Holter: Monitoring Continuous Integration Practices

Jadson Santos
Federal University of Rio Grande do Norte
Natal, Brazil
jadson.santos@ufrn.br

Daniel Alencar da Costa
University of Otago
Dunedin, New Zealand
danielcalencar@otago.ac.nz

Uirá Kulesza
Federal University of Rio Grande do Norte
Natal, Brazil
uira.kulesza@ufrn.br

ABSTRACT

Background: Despite widespread adoption and an extensive body of research evaluating Continuous Integration (CI), there is evidence that not all CI practices are fully adopted. ***Aims:*** To help the full adoption of CI, we present in this paper Holter - a tool that automatically calculates and monitors a suite of metrics associated with CI practices. The tool enables developers to continuously access these metrics and receive alerts regarding the evolution of CI practices. ***Method:*** We illustrate the usage of our tool for monitoring CI practices and compare the Holter monitoring dashboard with those of existing CI services and third-party tools. ***Results:*** Leading CI services and complementary tools still offer basic support for monitoring CI practices. Focusing on monitoring fundamental information related to the performance of the CI pipeline, Holter provides a more comprehensive perspective of CI that cannot be found in other tools. ***Conclusions:*** Holter can be used to monitor and improve CI practices in software development projects. Additionally, it can provide an overview of a project's CI maturity.

Video link: https://doi.org/10.6084/m9.figshare.25880083

KEYWORDS

CI Practices, Monitoring, Case Study, CI Maturity

1 INTRODUCTION

Continuous Integration (CI) is a software development practice where team members integrate their work frequently [7]. The integration process involves an automated build step, which includes compiling the code and running automated tests. This ensures the detection of integration errors as quickly as possible. Despite extensive research into the benefits and costs of Continuous Integration [1–3, 5, 10, 12–14, 17, 21, 22], most of studies focused in mining repositories and study the overall impact of CI. There is a lack of studies specifically addressing the monitoring of CI practices.

Soares et al. [16] conducted a systematic literature review that identified a lack of criteria for defining the effective use of CI. They found that 42.5% of the primary studies did not apply or establish any criteria to determine whether a project uses CI effectively. Among those projects that did use criteria, 56.25% of the studies relied on only one criterion. The most common criterion applied was adopting a CI service, such as Travis CI. In the same direction, other studies [6] have already demonstrated that some CI projects do not follow all CI practices. That study found that: (i) 60% of the projects had infrequent commits; (ii) 85% of the projects had at least one broken build that took a long time to be fixed; and (iii) most projects had builds that executed for more than 10 minutes. They refer to it as a CI Theater [6].

To facilitate the effective usage of CI, our paper aims to provide a tool that offers a more qualitative and comprehensive view of monitoring Continuous Integration (CI) practices. To demonstrate the advantages of our tool compared to existing monitoring tools, we compared our tool with existing CI monitoring dashboard tools using Grey Literature [9]. Researchers can use our tool to obtain an initial and primary overview of a project's CI maturity. In contrast, practitioners can use our tool to monitor and improve CI practices in their projects. The tool initially supports seven CI practices, described next. These practices were chosen because they cover most of the practices defined by Duvall et al [4] and Fowler [7]. As well as they cover most of the practices analyzed by Felidré et al [6], a work that motivated the development of our study about monitoring CI practices [15].

- **Commit Per Weekday/Commit Activity [6]**: Mean of the absolute number of commits according to the weekday of the analyzed period.
- **Coverage [6]**: It measures which parts of a program are executed when running the tests. Represents the percentage of the program covered by tests.
- **Build Duration [6]**: It measures the duration of the build (build finished at timestamp - build started at timestamp).
- **Build Activity/Build-frequency [19]**: It is a unit interval (i.e., a closed interval [0,1]) representing the rate of builds across days, i.e, if builds were made every day in the period of CI, the value would be 1. If builds were made in half of the days, the value would be 0.5. If there were no builds, the value would be 0.
- **Build Health/Build Quality [16]**: It is a unit interval representing the rate of build failures across days. If there were build failures every day, the value would be 0. if there were no build failures, the value would be 1.
- **Time to Fix a Broken Build [6]**: It consists of the median time in a period that builds remained broken. When a build breaks, we compute the time in seconds until the build returns to the "passed" status. If the CI period ends and the build did not return to the "passed" status, we consider the time since it was broken until the end of the period. When a period has no broken builds, the value would be 0.
- **Comments Per Change [18, 20]**: Mean of the number of comments grouped by Merge Request or Issue. This practice seeks to measure the level of communication between the team.

