Importing PMIF Models into PIPE2 using M2M Transformation

Pere Bonet
Computing and Maths Department
Universitat de les Illes Balears
07071 Palma de Mallorca, Spain

p.bonet@uib.cat

Catalina M. Lladó
Computing and Maths Department
Universitat de les Illes Balears
07071 Palma de Mallorca, Spain

cllado@uib.cat

ABSTRACT

Model-to-model (M2M) transformation is a key aspect of model-driven development (MDD), where importing and exporting models fits very well. A queueing network based metamodel (PMIF) and a Petri net metamodel are specified using the Eclipse Modelling Framework. The transformation from PMIF models to Petri net models is then build using ATL. This paper presents such a transformation and its integration into PIPE2, a Petri net modelling tool. It also illustrates the transformation by a simple example.

Categories and Subject Descriptors

B.8.2 [**Performance and Reliability**]: Performance Analysis and Design Aids

Keywords

M2M (model-to-model transformation), Performance models, Petri Nets, PMIF, Queueing networks

1. INTRODUCTION

Interchange formats have been defined for the interchange of queueing network models, Petri nets models, and others. However, there is still scope to extend their application to multiple formalisms, in particular interchanging models that can be a Petri net or a queueing network. This way a tool that analyses Petri nets can import, for example, a queueing network model and vice-versa.

Model-to-model (M2M) transformation can be done in many different ways. The advantages of using a transformation language instead of java-based transformations are discussed in [4]. Yet still, it has not been much used in performance engineering.

We present a Queueing Network (QN) based metamodel (PMIF, Performance Model Interchange Format) and a Petri net (PN) metamodel that are specified using Eclipse's EMF (Eclipse Modeling Framework) and its M2M transformation using ATL, which is part of EMF and provides a completely automated process for model to model transformations. This process needs the source model (which in our case the PMIF metamodel), the target model (in our case the PN one), and a transformation code expressed in ATL language (described by rules).

ICPE'12, April 22 – 25, 2012, Boston, Massachusetts, USA
ACM 978-1-4503-1202-8/12/04.

A different approach to multiformalism is offered by Mobius, OsMoSys and SIMTHESys [3]. All of them aim to provide a methodology and tool support for multi-formalism models' design and evaluation, and consider model composition and multiple solution methods.

2. M2M TRANSFORMATION

Our transformation's input is a PMIF [5] model, a common representation for system performance model that follows the QN paradigm. The PMIF metamodel espeficied in Eclipse is shown in Fig 1. On the other hand the output transformation is a PN model that follows the EMF metamodel shown in Fig 2.

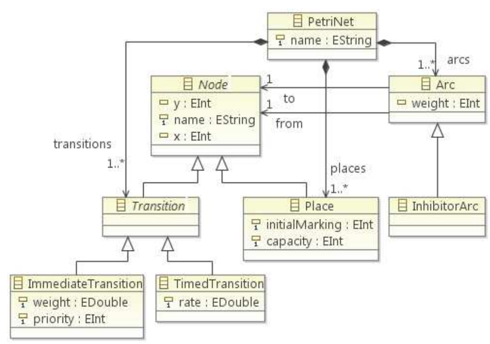

Figure 2: Petri net metamodel

The ATL transformation specifies how a PMIF model is transformed into a PN model. Each element in the QN has its correspondent element(s) in the PN. The most important elements and their counterparts are described next: (1) PMIF Servers and WorkUnitServers are nodes that provide some processing service for one or more Workloads. Its PN counterpart is a structure composed by a place, which represents the server queue, a timed transition whose firing rate is the inverse of the service time of the node, a place which represents the job exiting the node after having received its service and the necessary arcs to keep places and transitions connected. If there is more than one workload in the system (see below), this needs to be replicated as many times as workloads are served by one Server or WorkUnitServer since tokens are not distinguishable in a PN. This information is found in the ServiceRequest elements. (2) A PMIF OpenWorkload represents a workload with a potentially infinite population where transactions or jobs arrive from the

