Given $2n$ Eyeballs, All Quality Flaws Are Shallow

Panagiotis Gkikopoulos
pang@zhaw.ch
Zurich University of Applied Sciences
Winterthur, Switzerland

Josef Spillner
josef.spillner@zhaw.ch
Zurich University of Applied Sciences
Winterthur, Switzerland

Cristian Mateos
cristian.mateos@isistan.unicen.edu.ar
ISISTAN Research Institute
Tandil, Argentina

Alfredo Teyseyre
alfredo.teyseyre@isistan.unicen.edu.ar
ISISTAN Research Institute
Tandil, Argentina

Abstract

We demonstrate the capabilities of the Microservice Artefact Observatory (MAO), a federated software quality assessment middleware. MAO's extensible assessment tools continuously scan for quality flaws, defects and inconsistencies in microservice artefacts and observe runtime behaviour. The federation reduces bias and also increases the resilience and overcomes per-site failures, leading to a single, merged timeline of software quality. Already serving concurrently by $n = 3$ observant operators in Argentina and Switzerland, the federation is designed to become a community-wide consensus voting-based ground truth repository with query interfaces for large-scale software quality and evolution insights. These insights can be exploited for excluding buggy software before or after deployment, for optimised resource allocation, and further software management tasks.

ACM Reference Format:
Panagiotis Gkikopoulos, Josef Spillner, Cristian Mateos, and Alfredo Teyseyre. 2020. Given $2n$ Eyeballs, All Quality Flaws Are Shallow. In *21st International Middleware Conference Demos and Posters (Middleware '20 Demos and Posters), December 7–11, 2020, Delft, Netherlands.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3429358.3429371

1 Problem Description and Goals

Understanding complex software, especially composite cloud applications, means acquiring data via static analysis, testing, monitoring, observation and chaos engineering. As this enables data-driven software lifecycle management, engineers should be able to rely on verified software-related data (e.g. is a component affected by a vulnerability) as ground truth. This allows for informed and automated decision making (accept, reject, notify, commit, rollback) based on trusted and reproducible metrics delivered on demand, instead of spending their time with brittle experiments and finding out failures after the fact. The goal of a *federated* software artefact quality assessment framework is therefore to offer *reliable quality metrics as a service* in the context of testing and quality assessment. This service to highlight quality issues (and security/consistency flaws, depending on metric-producing *assessment tools*) is not performed by an untrusted and potentially biased vendor, but by independent operators (researchers, experts, companies) who are implicitly connected in a *web of trust* with software framework support by multiple *nodes* running the federated tools directed by *orchestrators*. For discrepancies among observations, confirmed and resolved with a variation of the well-known resolution *all bugs are shallow* by attentive eye pairs, *consensus voting* is employed to resolve any diverging assessment automatically, and in cases of persisting differences, researchers or citizen scientists such as developers from the respective communities inspect the deltas manually and make a decision on merging results. The outcome is *decentralised quantified knowledge* about software artefacts in the form of *ground truth* which can be exploited to determine software compositions, builds, deployments and other lifecycle matters through appropriate query interfaces. Fig. 1 summarises the envisioned environment and interfaces of the framework.

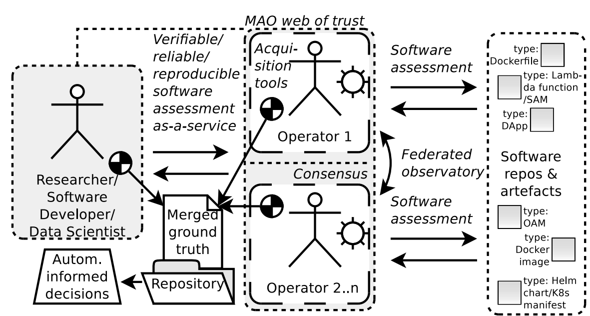

Figure 1. Web-of-trust based quality metrics as service

Middleware '20 Demos and Posters, December 7–11, 2020, Delft, Netherlands

ACM ISBN 978-1-4503-8202-1/20/12…$15.00
https://doi.org/10.1145/3429358.3429371

2 Research and Technical Approach

2.1 Contributions

The open source MAO framework[1] consists of an orchestrator with integrated scheduler for running artefact checkers (pluggable tools) and federation management based on a decentralised key-value store. Based on streams of metrics generated by the checkers, MAO learns about the value distribution and evolution. It contains generic detection of outliers and regressions (spike detection) and gaps (missing measurements, e.g. due to power failure), and furthermore about discrepancies to other measurements in the federation caused by different measurement methods, tools, parameters or times. Discrepancies are discovered by the framework on each node by iterating over all measurement streams of a representative sample of the respective other nodes. For each stream, both stream presence and timestamped metrics are compared. Deltas are expressed quantitatively. If for instance one node's Internet connection fails while retrieving an artefact for assessment, the resulting gap is quickly filled by the sample set of other nodes. These features offer cloud-native resilience and scalability not present in most long-term observation and testing frameworks. The framework is provided along with around a dozen checkers for assessing individual artefact types, covering primarily serverless (Lambda, SAM) and container technologies (Docker images, Kubernetes manifests), but also blockchains (DApps). Through software repositories, artefacts of those types are retrieved and assessed. Exemplary reference datasets spanning multiple months are further contributions in this context, representing joint observations as baseline for further nodes to join and widen the research community impact.