The tool was developed for a multiple case study research [1] involving three Brazilian public sector organizations. Our case study

[1]The study is being submitted for publication at the moment of this writing.

reveals that monitoring CI practices highlights potential issues, provides an overview of CI states, and motivates developers to improve their CI related processes. Developers strongly desire CI monitoring dashboards integrated with popular tools like GitLab, GitHub, and Jenkins, suggesting that seamless integration could optimize the development workflow. This study is part of a doctoral thesis in which we provide strong evidence of the need for monitoring CI practices and demonstrate the usefulness of our tool [15].

2 MONITORING CONTINUOUS INTEGRATION PRACTICES

Elazhary et. al. [5] investigated the extent to which three software organizations implement ten Continuous Integration (CI) practices defined by [7]. They explored the benefits these practices bring and the challenges encountered during their implementation. This inquiry was conducted through a multiple-case study with mixed methods, focusing on three small to medium-sized software-as-a-service organizations. The study revealed that four practices exhibited significant differences in adoption between organizations: (i) For *"Maintain a single source repository"* practice: dealing with merge conflicts had a higher priority than the unclear benefit of using a single source repository; (ii) For *"Make the build self-testing"* practice: the perceived work required for integration tests outweighs the perceived benefits; (iii) For *"Keep the build fast"* practice: doing comprehensive testing outweighs having shorter builds); and (iv) For *"Automate deployment"* practice: enhancing security by enforcing deployment privileges for some organizations outweighs a sense of change ownership.

Our paper aligns with previously cited work. However, we developed a dashboard tool that automates the collection and visualization of the CI practices, including sending alerts via email, facilitating the full adoption of CI. In addition, our tool calculates different CI practices than Elazhary et al.'s [5] work.

3 HOLTER AT A GLANCE

We developed a CI monitoring dashboard tool called Holter. It was designed as an extensible platform to collect and analyze data from a variety of CI tools used in the projects under investigation. The tool is designed to integrate seamlessly with various CI tools, including popular platforms like GitHub, GitLab, Jenkins, CodeCov, and SonarQube, among others. In addition, the tool offers flexibility in automatic data collection. Users can configure the tool to perform scheduled data mining, providing automatic updates on project status. There is also a functionality to send an alert email each time the collection runs.

3.1 Holter Architecture

Holter was developed using Vue.js in the front-end, offering a responsive and intuitive interface for users. On the back-end, the tool uses Kotlin and Spring Boot framework to ensure data collection and management efficiency, reliability, and scalability. To store and manage the collected data, the tool utilizes an embedded H2 database, providing a lightweight and robust solution for data storage that facilitates the implementation of the projects' environment. The tool also allows it to be run inside a Docker container, which makes it possible to install and start monitoring with a simple command. Figure 1 exemplifies the architecture schema of the tool.

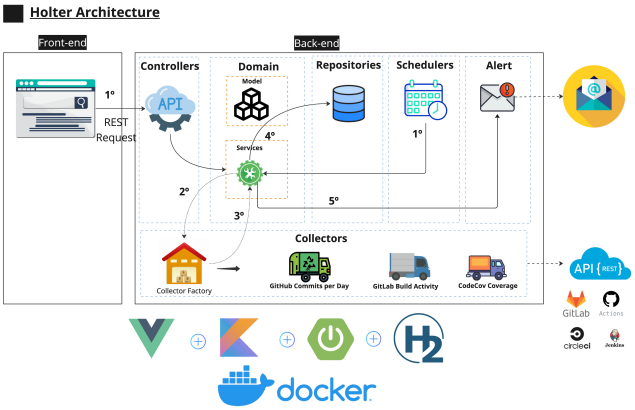

Figure 1: Architecture of CI Practice Monitoring Tool

Figure 1 shows that the actions executed by the tool can be triggered by the Controller layer from user web requests or automatically from the Schedulers layer. These actions are executed by the Services available in the Domain layer. A Service requests the appropriate collector from the Collectors layer using a Collectors Factory object. A Collector is an entity that collects a specific measurement associated with a CI practice (such as Build Duration, Coverage, Commit Activity, etc) in a particular repository (such as GitHub, GitLab, Jenkins, SonarQube, etc). Given the variety of repositories where information about a project can be stored, the tool was developed to support the implementation of new Collectors for CI practices in specific Repositories without having to make major changes to other parts of the tool. The "Collector part" uses the Factory and Template Method Design Patterns, as shown in the diagram in Figure 2. The abstract class defines a base algorithm to collect measurements associated with a CI practice in the `collect(p: Project)` method and an abstract method `calcMetricValue(p: Project) : BigDecimal`.

