When Pipelines Meet Things: Understanding CI/CD Practices in FLOSS IoT Projects through GitHub Actions

Igor Muzetti Pereira
Computing and Systems Department
Federal University of Ouro Preto
João Monlevade, Brazil
igormuzetti@ufop.edu.br

Tiago Garcia de Senna Carneiro
Computer Science Department
Federal University of Ouro Preto
Ouro Preto, Brazil
tiago@ufop.edu.br

Eduardo Figueiredo
Computer Science Department
Federal University of Minas Gerais
Belo Horizonte, Brazil
figueiredo@dcc.ufmg.br

Abstract

DevOps (Development and Operations) has been successfully adopted by large software companies, especially in mobile and cloud applications. This success has increased interest in DevOps adoption by small organizations and FLOSS (Free/Libre Open Source Software) communities developing IoT (Internet of Things)-related projects. CI/CD (Continuous Integration and Delivery) pipelines are among the most common DevOps practices for automating software development and delivery. Understanding their adoption is important for project planning and management. This study analyzes the CI/CD pipelines of twenty-five popular IoT-related GitHub repositories. We examine GitHub Actions workflow files and apply open coding to repository issues to investigate how these projects implement CI/CD practices. Results show a higher adoption of continuous integration than continuous delivery. Workflow executions are mainly triggered by technical events, such as Push and Pull Requests. We also observed frequent use of third-party Actions, while many steps rely on local commands and some Actions are not available on GitHub Marketplace, suggesting the use of community-specific solutions. The analysis of issues indicates that CI/CD discussions mainly focus on challenges and difficulties faced by contributors.

Keywords

RTOS, FLOSS, GitHub Actions, CI/CD, IoT-Related Projects

1 Introduction

Different applications of Internet of Things (IoT) systems are increasingly present in society, including smart cities, smart homes, industry 4.0, and wearable devices [13]. IoT systems consist of connected objects that share data and perform processing to generate value for users [1, 10]. These systems combine sensor networks and embedded system architectures, and their design varies according to application domains [12].

The quality of IoT systems depends on the development processes of both hardware and software components. Software engineering aims to improve these processes to deliver high-quality systems at satisfactory costs [16]. In this context, DevOps represents a collaborative and multidisciplinary approach that automates the continuous delivery of software updates while ensuring correctness and reliability [11]. Among its practices, Continuous Integration and Continuous Delivery (CI/CD) pipelines automate software building, testing, and delivery activities [8]. However, IoT projects introduce specific challenges due to embedded systems characteristics, such as continuous operation, limited hardware resources, reliability requirements, device heterogeneity, and deployment constraints that may require human intervention. Therefore, investigating CI/CD adoption in IoT-related projects is essential to understand challenges specific to this domain.

Automation tools are commonly integrated into source code repositories to support repetitive development activities. In 2019, GitHub introduced GitHub Actions (GHA), a service that enables repository contributors to create automated workflows triggered by repository events. In GHA, Actions consist of sets of commands executed as steps to automate tasks related to continuous integration, continuous delivery, and other DevOps activities. GitHub also supports integration with external automation tools, such as Circle CI, Travis, and Jenkins [6].

According to previous study, IoT project contributors need a holistic understanding of development and delivery processes [14]. Automation tools such as GHA make these processes more transparent, reducing release risks and disseminating knowledge among developer communities [20]. Therefore, understanding the GHA ecosystem in IoT-related projects and anticipating the effects of its adoption are important for project planning and management.

Despite advances in automation tools, developers still face challenges in implementing and maintaining CI/CD pipelines. Problems related to pipeline configuration, execution, and maintenance reveal gaps in knowledge and documentation within IoT-related project communities. These difficulties may result in delayed releases, inconsistent software quality, and reduced adoption of automation practices. Thus, investigating how IoT-related communities structure their CI/CD pipelines and address these challenges is necessary.

