# Main Problems Faced When Building GitHub Actions

Name of student 1: Kardo Marof
Name of student 2: Saif Sayed

Proposed academic supervisor's name (leave blank if you do not have an academic supervisor): Linda Erlenhov
Have your proposed academic supervisor **clearly** stated that he/she will supervise you: YES

Will the thesis work be conducted in collaboration with an external organization: NO
If yes, is your supervisor aware of the external organization and the contact person/advisor : YES / NO

## I. Introduction

Continuous Integration (CI) has become an integrated part of collaborative software development and DevOps practices. CI automates the quality of code checks, tests and integration of code changes in collaborative environments. The benefits of CI brings early detection of issues, fast feedback loops, increased code quality, reduced integration risks and continuous improvements. Famous examples of CI services include Jenkins, Travis, CircleCI and GitLab CI/CD [1]. GitHub Actions (abbreviated as GHA) was introduced to the public in 2019 as an alternative CI service for GitHub repositories. GitHub introduced its marketplace for sharing automation tools in an effort for developers to reuse workflow components [2].

The so called "Actions" refers to automated workflows triggered by specific events within a repository, including committing changes, opening pull requests, or creating new branches. These workflows streamline development processes by automating tasks and enhancing efficiency. GitHub's integration of GHA allows developers to define custom task sequences in response to events, simplifying collaboration and promoting a seamless development experience [3].

The growing popularity of GHA is immense, with on average more than 20 million GitHub Action minutes used per day in 2023. This growth leads to a 169% increase in the usage of automating tasks in public projects, pipelines and more [4]. Given its popularity, the increasing usage of GHA has lead to an emergence of its own ecosystem [5]. According to Decan [5], the growing ecosystem of GHA bears similarities to reusable software libraries distributed by package managers such as npm, Cargo, RubyGems, Maven and PyPI among others. Where these ecosystems are well known to suffer from variety of issues such as obsolescence, dependency issues, breaking changes and security vulnerabilities to name a few [5]. The authors go on to state "*The GHA ecosystem is likely to suffer from very similar issues and these issues will continue to become more important and more impactful, as the number of reusable Actions continues to grow at a rapid pace.*"

Given the concerns surrounding the GHA ecosystem, it is self-evident that developers will experience the effects of these issues. Moreover, it's worth noting that not all Actions are available on the GitHub marketplace. Many developers create and maintain their own Actions within local repositories, without making them available on the marketplace. Decan [5] conducted an analysis of prevalent automation practices on GitHub and discovered that 43.9% of repositories in their dataset reflected this behavior.

Due to its novelty, there is limited understanding of the challenges faced when implementing GHA. Therefore, by systematically analysing StackOverflow posts, GitHub Discussions threads, tags, and other pertinent repositories, alongside the utilization of database queries and APIs, we aim to quantitatively examine the questions, topics, and answers surrounding GHA. This endeavor not only facilitates the clarification of prevalent issues but also provides insights into potential solutions and areas requiring further research and development within the GHA landscape.

Through this research, we intend to answer the following questions:

**RQ1: To what extent do developers rely on locally maintained Actions within their repositories, as opposed to utilizing Actions available on the GitHub marketplace, and what factors contribute to this distribution pattern?**

We selected RQ1 in order to distinguish the problems faced between locally maintained and marketplace Actions. It is not currently known the exact factors that contribute to this distribution. There are also workflows that utilize both action types and knowing which action types are more locally maintained opposed to others, will give great insight into the

decision making of designing and developing a workflow.

**RQ2: What are the challenges and issues encountered during the implementation and usage of GHA?**

For RQ2 we want to determine what the chanllenges are for developing, maintaining and using GHA from a developers perspective. We intend to answer this question from several perspectives such as; complexity, scaling, ecosystem and tool integrations.

