A brief Introduction to MOSEL-2*

Patrick Wüchner[1], Hermann de Meer[1], Jörg Barner[2], and Gunter Bolch[2]

[1] Chair of Computer Networks and Computer Communications, Faculty of Mathematics and Computer Science, University of Passau,
Innstr. 33, 94032 Passau, Germany
{Patrick.Wuechner, Hermann.DeMeer}@Uni-Passau.de
[2] Research Group on Analytical Modeling, Chair of Distributed Systems and Operating Systems, Department of Computer Science, University of Erlangen-Nürnberg,
Martensstr. 1, 91058 Erlangen, Germany
{Barner, Bolch}@Informatik.Uni-Erlangen.de

Abstract. The versatile *MOdeling, Specification and Evaluation Language* is the core element of the MOSEL-2 tool. This *description languages* provides a high-level means for specifying models, performance measures, and the graphical presentation of results. The description languages is implemented in form of an *evaluation environment* that comprises translators to the modeling languages of several third-party performance evaluation tools that evaluate the specified model.

1 Introduction

While using performance modeling and evaluation tools, modelers from all application areas experience the same problems: Is the chosen tool able to evaluate the model at all? Will the tool still be able to evaluate the model after the stochastic distribution of, e.g., a service time is changed? Is it possible to switch to another evaluation method or to another tool without having to rewrite or redraw the whole model again?

Several performance modeling tools were developed during the last decades, some of them based on queueing networks, others on stochastic process algebras, stochastic Petri nets, or a mixture of those methods. For a modeler it is hard to choose the appropriate tool right from the beginning because all tools are different in some respect. Furthermore, even a slight change of model parameters during the cyclic modeling process may make it necessary to change the evaluation method. This method might not be provided by the tool currently in use and changing the tool implies redefining the model in another textual or graphical model description language.

MOSEL-2 has been developed to overcome these problems. During the design of MOSEL-2's textual model description language, particular emphasis was placed on keeping the MOSEL-2 model specifications compact.

* Parts of this work have been accomplished under support of the Euro–NGI – Network of Excellence, EC grant IST–507613.

```
// **** CONSTANT AND PARAMETER PART **********************************
CONST     mu_1  := 1/5;           /* service rate of Node1             */
PARAMETER t_2   := 3..7 STEP 0.5; /* service delay of Node2            */
CONST     K     := 10;            /* number of jobs in closed network  */
CONST     init1 := K;             /* initial number of jobs in Node1   */
// **** NODE PART ****************************************************
NODE Node1[K] := init1;        /* 'Node1' with capacity 'K' and        *
                                * 'init1' initial jobs                 */
NODE Node2[K] := K - init1;    /* 'Node2' with capacity 'K' and        *
                                * 'K - init1' initial jobs             */
// **** RULE PART ****************************************************
FROM Node1 TO Node2 RATE mu_1; /* expon. service at 'Node1' with       *
                                * parameter 'mu_1' and exit to 'Node2' */
FROM Node2 TO Node1 AFTER t_2; /* determ. service at 'Node2' with      *
                                * parameter 't_2' and exit to 'Node1'  */
// **** RESULT PART **************************************************
PRINT mean_number_of_jobs_in_Node1 := MEAN(Node1);
PRINT mean_number_of_jobs_in_Node2 := MEAN(Node2);
PRINT utilization_of_Node1         := PROB(Node1 != 0);
PRINT utilization_of_Node2         := UTIL(Node2);
PRINT DIST Node1;
// **** PICTURE PART *************************************************
PICTURE "Tandem Network Results"
PARAMETER t_2
CURVE mean_number_of_jobs_in_Node1
YLABEL "Mean Number of Jobs in Node 1"
XLABEL "Service Time of Node 2";
```

Fig. 1. MOSEL-2 model of closed tandem queueing network (tandem.mos).

MOSEL-2's evaluation environment is equipped with a set of model translators that allow the automatic translation of the MOSEL-2 model to several third-party performance evaluation tools. This allows the modeler to switch easily between different evaluation methods and tools. In doing so, the major part of the MOSEL-2 model can be reused and the model description languages of these tools do not have to be learned.

Furthermore, MOSEL-2 is equipped with syntactical and semantical preprocessors that provide more convenience by allowing compact specification of reoccurring elements, by automatic approximation of non-Markovian distributions using phase-type constructs, and by supporting multiple sets of parameters.

2 The MOSEL-2 Model Description Language

The MOSEL-2 model description is specified in plain text. Being familiar with its syntax, it is easy to write down the model description directly. The basic elements of a MOSEL-2 model are nodes and rules. *Nodes* are used to specify static model components. The *rules* describe the dynamic behavior of the model.

As an example, a MOSEL-2 specification of a closed tandem queueing network with $K = 10$ customers is modeled. The first node of this queueing network is an –/M/1–FCFS queueing system with constant service rate $\mu_1 = 1/5$ customers per time unit. The second node is an –/D/1–FCFS queueing system with deterministic service time $\overline{t_2}$ which is a variable system parameter. The analogous MOSEL-2 model is shown in Figure 1.