This motivated our research, which analyzes FLOSS (Free/Libre Open Source Software) repositories of IoT-related projects to understand how their communities structure CI/CD pipelines and handle challenges associated with these processes.

Our findings indicate most public or third-party Actions identified are highly specific, being used only once in the repositories. Several Actions may represent solutions developed for community needs and potentially become reusable by other projects. CI is the most common category among these Actions. Additionally, issue discussions frequently address difficulties and challenges faced by contributors when working with CI/CD pipelines.

These results suggest that IoT-related project communities are active but may face stability and functionality challenges that motivate contributors to create forks and implement alternative solutions. Encouraging Action reuse, continuous education on CI/CD practices, and community support may contribute to more collaborative and efficient development.

The remainder of this paper is organized as follows. Section 2 presents related work. Section 3 describes the research setup,

followed by the results in Section 4. Section 5 discusses the implications of our findings, Section 6 presents threats to validity and mitigation actions, and Section 7 concludes the paper with final considerations and future work.

2 Related work

Previous studies have investigated agile methodologies in embedded systems and automation tools across different software domains. Recent works analyzed the use of GitHub Actions (GHA) and its effects on collaborative software projects, mainly in web and mobile domains [9, 18]. However, this study focuses on embedded-layer software in IoT systems.

Evidence shows that Continuous Integration (CI) adoption in GitHub projects can improve collaborator productivity and software quality [17]. As a fundamental practice for CI/CD pipelines, CI has been investigated through different automation tools. Zhao et al. (2017) analyzed the impact of Travis CI using Regression Discontinuity Design (RDD), observing increases in merged commits, closed Pull Requests (PRs), and automated tests after adoption, while the number of closed issues decreased [21]. Beller et al. (2017) investigated Travis CI usage and showed that dependency on CI tests varies across programming languages. They also highlighted that GitHub integration enables parallel execution of integration tests in multiple environments [2].

With the introduction of GHA in late 2019, research began to investigate its adoption and impacts. Golzadeh et al. (2022) showed that GHA adoption contributed to reduced usage of other GitHub-integrated CI tools, such as Travis CI, Circle CI, Azure, and AppVeyor. They observed that GHA rapidly increased in popularity, surpassing Travis CI within eighteen months, and that repositories commonly combine multiple CI tools, especially Travis CI, Circle CI, and GHA [6].

Studies have also investigated the effects of GHA adoption on software development practices. Using RDD analysis, Kinsman et al. (2021) observed changes in Pull Request activities after GHA adoption, including an increase in rejected PRs and a decrease in accepted PR commits. They also identified CI, utilities, and CD as the most common categories of Actions [9]. Similarly, Wessel et al. (2023) found that GHA adoption is associated with more PR rejections, increased communication in accepted PRs, fewer commits in accepted PRs, and longer approval times [18].

Decan et al. (2022) empirically analyzed GHA usage in 68K GitHub repositories, finding that 43.9% used GHA workflows. Their study investigated automated workflows and identified that Action reuse is common, although concentrated on a limited number of Actions [5].

Other studies have examined CI impacts on PR delivery time. Bernardo et al., (2023) analyzed 162,653 PRs from 87 open-source projects and found that CI adoption only slightly reduced PR delivery time in some projects [3]. Guo and Leitner (2019) replicated this study using RDD and a control group, confirming limited improvements in individual PR delivery time while showing that CI enables projects to manage more PRs per release. They also highlighted the influence of release-cycle factors on delivery time [7].

Unlike previous studies that mainly analyze general open-source projects or PR delivery metrics, our work investigates CI/CD adoption and practices in IoT-related FLOSS projects using GHA. We analyze pipeline structures through YAML workflow files, community discussions through issues, and repository metrics. Future work includes applying longitudinal approaches, such as RDD, to evaluate the impact of GHA on IoT project activity metrics, including PRs, comments, and commits, as well as comparing GHA adoption with other CI/CD tools used in this domain.

