QPME - Queueing Petri Net Modeling Environment

Samuel Kounev
University of Cambridge
Computer Laboratory
Cambridge, CB3 0FD, UK
Email: skounev@acm.org

Christofer Dutz
Darmstadt University of Technology
Department of Computer Science
64289 Darmstadt, Germany
Email: dutz@c-ware.de

Alejandro Buchmann
Darmstadt University of Technology
Department of Computer Science
64289 Darmstadt, Germany
Email: buchmann@informatik.tu-darmstadt.de

***Abstract*— Queueing Petri nets are a powerful formalism that can be exploited for modeling distributed systems and analyzing their performance and scalability. However, currently available tools for modeling and analysis using queueing Petri nets are very limited in terms of the scalability of the analysis algorithms they provide. Moreover, tools are available only on highly specialized platforms unaccessible to most potential users. In this paper, we present QPME - a Queueing Petri Net Modeling Environment that supports the modeling and analysis of systems using queueing Petri nets. QPME runs on a wide range of platforms and provides a powerful simulation engine that can be used to analyze models of realistically-sized systems.**

I. Introduction

QPME (Queueing Petri Net Modeling Environment) is a performance modeling environment based on the Queueing Petri Net (QPN) modeling formalism. Introduced in 1993 by Falko Bause [1], the QPN formalism has a number of advantages over conventional modeling formalisms such as queueing networks and stochastic Petri nets. By combining the modeling power and expressiveness of queueing networks and stochastic Petri nets, Queueing Petri Nets (QPNs) enable the integration of hardware and software aspects of system behavior into the same model [2]. In [3], we showed how this advantage can be exploited for modeling distributed e-business applications. Building on this work, we have developed a methodology for performance modeling of distributed component-based systems using QPNs [4]. The methodology has been applied to model a number of systems ranging from simple systems to systems of realistic size and complexity. It can be used as a powerful tool for predicting the performance and scalability of distributed component-based systems.

However, while the QPN modeling paradigm provides many important benefits, there are currently few tools that support the modeling and analysis of systems using QPNs. Based on [5], the only tool that is available is the HiQPN-Tool [6], developed at the University of Dortmund. HiQPN can be used to build and analyze QPN models, however, it only supports analytical analysis techniques. As we demonstrated in [3], due to the state space explosion problem, QPN models of realistic systems are too large to be analyzable using analytical techniques. Another problem with HiQPN is that it is only available on Sun-OS 5.5.x / Solaris 2, which significantly limits its accessibility.

Recognizing the need for a tool to support the modeling and analysis of realistically-sized systems using QPNs, we have developed QPME - a QPN modeling environment with a user-friendly graphical user interface. In this paper, we present QPME, discussing its features and benefits. QPME is made of two major components, a QPN Editor (QPE) and a Simulator for QPNs (SimQPN). In the next sections, we take a closer look at these components.

II. QPE - Queueing Petri Net Editor

QPE (Queueing Petri Net Editor), the first major component of QPME, provides a graphical tool for modeling using QPNs [7]. It offers a user-friendly interface enabling the user to quickly and easily construct QPN models. QPE is based on GEF (Graphical Editing Framework) [8] - an Eclipse sub-project. GEF is an open source framework dedicated to providing a rich, consistent graphical editing environment for applications on the Eclipse platform. As a GEF application, QPE is written in pure Java and runs on all operating systems officially supported by the Eclipse platform. This includes Windows, Linux, Solaris, HP-UX, IBM AIX and Apple Mac OS among others, making QPE widely accessible.

Internally, being a GEF application, QPE is based on the model-view-controller (MVC) architecture. The model in our case is the QPN being defined, the views provide graphical representations of the QPN, and finally the controller connects the model with the views, managing the interactions among them. QPN models created with QPE can be stored on disk as XML documents. QPE uses its own XML schema based on PNML [9] with some changes and extensions to support the additional constructs available in QPN models.

A characterizing feature of QPE is that it allows token colors to be defined globally for the whole QPN instead of on a per place basis. This feature was motivated by the fact that in QPNs typically the same token color (type) is used in multiple places. Instead of having to define the color multiple times, the user can define it one time and then reference it in all places where it is used. This saves time, makes the model definition more compact, and last but not least, it makes the modeling process less error-prone since references to the same token color are specified explicitly.

III. SimQPN - Simulator for Queueing Petri Nets

The second major component of QPME is SimQPN - a discrete-event simulation engine specialized for QPNs. It is extremely light-weight and has been implemented 100% in Java

to provide maximum portability and platform-independence. SimQPN simulates QPNs using a sequential algorithm based on the event-scheduling approach for simulation modeling. Being specialized for QPNs, it simulates QPN models directly and has been designed to exploit the knowledge of the structure and behavior of QPNs to improve the efficiency of the simulation. Therefore, SimQPN provides much better performance than a general purpose simulator would provide, both in terms of the speed of simulation and the quality of output data provided.

SimQPN currently supports three different scheduling strategies for queues inside queueing places: First-Come-First-Served (FCFS), Processor-Sharing (PS) and Infinite Server (IS). A wide range of service time distributions are supported including Beta, BreitWigner, ChiSquare, Gamma, Hyperbolic, Exponential, ExponentialPower, Logarithmic, Normal, StudentT, Uniform, VonMises and Empirical. Timed transitions are currently not supported, however, in most cases a timed transition can be approximated by a serial network consisting of an immediate transition, a queueing place and a second immediate transition.