## II. Related Work

Previous studies on the use of GHA have shown its significant rise in popularity after only 18 months of its official release, thereby slowly replacing the use of traditional CI/CD services in GitHub repositories [6]. Unlike traditional CI/CD services, GHA assists the software development processes by improving code reviews, team communication and internal repository management in addition to automating the build and test procedures of software [7]. As mentioned in the 'Introduction' section, this emerging popularity on the use of GHA and the increasing number of reusable Actions that can be found on its marketplace led GHA to be considered similar to popular reusable software libraries distributed by package managers such as npm, Cargo, RubyGems, Maven and PyPI and so on. However, this also means GHA is more likely to face similar problems that are currently encountered by these reusable libraries [5]. Consequently, this would increase the chances of failure when building GitHub workflows in GitHub repositories leading to unsuccessful deployment of software packages.

According to Decan [5], the GHA ecosystem deserves to be studied as all other reusable software ecosystems that have made progress in finding issues related to them. Since GHA is a new emerging ecosystem, there is a lack of research and studies that have been carried out to identify the challenges and issues that are encountered in GHA. The authors point out that the GHA ecosystem is exposed to similar challenges, such as obsolescence [8] [9], dependency issues [10] [11] [12], breaking changes [13] [14], security vulnerabilities [15] [16], etc., that are encountered in well-researched reusable ecosystems. This section aims to briefly describe some of the issues pointed out above, using findings from related literatures, that are likely to appear in GHA.

### A. Obsolescence

Software codes are regularly updated to add features, fix bugs, etc. Sometimes functions or part of code becomes deprecated meaning they are outdated and no longer used; this usually happens after a long period of time when parts of the software code are replaced by an improved version. Generally, it is a good practice to avoid using deprecated code. Through the findings of Cogo et al. [9], deprecated code used in npm packages has shown to give rise to risks such as incompatibility between two different dependant libraries, presence/absence of features, bugs, security vulnerability and more. Cox et al. [17] found that deprecated systems are four time more likely to suffer from security issues and backward incompatibilities than systems that are up-to-date. In the context of GHA, there is a high possibility that a significant amount of code that are reused are deprecated which might consequently lead GitHub workflows to fail building.

### B. Dependency Issues

Software systems are usually developed using pre-existing and reusable packages such as modules, components, libraries, etc [10] [11]. Consequently, this leads to a large extent of dependencies between the reused packages and the original software code. According to Dietrich et al. [13], software developers struggle to choose which versions of a given package to use and mentions that a handful number of software systems encounter runtime version conflicts due to incompatibility with the versions of other dependant packages. Actions and workflows that depend on jobs from other Actions might fall on a similar pitfall of runtime version conflict and possibly other dependency issues.

### C. Security Vulnerabilities

Software that depends on open source and free reusable libraries provided by package managers like npm are more likely to be exposed to security vulnerabilities. Zimmermann et al. [15] quotes:

"*The open nature of npm has boosted its growth, providing over 800,000 free and reusable software packages. Unfortunately, this open nature also causes security risks, as evidenced by recent incidents of single packages that broke or attacked software running on millions of computers.*"

According to Koishybayev et al. [18], one example of a similar attack using the GHA ecosystem is to perform deployments based on the attacker's code by triggering a misconfigured workflow through a new pull request. Since GHA consists of many open-source reusable Actions, workflow and GHA developers need to be extra careful on choosing what packages they depend on for building jobs. It is a good practice to depend on first-party packages and packages from trustworthy maintainers [15].

Saroar et al. [2] mentions that even though the GHA platform provides a marketplace for sharing and reusing open-source Actions, there are still many repositories that prefer to maintain their own GitHub workflows. Although the survey analysis conducted by the authors resulted in identifying several challenges developers faced when using their own customised YAML files. The results indicated that developers found YAML files to be: (i) confusing and error-prone; (ii) difficult to test and debug; and (iii) hard to follow indentation rules. Another study by Valenzuela-Toledo and Bergel [19] showed that modifications to GitHub workflows are made to

correct YAML syntax and debug GA workflows which is in line with [2].