3 Research setup

Our methodology investigates CI/CD practices in twenty-five FLOSS IoT-related project repositories through the analysis of GitHub Actions (GHA) workflow YAML files and community discussions related to CI/CD pipelines. We followed the Goal-Question-Metric (GQM) approach proposed by Basili et al. (1994), defining the research goal at the conceptual level, research questions at the operational level, and metrics for quantitative and qualitative analyses.

3.1 Goal and research questions

The goal of this research is to investigate CI/CD pipeline practices in IoT-related FLOSS projects from the perspective of distributed software development communities. We formulated the following research questions:

- RQ1. How do FLOSS IoT-related projects structure their CI/CD pipeline?
- RQ2. What is discussed in FLOSS IoT-related project communities about the CI/CD pipeline?

RQ1 focuses on understanding how projects implement CI/CD pipelines using GHA, including workflow triggers and Actions used in automation processes. RQ2 investigates community discussions about CI/CD, identifying difficulties, improvements, and other relevant topics. Together, these questions provide insights into both the technical adoption and practical challenges of CI/CD in FLOSS IoT-related projects.

3.2 Selected IoT-related projects

The repository selection was performed using the GitHub API query: `https://api.github.com/search/repositories?q=topic:internetofthings,iot+stars:>1000` . This query aimed to identify popular FLOSS IoT-related projects with more than one thousand stars on GitHub. Then, we manually inspected each repository to confirm its relevance, excluded non-software items (e.g., books or tutorials), and kept only projects related to IoT embedded software or tools that support their development. We considered only repositories written in English.

Repositories with more than one thousand stars were selected because stars are widely recognized as an indicator of popularity and community interest on GitHub [4]. Most developers consider this metric before using or contributing to a project. However, this selection may introduce biases, favoring projects with active marketing, larger contributor bases, or popular languages.

Although IoT systems may include backend and frontend software deployed in clouds, servers, or personal computers, these components were outside the scope of this study. We focused on software closer to IoT devices, particularly those running on

resource-constrained hardware. GitHub was selected because of its popularity among developers, availability of FLOSS projects, and rich historical data. After the selection process and discussions among the authors when necessary, twenty-five repositories were included in the analysis.

3.3 Data collection and analysis

Data collection follows principles from Mining Software Repositories (MSR) research and ACM Empirical Standards. The collected data included GHA workflow configuration files in YAML format and repository discussions related to CI/CD pipelines. The qualitative analysis of discussions followed the open coding technique described by Stol et al. (2016) [15].

To answer RQ1, we automatically collected YAML workflow files from the .github/workflows directory of the selected repositories, which is the standard location for GHA workflow definitions. We analyzed these files to identify workflow triggers and the Actions used in Jobs, including both predefined and customized Actions. Python scripts (download_yml.py and verify_event.py are in our replication package) were developed to automate file collection and analysis. In total, we identified one hundred eighty-two YAML files from nineteen projects using GHA. The analysis focused on the frequency of workflow triggers and the most commonly used Actions.

To answer RQ2, we analyzed CI/CD-related discussions from repository issues. Issues were retrieved using the search expression: "continuous integration" OR delivery OR deploy OR ci/cd in title,body. The analysis aimed to identify themes and challenges discussed by project communities. Following open coding principles, we manually analyzed issue descriptions and comments, created initial classifications, and refined categories through iterative discussions among the authors. Each issue was classified into one of four categories: Difficulties or challenges, improvement, maintenance, and request or suggestion.

This methodology enabled the analysis of both the technical structure of CI/CD pipelines and the experiences reported by IoT-related FLOSS communities, providing evidence to answer the proposed research questions.

4 Results

In this section, we present the results of our study by first describing the analyzed dataset and then answering each research question.

4.1 Dataset overview

IoT-related projects differ from web and mobile projects due to their specific constraints and requirements. While web and mobile applications commonly use languages such as JavaScript, Python, Java, and Ruby in environments with greater computational resources, IoT-related projects frequently rely on C, C++, and Assembly due to hardware limitations, such as reduced processing power, memory, and storage. These projects also emphasize reliability and resilience because software often runs directly on embedded devices under restrictive conditions.

