Journal of Informatics and Web Engineering
Vol. 3 No. 2 (June 2024) eISSN: 2821-370X

Empirical Analysis of CI/CD Tools Usage in GitHub Actions Workflows

Adam Rafif Faqih[1*], Alif Taufiqurrahman[1], Jati H. Husen[1], Mira Kania Sabariah[1]
[1]Telkom University, Indonesia
**corresponding author: (adamrafif@student.telkomuniversity.ac.id, ORCiD: 0009-0005-6512-6151)*

Abstract - As software systems grow larger and more complex, with rapidly changing requirements, manually managing code integration, testing, and deployment becomes extremely challenging. Continuous Integration and Continuous Deployment (CI/CD) practices and tools have emerged to help automate these processes. This research explores the usage of different categories of CI/CD tools within GitHub Actions workflow configurations across GitHub repositories. The five-tool categories analyzed are Version Control Management, Static Code Analysis, Build Automation, Test Automation, and CI/CD Servers. The data used in this research is from a dataset of GitHub Actions workflow configuration files. From the data, the usage is extracted and the concurrent usage of the tools is calculated. Next, the tools are labeled based on their taxonomy. In our finding, the build automation has the biggest number of uses, while the test automation has the least number of uses. Our finding indicates the correlation between the tool category and the programming language used in the software project. Meanwhile, some tools cannot be classified into the existing taxonomy. This can lead to reevaluating the taxonomy structure of CI/CD tools.

Keywords—Empirical Software Engineering, CI/CD Tools, GitHub Actions Mining, Qualitative Analysis

Received: 15 March 2024; Accepted: 09 May 2024; Published: 16 June 2024

I. INTRODUCTION

Software development becomes more complex over time, with software requirements rapidly changing throughout development [1]. As the software system grows larger, managing code manually becomes extremely challenging, especially when it comes to reviewing, merging, and testing code from other contributors. This problem can lead to delays with every deliverable [2]. To address this issue, the concept of Continuous Integration and Continuous Deployment (CI/CD) has emerged as a solution. CI/CD enables the automatic integration, testing, and deployment of code changes, streamlining the entire software development lifecycle [1][3]. To facilitate the CI/CD process, many CI/CD tools emerge to implement CI/CD. These tools helping the developer to automate build, testing, and deploy their software [4].

One of the popular tools for implementing CI/CD is GitHub Actions, a hosted service provided by GitHub that allows developers to automate their software development workflows [4]. To support the development workflows, GitHub Actions provides tools to build, test, and deploy automatically [5]. These tools are provided by GitHub or another developer in GitHub. There are many CI/CD tools, both for GitHub Actions and other platforms in general. These tools can be categorized into five categories [6]. The given categories are as follows: Version Control Management, Static Code Analysis, Build Automation, Test Automation, and CI/CD Servers. Meanwhile, there is no understanding for the distribution of this categories in the real-world implementation of GitHub Actions.

Journal of Informatics and Web Engineering
https://doi.org/10.33093/jiwe.2024.3.2.18

Published by MMU Press. URL: https://journals.mmupress.com/jiwe

In this research, the goal is to explore the usage of these tools inside the workflow configuration of GitHub Actions. The main goal is to understand the usage frequency of these tools inside the workflow configuration of GitHub Actions. To achieve our goals, the following research questions (RQs) have been defined:

- RQ1. What type of CI/CD tools are the most used in GitHub Actions workflow configuration?
- RQ2. What type of CI/CD tools are rarely used in GitHub Actions workflow configuration?
- RQ3. What type of CI/CD tools are the most diverse in GitHub Actions workflow configuration?
- RQ4. What type of CI/CD tools are least diverse in GitHub Actions workflow configuration?

Our research has two contributions. First, an enhanced understanding of tools that are being used within GitHub Actions workflow configurations is provided. Second, and identification of tool categories that have lacked use within GitHub Actions workflow configurations can be presented.

The rest of this paper is structured as follows. Section 2 reviews related work about GitHub Repository Mining and CI/CD Tools Taxonomy. Section 3 talks about research method utilization in our research. Section 4 presents and discusses our findings. The conclusion is provided in section 5 with answers to our research questions and potential future steps for our work.

II. LITERATURE REVIEW

In this section, existing works that similar to our work on the empirical analysis of CI/CD tools usage in GitHub Actions workflows will be discussed. Various studies that have utilized repository mining techniques to collect valuable insight from repositories and a study used as references for categorization in this work will also be discussed. By using the comparison with other works, our research is positioned in the software repository mining scene for the understanding of GitHub Actions and CI/CD tools.

