A Comparison Framework for Runtime Monitoring Approaches (Journal-First Abstract)

Rick Rabiser*, Sam Guinea†, Michael Vierhauser‡, Luciano Baresi†, and Paul Grünbacher*
*Christian Doppler Lab MEVSS, ISSE, Johannes Kepler University Linz, Austria
rick.rabiser@jku.at
†Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria, Milano, Italy
sam.guinea@polimi.it
‡Computer Science and Engineering, University of Notre Dame, IN, USA
mvierhau@nd.edu

***Abstract*—This extended abstract summarizes our paper entitled "A Comparison Framework for Runtime Monitoring Approaches" published in the Journal on Systems and Software in vol. 125 in 2017 (https://doi.org/10.1016/j.jss.2016.12.034). This paper provides the following contributions: (i) a framework that supports analyzing and comparing runtime monitoring approaches using different dimensions and elements; (ii) an application of the framework to analyze and compare 32 existing monitoring approaches; and (iii) a discussion of perspectives and potential future applications of our framework, e.g., to support the selection of an approach for a particular monitoring problem or application context.**

***Index Terms*—Runtime monitoring, literature review, comparison framework**

I. Summary

Many of today's large-scale software systems are complex and heterogeneous, and need to be monitored since their full behavior often only emerges at runtime (e.g., when interacting with other systems or the environment). Diverse monitoring approaches for various kinds of systems and purposes have been proposed [1], e.g., requirements monitoring [2], monitoring of architectural properties [3], and runtime verification [4]. The desired runtime behavior is often formally expressed using temporal logic or through the use of domain-specific (constraint) languages. Defined constraints are checked based on events and data collected from systems at runtime, typically through instrumentation.

Existing monitoring approaches are very diverse: only few provide (end-user) tool support; some cover specific architectural styles (e.g., service-oriented architectures), while others are general-purpose; some automatically generate instrumentations based on models, while others require probes to be manually developed. Approaches also differ regarding their expressiveness [5], e.g., the degree of support to check the occurrence and/or order of runtime events (temporal behavior), the interactions occurring between different (sub-)systems (structural behavior), and/or the properties held by certain runtime data (data checks).

This diversity makes it difficult to analyze and compare existing approaches. We thus developed *a comparison framework for runtime monitoring approaches* [1] based on the results of a systematic literature review [2], building on existing taxonomies for monitoring languages and patterns [5], [6], and taking inspiration from comparison frameworks from other domains such as software architecture or software product lines [7], [8].

Specifically, our paper [1] provides the following contributions: (i) a framework that supports analyzing and comparing runtime monitoring approaches using different dimensions and elements; (ii) an application of the framework to analyze and compare 32 existing monitoring approaches; and (iii) a discussion of perspectives and potential future applications of our framework, e.g., to support the selection of an approach for a particular monitoring problem or application context.

The full version of this work was published as journal article [1].

Acknowledgments

This work has been partially supported by project EEB – Edifici A Zero Consumo Energetico In Distretti Urbani Intelligenti (Italian Technology Cluster For Smart Communities) – CTN01_0 0 034_594053, the Christian Doppler Forschungsgesellschaft Austria, and Primetals Technologies.

References

[1] R. Rabiser, S. Guinea, M. Vierhauser, L. Baresi, and P. Grünbacher, "A Comparison Framework for Runtime Monitoring Approaches," *Journal of Systems and Software*, vol. 125, pp. 309–321, 2017.
[2] M. Vierhauser, R. Rabiser, and P. Grünbacher, "Requirements Monitoring Frameworks: A Systematic Review," *Information and Software Technology*, vol. 80, pp. 89–109, 2016.
[3] H. Muccini, A. Polini, F. Ricci, and A. Bertolino, "Monitoring architectural properties in dynamic component-based systems," in *Int'l Symp. on Component-Based Software Engineering*, 2007, pp. 124–139.
[4] C. Ghezzi, A. Mocci, and M. Sangiorgio, "Runtime monitoring of component changes with spy@runtime," *Int'l Conf. on Software Engineering*, pp. 1403–1406, 2012.
[5] M. Dwyer, G. Avrunin, and J. Corbett, "Patterns in property specifications for finite-state verification," *Int'l Conf. on Software Engineering*, pp. 411–420, 1999.
[6] N. Delgado, A. Q. Gates, and S. Roach, "A taxonomy and catalog of runtime software-fault monitoring tools," *IEEE Trans. on Softw. Eng.*, vol. 30, no. 12, pp. 859–872, 2004.
[7] N. Medvidovic and R. Taylor, "A classificiation and comparison framework for software architecture description languages," *IEEE Trans. on Softw. Eng.*, vol. 26, no. 1, pp. 70–93, 2000.
[8] M. Matinlassi, "Comparison of Software Product Line Architecture Design Methods: COPA, FAST, FORM, KobrA and QADA," in *Int'l Conf. on Software Engineering*, 2004, pp. 127–136.