Revitalizing QPME: Modernizing a Queueing Petri Net Modeling and Simulation Environment

David Linke
david.linke@uni-wuerzburg.de
University of Würzburg, Germany

Nicolas Stadler
nicolas.stadler@stud-mail.uni-wuerzburg.de
University of Würzburg, Germany

Nikolas Herbst
nikolas.herbst@uni-wuerzburg.de
University of Würzburg, Germany

Samuel Kounev
samuel.kounev@uni-wuerzburg.de
University of Würzburg, Germany

Abstract

Queueing Petri net Modeling Environment[1] (QPME) has supported the modeling and simulation of Queueing Petri Nets (QPNs) for many years, but its desktop-based user interface has become increasingly difficult to maintain. Built on Java and the Eclipse Rich Client Platform, the tool now relies on a legacy software stack that creates usability and platform-specific issues and makes further development harder. At the same time, QPNs remain a useful formalism for performance modeling because they combine queueing behavior with synchronization and concurrency.

This talk presents our ongoing effort to revitalize QPME and establish a foundation for its continued development. As a first major step, we redesigned the user interface and developed a web-based frontend using Angular and TypeScript. QPME already separated model editing from simulation: the editor exchanged XML models with the SimQPN simulation engine. The modernized architecture retains this basic separation while exposing SimQPN through a lightweight Spring-based backend and a REST interface. The new frontend continues to support the existing QPME model format while providing a modern browser-based modeling environment. It also introduces a clearer separation at the user-interface level, with dedicated modeling and simulation workbenches between which users can switch depending on the task at hand.

We report on the current state of the modernization, including the architectural decisions behind the redesign, the implementation of the new modeling interface, and the integration with the existing QPME ecosystem. We also discuss initial evaluation results obtained through hands-on sessions with experienced QPME users and through comparisons with models created using the legacy interface. The evaluation indicates that common modeling tasks can be successfully performed using the redesigned interface and provides concrete feedback on areas that still require further development.

Finally, we outline the next steps toward broader web-based deployment and use of QPME. These include introducing optional authentication and user accounts, establishing a robust deployment architecture with appropriate load distribution for concurrent simulation workloads, and integrating contextual hints and user guidance directly into the frontend to lower the entry barrier for new users. Another key step is a systematic evaluation and testing of SimQPN, the existing simulation engine that remains at the core of QPME. While the frontend has now been substantially modernized, SimQPN represents a central legacy component whose correctness, robustness, and suitability for server-side operation need to be assessed before QPME can be offered as a dependable web-based service. Further work includes improving the presentation and analysis of simulation results, strengthening compatibility with existing models, and continuing to evolve the deployment and execution architecture. In the longer term, these steps may enable scalable server-side execution of simulations and easier integration of QPME into contemporary performance-engineering environments.

Overall, the modernization of QPME is less about replacing a mature research tool than about making it usable and maintainable again. The work highlights practical challenges that arise when legacy performance-engineering software is moved toward a modern, web-based architecture while its established models and simulation capabilities need to be preserved.

[1] QPME: `https://se.informatik.uni-wuerzburg.de/tools/qpme/`