A concrete collector class should then extend the abstract class `Collector` and implement `calcMetricValue(p: Project) : BigDecimal` method to provide a way to connect to a repository and calculate the CI measurement, returning a `BigDecimal` that represents the value associated with a CI practice that will be stored in the history for a specific project. One should implement a new concrete `Collector` class to support measuring new CI practices. All the logic and infrastructure necessary to collect and calculate measurements about CI practices must be contained within a concrete Collector. The tool supports 17 standard measurements (7 CI + 4 DORA + 6 Basics) and integrates with four repository types (GitHub/GHActions, GitLab, SonarQube, and CodeCov). Regardless, our work focuses specifically on measurements associated with CI practices (e.g., as opposed to DORA). Holter collects metrics from various CI tools using REST APIs[2]

[2]https://www.redhat.com/en/topics/api/what-is-a-rest-api

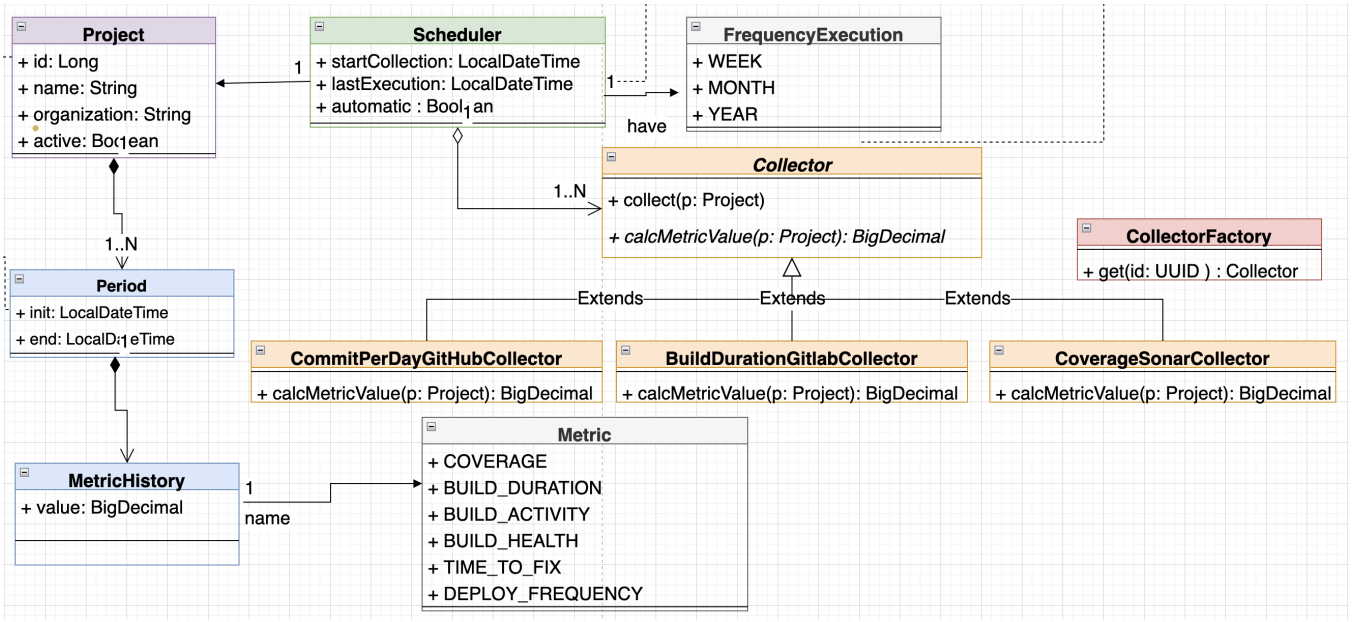

Figure 2: Class Diagram of CI Practice Monitoring Tool

In this way, Holter handles variations in CI environments, programming languages, and build systems by using a flexible architecture that supports the implementation of new Collectors for different CI practices and repositories. The tool seeks to be as adaptable as possible; all practices are configurable, and the user can choose which practices will be analyzed for a specific project.

3.2 Holter Scheduler

Holter allows you to make a series of configurations in the way it collects measurements for CI practices. As illustrated in Figure 3, the tool has the concept of Scheduler. Each project has its own Scheduler. Thus, it is possible to configure the collection execution period between 3 possible values: WEEKLY, MONTHLY, and ANNUAL. It is also possible to configure the date when the CI metrics will start to be collected and whether this collection will be carried out automatically or not.