The selected repositories include nine Real-Time Operating System (RTOS) projects and sixteen Device/Toolkit (DT) projects. The RTOS projects were: RT-Thread/rt-thread, zephyrproject-rtos/zephyr, RIOT-OS/RIOT, ARMmbed/mbed-OS, contiki-os/contiki, FreeRTOS/FreeRTOS, aws/amazon-freertos, apache/nuttx, and contiki-ng/contiki-ng. The DT projects were: tencent/ncnn, micropython/micropython, esp8266/Arduino, arduino/Arduino, rwaldron/johnny-five, espressif/arduino-esp32, espressif/esp-idf, hybridgroup/gobot, jerryscript-project/jerryscript, hybridgroup/cylon, adafruit/circuitpython, espressif/ESP8266_RTOS_SDK, jerryscript-project/iotjs, cesanta/mongoose-os, hybridgroup/artoo, and arduino/arduino-ide.

Table 1: The twelve kinds of events in workflows and repositories that used GHA

Event	% Workflows	% Repositories
push	74.73%	100.00%
pull_request	71.98%	100.00%
release	25.82%	61.10%
schedule	15.93%	50.00%
workflow_dispatch	38.80%	44.40%
issues	9.34%	38.80%
pull_request_target	22.20%	16.00%
workflow_run	2.20%	16.60%
issue_comment	1.10%	11.10%
repository_dispatch	1.65%	5.50%
workflow_call	1.65%	5.50%
pull_request_review	0.55%	5.50%

4.2 RQ1. How do FLOSS IoT-related projects structure their CI/CD pipeline?

To answer RQ1, we analyzed GHA workflow configuration files. Among the twenty-five repositories, nineteen use GHA, containing one hundred eighty-two YAML files. Therefore, the percentages presented in Tables 1 and 2 consider these nineteen repositories as the total.

GHA workflows are triggered by events, and we identified twelve event types used in the analyzed repositories. These events appeared four hundred and two times in workflows. Each Job contains, on average, four steps, which may execute reusable Actions or local commands.

Table 1 presents the most frequent workflow events. The most frequent events were push and pull_request, both used by more than half of the analyzed projects. These events are directly related to development activities: push triggers workflows after commits, while pull_request triggers workflows during code review and merging processes. These results are consistent with Decan et al. (2022), where these events were also the most commonly used [5].

Most repositories using GHA configure multiple workflows: fifteen of the nineteen projects contain two or more workflow files. This separation may facilitate maintenance and understanding by the community. We identified seventy-five reusable Actions among seven hundred sixty-six steps, and Table 2 presents the most frequent ones.

The most frequently used Action was actions/checkout, present in all repositories using GHA. This Action provides access to the repository source code and is commonly the first step in CI/CD workflows. Other frequent Actions support artifact management,

programming environments, dependency caching, code coverage, and container-based workflows.

Among the ten most frequent Actions, six are maintained by GitHub, three by Docker, and one by Codecov. Docker-related Actions indicate relevant adoption of containerization practices in CI/CD workflows.

To further analyze reusable Actions, we searched GitHub Marketplace. We identified sixty-five of the seventy-five Actions in the Marketplace. These Actions were used in seven hundred and seventy-eight of the one thousand and six hundred and twenty-seven analyzed steps (47.8%). The remaining steps execute local commands directly defined in YAML files.

Table 2: Top ten most frequent actions in steps and repositories using GHA

Action	% Steps	% Repositories
actions/checkout	47.17%	100.00%
actions/upload-artifact	12.34%	52.63%
actions/setup-python	8.74%	68.42%
actions/cache	7.84%	31.58%
actions/download-artifact	3.47%	26.32%
codecov/codecov-action	1.03%	15.79%
actions/setup-node	0.90%	10.53%
docker/login-action	0.77%	26.32%
docker/build-push-action	0.77%	21.05%
docker/setup-qemu-action	0.77%	15.79%

Table 3 presents the most frequent Action categories. Continuous Integration was the most common category, followed by Utilities and Deployment. These results indicate that projects frequently use Actions to automate building, testing, and workflow support activities.