A. GitHub Repository Mining

There have been several studies in the domain of repository mining. Barros et al. utilize repository mining to develop gthbmining to discover DevOps trends from public repositories [7]. Chatziasimidisy and Stamelos use GitHub rest API for collecting and analyzing GitHub data about users, including repositories and characteristics [8]. Hebig et al. use GitHub mining to get information about UML in many projects [9]. Peters and Zaidman used a repository mining approach in seven open-source systems to investigate the lifespan of code smells and the refactoring behavior of developers [10]. Lima et al. proposed the design suite of metrics for assessing developer contribution in software development projects through repository mining [11]. While the other studies explored various aspects such as DevOps trends, users and their repositories and characteristics, UML information, code smells, and developer contributions, this research focuses on collecting information about GitHub Actions from a dataset of GitHub actions workflow histories through GitHub repository mining. In addition, we conducted secondary data analysis that involves analyzing existing datasets as conducted by Hussain et al. in [12] that focused on predicting student performance through a data mining approach applied to the educational dataset, and Lew et al. in [13] that evaluate the effectiveness of an Adaptive Gaussian Wiener Filter for denoising CT scan images corrupted with Gaussian noise variance from the medical dataset. Our research applied a similar methodology to analyze the GitHub Actions workflow histories dataset to gain insight into the usage of CI/CD tools.

B. CI/CD Taxonomy

The study conducted by Cano et al. in [6] presents an insightful examination about a wide picture of taxonomy of CI/CD tools and framework. The Taxonomy in the study categorizes CI/CD tools and framework into five major categories, including version control management, static code analysis, build automation, test automation, and CI/CD servers. A version control management is a tool that allows all developers to track changes to one or more files over time and to track whether there are conflicts or errors have occurred in the production application, and it can be restored to a previous version without major changes or longer waiting times [14]. Static code analysis is a tool to detect code defects by inspecting the code without executing the code or program [15]. Build automation is a tool to process automating the steps of software build [16]. Test automation is a tool to make the process of automated tests of application faster and efficient [17]. CI/CD Servers is a system that automatically and regularly integrates, creates, and test software, and this system can simplify and automate performing other routine development tasks. This taxonomy serves as a guide to assign tools categories to each GitHub Actions in this research.

Taxonomy type	General characteristics / Tool	Programming language	Tasks				
			Build	Test	Version Control	Deployment	Configuration Management
Versión Control Management	Git	All languages			X		
	Mercurial	All languages			X		
	Apache Subversion	All languages			X		
	CVS	All languages			X		
Static Code Analysis	Veracode	All languages		X			
	CoverityScan	C/C++, C#, Frotan, Go, Java, JavaScript, PHP, Python, Ruby, Scala, Visual Basic and others		X			
	Code Sonar	C, C++, C#, Java and Python.		X			
Build Automation	Apache Ant	Java, C, C++	X				
	Apache Maven	Java	X				
	Make	All languages	X				
	Cheff	Imperative language	X				X
	Puppet	Declarative language	X				X
Test Automation	xUnit	Majority of programming languages		X			
	JUnit	Java		X			
	Selenium	C#, Groovy, Java, Perl, PHP, Python, Ruby and Scala		X			
	Cucumber	All languages		X			
	Apache JMeter	Scripting language		X			
	CodeCover	Java and Cobol		X			
	FrogLogic Coco	C, C++ and C#		X			
CI/CD Servers	Jenkins(CI/CD)	All languages	X	X		X	X
	Travis-CI	All languages	X	X		X	X
	The UrbanCodeBuild	Java	X	X			
	Octopus	PowerShell, Bash, Python, F# or C#				X	

Figure 1. CI/CD Tools Taxonomy [6]

III. RESEARCH METHODOLOGY

This study employed secondary data analysis using a structured approach to examine GitHub Actions tools usage in workflow configuration. Secondary data analysis involves analyzing existing data that was collected for another primary purpose [18]. The structured approach provides a systematic way to extract pertinent information from the secondary data sources to address the research questions, as per depicted in Figure 2.