A Scheduler has association `one to many` with Collectors. It is possible, at any time, to choose which practices are monitored for a given project by adding or removing Collectors associated with the project Scheduler.

3.3 Holter Dashboard

Holter has a dashboard that allows the project team to monitor the evolution of CI practices, Figure 4 shows a part of the tool's dashboard screen.

Each box at the bottom of the Figure 4 corresponds to a CI practice. Inside the box, the participants can view the current measurement value associated with the practice (the average of all entries collected in the last period) and the project reference value for that practice. The reference values are used as parameters by the tool to issue alerts if any practice measurement do not comply with the expected value. The reference values are editable and can be assigned values that best suit the project context. In the upper part of Figure 4, a status bar is shown, indicating the number of practices that have reached the reference value, serving as a primitive gauge for the project's level of CI maturity.

Inside each box, there is also a button, which, when clicked, shows to the participants a modal panel with details of the practice measurement's evolution. Figure 5 shows an example of the Build Duration CI practice. The modal panel shows the complete graph of the evolution of the CI practice measurement during the selected period, the description of the practice measurement, the formula used for its calculation, the unit in which it is calculated, the current value, the mean value for all period, some categories where the practice measurement is classified, and the last five values of the practice measurement. Furthermore, the tool calculates the practice measurement trend (a delta that indicates in percentage how much the value has grown or decreased in relation to the previous period[3]), an indication of whether the practice measurement values inclined to increase or decrease over time, and the stability of the practice measurement in three possible values, represented by icons: SUNNY, CLOUDY, or RAINING.

To calculate the CI practice measurement trend, we use the formula described in the CircleCi CI service documentation[4] : *"Trends displayed in the CircleCI UI are calculated as 100 * (current value - previous value) / prior-value"*. For the practice stability, we compare Mean Absolute Deviation [5] values ($MAD = \frac{\sum |x_i - \bar{x}|}{n}$) with quartiles of measurement values to classify its stability. When the MAD value is below the first quartile, it is considered low (SUNNY). Between the second and third quartiles is indicated as medium (CLOUDY).

[3]https://circleci.com/docs/insights-glossary/#trends
[4]https://circleci.com/docs/insights-glossary/
[5]https://www.khanacademy.org/math/statistics-probability/summarizing-quantitative-data/other-measures-of-spread/a/mean-absolute-deviation-mad-review

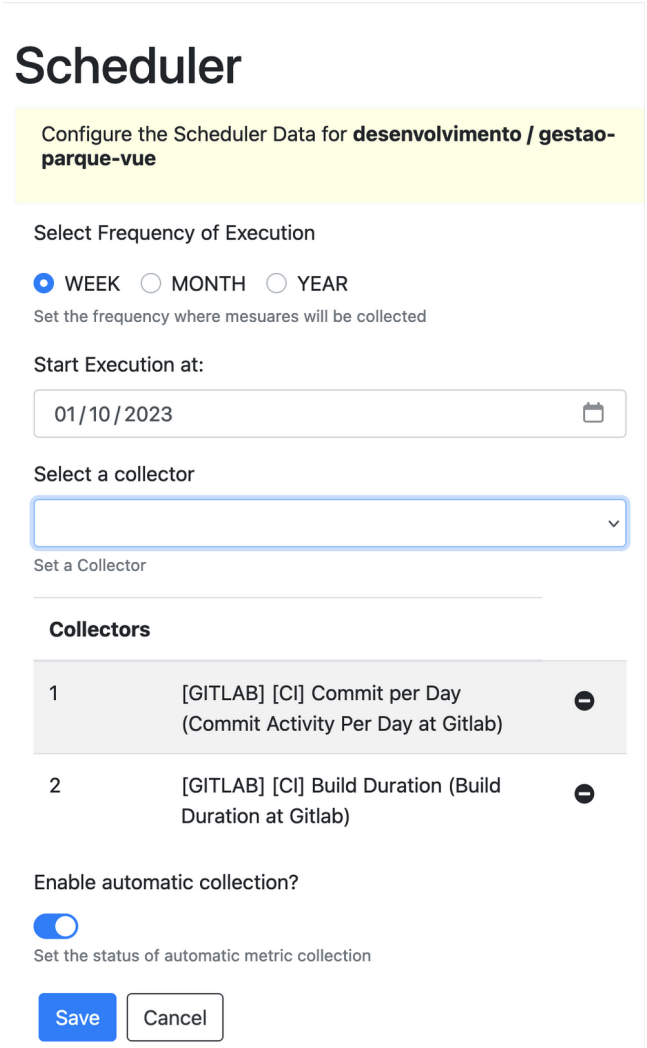

Figure 3: Scheduling CI practice measurement collection

Finally, above the third quartile is considered high (RAINING). A lower MAD suggests higher data stability. This association with the weather is inspired by how Jenkins shows the status of building jobs [6]. This approach can provide a contextually relevant way to understand the variability of our data.