Table 3: Most frequent categories and the proportion of actions using them

Action category	% Actions
Continuous Integration	26.67%
Utilities	20.00%
Deployment	6.67%
Dependency Management	5.53%
Testing	5.53%
Project Management	5.53%
Code Quality	4.00%
Code Review	2.67%
Open Source Management	2.67%
Chat	2.67%
Security	2.67%
Publishing	1.33%
Container CI	1.33%
Uncategorised	13.33%

Regarding CI/CD tools, eleven of the twenty-five repositories use only GHA. Five projects combine GHA with other tools, such as Jenkins, GitLab CI, and AppVeyor. Four projects use Travis CI exclusively, two use non-GHA combinations, and one project does not use pipeline automation. All analyzed RTOS projects use GHA.

Compared with previous studies, GHA adoption in our dataset is high: nineteen of twenty-five repositories (76%) use GHA. Previous studies reported adoption rates ranging from approximately 30% to 43.9% in general FLOSS repositories [5, 9, 18]. This adoption may be facilitated by GitHub-native workflow configuration, visualization tools, and Action reuse.

4.3 RQ2. What is discussed in FLOSS IoT-related projects communities about the CI/CD pipeline?

To answer RQ2, we analyzed one hundred eighty-six GitHub issues related to continuous integration, delivery, or deployment using the search expression: "continuous integration" OR delivery OR deploy OR ci/cd in title,body. Using open coding inspired by Grounded Theory principles (Stol et al., 2016), we classified issues into four categories after iterative analysis among the authors. Table 4 presents the results.

Open coding was conducted by two researchers who iteratively analyzed the descriptions and comments of CI/CD-related issues. Initially, one researcher read the issues in their entirety and proposed an initial coding scheme based on emerging categories. Subsequently, the second researcher independently reviewed this coding, evaluating the appropriateness of the categories and the assigned classifications. Discrepancies were discussed in meetings among the authors until a consensus was reached, iteratively refining category definitions and the final classification of each issue. As the analysis was exploratory and driven by the identification of emerging themes, no statistical measure of inter-coder agreement was calculated. Instead, we sought to enhance the analysis's reliability through successive rounds of discussion and category consolidation among the researchers.

Table 4: Issue discussion categories

Categories of the Issues	# Issues	% Issues
Difficulties or challenges	83	43.62%
Improvement	44	23.66%
Maintenance	32	17.20%
Request or suggestion	27	14.52%
Total	186	100.00%

Difficulties or challenges. This was the most frequent category, including discussions about broken builds, configuration problems, and failures during CI/CD activities. Contributors frequently reported difficulties adapting workflows to new dependencies, environments, or hardware platforms.

Improvement. This category includes discussions about adding workflows, improving documentation, and extending CI/CD support for new boards and environments.

Maintenance. Issues classified in this category involved updates to workflow resources, configuration changes, and minor adjustments to improve pipeline execution.

Request or suggestion. This category includes requests for new CI/CD capabilities or modifications to existing processes, such as remote deployment support and corrections related to hardware-specific configurations.

Unlike Kinsman et al. (2021), who identified maintenance-related discussions as predominant, our results show that IoT-related projects mainly discuss difficulties and challenges [9]. Wessel et al. (2023) also observed discussions related to configuration problems and requests for help, reinforcing that CI/CD adoption may generate specific challenges depending on the project context [18].

