A dataset of GitHub Actions workflow histories

Guillaume Cardoen
Guillaume.CARDOEN@umons.ac.be
University of Mons
Belgium, Mons

Tom Mens
Tom.MENS@umons.ac.be
University of Mons
Belgium, Mons

Alexandre Decan*
Alexandre.DECAN@umons.ac.be
University of Mons
Belgium, Mons

ABSTRACT

GitHub Actions is the *de facto* workflow automation tool for GitHub repositories. Its popularity has increased dramatically over the recent years, opening up opportunities for empirical studies related to its usage. To enable such studies, we implemented *gigawork*, an open source tool for extracting the commit histories of changes to workflow files in GitHub repositories. Using this tool we collected and publicly released a dataset of 160K+ commit histories of workflow files in 32K+ public GitHub repositories, covering 1.5M+ workflow file versions. In order to facilitate its use by other researchers, the dataset includes relevant metadata related to workflow file changes in each commit. gigawork is publicly released on PyPi. Its associated dataset can be found on Zenodo (DOI: 10.5281/zenodo.10259013).

ACM Reference Format:
Guillaume Cardoen, Tom Mens, and Alexandre Decan. 2024. A dataset of GitHub Actions workflow histories. In *21st International Conference on Mining Software Repositories (MSR '24), April 15–16, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3643991.3644867

1 INTRODUCTION

CI/CD tools automate many repetitive activities that reduce workload and increase quality during collaborative software development. Until 2020, Travis used to be the most prevalent CI/CD service on GitHub [10], resulting in many empirical studies [1, 6, 17, 18, 20–22], partly thanks to datasets like TravisTorrent [2]. In November 2019, GitHub integrated GitHub Actions as a new workflow automation service. Only 18 months after its introduction, it overtook Travis and its competitors for a diverse set of reasons [15].

GitHub repository maintainers that want to automate activities with GitHub Actions can define *workflows* in the .github/workflows directory. Such workflows must respect the YAML-based syntax described in GitHub's online documentation.[1] To facilitate the development and use of workflows, one can rely on reusable workflow components called *Actions*, most of which are publicly available on the GitHub Marketplace.[2] Because empirical research on GitHub Actions workflows is on the rise [3, 8, 13, 16, 19], there is a need for historical datasets related to these workflows, allowing researchers to conduct large-scale empirical studies. Such studies require a large quantity of workflow files from a variety of repositories.

Previous research on GitHub Actions already used some limited workflow file datasets. In a preliminary analysis on the usage of GitHub Actions workflows in early 2021, Kinsman et al. [13] assembled a dataset of workflow files and their history collected from 3,190 GitHub repositories. Decan et al. [9] created a dataset of nearly one million workflow files obtained from monthly snapshots of 22K+ repositories to study the outdatedness of the GitHub Actions ecosystem. In a more detailed analysis of GitHub Actions workflows, Decan et al. [8] built and released a dataset of monthly snapshots of workflow files until January 2022, coming from nearly 30K repositories. The snapshot-based nature of these datasets does not allow to obtain the full workflow history of individual repositories. Recent research using one of these datasets [5, 11] revealed the need for a more complete, maintainable and extendable dataset of GitHub Actions workflow files and their history.

Hence, this paper presents a new dataset of GitHub Actions workflow files, that is more recent (October 2023), covers a larger amount of repositories (32,882 to be precise), and contains their full workflow histories and other relevant metadata about workflow changes. We also present an open-source tool allowing to update and further expand this dataset.

Section 2 presents *gigawork*, a tool for extracting GitHub Actions workflow histories from GitHub repositories. Section 3 presents the dataset that has been obtained by means of this tool. Section 4 illustrates through an exploratory empirical analysis how the dataset can be used. Section 5 discusses some limitations of the dataset, and Section 6 concludes.