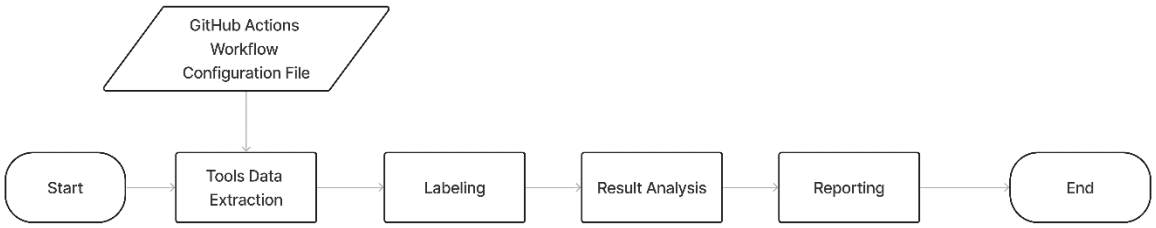

Figure 2. Research Flow

A. Data Sources

The secondary data was obtained from the existing dataset [19]. These sources were selected for two reasons. The first one is their recentness, the dataset contains data as recent as October 2023 and as late of July 2019. Arguably, the data is representative of recent implementation of CI/CD pipeline inside the Github Action workflows, which leads to findings that reflect real practice conducted by software developers. The second reason is their size, which amounts to 32,886 repositories, each containing workflow files totaling 1,526,475 files. The massive number of workflow files should satisfy the required amount needed for a reliable finding. Table 1 summarizes the characteristics of the dataset used.

Table 1. Data Characteristics

Characteristic	**Value**
#GitHub repositories	32886
#repositories containing workflow files	32886
#commit touching workflow files	1004202
Earliest commit date containing a workflow	2019/07/11
Latest commit date containing a workflow	2023/10/12
#workflow histories	160443
#workflow files	1526475
#modification	1342662
#additions	147978
#deletions	35835
#auxiliary files	75234

B. Data Extraction

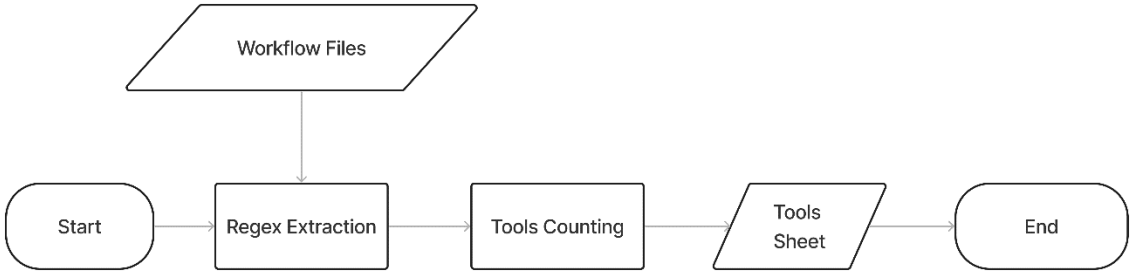

Figure 33. Tools Extraction Flow

To get the tools that developers use in GitHub Actions workflow configuration, the process (refer Figure 3-4) begins by taking one example of the workflow configuration to get the Regular Expression pattern. Once the pattern is obtained, it is applied to each workflow configuration file by iterating over each workflow configuration file. The following Regular Expression pattern is used to identify the usage of the CI/CD tools in GitHub Actions workflow configuration files: `r'uses:\s*(?P<tool>[^\s@]+)'`

This pattern is designed to match and capture the names of the tools used in the GitHub Actions workflow configuration file after the "uses:" keyword, ignoring any leading whitespace. By applying this pattern iteratively to each configuration file, the names of CI/CD tools employed in the GitHub Actions workflows can be extracted. To analyze the prevalence and distribution of these tools, their frequency of occurrence was tallied. Then, only the tools that have usages above 1500 uses is being used as the final data with the assumption the usage below it is not significant. This frequency count provided quantitative insights into the popularity and adoption rates of different CI/CD tools within the GitHub Actions workflows.

With the data successfully extracted and frequencies recorded, the next step involved preparing the data for labeling. This labeling process involved categorizing or tagging each CI/CD tool based on its taxonomy. By assigning labels to the extracted CI/CD tools, the data was structured in a manner conducive to further analysis and interpretation.

```yaml
name: Update Dependencies

on:
  workflow_dispatch:
  schedule:
    # every Monday morning
    - cron:  '0 3 * * 1'

jobs:
  update-dependencies:
    strategy:
      fail-fast: false
      matrix:
        os: ["ubuntu-latest"]
        ruby: ["2.7.1"]
        bundler: ["2.1.4"]
    runs-on: ${{ matrix.os }}
    name: Update Dependencies

    steps:
      - name: Checkout ${{ github.sha   }}
        uses: actions/checkout@v2

      - name: Setup Ruby ${{ matrix.ruby }} using Bundler ${{ matrix.bundler }}
        uses: ruby/setup-ruby@v1
        with:
          ruby-version: ${{ matrix.ruby }}
          bundler: ${{ matrix.bundler }}

      - name: Cache dependencies
        uses: actions/cache@v2
        with:
          path: ~/vendor/bundle
          key: ${{ runner.os }}-${{ matrix.ruby }}-gems-${{ hashFiles('apps/*/Gemfile.lock') }}

      - name: Setup Bundler
        run: |
          bundle config path ~/vendor/bundle
          bundle install

      - name: Run Rake Update
        run: bundle exec rake update

      - name: Create Pull Request
        uses: peter-evans/create-pull-request@v3
        with:
          token: ${{ secrets.OSC_ROBOT_GH_PUB_REPO_TOKEN }}
          commit-message: update dependencies
          committer: OSC ROBOT <osc.robot@gmail.com>
          author: osc-bot <osc.robot@gmail.com>
          delete-branch: true
          title: 'Update Dependencies'
          push-to-fork: osc-bot/ondemand
          branch: osc-bot/dep-updates
          body: |
            Update dependencies generated from `rake update`
```