## III. Research Methodology

With this study we aim to answer some of the challenges and problems developers face when building GHA. Our goal is to evaluate whether similar challenges and issues from well-researched reusable libraries can be spotted in the GHA ecosystem. Overall, we are using a combination of methodologies to answer our research questions.

In this section, we describe the research questions we are trying to answer and the research strategies which will be implemented to facilitate our explorative study. Finally, we identify and explain the limitations relevant to our research.

### *A. Research questions and methodology*

**RQ1: To what extent do developers rely on locally maintained Actions within their repositories, as opposed to utilizing Actions available on the GitHub marketplace, and what factors contribute to this distribution pattern?**

Research Method: Survey

This research question is answered by finding out whether developers favour to build their own GitHub workflows versus reusing Actions from the marketplace. In turn, it also explains the reason for the differences between them.

To address RQ1, we take surveys of individuals from a large group of GitHub users, utilizing GHA. A questionnaire is designed to obtain statistical data about their preferences. Using online surveys makes it possible to collect data for a large population by generalizing our findings based on a sample [20].

1) **Data Collection**:

- **Sampling**: Administer online surveys targeting active GitHub users involved in software development and GHA workflows.
- **Instrumentation**: Design a questionnaire to gather demographic information, usage patterns of GHA, and factors influencing action selection.
- **Procedures**: Distribute surveys through GitHub platforms, developer forums, and social media.
- **Time Schedule**: Plan for 1 month to design, distribute, and collect responses.

2) **Data Analysis**:

- **Procedures**: Employ descriptive and inferential statistics to analyse survey responses.
- **Criteria**: Define metrics such as frequency of marketplace action usage, reasons for selection, and perceived advantages/disadvantages.
- **Evaluation**: Identify trends and factors influencing distribution patterns.

**RQ2: What are the challenges and issues encountered during the implementation and usage of GHA?**

Research Method: Repository Mining and Interview

Here we already know the problems that are encountered in other ecosystems. Our goal is to locate these similar challenges and/or issues in GHA and evaluate the effect of these problems on the GHA ecosystem.

We plan to carry out repository mining and interviews to address RQ2. We are using repository mining which is a type of case study which is followed by a semi-structured interview. We can also automate the process of mining data by developing scripts and easing the process of going through multiple repositories [21].

1) **Data Collection**:

- **Repository Mining**: Utilize APIs and database queries to collect relevant data from StackOverflow, GitHub Discussions, and other pertinent repositories.
- **Interviews**: Conduct semi-structured interviews with developers experienced in GHA usage and are actively devoloping.
- **Procedures**: Develop interview protocols focusing on specific topics related to challenges, issues, and comparisons.
- **Time Schedule**: Allocate 1.5 month for data collection, including both repository mining and interviews.

2) **Data Analysis**:

- **Repository Mining Analysis**: Employ repository mining techniques to analyse patterns and trends in StackOverflow posts, GitHub Discussions threads and repository data obtained through APIs and database queries.
- **Interview Analysis**: Utilize qualitative analysis techniques to analyse interview transcripts, identifying recurring themes and insights.
- **Integration**: Integrate findings from repository mining and interviews with the results obtained from RQ1 and RQ2 to measure and categorize the challenges and issues encountered during the implementation and usage of GHA. The analysis will consider how distribution patterns of Actions (RQ1) influence the identified challenges and issues, providing a comprehensive understanding on the application of GHA based on a subset of

factors.

These methods ensure a systematic approach to address each research question effectively.

### B. Threats to Validity

In this section, we discuss the limitations and threats to validity and how we can mitigate them.

**External Validity:** Since our repository mining procedure is supposed to collect data from diverse GitHub repositories with different implementation of workflows, programming languages and size of the project, the results we obtain might differ depending on these factors and our findings cannot be generalized for all types of GitHub projects. However, it is possible to group the projects based on their size, programming language, etc. After grouping the projects, we could try to analyse data from each group and identify a common trend between the results to generalize our findings.