5 Discussion

In this work, we provide evidence about CI/CD pipeline automation practices in IoT-related FLOSS projects. Among the twenty-five analyzed repositories, nineteen showed evidence of GHA usage. All RTOS projects adopted GHA, except contiki-os/contiki, which uses Travis CI; however, its successor, contiki-ng/contiki-ng, migrated to GHA. Among Device/Toolkit (DT) projects, eight repositories use GHA, while others still rely on tools such as Circle CI, Jenkins, Travis CI, GitLab CI, and AppVeyor. Five DT projects did not use GHA at the time of our analysis. These results suggest that GHA adoption is becoming a relevant trend in IoT-related projects hosted on GitHub, although communities may still be transitioning to this relatively recent tool.

Although previous studies have demonstrated the increasing adoption of GHA in FLOSS projects across different domains, our study focused on observing its usage in IoT-related repositories at a specific point in time [5, 9, 18]. Therefore, we did not investigate the motivations behind GHA adoption. Previous literature suggests that its popularity may be related to its native integration with GitHub, YAML-based configuration, documentation, and reusable Actions. Investigating how these characteristics influence adoption in IoT communities remains an opportunity for future research.

We identified an average of thirty reusable or third-party Actions per repository, which is higher than values reported in previous studies. Wessel et al. (2023) reported an average of 1.5 third-party Actions per workflow, while Kinsman et al. (2021) found an average of 4.5 third-party Actions per repository. These differences may be related to the specific characteristics of our IoT dataset and the maturity of the analyzed projects [9, 18].

The analyzed repositories used twelve different workflow events, mainly related to technical development activities, such as push, pull_request, and release. We also identified seventy-five reusable Actions, with Continuous Integration being the most frequent category. These Actions mainly support activities related to building, testing, and code integration. The second most frequent category was Utilities, which provides additional support for repository interaction and management. Therefore, the most frequently used Actions address both technical and organizational aspects of development workflows.

Although 7.5% of the identified Actions were not available on GitHub Marketplace, they may represent solutions developed for specific community needs that could become reusable in the future. Additionally, 52.2% of workflow steps execute local commands instead of reusable Actions, suggesting opportunities for future Action development and publication. Considering the diversity of microcontrollers and embedded platforms in IoT environments, reusable Actions for building and configuring these environments may represent a relevant opportunity for the community.

Our findings have implications for both practitioners and researchers. IoT developers and communities need to evaluate tools that support CI/CD automation, and GHA represents a relevant alternative due to its integration with GitHub and support for reusable automation components. Beyond technical activities, workflows can automate project management tasks, social interactions, and integration with external tools. Understanding how these automation mechanisms influence collaboration and development practices is an important direction for future research. This study represents an initial step toward understanding the role of GHA in IoT-related FLOSS communities.

6 Threats to validity

In this section, we discuss the threats to validity of our study and the mitigation strategies adopted, following the guidelines proposed by Wohlin et al. (2012) [19].

External validity. Threats to external validity concern the generalization of our findings beyond the analyzed context. Our study focused on FLOSS IoT-related repositories, and the results may not apply to other domains due to differences in programming languages, developer profiles, and project characteristics. To mitigate this threat, we clearly defined and applied selection criteria to ensure consistency during repository selection.

Construct validity. Threats to construct validity concern whether the collected data and analysis represent the concepts investigated. We interpreted workflow events, labels, and job identifiers based on GitHub documentation; however, communities may use these concepts differently. We selected GHA because of its growing adoption, GitHub integration, large repository ecosystem, and available APIs. For RQ2, the use of keyword-based issue searches may have excluded discussions using different terminology. Furthermore, our point-in-time analysis of YAML workflow files may not capture changes in CI/CD tool usage over time.

Internal validity. Threats to internal validity relate to possible factors affecting our analysis. Actions were categorized according to their GitHub Marketplace primary and secondary categories, assuming primary categories represent their main purpose. When categories were unavailable, we analyzed Action names and workflow steps to identify the most appropriate classification. The qualitative coding relied on consensus among the authors instead of a formal inter-rater agreement measure. We mitigated this threat through iterative discussions and by making the complete coding spreadsheet publicly available in the replication package.