In Figure 5, we notice that the Build Duration practice measurement has increased greatly in the latest collected period, more than 5000% (using the CircleCI formula). As for specific Build Duration measurements, given that an increase is considered bad, the tool highlights this value in red. As it constantly fluctuates, one week it is low, the next it increases; stability is shown with the CLOUD icon.

[6]https://www.jenkins.io/doc/book/blueocean/dashboard/#pipeline-health

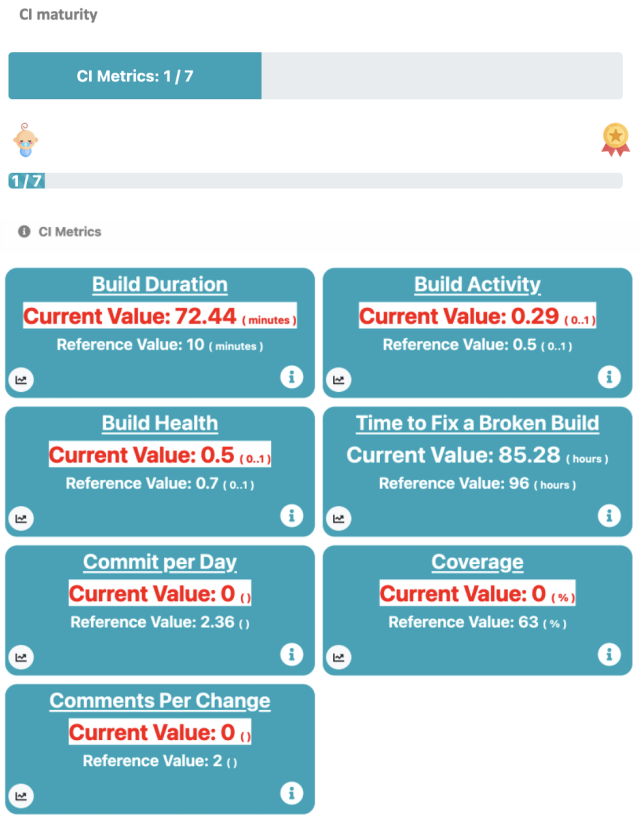

Figure 4: CI Practice Monitoring Tool Dashboard

4 COMPARATIVE ANALYSIS WITH OTHER CI TOOLS

We compared our tool with existing CI monitoring dashboard tools using Grey Literature [9]. We conducted an analysis using Grey Literature, as we believe that the official documentation of CI tools offers a more comprehensive description of their monitoring capabilities than academic papers.

As an initial step, we conduct a search on the official documentation of the most popular CI services, as this is the most intuitive thing a developer does when she/he wants to learn about these CI services. In our analysis, we focused on the seven most popular CI services, covering 99% of all observed instances of CI usage as described by [11]: TravisCI, GHActions, CircleCI, AppVeyor, Azure, GitLab and Jenkins.

According to [8], these documents can be classified as "2nd tier Grey Literature", which provides "Moderate credibility/Moderate outlet control". Using a specific data source also allows the study to be reproducible. Subsequently, we meticulously examined the documentation and selected documents that explicitly showed images of CI-related dashboards. Next, we included the analysis of third-party tools, as these tools may have features not present in the main CI services. We collect all references to third-party tools cited in the official CI services documentation obtained from the previous step. Then, similar to the first step, we provided URLs of documentation

