An Empirical Analysis of the GitHub Actions Language Usage and Evolution

Aref Talebzadeh Bardsiri[1], Alexandre Decan[1,2] and Tom Mens[1]

[1]Software Engineering Lab, University of Mons, Belgium
[2]F.R.S.-FNRS Research Associate

With the increasing demand for efficient and high-quality software systems, the practice of Continuous Integration, Deployment and Delivery (CI/CD) has become mainstream in software projects to streamline their development pipelines. CI/CD services automate repetitive tasks such as building code, running tests, and deploying applications. They have become an integral part of software development because they enhance productivity, improve efficiency, and reduce the likelihood of human errors [1].

In the past, different CI/CD services (e.g., Travis, CircleCI and Jenkins) were frequently used in GitHub repositories. Since the public release in 2019 of GitHub's own integrated CI/CD solution called GitHub Actions (hereafter shortened to GHA), it has become the most popular CI/CD tool on GitHub [2]. GHA allows repository maintainers to automate numerous activities, through YAML-based workflow configuration files.

For writing workflows, GHA provides a rich set of language constructs (i.e., keys, structures, values, etc.). According to Mernik's definition of a domain-specific language (DSL) [3], the GHA workflow syntax[1] is a DSL that allows workflow maintainers to define workflows, jobs, steps, and more. Its seamless integration with GitHub, its large marketplace of Actions, and its free plan for running workflows for public repositories, have made GHA a compelling choice among developers [2, 4].

However, beyond adoption, researchers have highlighted that its use comes with multiple challenges. Practitioners reported difficulties in understanding and writing workflow files. As an example, a workflow maintainer said *"YAML is untyped, which frequently leads to serious bugs in configuration code. I wish there was a statically-typed and more reliable alternative to YAML available and officially supported by GitHub Actions"* [5]. Ghaleb et al. [6] further found that GHA workflow files are among the most complex CI/CD automation services and can have high maintenance effort, while Zheng et al. [7] observed that GHA workflow files frequently fail during execution, highlighting challenges related to reliability and efficiency.

These findings suggest that GHA syntax and semantics may be poorly understood and insufficiently mastered by workflow maintainers. Despite the widespread adoption of GHA, there is a lack of comprehensive empirical studies that analyze the language constructs used in GHA workflow files, their usage patterns, and their evolution over time. We believe that addressing this gap is the first step for researchers to study current challenges in GHA workflows, such as their complexity and maintainability, and to provide a set of best practices to overcome these challenges.

In this empirical study of GHA language and its usage, we therefore aim to answer the following research questions:

RQ1 *What are the constructs of the GHA language?* A first step towards understanding the usage of GHA is to identify its language constructs. To this end, we conduct a large-scale empirical analysis of workflows and used the results as a proxy to enumerate the constructs of GHA. As an outcome of this RQ, we identify 197 constructs.

BENEVOL 2025: The 24th Belgium-Netherlands Software Evolution Workshop Enschede, 17-18 November 2023
aref.talebzadehbardsiri@umons.ac.be (A. Talebzadeh Bardsiri); alexandre.decan@umons.ac.be (A. Decan); tom.mens@umons.ac.be (T. Mens)
0009-0005-3719-9716 (A. Talebzadeh Bardsiri); 0000-0002-5824-5823 (A. Decan); 0000-0003-3636-5020 (T. Mens)

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

RQ2 *Which constructs are used in practice?* Understanding the usage frequency of the different GHA language constructs can reveal which constructs are central to workflow configurations and which ones are more specialized. To answer this question, we analyze the frequency of all constructs extracted from a large corpus of workflow snapshots. We conclude that only a small subset of them are frequently used in practice, while the majority occur rarely.

RQ3 *Which constructs contribute to GHA features?* To better understand the purpose of the different GHA constructs and to gain a higher-level view of the GHA language, we group constructs based on the features they contribute to. We refer to *features* as logical groupings of language constructs. For example, the *matrix strategy* is a feature that enables maintainers to run multiple instances of a job with different configurations and it involves several constructs. The outcome of this RQ is a mapping from constructs to features, along with an assessment of the number and nesting of constructs for each feature.

RQ4 *How are GHA features used in practice?* From RQ2, we observed that a small subset of GHA constructs are frequently used in practice. Building on this, understanding the frequency of usage of GHA *features* provides a higher-level perspective on how workflows are configured and implemented. We identify which features are commonly used and which ones are rarely observed in the workflows. In addition, we analyze to what extent workflows use the available constructs for each feature, and which constructs are most frequently employed in their implementation.

RQ5 *How does the GHA language usage evolve over time?* We analyze the evolution of GHA language constructs and workflow features to understand how its use changed over time. This knowledge can help us identify trends and change patterns in the usage of GHA.

To answer our research questions, we conducted an empirical study on a large-scale dataset of GHA workflow histories. The dataset, provided by Cardoen et al. [8], contains over 3 million workflow snapshots from 49K popular and active GitHub repositories related to software development, covering the period from July 2019 to August 2025. Such results pave the way for a more in-depth study on the complexity of writing and maintaining GHA workflow files.

References

[1] F. Zampetti, S. Geremia, G. Bavota, M. Di Penta, CI/CD pipelines evolution and restructuring: A qualitative and quantitative study, in: Int'l Conf. Software Maintenance and Evolution (ICSME), 2021.

[2] M. Golzadeh, A. Decan, T. Mens, On the rise and fall of CI services in GitHub, in: Int'l Conf. Software Analysis, Evolution and Reengineering (SANER), IEEE, 2022, pp. 662–672. doi:10.1109/SANER53432.2022.00084.

[3] M. Mernik, J. Heering, A. M. Sloane, When and how to develop domain-specific languages, ACM computing surveys (CSUR) 37 (2005) 316–344.

[4] A. Decan, T. Mens, P. Rostami Mazrae, M. Golzadeh, On the use of GitHub Actions in software development repositories, in: Int'l Conf. Software Maintenance and Evolution (ICSME), IEEE, 2022. doi:10.1109/ICSME55016.2022.00029.

[5] S. G. Saroar, M. Nayebi, Developers' perception of GitHub Actions: A survey analysis, in: Int'l Conf. Evaluation and Assessment in Software Engineering, 2023. doi:10.1145/3593434.3593475.

[6] T. Ghaleb, O. Abduljalil, S. Hassan, Ci/cd configuration practices in open-source android apps: An empirical study, ACM Trans. Softw. Eng. Methodol. (2025). URL: https://doi.org/10.1145/3736758. doi:10.1145/3736758, just Accepted.

[7] L. Zheng, S. Li, X. Huang, J. Huang, B. Lin, J. Chen, J. Xuan, Why do github actions workflows fail? an empirical study, ACM Trans. Softw. Eng. Methodol. (2025). doi:10.1145/3749371.

[8] G. Cardoen, T. Mens, A. Decan, A dataset of GitHub Actions workflow histories, in: Int'l Conf. Mining Software Repositories (MSR), ACM, 2024, pp. 677–681. doi:10.1145/3643991.3644867.