j.infsof.2020.106274
5. Decan, A., Mens, T., Claes, M.: An empirical comparison of dependency issues in oss packaging ecosystems. In: 2017 IEEE 24th international conference on software analysis, evolution and reengineering (SANER). pp. 2–12. IEEE (2017)
6. Elazhary, O., Werner, C., Li, Z.S., Lowlind, D., Ernst, N.A., Storey, M.A.: Uncovering the benefits and challenges of continuous integration practices. IEEE Transactions on Software Engineering (2021)
7. Fowler, M., Foemmel, M.: Continuous integration (2006)
8. Golzadeh, M., Decan, A., Mens, T.: On the rise and fall of CI services in GitHub (2022)
9. Kavaler, D., Trockman, A., Vasilescu, B., Filkov, V.: Tool choice matters: JavaScript quality assurance tools and usage outcomes in GitHub projects. In: International Conference on Software Engineering (ICSE). pp. 476–487. IEEE (2019)
10. Savor, T., Douglas, M., Gentili, M., Williams, L., Beck, K., Stumm, M.: Continuous deployment at facebook and oanda. In: International Conference on Software Engineering. pp. 21–30. IEEE (2016)
11. Soares, E., Sizilio, G., Santos, J., Alencar, D., Kulesza, U.: The effects of continuous integration on software development: a systematic literature review. arXiv preprint arXiv:2103.05451 (2021)
12. Vasilescu, B., Yu, Y., Wang, H., Devanbu, P., Filkov, V.: Quality and productivity outcomes relating to continuous integration in GitHub. In: Joint Meeting on Foundations of Software Engineering (FSE). pp. 805–816 (2015)
13. Wessel, M., De Souza, B.M., Steinmacher, I., Wiese, I.S., Polato, I., Chaves, A.P., Gerosa, M.A.: The power of bots: Characterizing and understanding bots in oss projects. International Conference on Human-Computer Interaction (CSCW) **2**, 1–19 (2018)
14. Widder, D., Vasilescu, B., Hilton, M., Kästner, C.: I'm leaving you, travis: a continuous integration breakup story. In: 2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR). pp. 165–169. IEEE (2018)
15. Zampetti, F., Geremia, S., Bavota, G., Di Penta, M.: Ci/cd pipelines evolution and restructuring: A qualitative and quantitative study. In: International Conference on Software Maintenance and Evolution (ICSME). pp. 471–482. IEEE (2021)
16. Zerouali, A., Mens, T., Gonzalez-Barahona, J., Decan, A., Constantinou, E., Robles, G.: A formal framework for measuring technical lag in component repositories—and its application to npm. Journal of Software: Evolution and Process **31**(8), e2157 (2019)