2 WORKFLOW EXTRACTION TOOL

To enable the extraction of workflows from GitHub repositories, we developed *gigawork*, which is an acronym for "**Gi**ve me **G**itHub **A**ctions **Work**flows". *gigawork* comes as an open source tool, with a command-line interface, hosted on GitHub.[3] It has been implemented in Python 3 and relies on the open source module gitpython.[4]

gigawork is distributed through PyPI and can be installed via pip (using `pip install gigawork`). After installation, one can extract the workflows of a repository stored at `repoPath` by issuing the command `gigawork repoPath`.

gigawork iterates through all ancestor commits of a given git reference, git HEAD by default, following the first-parent rule.[5] Whenever *gigawork* finds such commits, it extracts the workflow files that were touched, and saves them in the `workflows` directory of the current working directory, unless asked otherwise. Metadata

*F.R.S.-FNRS Research Associate

[1] https://docs.github.com/en/actions/using-workflows/workflow-syntax-for-github-actions

[2] https://github.com/marketplace

[3] https://github.com/cardoeng/gigawork

[4] https://github.com/gitpython-developers/GitPython

[5] This rule dictates that, when traversing the git history, git only follows the first parent of each commit. In case of merge commit, it thus only follows one branch.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MSR '24, April 15–16, 2024, Lisbon, Portugal
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0587-8/24/04…$15.00
https://doi.org/10.1145/3643991.3644867

about the commits and extracted files are, by default, sent to the standard output. The use of *gigawork* can be customised through optional arguments. Below we provide an example:

```
gigawork https://github.com/cardoeng/gigawork
  --save-repository repositoryName --output file.csv
  --workflows extractedWorkflows --after v1.0.0
```

`--save-repository` specifies the name under which a remote repository will be saved on disk. If not provided for a remote repository, a temporary directory will be created for fetching it and will be deleted after the extraction.
`--update` allows to specify that a (local) repository will be updated prior to analysing the repository.
`--output` allows the user to redirect the metadata output to a given CSV file. By default, the standard output is used.
`--workflows` allows to specify the directory where all extracted files will be stored. By default, a directory `workflows` will be created and used.
`--branch` allows to specify the git reference starting from which *gigawork* will extract ancestor commits. This reference can notably be used to specify the branch in which commits will be considered for extraction. By default, this value equals to the git `HEAD`.
`--after` specifies the git reference at which *gigawork* will stop the extraction. By default, *gigawork* stops once it reaches a commit with no ancestors. `--after` can be combined with `--branch` to specify the interval of commits that should be considered. They can be used to explore parallel branches or to extend and update an already existing dataset.
Other arguments exist and are described in the help documentation of the tool.

3 DATASET OF GITHUB ACTIONS WORKFLOWS

Using *gigawork* (version 1.2.0), we extracted 1,601,709 files stored in the `.github/workflows` directory of 32,886 GitHub repositories. The extracted files and related metadata are publicly available on https://zenodo.org/doi/10.5281/zenodo.10259013. Next to workflow files, the `.github/workflows` may also include auxiliary files. Because such files could be relevant for certain types of empirical analyses, we decided to include them as well. We consider a file to be a workflow if the following conditions are met: (1) the file extension is either yaml or yml; (2) the file is not contained in some subdirectory of `.github/workflows`. Following the aforementioned heuristic, the extracted files were subdivided in 1,526,475 workflow files and 75,234 auxiliary files.

To create the dataset, we used the SEART search engine [7] on 11 October 2023 to obtain an initial list of candidate GitHub repositories, by applying the following filters: (1) at least 300 commits; (2) at least 100 stars; (3) the repository is not a fork; (4) repository creation date before January 1st, 2023; (5) at least one commit after January 1st, 2023. These criteria aimed to exclude personal and experimental repositories [12], focusing on sufficiently popular and recently active repositories, assuming that such repositories are more likely to rely on automated workflows.

46,281 GitHub repositories matching these criteria were obtained. For all these repositories, we checked whether they contained a `.github/workflows` directory, which is a necessary condition to use GitHub Actions workflows. Only 33,209 repositories (71.8% of the candidate list) matched this condition. We cloned each of these repositories between 11th and 13h of October 2023. 168 repositories could not be cloned because of errors during the cloning process. We then applied *gigawork* on the default branch[6] of the remaining 33,041 GitHub repositories. 159 of these repositories did not contain any files within their `.github/workflows` directory and were therefore excluded, resulting in a final dataset of 32,886 repositories. While 4 repositories did not contain any workflow file, they are still included in the dataset because they do contain auxiliary files.

Table 1 summarises the characteristics of the extracted data. 1,004,202 of the commits touch at least one of the 1,526,475 extracted workflow files. These commits include a total of 1,342,662 workflow file modifications, 147,978 additions and 35,835 deletions. The earliest commit date affecting a workflow file is July 11th, 2019. The last commit for each repository contains 3.46 workflow files on average (with a median of 2 and a maximum of 178).