**Internal validity:** Collecting data from online surveys is a risky approach and might give anomalies in our result, especially distributing the surveys through social media might result in invalid responses. There is even a chance that we lack sufficient answers to come to a conclusion. To mitigate this threat, we could distribute the surveys through highly maintained development platforms or reach out to potential companies in person.

## REFERENCES


[1] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb. Social coding in GitHub: Transparency and collaboration in an open software repository. In *International Conference on Computer Supported Cooperative Work (CSCW)*, pages 1277–1286. ACM, 2012.

[2] S. Saroar and M. Nayebi. Developers' perception of GitHub Actions: A survey analysis. *IEEE Software*, 2023.

[3] C. Chandrasekara and P. Herath. Getting started with GitHub Actions workflows. In *Hands-on GitHub Actions*. Springer, 2021.

[4] GitHub. Octoverse: The state of open source and rise of ai in 2023, 2023.

[5] A. Decan, T. Mens, P. Mazrae, and M. Golzadeh. On the use of GitHub Actions in software development repositories. In *2022 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 235–245, 2022.

[6] M. Golzadeh, A. Decan, and T. Mens. On the rise and fall of ci services in GitHub. In *29th IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2021.

[7] C. Chandrasekara and P. Herath. *Hands-on GitHub Actions: Implement CI/CD with GitHub Action Workflows for Your Applications*. Apress, 2021.

[8] A. Decan, T. Mens, and E. Constantinou. On the evolution of technical lag in the npm package dependency network. In *IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 404–414, 2018.

[9] F. Cogo, G. Oliva, and A. E. Hassan. Deprecation of packages and releases in software ecosystems: A case study on npm. *IEEE Transactions on Software Engineering*, 2021.

[10] A. Decan, T. Mens, and P. Grosjean. An empirical comparison of dependency network evolution in seven software packaging ecosystems. *Empirical Software Engineering*, 24(1):381–416, 2019.

[11] C. Soto-Valero, N. Harrand, M. Monperrus, and B. Baudry. A comprehensive study of bloated dependencies in the maven ecosystem. *Empirical Software Engineering*, 26(3):45, 2021. [Online]. Available: https://doi.org/10.1007/s10664-020-09914-8.

[12] A. Decan and T. Mens. What do package dependencies tell us about semantic versioning? *IEEE Transactions on Software Engineering*, 47(6):1226–1240, 2019.

[13] J. Dietrich, D. Pearce, J. Stringer, A. Tahir, and K. Blincoe. Dependency versioning in the wild. In *16th International Conference on Mining Software Repositories (MSR)*, pages 349–359, 2019.

[14] A. Decan, T. Mens, and E. Constantinou. On the impact of security vulnerabilities in the npm package dependency network. In *15th international conference on mining software repositories*, pages 181–191, 2018.

[15] M. Zimmermann, C.-A. Staicu, C. Tenny, and M. Pradel. Small world with high risks: A study of security threats in the npm ecosystem. In *28th USENIX Security Symposium*, pages 995–1010, 2019.

[16] R. G. Kula, D. M. German, A. Ouni, T. Ishio, and K. Inoue. Do developers update their library dependencies? *Empirical Software Engineering*, 23(1):384–417, 2018.

[17] J. Cox, E. Bouwers, M. van Eekelen, and J. Visser. Measuring dependency freshness in software systems. In *Int'l Conf. Software Engineering*, pages 109–118. IEEE Press, 2015.

[18] I. Koishybayev, A. Nahapetyan, R. Zachariah, S. Muralee, B. Reaves, A. Kapravelos, and A. Machiry. Characterizing the security of github ci workflows. In *USENIX Security*, 2022.

[19] P. Valenzuela-Toledo and A. Bergel. Evolution of github action workflows. In *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 123–127, 2022.

[20] J. Linºaker, S. Sulaman, and M. Höst. Guidelines for conducting surveys in software engineering. Technical report, Version 1.1, 2013.

[21] K. K. Chaturvedi, V. B. Sing, and P. Singh. Tools in mining software repositories. In *Proceedings of the 2013*, 2013.