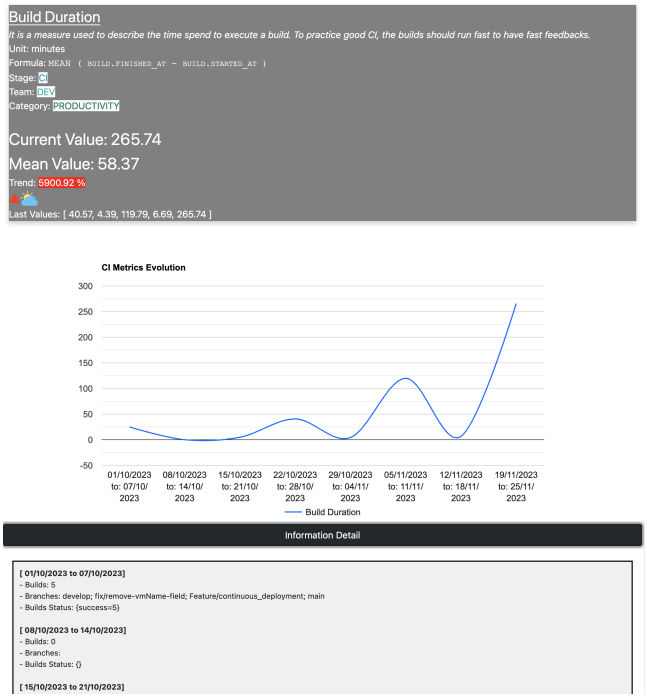

Figure 5: CI Practice's Evolution Details Panel

for third-party tools. We discover monitoring dashboard evidence in 14 documents, with 11 being primary documents and 3 being snowballing third-party tools documents.

As a complementary step, given the low number of results returned by our first search criteria, we use the query string *" (Dashboard) AND ("continuous integration" OR "CI") "* on google.com, using incognito windows to collect official documentation from third-party tools. We then performed a Grey Literature review within the official tools documentation that appeared on the first 20 research results. We increased our list to 22 documents by adding more than 8 documents obtained using query string search.

4.1 Comparative Analysis Results

This subsection describes our main findings about state-of-the-art CI monitoring practices dashboard tools, comparing them to Holter.

Most of the dashboards found in our comparative analysis provide fairly rudimentary and limited metrics for accurately assessing CI practices. For example, the Sumo Logic Dashboard from Circle CI, shown in Figure 6, offers an overview of the health and usage of a project's build processes, including time-series and aggregated data on credit usage, success rates, and pipeline duration. Similarly, tools like the Build Monitor Dashboard, CloudBees CI dashboard, and Jenkins - Blue Ocean Dashboard follow the same approach.

Other tools like Gitlab and Build Pulse do not focus on CI practices. They prioritize monitoring metrics directly related to

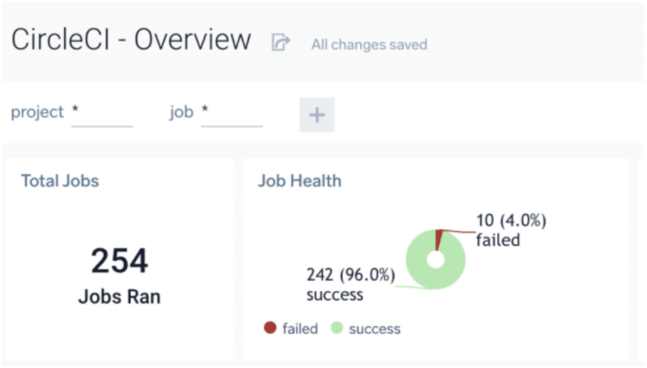

Figure 6: Circle CI - Partial Sumo Logic Dashboard

Continuous Delivery (CD) or DevOps, such as the DORA metrics[7]. Figure 7 shows partially the " GitLab - Value Streams Dashboard", which is a customizable tool used to identify trends, patterns, and opportunities for digital transformation improvements. The Value Streams Dashboard includes several panels that visualize the following metrics: DORA metrics, Value Stream Analytics (VSA) - flow metrics, Vulnerabilities metrics, and GitLab Duo AI Code Suggestions.

Figure 7: Gitlab - Partial Value Streams Dashboard

The "AWS - Code Changes Dashboard" provides a different perspective on the developer area. It displays the number of code changes made by the author and repository, allowing users to answer questions such as who makes the most code changes and which repositories are the most active over time. This perspective can help provide insights into CI practices related to developers' tasks not covered by other dashboard tools.

Figure 9 shows components of the "DataDog CI Visibility Dashboard", which integrates information about CI tests and pipeline results, along with data on CI performance (error rates or long build duration), trends, and reliability. The DataDog dashboard also allows you to monitor your tests across all builds, identify common errors, and visualize test performance over time.