Table 1: Characteristics of the dataset of workflow files

characteristic	value
# GitHub repositories	32,886
# repositories containing workflow files	32,882
# commits touching workflow files	1,004,202
earliest commit date containing a workflow	2019-07-11
latest commit date containing a workflow	2023-10-12
# workflow histories	160,443
# workflow files	1,526,475
# modifications	1,342,662
# additions	147,978
# deletions	35,835
# auxiliary files	75,234

Based on the data extracted using *gigawork*, we created a dataset composed of extracted workflow and their metadata. The dataset content is described at Table 2.

Table 2: Structure of the dataset

file	description
`workflows.tar.gz`	Compressed archive containing the extracted files found in the `.github/workflows` directory of each repository (cf. Table 1).
`workflows.csv.gz`	Compressed CSV file containing, for each repository, relevant metadata for each extracted workflow file, such as the name, email and date of the author and committer for each commit (cf. Table 3).
`auxiliaries.csv.gz`	A similar compressed CSV file for the extracted auxiliary files.
`repositories.csv.gz`	Compressed CSV file provided by SEART [7] containing metadata of every considered repository such as the number of stars, contributors, main language, …

Table 4 provides an excerpt of the output generated by *gigawork* on a GitHub repository that uses GitHub Actions. The value of the change_type field in the metadata can be either A, D or M representing respectively an Added, Deleted or Modified file. These letters are the ones used by git when detecting the change type. If a file is renamed, it will appear as Modified.

To avoid storing identical file contents multiple times, a SHA-256 value is computed for each workflow file, and files with the

[6] As reported by SEART.

Table 3: Metadata stored for each file change.

column	description
repository	The repository from which the workflow was extracted
commit_hash	The commit hash returned by git
author_name	The name of the author that changed this file
author_email	The email of the author that changed this file
committer_name	The name of the committer
committer_email	The email of the committer
committed_date	The committed date of the commit
authored_date	The authored date of the commit
file_path	The path to this file
previous_file_path	The path to this file before it has been touched
file_hash	The name of the related workflow file in the dataset
previous_file_hash	The name of the related workflow file in the dataset, before it has been touched
change_type	A single letter (A,D or M) representing the type of change made to the workflow (**A**dded, **D**eleted or **M**odified)

same hash are stored only once. In this way, we avoided storing 156,933 identical workflow files, representing 10.36% of the complete dataset.

4 EXPLORATORY EMPIRICAL ANALYSIS

The presented dataset can be used by researchers as a starting point for empirical studies on GitHub Actions workflows, such as [8, 9, 13]. To illustrate how to use our dataset for such empirical research, we conducted an exploratory analysis. We focused on two of Lehman's laws of software evolution [14]: *increasing growth* and *continuing change*, hypothesising that both laws should apply for workflows, given that these are code-related software artefacts.

Increasing growth. Figure 1 shows the evolution of the number of workflows and repositories in the dataset, as well as the evolution of workflow-to-repository ratio. One can observe an increase over time in the number of workflows being used by GitHub repositories. This increase is likely due to an overall increase in the number of GitHub repositories over time combined with an increase in GitHub Actions usage by GitHub repositories [10]. The workflow-to-repository-ratio shows an increasing growth, confirming that workflows are indeed becoming increasingly used by repository maintainers. A linear regression analysis (dashed line in Figure 1) strongly suggests a linear growth trend (coefficient of determination is $R^2 = 0.995$). One can observe a variation in the growth trend of workflows near the end of 2020. It is difficult to pinpoint the exact cause of this variation, but it coincides with reported evidence of many public repositories switching from Travis to GitHub Actions around this time, a consequence of the restrictions imposed by Travis on its free plan [10].

A complementary view on *increasing growth* pertains to how the size of workflows evolves over time. Figure 2 shows the evolution of the median size of workflow files measured as: (1) the number of non-empty lines in the workflow file; (2) the number of steps contained in the workflow file. Steps are the main building blocks of workflows, allowing to run specific tasks, either by relying on a reusable Action, or by running commands directly in the runner environment.

