MicroSensor: Towards an Extensible Tool for the Static Analysis of Microservices Systems in Continuous Integration

Edson Soares
Instituto Atlântico
State University of Ceará
Fortaleza, Brazil
edson_soares@atlantico.com.br

Matheus Paixao
State University of Ceará
Fortaleza, Brazil
matheus.paixao@uece.br

Allysson Allex Araújo
Federal University of Cariri
Juazeiro do Norte, Brazil
allysson.araujo@ufca.edu.br

ABSTRACT

In the context of modern Continuous Integration (CI) practices, static analysis sits at the core, being employed in the identification of defects, compliance with coding styles, automated documentation, and many other aspects of software development. However, the availability of ready-to-use static analyzers for microservices systems and focused on developer experience is still scarce. Current state-of-the-art tools are not suited for a CI environment, being difficult to setup and providing limited data visualization. To address this gap, we introduce our software product under progress called μSensor: a new open-source tool to facilitate the integration of microservice-based static analyzers into CI pipelines as modules with minimal setup, where the resulting reports can be viewed on a webpage. By doing so, μSensor contributes to data visualization and enhances the developer experience.

CCS CONCEPTS

• **Software and its engineering → Software maintenance tools**.

KEYWORDS

Microservices; Continuous Integration; Static Analysis

ACM Reference Format:
Edson Soares, Matheus Paixao, and Allysson Allex Araújo. 2024. MicroSensor: Towards an Extensible Tool for the Static Analysis of Microservices Systems in Continuous Integration. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering (FSE Companion '24), July 15–19, 2024, Porto de Galinhas, Brazil.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3663529.3663874

1 INTRODUCTION

With a global market size valued at 2,073 million in 2018, and projected to reach 8,073 million by 2026 [4], the microservices architecture stands out in the software engineering landscape. Microservices projects require a great deal of supporting tools, making use of both Continous Integration (CI) and Continuous Delivery (CD) approaches in their development lifecycle.

CI and static analysis walk hand in hand in modern software development. By leveraging CI's infrastructure to easily build a project's source code, static analyzers can be integrated to evaluate the project's quality and provide automated feedback for developers. Although the modernization of CI and static analysis brought to life some notable open-source and commercial tools such as SonarQube, Semgrep, Checkmarx, Snyk, CodeGuru, and Coverity, none deal with microservices-specific smells.

In this context, the development of state-of-the-art static analysis tools has flourished, where the microservices domain has not been left behind. One can find static analysis tools for the identification of microservice-specific code smells [5], maintaining microservice-based systems [1], and migration towards a microservice architecture [3]. Nevertheless, the current state-of-the-art static analyzers for microservices are not yet ready to be integrated into a CI environment.

To address this gap, we propose μSensor[1], a new open-source tool aimed at facilitating the integration of microservices-related static analyzers into CI pipelines. μSensor runs on the GitHub Actions[2] platform. GitHub Actions allow the automation of tasks based on various triggers (*e.g.*, commits, pull requests, issues, etc.), making it easier for developers to automatically build, test, and deploy software projects. In addition, GitHub Actions does not require the installation of any additional software and can be integrated into several other resources in the GitHub ecosystem. By leveraging GitHub Actions, μSensor can be integrated into the CI pipeline of any project hosted on GitHub without any local installation.

To showcase our tool's architecture and features, we first incorporated a state-of-the-art microservices code smells detector [5] into μSensor. Next, we selected 9 GitHub projects and integrated μSensor into the CI pipelines of all selected projects.

2 THE μSENSOR TOOL

μSensor serves as a platform to make static analysis tools readily available for CI pipelines. By incorporating a new (or existing) static analyzer into μSensor, the approach will be able to enjoy all of μSensor's existing CI-related infrastructure, such as building capabilities, trigger configuration, self-hosting, and web-based visualization. μSensor can bridge the gap between a static analyzer prototype and its release to real-world practitioners in a modern CI infrastructure. μSensor's architecture is organized into two main components: the `Scanner` and the `Renderer`.

[1]https://github.com/microsensorproject/microsensor
[2]https://github.com/features/actions

2.1 Static Analysis `Scanner`

The responsibilities of this component are to allow multiple analyzers to be plugged in as modules, execute all of μSensor's static analyzers, and collect their results. To tackle the difficulty of coordinating the proper execution of several static analyzers, each having its own running environment, μSensor makes use of container-based solutions. For each static analyzer in μSensor, the `Scanner` has a Docker image with the necessary running environment for the static analyzer to properly execute and generate its results.

To incorporate a static analyzer into μSensor's `Scanner`, one needs to: (i) create a Docker image with the running (and possibly building) environments for the analyzer, and (ii) write a few lines of shell script within the `Scanner`'s code to execute the analyzer in the `Scanner`'s pipeline and export the results.

