TQPropRefiner: How to Interactively Comprehend and Refine Specifications on Transient Software Quality Properties?

Extended Abstract

Sebastian Frank
sebastian.frank@uni-hamburg.de
University of Hamburg, Hamburg

Julian Brott
julian.brott@uni-hamburg.de
University of Hamburg, Hamburg

Alireza Hakamian
mir-alireza.hakamian@iste.uni-stuttgart.de
University of Stuttgart, Stuttgart

André van Hoorn
andre.van.hoorn@uni-hamburg.de
University of Hamburg, Hamburg

1 Context and Problem

Microservice-based systems are exposed to transient behavior caused, for example, by (frequent) deployments, failures, or self-adaption. The potential complexity of transient behavior scenarios makes specifying flawless transient behavior requirements challenging. Still, the required approaches and tooling to comprehend transient behavior and refine the requirements are lacking. It is important to make quality requirements and expectations regarding transient behavior explicit and to (in-)validate them [5]. For example, a too-long service recovery time may lead to customer frustration. Furthermore, disproving the expectation of the reaction time of an autoscaler can indicate severe problems in the system design and configuration. However, flawlessly specifying transient behavior is challenging. This is due to the complexity involved in the changes triggering transient behavior and the dynamic nature of transient behavior.

2 Objective

Our work aims to address this gap by providing a structured interactive approach that assists software architects in comprehending transient behavior and refining requirements. TQPropRefiner — the early prototypical implementation of our approach — guides software architects through the three steps of (i) specifying transient behavior requirements, (ii) validating the requirements against runtime data and visualizing the requirement satisfaction, and (iii) refining the requirements by altering time constraints.

3 Method

As a formalism for specifying requirements, we use Property Specification Patterns (PSP) [4] to transform human-readable Structured English Grammar (SEG) specifications into testable Metric Temporal Logic (MTL) [1] formulas. TQPropRefiner uses runtime verification [2] to evaluate requirement satisfaction on time-series data, e.g., from Chaos Experiments. TQPropRefiner visualizes the system's behavior and requirement satisfaction to foster comprehension. Based on the gathered insights, users can refine their requirements. In particular, TQPropRefiner currently supports refining timing constraints for the use case in which a requirement is not satisfied.

We conducted an expert user study with five industry participants to gather early feedback on our approach and TQPropRefiner, despite limitations in the supported PSP and refinement strategies. The participants had to solve two tasks regarding comprehension and refinement capabilities of TQPropRefiner, answer a questionnaire, and participate in an interview.

4 Result

The participants were able to solve the tasks, and their answers indicate that our approach was easy to use. However, further improvements are necessary for use in practice, e.g., alignment of the shown visualizations and closer integration with system monitoring. Furthermore, the time constraint refinement needs more explanation. Despite currently limited support of PSP and refinement strategies, the preliminary results indicate that the approach can facilitate understanding transient behavior requirements among software architects and assist in the refinement process. Thus, our work is a first step toward facilitating the comprehension of transient behavior and refinement of requirements.

5 Talk Outline and Additional Resources

In our talk, we will present (i) our vision and initial concept of refining transient behavior requirements, (ii) our approach and the TQPropRefiner tool, and (iii) the results of the expert user study.

This work is embedded into a research endeavor introduced at SSP '21 [6] and based on a work regarding verifying transient behavior specifications presented at SSP '22 [7]. This talk presents contents of a paper accepted at the QUALIFIER workshop at ECSA '23 [3].

6 Acknowledgment

The authors thank the German Federal Ministry of Education and Research (dqualizer FKZ: 01IS22007B and Software Campus 2.0 — Microproject: DiSpel, FKZ: 01IS17051) for supporting this work. The work was conducted in the context of the SPEC RG DevOps Performance Working Group[1].

References

[1] R. Koymans. "Specifying real-time properties with metric temporal logic". In: *Real-time systems* 2.4 (1990), pp. 255–299.

[2] M. Leucker and C. Schallhart. "A brief account of runtime verification". In: *The journal of logic and algebraic programming* 78.5 (2009), pp. 293–303.

[3] S. Frank et al. "TQPropRefiner: Interactive Comprehension and Refinement of Specifications on Transient Software Quality Properties". In: *ECSA '23 Companion.* https://dqualizer.github.io/files/QUALIFIER2023-TB-Refinement.pdf (in press). 2013.

[4] M. Autili et al. "Aligning qualitative, real-time, and probabilistic property specification patterns using a structured english grammar". In: *IEEE Transactions on Software Engineering* 41.7 (2015), pp. 620–638.

[5] J. Eckhardt, A. Vogelsang, and D. M. Fernández. "Are "non-functional" requirements really non-functional? an investigation of non-functional requirements in practice". In: *Proceedings of the 38th international conference on software engineering.* 2016, pp. 832–842.

[6] S. Frank et al. "Scenario-Based Elicitation, Specification, and Comprehension of Transient Software Behavior". In: *SSP 2021* (2021). http://ceur-ws.org/Vol-3043/poster2.pdf.

[7] D. Zahariev et al. "Supporting and Verifying Transient Behavior Specifications in Chaos Engineering". In: *SSP 2022* (2022). https://dqualizer.github.io/files/SSP22_TransientBehaviorVerificationAsChaosExtension_ExtendedAbstract.pdf.

[1] https://research.spec.org/devopswg