Figure 4. Workflow File Example

C. Coding

To obtain the final data, the list of CI/CD tools obtained in the previous step of this research was categorized. Our categorization of CI/CD tools involved classifying them into five distinct labels based on their taxonomy: "Build Automation," "Test Automation," "Static Code Analysis," "Version Control Management," and "CI/CD Servers." Each label serves a specific purpose in categorizing tools according to their primary contributions to the software development life cycle. A Version Control System is a tool where the developer can track all the changes in a file or set of files that occur in any stage of development, also it can restore to the previous version of development. Static Code Analysis is a tool to examine code without running the code to find any defect in the code itself. Build Automation is a tool to automate the process of building the software into a release version of the software project, which means this tool automates the step by step of converting a source code into binary code by compiling it. Test Automation is a tool required to do continuous integration, while continuous integration itself requires every commit to be built, and it runs against a set of test cases. Test Automation involves selecting test cases to be run against the new build also setting the environment variable for the test to run. CI/CD servers are a tool that can simplify the execution process or task in development, also these tools usually provide an interactive dashboard where a summary of the tasks that have been run is displayed. By categorizing CI/CD tools into these labels, the aim is to provide a comprehensive understanding of the diverse range of tools available for automating various aspects of the software development life cycle.

To do the labeling process, author one and author two do their labeling, then the result is compared to another. In instances where uncertainty or disagreement arose, a systematic conflict resolution approach is used. Author One and Author Two engaged in a collaborative discussion session to clarify the specific functionalities and features of the tools in question, ensuring alignment with the predefined categorization criteria. If discrepancies persisted, consensus was sought through a thorough review of the tool's documentation capabilities, use cases, and industry standards. Additionally, expert insight from with specialized knowledge in CI/CD practices was leveraged to inform categorization decisions and resolve conflicts effectively.

D. Analysis

In this research, a descriptive statistical analysis was conducted to analyze the distribution of data represented in the dataset. The data consists of CI/CD tools that have been labeled before. The label frequency and percentage are then calculated based on the total number of occurrences. It involves dividing the frequency and multiplying by 100 to obtain the percentage of every category. The mode, indicating the category with the highest frequency, was identified to assess the central tendency of the data. It involves identifying the category that appeared most frequently, providing insight into the predominant aspects of the category within the data. The identification of discrepancies or imbalances in the distribution was examined by analyzing the distribution of frequency across the category. It can highlight categories that exhibited prominence or prevalence compared to others. To understand the overall pattern of category, the shape of data distribution was examined. Any skewer in distribution is considered, therefore indicating the concentration of value within any specific categories and potential deviations from a uniform distribution. To offer a clear intuitive and intuitive visualization of the data, pie chart are used to visualize the data. The size of the slice in the pie chart corresponds to the percentage or frequency of the respective category.

IV. RESULTS AND DISCUSSIONS

This section presents the key findings and analysis derived from our research on CI/CD Tools Usage in GitHub Actions Workflow. The data was collected through the dataset of workflow configuration and extracted the tools. The result is that over 20 thousand tools that have been used in GitHub Action workflow configuration. The results provide insight into the distribution of CI/CD tools, as well as potential threats to the validity of our study.

A. Data extraction result

After the data extraction process, a total over 5 million number of tool usage data was found. Figure 5 shows the results of data extracted from the dataset. It shows the frequency of each individual tool identified inside the dataset. Some tools, such as *actions/checkout* show a high level of usage, while some other tools such as *imjasonh/setup-crane* have a low frequency of usage. As seen in Figure 5, 15,74% amount of data is removed due to extreme low frequency of usage. This removed data as sentenced in methodology with only the tools with usage above 1500 being used for the next process. In total, 203 number of tools from the extraction process were identified to be included in the coding

process. The 203 tools included in the coding process contains to x number of occurrences. One important thing discovered in this process is the huge discrepancy between actions-checkout with the rest of the tools.