Figure 10 gives an overview of CI tool areas related to support for dashboard tools. We observed that most tools focus on monitoring

[7]https://cloud.google.com/blog/products/devops-sre/using-the-four-keys-to-measure-your-devops-performance

Figure 8: AWS - Partial Code Changes dashboard

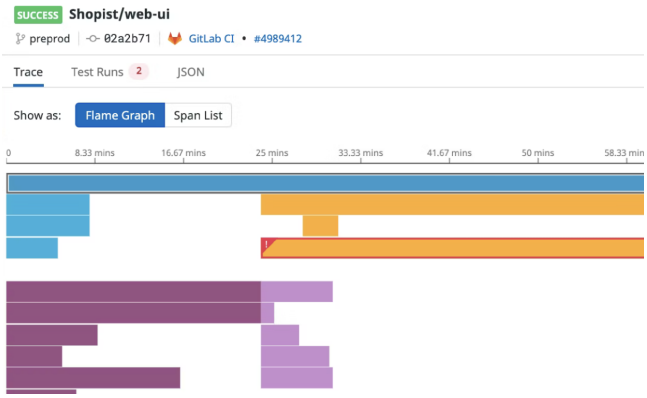

Figure 9: Data Dog - Partial CI Visibility Dashboard

CI pipeline aspects, while some concentrate on aspects related to source code, such as test writing and coverage. Few tools address both of these practice areas. However, there is a notable absence of tools specifically targeting monitoring practices related to developers, such as commit frequency. The Holter tool aims to cover all these areas and provide a broader view of monitoring CI practices. Holter monitors pipeline-related practices like Build Duration and Build Health, source code-related practices like Coverage, and developer-related practices like Commit Activity. Integrating these areas can offer a broader and more comprehensive view of CI.

5 LIMITATIONS

Holter currently implements 7 CI practices. While we aim to provide diversity to address different areas of CI, it is important to notice that the practices supported by Holter may not cover all aspects of CI. The inclusion criteria for dashboard tools used in our Comparative Analysis may not include all existing CI monitoring tools, and furthermore, some relevant documents may not have been included in the review. These factors can affect the robustness of our evaluation.

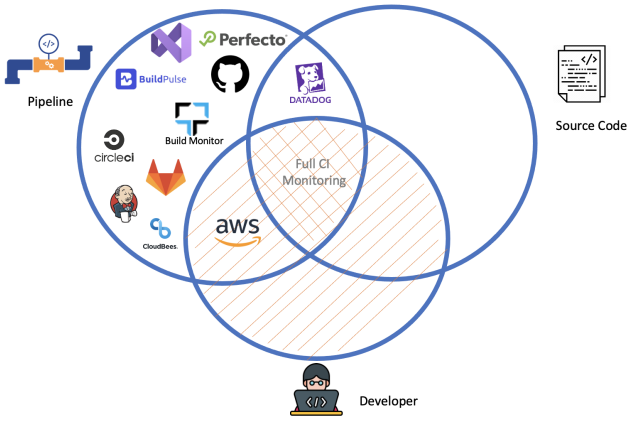

Figure 10: CI Areas according to monitoring dashboard

6 CONCLUSION

In this paper, we introduced the Holter - a CI practices monitoring tool. The tool provides a dashboard where users can monitor the evolution of seven CI practices. It automatically executes data collection and can send an alert email each time the collection runs.

We plan to integrate Holter with Continuous Integration (CI) services in future work. For example, creating Actions for GitHub Actions to enhance CI practices for a wider range of projects, thus facilitating the adoption of CI practice monitoring.

AVAILABILITY OF ARTIFACTS

We have made available the documentation generated by this paper, the Grey Literature review URLs, and the tool source code at: https://doi.org/10.6084/m9.figshare.25883674. The tool is available under the MIT License[8]. URL of the tool code repository [9].

ACKNOWLEDGMENTS

This work is partially supported by INES (www.ines.org.br), CNPq grant 465614/2014-0, CAPES grant 88887.136410/2017-00, and FACEPE grants APQ-0399-1.03/17 and PRONEX APQ/0388-1.03/14.

REFERENCES

[1] Thomas Bach, Artur Andrzejak, Ralf Pannemans, and David Lo. 2017. The Impact of Coverage on Bug Density in a Large Industrial Software Project. In *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 307–313. https://doi.org/10.1109/ESEM.2017.44

[2] João Helis Bernardo, Daniel Alencar da Costa, and Uirá Kulesza. 2018. Studying the Impact of Adopting Continuous Integration on the Delivery Time of Pull Requests. In *Proceedings of the 15th International Conference on Mining Software Repositories* (Gothenburg, Sweden) *(MSR '18)*. Association for Computing Machinery, New York, NY, USA, 131–141. https://doi.org/10.1145/3196398.3196421

[3] Nathan Cassee, Bogdan Vasilescu, and Alexander Serebrenik. 2020. The Silent Helper: The Impact of Continuous Integration on Code Reviews. In *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 423–434. https://doi.org/10.1109/SANER48275.2020.9054818

[8] https://opensource.org/license/mit
[9] https://github.com/jadsonjs/holter-devops

[4] Paul Duvall, Stephen M. Matyas, and Andrew Glover. 2007. *Continuous Integration: Improving Software Quality and Reducing Risk (The Addison-Wesley Signature Series)*. Addison-Wesley Professional.