SimQPN offers the ability to configure what data exactly to collect during the simulation and what statistics to provide at the end of the run. This can be specified for each place (ordinary or queueing) of the QPN. The user can choose one of four modes of data collection. The higher the mode, the more information is collected and the more statistics are provided. Since collecting data costs CPU time, the more data is collected, the slower the simulation would run. Therefore, by configuring data collection modes, the user can make sure that no time is wasted collecting unnecessary data and, in this way, speed up the simulation.

SimQPN supports two methods for estimation of the steady state mean residence times of tokens inside the queues, places and depositories of the QPN. These are the well-known method of independent replications (in its variant referred to as replication/deletion approach) and the classical method of non-overlapping batch means. Both of them can be used to provide point and interval estimates of the steady state mean token residence time. The method of Welch is used for determining the length of the initial transient (warm-up period).

We have validated the analysis algorithms implemented in SimQPN by subjecting them to a rigorous experimental analysis and evaluating the quality of point and interval estimates [10]. In particular, the variability of point estimates provided by SimQPN and the coverage of confidence intervals reported were quantified. A number of different models of realistic size and complexity were considered. Our analysis showed that data reported by SimQPN is very accurate and stable. Even for residence time, the metric with highest variation, the standard deviation of point estimates did not exceed 2.5% of the mean value. In all cases, the estimated coverage of confidence intervals was less than 2% below the nominal value (higher than 88% for 90% confidence intervals and higher than 93% for 95% confidence intervals). However, still in case the user wants to apply a different technique for steady state analysis this is also possible. SimQPN can be configured to output observed token residence times to files (mode 4), which can then be used as input to external analysis tools.

A novel feature of SimQPN is the introduction of the so-called *departure disciplines*. The latter are defined for ordinary places or depositories and determine the order in which arriving tokens become available for output transitions. We define two departure disciplines, Normal (used by default) and First-In-First-Out (FIFO). The former implies that tokens become available for output transitions immediately upon arrival just like in conventional QPN models. The latter implies that tokens become available for output transitions in the order of their arrival, i.e. a token can leave the place/depository only after all tokens that have arrived before it have left, hence the term FIFO. For an example of how this feature can be exploited and the benefits it provides we refer the reader to [4], [11].

IV. Summary

QPME provides a robust and powerful tool for performance analysis making it possible to exploit the modeling power and expressiveness of queueing Petri nets to their full potential. The tool is available free of charge for non-profit use. Further information can be obtained by contacting the first author.

References

[1] F. Bause, "Queueing Petri Nets - A formalism for the combined qualitative and quantitative analysis of systems," in *Proceedings of the 5th International Workshop on Petri Nets and Performance Models, Toulouse, France, October 19-22*, 1993.

[2] F. Bause, P. Buchholz, and P. Kemper, "Integrating Software and Hardware Performance Models Using Hierarchical Queueing Petri Nets," in *Proceedings of the 9. ITG / GI - Fachtagung Messung, Modellierung und Bewertung von Rechen- und Kommunikationssystemen, (MMB'97), Freiberg (Germany)*, 1997.

[3] S. Kounev and A. Buchmann, "Performance Modelling of Distributed E-Business Applications using Queuing Petri Nets," in *Proceedings of the 2003 IEEE International Symposium on Performance Analysis of Systems and Software - ISPASS2003, Austin, Texas, USA, March 20-22*, 2003.

[4] S. Kounev, *Performance Engineering of Distributed Component-Based Systems - Benchmarking, Modeling and Performance Prediction*. Shaker Verlag, Dec. 2005, ISBN: 3832247130.

[5] University of Aarhus, "Petri Net Tool Database," Department of Computer Science - DIAMI, 2004, http://www.daimi.au.dk/PetriNets/tools/.

[6] F. Bause, P. Buchholz, and P. Kemper, "QPN-Tool for the Specification and Analysis of Hierarchically Combined Queueing Petri Nets," in *Quantitative Evaluation of Computing and Communication Systems*, ser. Lecture Notes in Computer Science, H. Beilner and F. Bause, Eds., vol. 977. Springer-Verlag, 1995.

[7] C. Dutz, "QPE - A Graphical Editor for Modeling using Queueing Petri Nets," Master Thesis, Technical University of Darmstadt, Apr. 2006.

[8] The Eclipse Foundation, "Graphical Editing Framework (GEF)," 2006, http://www.eclipse.org/gef/.

[9] J. Billington, S. Christensen, K. van Hee, E. Kindler, O. Kummer, L. Petrucci, R. Post, C. Stehno, and M. Weber, "The Petri Net Markup Language: Concepts, Technology, and Tools," in *Proceedings of the 24th International Conference on Application and Theory of Petri Nets, June 23-27, Eindhoven, Holland*, June 2003.

[10] S. Kounev and A. Buchmann, "SimQPN - a tool and methodology for analyzing queueing Petri net models by means of simulation," *Performance Evaluation*, vol. 63, no. 4-5, pp. 364–394, May 2006, doi:10.1016/j.peva.2005.03.004.

[11] S. Kounev, "Performance Modeling and Evaluation of Distributed Component-Based Systems using Queueing Petri Nets," *IEEE Transactions on Software Engineering*, 2006, to appear.