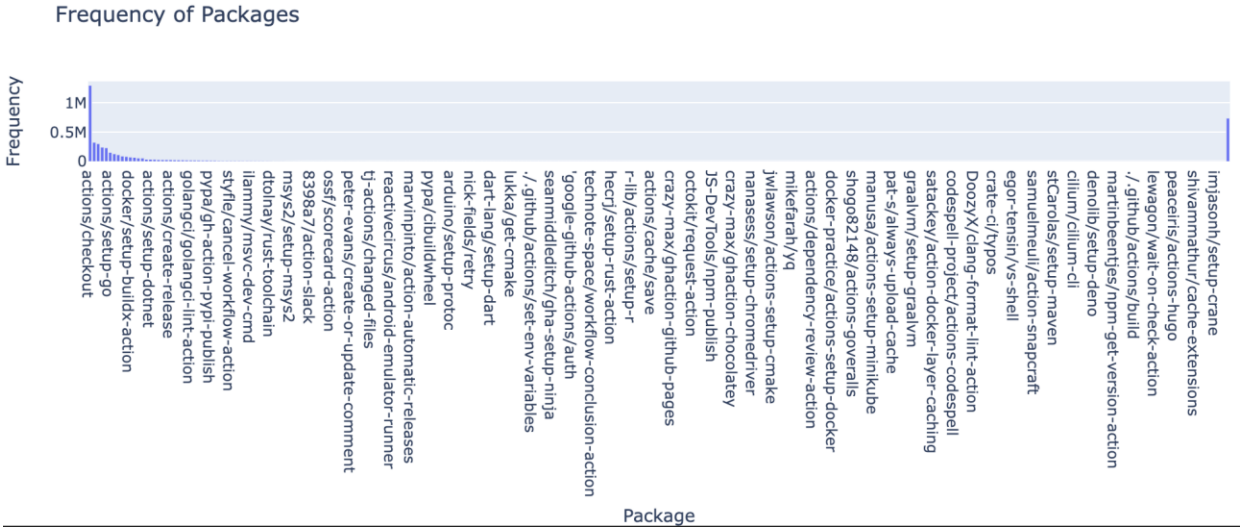

Figure 4. Frequency of CI/CD Tools in GitHub Actions

B. Coding result

Table 2. Samples of Coding Results

No	Packages	Label
1	actions/checkout	Version Control Management
2	actions/upload-artifact	Version Control Management
3	actions/cache	Version Control Management
4	actions/setup-node	Build Automation
5	actions/setup-python	Build Automation
6	actions/setup-go	Build Automation
7	docker/login-action	CI/CD Servers
8	actions/github-script	CI/CD Servers
9	actions/create-release	CI/CD Servers
10	golangci/golangci-lint-action	Static Code Analysis
11	github/codeql-action/autobuild	Static Code Analysis
12	github/codeql-action/upload-sarif	Static Code Analysis
13	coverallsapp/github-action	Test Automation
14	cypress-io/github-action	Test Automation
15	helm/kind-action	Test Automation
16	8398a7/action-slack	Etc
17	rtCamp/action-slack-notify	Etc
18	slackapi/slack-github-action	Etc

Table 2 shows some samples of the coding results based on the CI/CD tools taxonomy. An extra label, "etc" is assigned to tools that don't fit into the taxonomy. A quick analysis of tools that falls into this category shows that it mostly contains automation of notifications, namely on slack. This type of the tools is not identified inside the taxonomy, likely due to the overlook of such process in the common scheme of CI/CD. This raises a question whether communication should be emphasized more in the discussion of implementation of CI/CD pipeline, as it exists inside the existing projects, yet doesn't get mapped into the taxonomy. This guides our future work into exploration of this type of tool and its position inside the CI/CD pipeline. The frequency of each class will be populated to address the research questions that have been defined.