We found a strong correlation between both size measures (Pearson correlation coefficient of 0.91). The median number of steps seems to be quite stable over time (between 5 and 6 steps). The

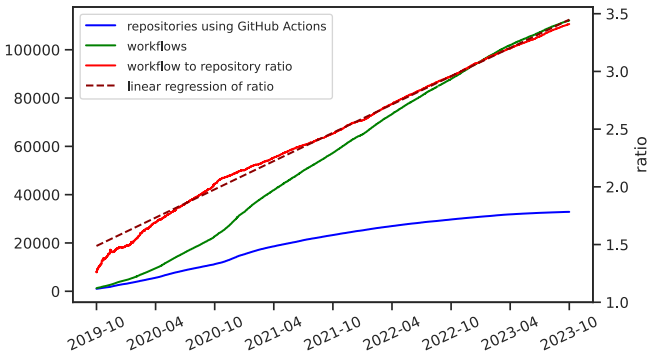

Figure 1: Evolution of the number (left axis) and ratio (right axis) of workflows and repositories.

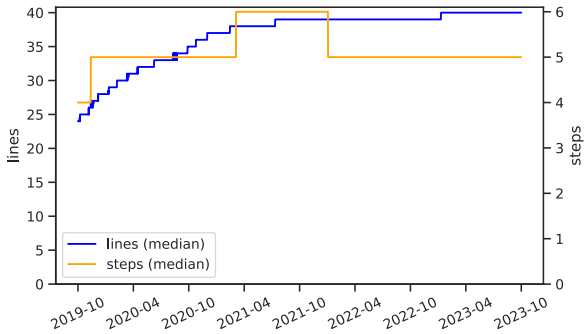

Figure 2: Evolution of the median number of lines (left y-axis) and steps (right y-axis) of workflow files.

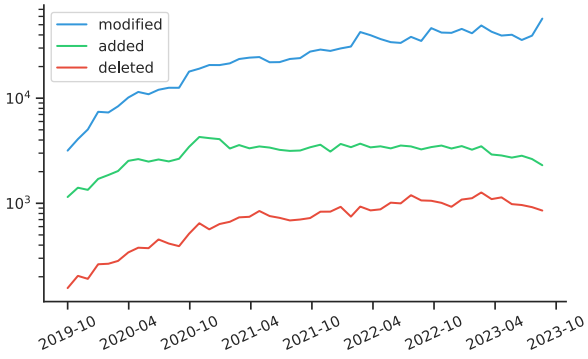

Figure 3: Number of workflow files (on log-scale) touched by commits on a monthly basis (based on the commit date), grouped by change type (added, deleted or modified).

median number of lines increased until August 2021, after which it stabilises at 39-40 lines. We also computed the median number of *jobs* over time, but this value remained equal to 1 during the entire observation period, signifying that the majority of workflows only specify a single *job* (containing multiple *steps*).

Continuing change. Figure 3 shows the monthly evolution of the number of workflow files that are added, deleted or modified in the dataset. If a same workflow file was modified multiple times in

Table 4: Example of a real CSV output file accompanying the extracted workflows (ungc* = users.noreply.github.com)

commit	author name	author email	committer name	committer email	committed date	authored date	file_path	file_hash	change type
82c0ae	Tom Mens	tom.mens@umons.ac.be	Tom Mens	tom.mens@umons.ac.be	1671725010	1671708385	coverage.yml	01a7cb	D
82c0ae	Tom Mens	tom.mens@umons.ac.be	Tom Mens	tom.mens@umons.ac.be	1671725010	1671708385	maven.yml	e9d48c	M
6c97b6	Tom Mens	tommens@ungc*	GitHub	noreply@github.com	1611568580	1611568580	coverage.yml	01a7cb	A
156661	Tom Mens	tom.mens@umons.ac.be	Tom Mens	tom.mens@umons.ac.be	1610144869	1610144869	codecov.yml	f5b2c2	D
4ed007	Tom Mens	tommens@ungc*	GitHub	noreply@github.com	1610141430	1610141430	codecov.yml	f5b2c2	A
a68b53	Tom Mens	tom.mens@umons.ac.be	Tom Mens	tom.mens@umons.ac.be	1610128723	1610128723	maven.yml	2bb6d2	A

the same month, it is only counted once. The monthly number of added workflows continues to increase until between November 2020 and January 2021 where it reaches its highest value: 4,277, 4,169 and 4,077 workflows, respectively for November, December and January. This again coincides with the restrictions imposed on Travis' free plan for public repositories [10], that might have lead repositories to switch to GitHub Actions instead.[7] From February 2021 until March 2023, the monthly number of workflow file additions remains more or less stable, fluctuating between 3,104 and 3,693. After March 2023, the monthly number of additions decreases again, down to 2,302 in September 2023. The monthly number of deletions gradually increases up to 1,268 in March 2023, followed by a decrease down to 853 in September 2023.