Conclusion validity. Threats to conclusion validity concern the reliability of our interpretations. For RQ1, the analysis was mainly descriptive and based directly on workflow configuration files, reducing the impact of interpretation bias. However, RQ2 involved qualitative analysis of issue discussions using open coding, which may introduce subjectivity. To mitigate this threat, multiple authors iteratively analyzed and discussed the categories until reaching agreement. Additionally, we provide quantitative distributions of classified issues and examples from each category to support the consistency of our interpretations.

7 Final considerations

In this work, we investigated CI/CD pipeline practices in FLOSS IoT-related projects by mining repositories, analyzing workflow configuration files, and examining community discussions. We developed scripts to analyze YAML files and applied open coding, inspired by Grounded Theory, to classify discussions related to CI/CD pipelines.

Our results show that FLOSS IoT-related projects commonly adopt CI/CD practices, with a predominance of Continuous Integration and GHA as the automation platform. We also observed differences between RTOS and Device/Toolkit (DT) projects regarding usage patterns and community discussions. Despite the adoption of CI/CD practices, contributors frequently discuss difficulties and challenges related to pipeline configuration and maintenance.

Most workflows are triggered by technical repository events, especially push, pull_request, and release. Reusable Actions from third-party providers, particularly GitHub, Codecov, and Docker, are widely used. On average, workflows use four third-party Actions, mainly related to Continuous Integration and Utilities that support developer interaction with GitHub. However, Action reuse remains concentrated in a small number of categories, and few security-related Actions were identified, despite security being an important concern in IoT systems.

The analysis of community discussions indicates that contributors mainly report difficulties and challenges when working with CI/CD pipelines. Although projects generally provide documentation about software usage and contribution processes, only a few provide detailed documentation about CI/CD workflows. More complete and updated documentation about pipeline configuration and execution may help maintainers better understand automated processes and focus on other project activities.

As future work, we plan to extend this observational analysis by investigating the impact of GHA adoption on project activity indicators, such as pull requests, comments, commits, and pull request resolution time. We intend to apply Regression Discontinuity Design (RDD) in longitudinal analyses comparing project behavior before and after GHA adoption. This analysis will focus on IoT-related projects that have used GHA for at least twelve months, aiming to evaluate whether CI/CD adoption influences collaboration patterns, pull request outcomes, and project sustainability.

Artifact Availability

In accordance with the VEM 2026 Open Science policy, the complete dataset, the Python scripts for extracting YAML files, and the analysis spreadsheets are available in a public, anonymized replication repository[1].

Acknowledgements

This research was partially supported by the Brazilian funding agencies CAPES, CNPq, FAPEMIG, and the universities where the authors are assigned.

[1]https://zenodo.org/records/21286530

References

[1] Luigi Atzori, Antonio Iera, and Giacomo Morabito. 2010. The Internet of Things: A survey. *Computer Networks* 54, 15 (2010), 2787–2805. doi:10.1016/j.comnet.2010.05.010

[2] Moritz Beller, Georgios Gousios, and Andy Zaidman. 2017. TravisTorrent: Synthesizing Travis CI and GitHub for Full-Stack Research on Continuous Integration. In *IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. 447–450. doi:10.1109/MSR.2017.24

[3] João Helis Bernardo, Daniel Alencar da Costa, Uirá Kulesza, and Christoph Treude. 2023. The impact of a continuous integration service on the delivery time of merged pull requests. *Empirical Software Engineering* 28, 4 (2023). doi:10.1007/s10664-023-10327-6

[4] Hudson Borges and Marco Tulio Valente. 2018. What's in a GitHub Star? Understanding Repository Starring Practices in a Social Coding Platform. *Journal of Systems and Software* 146 (2018), 112–129. doi:10.1016/j.jss.2018.09.016

[5] Alexandre Decan, Tom Mens, Pooya Rostami Mazrae, and Mehdi Golzadeh. 2022. On the Use of GitHub Actions in Software Development Repositories. In *IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 235–245. doi:10.1109/ICSME55016.2022.00029

