# Main Problems Faced When Building GitHub Actions

1$^{st}$ Kardo Marof
*Department of Computer Science and Engineering*
*Software Engineering and Management*
Guthenberg, Sweden
gusmaroka@student.gu.se

## 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.
Throught 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 GitHub Actions (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 GitHub Actions (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

The related work section provides the background and context for the research problem. It should establish the need for the research and indicate that the writer is knowledgeable about the area. The related work section:

- Describes the results of other studies that are closely related to the study being proposed

- Relates a study to the larger, ongoing dialogue in the literature about a topic, filling in gaps and extending prior studies
- Provides a framework for establishing the importance of the study, as well as a benchmark for comparing the results of a study with other findings
- ”Frames” the problem earlier identified

The related work section shall demonstrate to the reader that you have a comprehensive grasp of the field and are aware of important recent substantive and methodological developments. Define the starting point for your study - how will your study refine, revise, or extend what is now known?

In a proposal, the related work section is generally brief and to the point. However, it shall be more extensive than the brief review in the introduction, both in terms of information about the related research that was mentioned in the introduction and by using more references. Select and reference only the more appropriate citations. Make key points clearly and succinctly. Later in your thesis, you will elaborate on this section.

## III. Research Methodology

Start by stating the aim/purpose of the research study.

### A. Research questions and/or hypotheses

Questions are relevant to descriptive, normative or census type research. (What are relevant factors? How many of them are there? Is there a relationship between them?) Hypotheses are relevant to theoretical research, and when you state hypotheses the reader is entitled to have an exposition of the theory that lead to them (and the assumptions underlying the theory).

In general, you should be prepared to interpret any possible outcome with respect to the questions or hypotheses. Try to visualize in your mind tables or other summary devices, which you expect to come out of the research, short of the actual data.

Describe the main research question(s) that you intend to answer with your thesis project. Use sub-questions if needed.

### B. Research methodology to be used

Any research or problem solving requires a systematic approach with methods and procedures. Indicate the steps you will take to answer every question or to test every hypothesis indicated in Section III.A, to solve the problem that you are addressing. There are several research methods, e.g. design research [3], case study [5], and Survey [2], [4], just to mention a few. Different research methods and procedures require different descriptions.

For example, for a survey, it becomes vital to describe sampling and instrumentation. The sampling, i.e. the population and how the sample has been drawn from that, needs to be described to clarify to what extent the findings of a study can be generalized to people or situations. You shall also outline the instruments you propose to use (surveys, scales, interview protocols, observation grids). For a case study or a design research, other aspects become vital.

*1) Data collection:* For all studies, you need to have a systematic approach for data collection. Outline the general plan for what data to collect, and how. This may include survey administration procedures, interview or observation procedures. Also, provide a general outline of the time schedule you expect to follow.

*2) Data analysis:* For all studies, you need to have a systematic approach for data analysis. Specify the procedures you will use to analyze your data. If coding procedures are to be used, describe these in reasonable detail. For evaluations, describe the criteria to be used in reasonable detail.

### C. Threats to Validity

A limitation identifies potential weaknesses of the study. Think about your analysis, the nature of self-report, your instruments, and the sample. Think about threats to external or internal validity that may have been impossible to avoid or minimize and explain these.

## References

Reference all sources that are cited in your proposal.

## 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

[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.