Since 1 November 2019 the monthly number of modifications increased from 3,170 to 57,207 in September 2023. The monthly number of modifications thus tends to increase with time, which is expected since all of the workflows that are being added over time need to be maintained. This increase over time and the number of modifications confirms that GitHub Actions workflows follow the law of continuing change

We can therefore conclude that we found empirical evidence in favour of the laws of increasing growth and continuing change of GitHub Actions workflow files.

5 LIMITATIONS

The dataset of GitHub Actions workflow histories is subject to some limitations. First, as the extraction of our dataset is based on git, we are subject to the intrinsic limitations of mining git histories [4].

Second, we only considered repositories given by the SEART GitHub search tool [7], limiting the scope to GitHub repositories with at least 300 commits and 100 stars. This allowed us to exclude abandoned, personal or experimental repositories, but may have excluded smaller or less active projects relying on GitHub Actions.

Third, before cloning the repositories, we checked whether they contained a `.github/workflows` directory. However, this implies that we potentially excluded repositories that were making use of GitHub Actions workflows but stopped using it and deleted this directory.

Finally, we only considered commits occurring on the default branch following the first-parent rule. As such, commits occurring in parallel branches were not considered. On the other hand, the changes made to workflow files in these commits are still visible in the dataset when they are eventually merged back to the default branch. It is worth mentioning that *gigawork* can be applied on these parallel branches, and thus, overcomes this limitation if necessary.

[7]We could not verify this assumption since our dataset does not provide any information about Travis usage.

6 CONCLUSION

GitHub Actions is a widely used CI/CD service [15]. Its popularity leads to many empirical researches [3, 8, 13, 16, 19], often requiring large datasets of GitHub Actions workflow files and their history. We developed *gigawork*, an open-source command-line tool for extracting (the history of) GitHub Actions workflows and their metadata. We applied this tool on a large set of repositories, and released a publicly accessible dataset of the evolution of workflow files used in GitHub repositories.

The dataset contains 160K+ workflow histories (totalling 1,5M+ distinct workflow files) obtained from 32K+ public GitHub repositories and reflecting the evolution of workflow usage since the introduction of GitHub Actions in November 2019. Such a dataset is valuable for researchers as it enables future in-depth empirical research on GitHub Actions workflows without requiring to perform the time-consuming and error-prone task of collecting large collections of workflow histories. We illustrated the usability of our dataset by providing empirical evidence that workflow files follow Lehman's laws of increasing growth and continuing change.

ACKNOWLEDGMENTS

This work is supported by the Fonds de la Recherche Scientifique - FNRS under grant numbers T.0149.22, F.4515.23 and J.0147.24.

REFERENCES

[1] Moritz Beller, Georgios Gousios, and Andy Zaidman. 2017. Oops, My Tests Broke the Build: An Explorative Analysis of Travis CI with GitHub. In *International Conference on Mining Software Repositories (MSR)*. 356–367. https://doi.org/10.1109/MSR.2017.62

[2] Moritz Beller, Georgios Gousios, and Andy Zaidman. 2017. TravisTorrent: Synthesizing Travis CI and GitHub for full-stack research on continuous integration. In *International Conference on Mining Software Repositories (MSR)*. IEEE, 447–450.

[3] Giacomo Benedetti, Luca Verderame, and Alessio Merlo. 2022. Automatic Security Assessment of GitHub Actions Workflows. In *Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses*. ACM, 37–45. https://doi.org/10.1145/3560835.3564554

[4] Christian Bird, Peter C. Rigby, Earl T. Barr, David Hamilton, Daniel M. German, and Prem Devanbu. 2009. The Promises and Perils of Mining Git. In *Working Conference on Mining Software Repositories (MSR)*. IEEE Computer Society. https://doi.org/10.1109/MSR.2009.5069475

[5] Islem Bouzenia and Michael Pradel. 2024. Resource Usage and Optimization Opportunities in Workflows of GitHub Actions. In *International Conference on Software Engineering (ICSE)*. IEEE Computer Society, 268–279.