[6] Mehdi Golzadeh, Alexandre Decan, and Tom Mens. 2022. On the rise and fall of CI services in GitHub. In *IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 662–672. doi:10.1109/SANER53432.2022.00084

[7] Yunfang Guo and Philipp Leitner. 2019. Studying the impact of CI on pull request delivery time in open source projects—a conceptual replication. *PeerJ Computer Science* 5 (2019). doi:10.7717/peerj-cs.245

[8] Jez Humble and Dave Farley. 2010. *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation.* Pearson Education.

[9] Timothy Kinsman, Mairieli Wessel, Marco A. Gerosa, and Christoph Treude. 2021. How Do Software Developers Use GitHub Actions to Automate Their Workflows?. In *IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*. 420–431. doi:10.1109/MSR52588.2021.00054

[10] Surapon Kraijak and Panwit Tuwanut. 2015. A survey on internet of things architecture, protocols, possible applications, security, privacy, real-world implementation and future trends. In *IEEE 16th International Conference on Communication Technology (ICCT)*. 26–31. doi:10.1109/ICCT.2015.7399787

[11] Leonardo Leite, Carla Rocha, Fabio Kon, Dejan Milojicic, and Paulo Meirelles. 2019. A Survey of DevOps Concepts and Challenges. *ACM Comput. Surv.* 52, 6 (2019). doi:10.1145/3359981

[12] Rebeca Campos Motta, Valéria Silva, and Guilherme Horta Travassos. 2019. Towards a more in-depth understanding of the IoT Paradigm and its challenges. *Journal of Software Engineering Research and Development* 7 (2019), 3:1 – 3:16. doi:10.5753/jserd.2019.14

[13] Elisa Yumi Nakagawa, Pablo Oliveira Antonino, Frank Schnicke, Thomas Kuhn, and Peter Liggesmeyer. 2021. Continuous Systems and Software Engineering for Industry 4.0: A disruptive view. *Information and Software Technology* 135 (2021), 106562. doi:10.1016/j.infsof.2021.106562

[14] Igor Muzetti Pereira, Tiago Garcia de Senna Carneiro, and Eduardo Figueiredo. 2021. Investigating Continuous Delivery on IoT Systems. In *Proceedings of the XX Brazilian Symposium on Software Quality (SBQS '21)*. Association for Computing Machinery. doi:10.1145/3493244.3493261

[15] Klaas-Jan Stol, Paul Ralph, and Brian Fitzgerald. 2016. Grounded theory in software engineering research: a critical review and guidelines. In *38th International Conference on Software Engineering (ICSE)*. 120–131. doi:10.1145/2884781.2884833

[16] Marco Tulio Valente. 2024. *Software Engineering: A Modern Approach.* Self-published.

[17] Bogdan Vasilescu, Yue Yu, Huaimin Wang, Premkumar Devanbu, and Vladimir Filkov. 2015. Quality and productivity outcomes relating to continuous integration in GitHub *(ESEC/FSE 2015)*. Association for Computing Machinery, 805–816. doi:10.1145/2786805.2786850

[18] Mairieli Wessel, Joseph Vargovich, Marco A. Gerosa, and Christoph Treude. 2023. GitHub Actions: The Impact on the Pull Request Process. *Empirical Software Engineering* 28, 6 (2023). doi:10.1007/s10664-023-10369-w

[19] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in Software Engineering.* Springer Berlin Heidelberg.

[20] Fiorella Zampetti, Damian Tamburri, Sebastiano Panichella, Annibale Panichella, Gerardo Canfora, and Massimiliano Di Penta. 2023. Continuous Integration and Delivery Practices for Cyber-Physical Systems: An Interview-Based Study. *ACM Transactions Software Engineering Methodology* 32, 3 (2023). doi:10.1145/3571854

[21] 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 *32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 60–71. doi:10.1109/ASE.2017.8115619