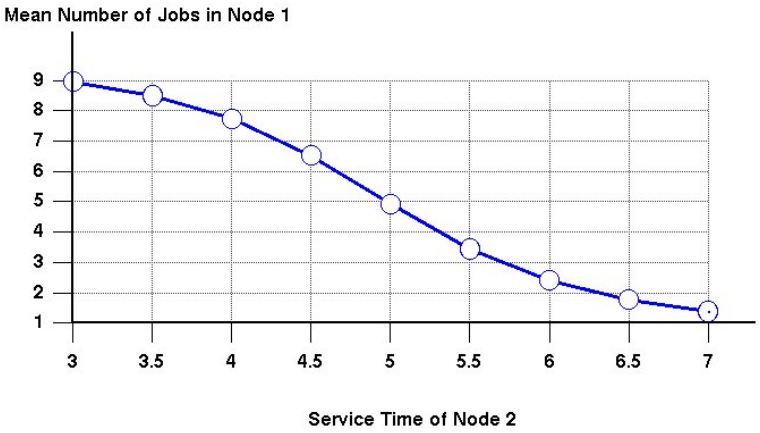

Fig. 2. Graphical result file (tandem.igl).

A MOSEL-2 specification can consist of up to six parts: the optional *constant and parameter part*, the *node part*, the optional *function and condition part*, the *rule part*, the *result part*, and the optional *picture part*. The MOSEL-2 model given in Figure 1 consists of 5 parts:

In the *constant and parameter part* (lines 2 to 5) several constants and one parameter is defined. These can be reused in the remainder of the model. Thus, it is very easy to change the model's settings without having to reconsider the whole model. Parameters are useful to evaluate the same model automatically with different settings. The output of all evaluation runs is collected in a single graph to compare the effect of different settings directly. When several parameters are defined, all combinations of these will be evaluated.

The two nodes are defined in the *node part* (lines 6 to 10). The maximum number of jobs within each node and the initial number of jobs in each system can be assigned.

The dynamic behavior of the model is specified using MOSEL-2 rules. These rules are located in the *rule part* (lines 11 to 15). MOSEL-2 rules are only able to transfer jobs between nodes with the given distribution of the holding time as long as they are enabled, i.e., all implicit and explicit rule conditions are fulfilled.

The modelers can specify the results they are interested in using the *result part* (lines 16 to 21). For this, there exists a set of basic performance measures that can be combined arbitrarily using mathematical equations. These performance measures include the mean number of jobs within a certain node (MEAN), the probability that there is a certain number of jobs within a node (PROB), and the utilization of a certain node (UTIL).

Graphical representations of these results can be specified in the *picture part* (lines 22 to 27). The graphical results obtained from the tandem queueing model are shown in Figure 2.

3 The MOSEL-2 Evaluation Environment

The evaluation of the MOSEL-2 model is done predominantly automatically by the MOSEL-2 evaluation environment following these steps:

1. After describing the model using the MOSEL-2 language (Sec. 2), the modeler passes the model to the evaluation environment via command line calling, e.g., `mosel2 -Ts tandem.mos`.
2. MOSEL-2 translates the model to the model description language of the tool TimeNET (option `T`) which is then started (option `s`) by MOSEL-2 automatically – once for each parameter setting. By default, TimeNET solves the given model using numerical steady-state analysis and saves the results to a file with TimeNET specific structure.
3. The MOSEL-2 environment parses the tool's output and generates a textual result file (tandem.res) containing the performance and reliability measures specified by the user. The second result file (tandem.igl) contains the requested graphical representations of the results and can be viewed using the IGL interpreter that is distributed with MOSEL-2 (see Fig. 2).

The MOSEL-2 environment is currently able to generate input files for the tools SPNP[1], TimeNET[2], and MOSES[3]. The evaluation methods provided by these tools include numerical solvers for Markovian models, numerical solvers for a restricted class of models with non-exponentially distributed state transitions (eDSPNs), and discrete-event simulation for non-Markovian models.

4 Conclusion

MOSEL-2 provides a textual model specification language, that is very compact and easy to learn. The model translators of MOSEL-2 give convenient access to various performance evaluation tools and their evaluation methods. Several concepts of MOSEL-2 prevent the user from tedious tasks while modeling complex systems.

MOSEL-2 is distributed under the GNU Public License (GPL). It can be downloaded from `http://www.mosel2.net.fmi.uni-passau.de/`.

References

1. Hirel, C., Tuffin, B., Trivedi, K.S.: SPNP: Stochastic Petri Nets. version 6.0. In: Proceedings of the 11th International Conference on Computer Performance Evaluation: Modelling Techniques and Tools, Springer-Verlag, London, UK (2000) 354–357
2. Zimmermann, A., Freiheit, J., German, R., Hommel, G.: Petri Net modelling and performability evaluation with TimeNET 3.0. In: Proc. 11th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation (TOOLS'2000). Volume 1786 of LNCS., Springer (2000) 188–202
3. Greiner, S., Bolch, G.: Modeling production lines with blocking, batch processing and unreliable machines using the Markov analyzer MOSES. In: Proc. European Simulation Symp. - ESS '95. (1995) 303–309