[5] Omar Elazhary, Colin Werner, Ze Shi Li, Derek Lowlind, Neil A Ernst, and Margaret-Anne Storey. 2021. Uncovering the benefits and challenges of continuous integration practices. *IEEE Transactions on Software Engineering* 48, 7 (2021), 2570–2583.

[6] Wagner Felidré, Leonardo Furtado, Daniel A. da Costa, Bruno Cartaxo, and Gustavo Pinto. 2019. Continuous Integration Theater. In *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 1–10. https://doi.org/10.1109/ESEM.2019.8870152

[7] Martin Fowler. 2006. *Continuous Integration*. Retrieved november 24, 2021 from https://martinfowler.com/articles/continuousIntegration.html

[8] Vahid Garousi, Michael Felderer, and Mika V Mäntylä. 2019. Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Information and software technology* 106 (2019), 101–121.

[9] Vahid Garousi, Michael Felderer, Mika V Mäntylä, and Austen Rainer. 2020. Benefitting from the grey literature in software engineering research. In *Contemporary Empirical Methods in Software Engineering*. Springer, 385–413.

[10] Taher Ahmed Ghaleb, Daniel Alencar Da Costa, and Ying Zou. 2019. An empirical study of the long duration of continuous integration builds. *Empirical Software Engineering* 24 (2019), 2102–2139.

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

[12] Michael Hilton, Nicholas Nelson, Timothy Tunnell, Darko Marinov, and Danny Dig. 2017. Trade-Offs in Continuous Integration: Assurance, Security, and Flexibility. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3106237.3106270

[13] Michael Hilton, Timothy Tunnell, Kai Huang, Darko Marinov, and Danny Dig. 2016. Usage, costs, and benefits of continuous integration in open-source projects. In *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 426–437.

[14] Eero Laukkanen, Maria Paasivaara, and Teemu Arvonen. 2015. Stakeholder Perceptions of the Adoption of Continuous Integration – A Case Study. In *2015 Agile Conference*. 11–20. https://doi.org/10.1109/Agile.2015.15

[15] Jadson Santos. 2024. *A Deep Dive into Continuous Integration Monitoring Practices*. Ph.D. Dissertation. PPGSC/UFRN.

[16] Eliezio Soares, Gustavo Sizílio, Jadson Santos, Daniel Alencar da Costa, and Uirá Kulesza. 2021. The Effects of Continuous Integration on Software Development: a Systematic Literature Review. *CoRR* abs/2103.05451 (2021). arXiv:2103.05451 https://arxiv.org/abs/2103.05451

[17] Daniel Ståhl and Jan Bosch. 2013. Experienced benefits of continuous integration in industry software product development: A case study. In *The 12th iasted international conference on software engineering,(innsbruck, austria, 2013)*. 736–743.

[18] Daniel Ståhl and Jan Bosch. 2013. Experienced benefits of continuous integration in industry software product development: A case study. In *The 12th iasted international conference on software engineering,(innsbruck, austria, 2013)*. 736–743.

[19] Daniel Ståhl and Jan Bosch. 2014. Modeling continuous integration practice differences in industry software development. *Journal of Systems and Software* 87 (2014), 48–59. https://doi.org/10.1016/j.jss.2013.08.032

[20] Christopher Thompson. 2017. Large-Scale Analysis of Modern Code Review Practices and Software Security in Open Source Software. In *Proceedings of the 13th International Conference on Predictive Models and Data Analytics in Software Engineering (2017)*. 83–92.

[21] Bogdan Vasilescu, Yue Yu, Huaimin Wang, Premkumar Devanbu, and Vladimir Filkov. 2015. Quality and Productivity Outcomes Relating to Continuous Integration in GitHub. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering* (Bergamo, Italy) *(ESEC/FSE 2015)*. Association for Computing Machinery, New York, NY, USA, 805–816. https://doi.org/10.1145/2786805.2786850

[22] David Gray Widder, Michael Hilton, Christian Kästner, and Bogdan Vasilescu. 2019. A Conceptual Replication of Continuous Integration Pain Points in the Context of Travis CI. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3338906.3338922