

# 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 [14]. 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.

The growing popularity of GHA is immense, with more than 20 million GitHub Action minutes used per day on average in 2023. Leading to a 169% increase of usage to automate tasks in public projects, pipelines and more [3]. Given its popularity, the increasing usage of GHA has lead to an emergence of its own ecosystem [4]. According to Decan et al. [4], 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 [5], [6], dependency issues [7], [8], [9], breaking changes [10], [11], and security vulnerabilities [12], [13] to name a few. Decan et al. [4] 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 et al. [4] 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:

- 1) What are the prevalent automation practices and patterns observed within GitHub repositories, particularly concerning the adoption and utilization of GHA?
- 2) 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?

- 3) What are the primary challenges and issues encountered during the implementation and usage of GHA, and how do these challenges manifest in the evolving ecosystem surrounding GHA, particularly in comparison to established software library ecosystems like npm, Cargo, RubyGems, Maven, and PyPI?

## 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 [15]. 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 [16]. 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 [4]. However, this also means GHA is more likely to face similar problems that are currently encountered by these reusable libraries. 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 et al. [4], it is reasonable enough to consider GHA as its own software ecosystem consisting of more than 12K reusable actions. The paper claims that 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. Due to insufficient research and unaddressed questions regarding the GHA ecosystem, it is high time we set out to look for issues and challenges faced during the implementation and usage of GHA. Decan et al. [4] points out that the GHA ecosystem is exposed to similar challenges, such as obsolescence, dependency issues, breaking changes, security vulnerabilities, 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 and so on. 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. [6], 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 [7] [8]. Consequently, this leads to a large extent of dependencies between the reused packages and the original software code. According to Dietrich et al. [10], 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. GitHub 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. [12] 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 Zimmermann et al. [12], one way of exposing vulnerability on software is by publishing malicious packages and guiding the client to depend on these harmful packages. 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 [12].

## III. RESEARCH METHODOLOGY

Our findings from several scientific papers relevant to the topic led us to identify the gap in knowledge regarding the new emerging GHA ecosystem. To address the contemporary lack of research and studies within this area, we aim to answer some of the challenges and problems developers face when building GitHub Actions (GHA). Our goal is to evaluate whether similar challenges and issues from well-researched reusable libraries can be spotted in the GHA ecosystem. We also intend to mine software repositories on GitHub to discover further problems development teams encounter while building workflows specific to their project and address the challenges surrounding the development of workflows.

While doing so, we expect our research to provide insights on potential solutions and identify areas surrounding the GHA ecosystem that require further research.

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/or hypotheses*

Throughout this study, we aim to obtain answers and data to the three research questions we mentioned in the Introduction section. For each of the questions, we elaborate it further and describe the research method we believe is suitable for our study.

**RQ1:** What are the prevalent automation practices and patterns observed within GitHub repositories, particularly concerning the adoption and utilization of GHA?

Our intention is to pick a wide variety of public repositories on GitHub and identify how developers mitigate problems faced when utilizing GHA. Consequently, we recognise the common trend of practices developers follow to prevent automation failures.

To address RQ1, we carry out our research through a case study. The study should explore several diverse GitHub repositories where data is collected and analysed from their respective builds of workflows, comments and implementation of workflows, GitHub documentations concerning the development of workflows and more. Case study aligns with our aim to discover the present problems within GHA ecosystem in real-time [18].

**RQ2:** 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?

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

To address RQ2, we take surveys of individuals from a large group of GitHub users, utilizing GitHub actions, from all over Gothenburg, Sweden. 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 [19].

**RQ3:** What are the primary challenges and issues encountered during the implementation and usage of GHA, and how do these challenges manifest in the evolving ecosystem surrounding GHA, particularly in comparison to

established software library ecosystems like npm, Cargo, RubyGems, Maven, and PyPI?

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 data mining and interviews to address RQ3. We are using data 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 [20].

#### *B. Research methodology to be used*

This section outlines the methodology to address each research question outlined in Section III.A using appropriate research methods. Each method is accompanied by steps for data collection and analysis.

**Research Question 1:** Prevalent automation practices and patterns observed within GitHub repositories, particularly concerning the adoption and utilization of GHA.

**Research Method:** Case Study

##### **1) Data Collection:**

- **Sampling:** Select diverse GitHub repositories based on programming languages, project sizes, and industries.
- **Data Sources:** Extract data from GitHub repositories, focusing on GHA workflow configurations, historical builds, and project documentation.
- **Procedures:** Develop a scriptive program to extract and analyse repository information in a cloud based environment.
- **Time Schedule:** Allocate approximately 1 month for metric design and data collection.