[6] Nathan Cassee, Bogdan Vasilescu, and Alexander Serebrenik. 2020. The Silent Helper: The Impact of Continuous Integration on Code Reviews. In *International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 423–434. https://doi.org/10.1109/SANER48275.2020.9054818

[7] Ozren Dabic, Emad Aghajani, and Gabriele Bavota. 2021. Sampling Projects in GitHub for MSR Studies. In *International Conference on Mining Software Repositories (MSR)*. IEEE, 560–564.

[8] Alexandre Decan, Tom Mens, Pooya Rostami Mazrae, and Mehdi Golzadeh. 2022. On the Use of GitHub Actions in Software Development Repositories. In *International Conference on Software Maintenance and Evolution (ICSME)*. IEEE. https://doi.org/10.1109/ICSME55016.2022.00029

[9] Alexandre Decan, Tom Mens, and Hassan Onsori Delicheh. 2023. On the outdatedness of workflows in the GitHub Actions ecosystem. *Journal of Systems and Software* 206 (2023). https://doi.org/10.1016/j.jss.2023.111827

[10] Mehdi Golzadeh, Alexandre Decan, and Tom Mens. 2022. On the rise and fall of CI services in GitHub. In *International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE. https://doi.org/10.1109/SANER53432.2022.00084

[11] Jiangnan Huang and Bin Lin. 2023. CIGAR: Contrastive Learning for GitHub Action Recommendation. In *International Working Conference on Source Code Analysis and Manipulation (SCAM)*. IEEE, 61–71.

[12] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. German, and Daniela Damian. 2014. The Promises and Perils of Mining GitHub. In *International Conference on Mining Software Repositories (MSR)*. ACM, 92–101. https://doi.org/10.1145/2597073.2597074

[13] Timothy Kinsman, Mairieli Wessel, Marco A Gerosa, and Christoph Treude. 2021. How do software developers use GitHub Actions to automate their workflows?. In *International Conference on Mining Software Repositories (MSR)*. IEEE, 420–431.

[14] M.M. Lehman, J.F. Ramil, P.D. Wernick, D.E. Perry, and W.M. Turski. 1997. Metrics and laws of software evolution - the nineties view. In *Fourth International Software Metrics Symposium*. 20–32. https://doi.org/10.1109/METRIC.1997.637156

[15] Pooya Rostami Mazrae, Tom Mens, Mehdi Golzadeh, and Alexandre Decan. 2023. On the usage, co-usage and migration of CI/CD tools: A qualitative analysis. *Empirical Software Engineering* 28, 2 (2023), 52. https://doi.org/10.1007/s10664-022-10285-5

[16] Pablo Valenzuela-Toledo and Alexandre Bergel. 2022. Evolution of GitHub Action Workflows. In *International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE.

[17] Bogdan Vasilescu, Stef van Schuylenburg, Jules Wulms, Alexander Serebrenik, and Mark G.J. van den Brand. 2014. Continuous Integration in a Social-Coding World: Empirical Evidence from GitHub. In *International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 401–405. https://doi.org/10.1109/ICSME.2014.62

[18] Bogdan Vasilescu, Yue Yu, Huaimin Wang, Premkumar Devanbu, and Vladimir Filkov. 2015. Quality and productivity outcomes relating to continuous integration in GitHub. In *Joint Meeting on Foundations of Software Engineering (FSE)*. 805–816.

[19] Mairieli Wessel, Tom Mens, Alexandre Decan, and Pooya Rostami Mazrae. 2023. The GitHub Development Workflow Automation Ecosystems. In *Software Ecosystems: Tooling and Analytics*. Springer.

[20] Yue Yu, Huaimin Wang, Vladimir Filkov, Premkumar Devanbu, and Bogdan Vasilescu. 2015. Wait for It: Determinants of Pull Request Evaluation Latency on GitHub. In *Working Conference on Mining Software Repositories (MSR)*. 367–371. https://doi.org/10.1109/MSR.2015.42

[21] Fiorella Zampetti, Salvatore Geremia, Gabriele Bavota, and Massimiliano Di Penta. 2021. CI/CD pipelines evolution and restructuring: A qualitative and quantitative study. In *International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 471–482.

[22] Yangyang Zhao, Alexander Serebrenik, Yuming Zhou, Vladimir Filkov, and Bogdan Vasilescu. 2017. The impact of continuous integration on other software development practices: A large-scale empirical study. In *International Conference on Automated Software Engineering (ASE)*. 60–71. https://doi.org/10.1109/ASE.2017.8115619