2.2 Visualization `Renderer`

The responsibilities of this component are to process the outputs of each static analyzer and provide a visualization of the results. To address this goal, μSensor uses micro frontends [2] and leverages resources from the GitHub ecosystem. After the static analysis results are made available by the `Scanner`, the `Renderer` will process the outputs and automatically generate a web app. Next, the web app is deployed into GitHub Pages. To create the results' visualization for a new (or existing) static analyzer within the `Renderer`, one needs to provide a micro frontend that processes the results and creates the visualization. Each micro frontend is an independent web project within the `Renderer`. From a UI perspective, the website will have different tabs for accessing the visualization of each static analyzer's results.

Figure 1 summarizes μSensor's execution flow. After the workflow event is triggered, μSensor will check out the latest version of the project's source code. Next, the `Scanner` will create Docker containers for each static analyzer, where the analyses will be executed and the results will be exported. The `Renderer` will then process the static analysis results, create a web visualization, and automatically publish it into the repository's GitHub Pages.

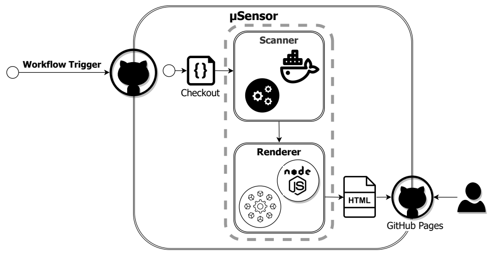

Figure 1: μSensor Architecture and Execution Flow

3 EVALUATION

3.1 Integrating a Static Analyzer into μSensor

To showcase μSensor's ability to incorporate static analyzers into its architecture, we chose MSANose [5], a state-of-the-art code smells detector for microservices systems. Even though MSANose is capable of detecting a wide array of microservices code smells, it presents a few shortcomings regarding its developer experience in a CI environment. MSANose requires a local Java and Spring Boot installation for its execution. In addition, MSANose results are presented as a JSON file, lacking any data visualization. To incorporate MSANose into μSensor, we followed the instructions depicted in Section 2.

3.2 Integrating μSensor into CI pipelines

Due to MSANose's limitations, we selected Java-based systems based on Spring Boot to develop the microservices. To this end, we explored GitHub using keywords such as '`microservices`' and '`spring boot`'. We made sure that all projects we looked at had a reasonable number of stars as a proxy for non-trivial projects. As a result, we selected nine projects, where we integrated μSensor to the CI pipelines to each of them, respectively: Apollo (26.8k stars), Conductor (4.5k stars), Spring Cloud Alibaba (24.1k stars), Spring Consul (744 stars), Micro-Company (333 stars), Sitewhere (889 stars), MyCollab (1.1k stars) for project and document management, Abixen (640 stars), and Genie (1.6k stars). For each one of these projects we have a specific μSensor report (see this example[3] for Apollo, in particular). Due to space limitations, we can't display all the experiment information. However, all other reports, including the GitHub links and project descriptions, are open and available in our repository[4] as evidence concerning the viability of our proposal.

4 FINAL REMARKS AND NEXT STEPS

This paper aims to introduce μSensor, an open-source tool to facilitate the integration of microservices-specific static analyzers into CI pipelines. Our future work includes improving integration with multiple state-of-the-art tools with the adoption of integration standard formats (e.g., GitHub SARIF), expanding support to other CI platforms, and offering customization for practitioners.

ACKNOWLEDGMENTS

This work received funding from CNPq - Brazil, Universal grant 404406/2023-8.

REFERENCES

[1] Giona Granchelli, Mario Cardarelli, Paolo Di Francesco, Ivano Malavolta, Ludovico Iovino, and Amleto Di Salle. 2017. Microart: A software architecture recovery tool for maintaining microservice-based systems. In *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*. IEEE, 298–302.
[2] Severi Peltonen, Luca Mezzalira, and Davide Taibi. 2021. Motivations, benefits, and issues for adopting Micro-Frontends: A Multivocal Literature Review. *Information and Software Technology* 136, July 2020 (aug 2021), 106571.
[3] Ilaria Pigazzini, Francesca Arcelli Fontana, and Andrea Maggioni. 2019. Tool support for the migration to microservice architecture: An industrial case study. In *European Conference on Software Architecture*. Springer, 247–263.
[4] Vineet Kumar Rachita Rake. 2020. Microservices Architecture Market Statistics - 2026. https://www.alliedmarketresearch.com/microservices-architecture-market. Accessed: 2022-05-24.
[5] Andrew Walker, Dipta Das, and Tomas Cerny. 2020. Automated Code-Smell Detection in Microservices Through Static Analysis: A Case Study. *Applied Sciences* 10, 21 (2020). https://doi.org/10.3390/app10217800

Received 2024-03-15; accepted 2024-04-26

[3] https://microsensorproject.github.io/apollo
[4] https://github.com/microsensorproject/microsensor/blob/development/evaluation.md