##### **2) Data Analysis:**

- **Coding Procedures:** Employ thematic analysis to identify prevalent automation practices and patterns within GHA workflows.
- **Criteria:** Define coding categories based on workflow structures, triggers and actions used.
- **Evaluation:** Assess frequency and distribution of identified patterns across repositories.

**Research Question 2:** Distribution pattern and factors influencing reliance on locally maintained actions versus marketplace actions.

Research Method: Survey

### 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.

Research Question 3: Primary challenges and issues encountered during the implementation and usage of GHA, and comparison with established software library ecosystems.

Research Method: Data Mining and Interview

### 1) Data Collection:

- **Data 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 developing.
- **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 data mining and interviews.

### 2) Data Analysis:

- **Data Mining Analysis:** Employ data mining techniques to analyze 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 analyze interview transcripts, identifying recurring themes and insights.
- **Integration:** Integrate findings from data 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 prevalent automation practices (RQ1) and distribution patterns of actions (RQ2) 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.

### C. Threats to Validity

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

**External Validity:** Since our data 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 can't 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 analyze 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. 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)*. ACM, 2012, pp. 1277–1286.
- [2] Saroar, Sk Golam and Nayebi, Maleknaz, "Developers' Perception of GitHub Actions: A Survey Analysis," 2023
- [3] GitHub, "Octoverse: The state of open source and rise of AI in 2023," 2023. [Online]. Available: [octoverse.github.com](https://octoverse.github.com)
- [4] Decan, A., Mens, T., Mazrae, P., & Golzadeh, M. (2022). On the Use of GitHub Actions in Software Development Repositories. In *2022 IEEE International Conference on Software Maintenance and Evolution (ICSME)* (pp. 235-245).
- [5] A. Decan, T. Mens and E. Constantinou, "On the evolution of technical lag in the npm package dependency network", 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 404-414, 2018.
- [6] 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.

- [7] A. Decan, T. Mens and P. Grosjean, "An empirical comparison of dependency network evolution in seven software packaging ecosystems", *Empirical Software Engineering*, vol. 24, no. 1, pp. 381-416, 2019.
- [8] C. Soto-Valero, N. Harrand, M. Monperrus and B. Baudry, "A comprehensive study of bloated dependencies in the Maven ecosystem", *Empirical Software Engineering*, vol. 26, no. 3, pp. 45, 2021, [online] Available: <https://doi.org/10.1007/s10664-020-09914-8>.
- [9] A. Decan and T. Mens, "What do package dependencies tell us about semantic versioning?", *IEEE Transactions on Software Engineering*, vol. 47, no. 6, pp. 1226-1240, 2019.
- [10] J. Dietrich, D. Pearce, J. Stringer, A. Tahir and K. Blincoe, "Dependency versioning in the wild", 16th International Conference on Mining Software Repositories (MSR), pp. 349-359, 2019.
- [11] A. Decan, T. Mens and E. Constantinou, "On the impact of security vulnerabilities in the npm package dependency network", 15th international conference on mining software repositories, pp. 181-191, 2018.
- [12] M. Zimmermann, C.-A. Staicu, C. Tenny and M. Pradel, "Small world with high risks: A study of security threats in the npm ecosystem", 28th USENIX Security Symposium, pp. 995-1010, 2019.
- [13] R. G. Kula, D. M. German, A. Ouni, T. Ishio and K. Inoue, "Do developers update their library dependencies?", *Empirical Software Engineering*, vol. 23, no. 1, pp. 384-417, 2018.
- [14] Chaminda Chandrasekara and Pushpa Herath. 2021. Getting Started with GitHub Actions Workflows. In *Hands-on GitHub Actions*. Springer.
- [15] 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.
- [16] C. Chandrasekara and P. Herath, *Hands-on GitHub Actions: Implement CI/CD with GitHub Action Workflows for Your Applications*. Apress, 2021.
- [17] J. Cox, E. Bouwers, M. van Eekelen, and J. Visser, "Measuring dependency freshness in software systems," in *Int'l Conf. Software Engineering*. IEEE Press, 2015, pp. 109-118.
- [18] Runeson, Per, and Martin Höst. "Guidelines for conducting and reporting case study research in software engineering." *Empirical software engineering* 14.2 (2009): 131.
- [19] Johan Linäker, Sardar Muhammad Sulaman, Martin Höst, *Guidelines for Conducting Surveys in Software Engineering*, v. 1.1.
- [20] Chaturvedi et al., (2013). *Tools in Mining Software Repositories*. Proceedings of the 2013.