As long-lasting impact of our work, scientific software studies will become more comparable by standardising the empirical evidence on software quality, and follow-up works to already published papers on microservice quality can be underpinned with artefact views. A video presentation of the consensus algorithm acting in a realistic scenario is available online[2].

2.2 Capabilities and Maturity

We have performed regular artefact checks for over two years, in the recent year with MAO. On a technical level, the framework is entirely containerised, making it easy to redeploy it into any managed runtime environment supporting container execution and basic network port access. On a process level, the open federation ensures that upon invitation by existing nodes, new nodes can join and thus the research community at large can participate as trust anchor over a long period of time. Yet a governance model still needs to be found. Who owns the data when multiple nodes discover the same metrics? What happens in case of a split when two leaders emerge from the voting? According to our preliminary results, MAO offers substantial potential to conduct research on software quality assessment and management and to rethink the notion of reliable quality metrics for software artefacts.

[1]Code: https://github.com/serviceprototypinglab/mao-orchestrator

[2]Video: https://www.youtube.com/watch?v=6ELYkZijfu8

3 Related Work

MAO is not the first framework aiming to overcome reproducibility and reliability limitations by automated checks. The need to understand software quality has led to some specialised tools, ranging from source code analysis (e.g. SonarQube [5]) to runtime tracing and anomaly detection (e.g. FRAP [4]). ARRESTT [2] covers reproducible software testing methodologies, and Elastest [1] covers programmable end-to-end testing with failure injection in the cloud. Further techniques have been proposed to continuously monitor runtime and security properties of composite cloud applications but without capturing the observations as datasets [3].

All of these tools assume single ownership, and consequently most studies present the results from single instances, making them subject to potential bias. Moreover, their results are hard to compare because quality checks and studies are tightly coupled; the checks end when the studies are published, limiting comparability and reproducibility. In contrast, MAO advocates for a merged view on multiple instances running continuously and providing citeable observation windows and trusted snapshots for empirical studies.

References

[1] Antonia Bertolino, Antonello Calabrò, Guglielmo De Angelis, Micael Gallego, Boni García, and Francisco Gortázar. 2018. When the testing gets tough, the tough get ElasTest. In *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018*. ACM, 17–20. https://doi.org/10.1145/3183440.3183497

[2] Iaron da Costa Araújo, Wesley Oliveira da Silva, José B. de Sousa Nunes, and Francisco Oliveira Neto. 2016. ARRESTT: A framework to create reproducible experiments to evaluate software testing techniques. In *Proceedings of the 1st Brazilian Symposium on Systematic and Automated Software Testing, SAST 2016, Maringa, Parana, Brazil, September 19-20, 2016*. ACM, 1:1–1:10. https://doi.org/10.1145/2993288.2993303

[3] Holger Gantikow, Christoph Reich, Martin Knahl, and Nathan L. Clarke. 2019. Rule-based Security Monitoring of Containerized Workloads. In *Proceedings of the 9th International Conference on Cloud Computing and Services Science, CLOSER 2019, Heraklion, Crete, Greece, May 2-4, 2019*. SciTePress, 543–550. https://doi.org/10.5220/0007770005430550

[4] Xueyuan Han, Thomas F. J.-M. Pasquier, Tanvi Ranjan, Mark Goldstein, and Margo Seltzer. 2017. FRAPpuccino: Fault-detection through Runtime Analysis of Provenance. In *9th USENIX Workshop on Hot Topics in Cloud Computing, HotCloud 2017, Santa Clara, CA, USA, July 10-11, 2017*. USENIX Association.

[5] Diego Marcilio, Rodrigo Bonifácio, Eduardo Monteiro, Edna Dias Canedo, Welder Pinheiro Luz, and Gustavo Pinto. 2019. Are static analysis violations really fixed?: a closer look at realistic usage of SonarQube. In *Proceedings of the 27th International Conference on Program Comprehension, ICPC 2019, Montreal, QC, Canada, May 25-31, 2019*. IEEE / ACM, 209–219. https://doi.org/10.1109/ICPC.2019.00040