C. Distribution of CI/CD Tools Varieties and Their Usage

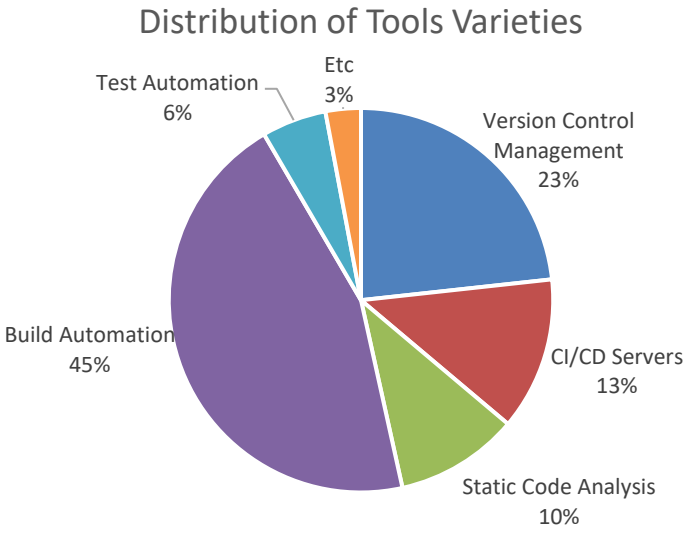

Figure 5. Tools Category Distribution

Figure 6 shows the distribution of varieties of tools of each category. Overall, it illustrates build automation dominating the tools category followed by version control management. Build automation has the highest frequency of 45% indicating a reliance on automation processes for software development. Followed by the version control management category which comes in second as the most used tools category at 23%, signifying the importance of version control systems in effectively managing code repositories. CI/CD server's category have a frequency of 13% indicating a moderate level of usage. However, other categories such as static code analysis and test automation contribute to a smaller extent, 10% and 6% respectively indicating low usage of the tools. Also, there is a 3% usage of tools with categories that are not included in the CI/CD tools taxonomy. We would like to argue that the 3% usage is quite a significant amount tools that not included inside the taxonomy to be used inside the CI/CD pipeline.

The distribution of usage is shown in Figure 7. While both build automation and version control management still tops the distribution, their position flips when compared to the variety shown in Figure 6. It shows that version control management is used most prominently even with less variety compared to build automation tools. As the *actions/checkout* tool comes from this side, it has become the contributor for high usage of version control management tools. The most interesting findings from this data is the extreme low usage of test automation tools, despite high variety of it. This finding raises a question of position of test automation inside GitHub Action as CI/CD pipeline and requires further investigation to answer the question.

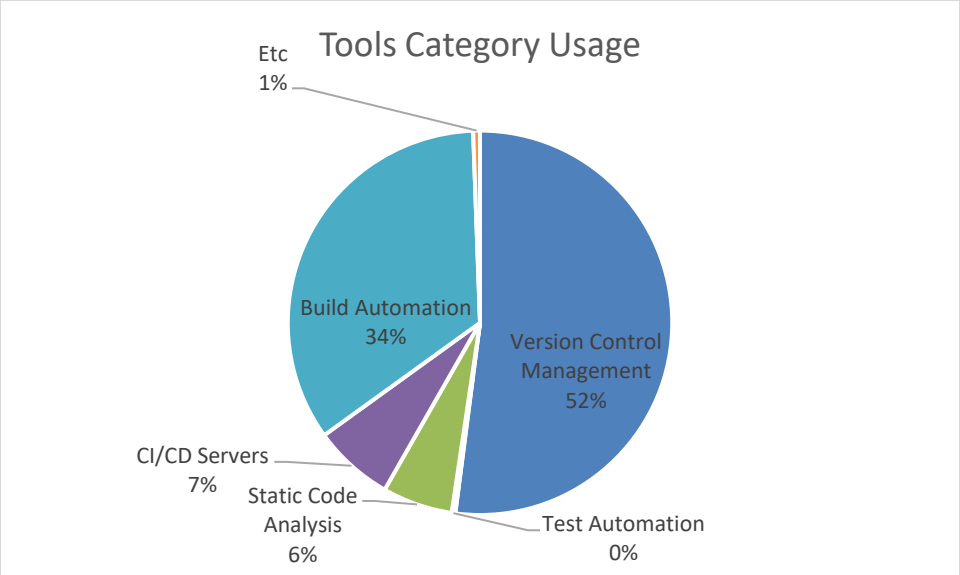

Figure 6. Tools Category Usage

D. Answers to The Research Questions

In answering the research questions, we derived insights from Amazon Web Services' paper on Practicing Continuous Integration and Continuous Delivery (CI/CD) on AWS [20], which provides a valuable perspective on the utilization of CI/CD tools in software development environments.

RQ1. What type of CI/CD tools are the most used in GitHub Actions workflow configuration?

Version control management tools are the most used type of tools in GitHub Actions. It comes with little surprise as the integrated utilization of GitHub as versioning tool likely drive this. Build automation tools comes second which indicates the basic function of automatic version controlling and integration as the main features of CI/CD implemented the most in GitHub Action.

RQ2. What type of CI/CD tools are rarely used in GitHub Actions workflow configuration?