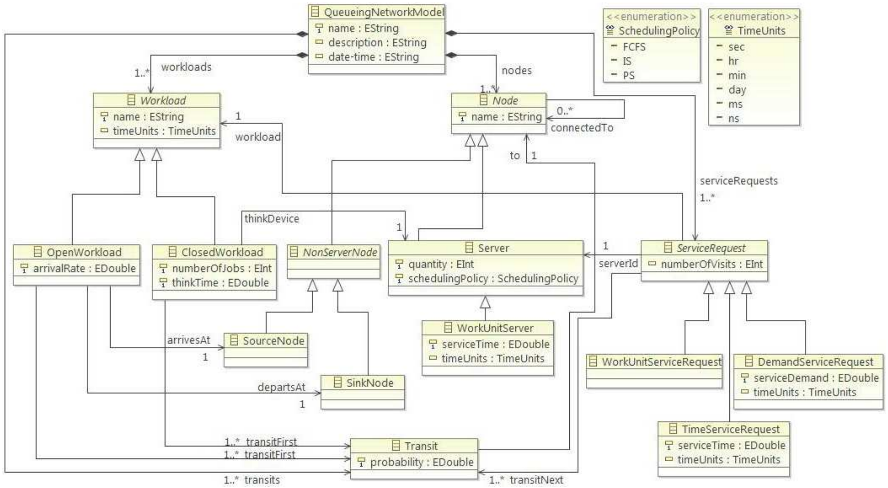

Figure 1: PMIF metamodel

outside world, receive service, and exit. The PN representation of this element consists of a timed transition whose firing rate is equal to the open workload arrival rate, a place that represents the arrival of the jobs and an arc connecting the transition and the place. (3) A PMIF ClosedWorkload represents a workload with a fixed population that circulates among the Servers, so it is represented in the PN has a place with the population as its initial marking. A closed workload has a ThinkDevice or independent delay node (for example, to model finite collections of users) characterized by its ThinkTime (average interval of time that elapses between the completion of a transaction or job and the submission of the next transaction or job). The ThinkDevice counterpart is as the server nodes.

Once the ATL transformation is done and we have a PN model, an XSLT transformation is applied, so the model is specified with PIPE2's [1] format, which is a pseudo PNML [2]. The original intention was to use a model to text transformation. However, all the ones we could find would possibly work in the Eclipse environment but could not be exported to be build as part as an existing application as PIPE2. This way, using PIPE2 we can open a PMIF model, as shown in Fig. 3. Clearly PIPE2 has been updated so it allows for the importing option and the transformation process.

3. CASE STUDY AND CONCLUSIONS

As a case study we use the Oracle example described in [6], with 3 servers (CPU, UserThink and Delay). Its PMIF model is imported into PIPE2 and the PN seen for this example in as shown in Fig. 3, only with some transitions and places moved a little bit so the drawing is clearer (the transformation output leaves some arcs crossed). In fact, improving the automatic drawing of the nets is part of our future work as well as the import of more complex QN models, for example, allowing for different scheduling policies. Performance indexes obtained for this example are exactly as shown in [6], so it demostrates the correct transformation of our tool.

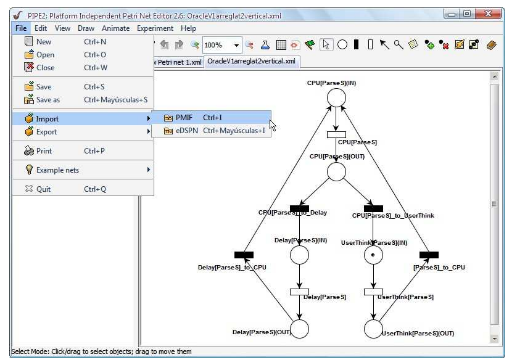

Figure 3: Oracle case study

Acknowledgments

This work is partially funded by the TIN2010-16345 EIGER project of the *Ministerio de Educacion y Ciencia*, Spain.

4. REFERENCES

[1] Platform Independent Petri net Editor 2. `http://pipe2.sourceforge.net/`.

[2] PNML, Petri Net Markup Language. `www.pnml.org`.

[3] E. Barbierato, M. Gribaudo, and M. Iacono. Exploiting multiformalism models for testing and performance evaluation in simthesys. In *Valuetools*, 2011.

[4] V. Cortellessa, S. Di Gregorio, and A. Di Marco. Using atl for transformations in software performance engineering: a step ahead of java-based transformations? In *Proc. of the 7th Int. Workshop on Software and Performance*, pages 127–132. ACM, 2008.

[5] C. U. Smith, C. M. Lladó, and R. Puigjaner. Performance Model Interchange Format (PMIF 2): A Comprehensive Approach to Queueing Network Model Interoperability. *Performance Evaluation*, 67(7):548 – 568, 2010.

[6] C.U. Smith and C. Milsap. Software performance engineering for Oracle applications: Measurements and models. In *Proc. Computer Measurement Group*, Las Vegas, NV, USA, 7-12 December 2008.