Automatic testing surprisingly comes low in term of usage, despite the high variety. One hypothesis we can raise is that the usage of automatic testing in a single pipeline is still hard to achieve, mostly in level higher than unit testing. More exploration inside the implementation of automatic testing inside the CI/CD pipeline should be conducted to understand the reason behind the phenomena. Moreover, testing practices should be more frequent in conducting CI/CD, as there are numerous mentions of testing in our industry standard. In contrast, when comparing with our industry reference, communication tools that are not included in the taxonomy should have been considered, as there is a need for such tools evident in the industry standard.

RQ3. What type of CI/CD tools are the most diverse in GitHub Actions workflow configuration?

Based on the frequency distribution depicted in Figure 6, the most used category of CI/CD tools in the GitHub Actions workflow configuration is build automation tools. With a frequency of 45%, build automation tools underscore the essential role of automation processes in the GitHub Actions workflow, highlighting the importance of efficient and automated build processes in GitHub Actions workflows. Despite potential duplication, this dominance is nonetheless apparent, emphasizing the presence of build automation in this context. We can argue that the developers are focusing the usage of GitHub Actions mostly for automating build and the tools are highly adapting to the needs, resulting in high diversity of tools inside this type.

RQ4. What type of CI/CD tools are least diverse in GitHub Actions workflow configuration?

Based on the frequency distribution depicted in Figure 6, the rarely used category of CI/CD tools in the GitHub Actions workflow configuration is test automation tools. Tools in this category were used only 6% of the overall frequency, indicating a minimal emphasis on automated testing processes in the GitHub Actions workflow. Possible contributing factors to this low usage include the flexibility of test automation tools across different programming languages or the workflows do not carry out testing.

E. Threats to Validity

While the analysis provides valuable insights into the frequency distribution of CI/CD tool categories within GitHub Actions workflows, there are certain threats to the validity of our findings that must be acknowledged.

- **Secondary Data Quality,** this research relied on a secondary dataset of GitHub repositories that uses GitHub Actions workflows. While efforts were made to ensure the quality and representativeness of the dataset, there is a possibility that the data may not accurately reflect the entire population of GitHub Actions users or may contain inconsistencies or errors, also may not cover the entire usage pattern of GitHub Actions.
- **Lack of Qualitative Insights,** *o*ur research focused on a quantitative analysis of the CI/CD tools used in the GitHub Actions workflow. However, this study lacks qualitative insights into the rationale behind the choice of tools, challenges faced during implementation, and specific use cases or workflows that utilize these tools.
- **Tool Categorization Subjectivity***, t*he categorization of CI/CD tools used in our analysis may be subject to interpretation and potential biases. Different researchers or organizations may categorize tools differently, which could impact the observed distributions and patterns.

To address these threats and strengthen the validity of our findings, we involve a combination of the following approaches. Firstly, we use the GitHub Actions workflow history dataset from an existing dataset. In addition to the quantitative analysis, we conducted focus group discussions to gather qualitative insights regarding the selection of tool categories to enrich our understanding. To reduce subjective categorization, two researchers independently classified the CI/CD tools, resolving any differences through structured discussion, thus ensuring alignment with the predefined categorization and consistency in categorization. Through these steps, we aim to increase the validity of our findings to contribute a more comprehensive understanding of CI/CD tool usage of GitHub Actions workflows.

V. CONCLUSION

In this paper, we have provided an empirical analysis of CI/CD tools usage within GitHub Actions workflows, aiming to understand the frequency distribution of CI/CD tools. After conducting an analysis of more than 30 thousand repositories, we found that the build automation tool is the most dominant category in the GitHub Actions workflow configuration, indicating the important role of automation processes in software development workflows. Also, we identified the test automation tool as a rarely utilized category in GitHub Actions workflow configuration, indicating a low level of usage in automated testing in the workflows.

We have several directions we would like to explore in the future. First, we will revisit the taxonomy and expand it to facilitate the tools that do not fit into the taxonomy, thus enhancing the comprehensiveness of our research, and further informing CI/CD practices. Second, we will explore the cause of the usage frequency of each category of the tools to understand better the nature of CI/CD implementation within Github Actions. In addition, to deepen the understanding of the CI/CD tools usage within GitHub Actions workflows, we propose an exploration using a questionnaire survey to gather insights directly and more deeply.

ACKNOWLEDGEMENT

The research for this paper was not funded.

AUTHOR CONTRIBUTIONS

ARF: Data Visualization, Data Analysis, Writing, Labeling
ATR: Data Filtering, Data Analysis, Writing, Labeling
JHH: Direction, Supervision, Review
MKS: Direction, Supervision, Review

CONFLICT OF INTERESTS

No conflict of interests were disclosed.

REFERENCES

[1] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, "Agile Software Development Methods: Review and Analysis," Sep. 2017.

[2] M. Shahin, M. Ali Babar, and L. Zhu, "Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices," *IEEE Access*, vol. 5, pp. 3909–3943, 2017, doi: 10.1109/ACCESS.2017.2685629.

[3] Y. Ska and R. Publications, "A STUDY AND ANALYSIS OF CONTINUOUS DELIVERY, CONTINUOUS INTEGRATION IN SOFTWARE DEVELOPMENT ENVIRONMENT," *SSRN Electron. J.*, vol. 6, pp. 96–107, Apr. 2019.

[4] P. Rostami Mazrae, T. Mens, M. Golzadeh, and A. Decan, "On the usage, co-usage and migration of CI/CD tools: A qualitative analysis," *Empir. Softw. Eng.*, vol. 28, no. 2, p. 52, Mar. 2023, doi: 10.1007/s10664-022-10285-5.

[5] M. Wessel, T. Mens, A. Decan, and P. R. Mazrae, "The GitHub Development Workflow Automation Ecosystems," in *Software Ecosystems*, Cham: Springer International Publishing, 2023, pp. 183–214.

[6] P. O. Cano, A. M. Mejia, S. De Gyves Avila, G. E. Z. Dominguez, I. S. Moreno, and A. N. Lepe, "A Taxonomy on Continuous Integration and Deployment Tools and Frameworks," 2021, pp. 323–336.

[7] D. D. R. Barros, F. Horita, and D. G. Fantinato, "Data mining tool to discover DevOps trends from public repositories," in *Proceedings of the XXXIV Brazilian Symposium on Software Engineering*, Oct. 2020, pp. 658–663, doi: 10.1145/3422392.3422501.

[8] F. Chatziasimidis and I. Stamelos, "Data collection and analysis of GitHub repositories and users," in *2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA)*, Jul. 2015, pp. 1–6, doi: 10.1109/IISA.2015.7388026.

[9] R. Hebig, T. H. Quang, M. R. V. Chaudron, G. Robles, and M. A. Fernandez, "The quest for open source projects that use UML," in *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*, Oct. 2016, pp. 173–183, doi: 10.1145/2976767.2976778.

[10] R. Peters and A. Zaidman, "Evaluating the Lifespan of Code Smells using Software Repository Mining," in *2012 16th European Conference on Software Maintenance and Reengineering*, Mar. 2012, pp. 411–416, doi: 10.1109/CSMR.2012.79.

[11] J. Lima, C. Treude, F. F. Filho, and U. Kulesza, "Assessing developer contribution with repository mining-based metrics," in *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Sep. 2015, pp. 536–540, doi: 10.1109/ICSM.2015.7332509.

[12] M. M. Hussain, S. Akbar, S. A. Hassan, M. W. Aziz, and F. Urooj, "Prediction of Student's Academic Performance through Data Mining Approach," *J. Informatics Web Eng.*, vol. 3, no. 1, pp. 241–251, Feb. 2024, doi: 10.33093/jiwe.2024.3.1.16.

[13] K. L. Lew, C. Y. Kew, K. S. Sim, and S. C. Tan, "Adaptive Gaussian Wiener Filter for CT-Scan Images with Gaussian Noise Variance," *J. Informatics Web Eng.*, vol. 3, no. 1, pp. 169–181, Feb. 2024, doi: 10.33093/jiwe.2024.3.1.11.

[14] J. Loeliger and M. McCullough, *Version Control with Git: Powerful tools and techniques for collaborative software development*. " O'Reilly Media, Inc.," 2012.

[15] C. Artho and A. Biere, "Combined Static and Dynamic Analysis," *Electr. Notes Theor. Comput. Sci.*, vol. 131, pp. 3–14, 2005, doi: 10.1016/j.entcs.2005.01.018.

[16] M. Prakash, "Software Build Automation Tools a Comparative Study between Maven, Gradle, Bazel and Ant," *Int. J. Softw. Eng. \& Appl. DOI https//doi. org/10.5121/ijsea*.

[17] G. Mohan, *Full Stack Testing*. " O'Reilly Media, Inc.," 2022.

[18] J. Heaton, "Secondary analysis of qualitative data: An overview," *Hist. Soc. Res. Sozialforsch.*, pp. 33–45, 2008.

[19] G. Cardoen, "A dataset of GitHub Actions workflow histories." Zenodo, 2024, doi: 10.5281/ZENODO.10566003.

[20] Amazon Web Services (2023, July 24). *Practicing Continuous Integration and Continuous Delivery on AWS*, AWS Whitepaper. Accessed on: April 30, 2024. [Online]. Available: https://docs.aws.amazon.com/pdfs/whitepapers/latest/practicing-continuous-integration-continuous-delivery/practicing-continuous-integration-continuous-delivery.pdf