مجلة العلوم البحثة والتطبيقية
Journal of Pure & Applied Sciences
www.Suj.sebhau.edu.ly ISSN 2521-9200
Received 20/03/2016 Revised 29/07/2017 Published online 01/08/2017

Modelling, Specification, and Evaluation Language (MOSEL-2) Overview

*Wael S. Abughres[1] , Mohamed A. Mgheder[2], Ahmed B. Abdurrman[3]
[1] Department of Electrical & Electronic Engineering, Faculty of engineering, University of Tripoli, Libya
[2] Department of Computer Science, Faculty of Information Technology, University of Tripoli, Libya
[3] Department of Electrical & Electronic Engineering, Faculty of engineering, University of sebha, Libya
*Corresponding Author: W.abughres @uot.edu.ly

Abstract Modeling, Specification & Evaluation Language (MOSEL), tool used for the performance and reliability modeling of communication systems, computers, and manufacturing systems, once the system is specified using this language. The modeling language is part of the evaluation environment. Once the system is specified using the language, the evaluation environment takes place by executing the performance analyses of the model, and calculating the steady state probabilities . After this stage, results can be collected in the result file or in graphics mode using the Intermediate Graphical Language (IGL), where the aim of this paper is to give an overview of mosel and to show a real example under windows platform.

Keywords: Modelling Evaluation, MOSEL-2

نظرة عامة على لغة النمذجة والمواصفات والتقييم (MOSEL-2)

وائل صالح أبوغريس[1] و محمد أحمد مغيدر[2] و أحمد أبوسيف[3]
[1] قسم الهندسة الكهربائية والإلكترونية – كلية الهندسة – جامعة طرابلس، ليبيا
[2] قسم البرمجيات ، كلية التقنية ، جامعة طرابلس، ليبيا
[3] قسم الهندسة الكهربائية والإلكترونية ، كلية الهندسة ، جامعة سبها، ليبيا
*للمراسلة: W.abughres @uot.edu.ly

الملخص تستخدم لغة النمذجة والتوصيف والتقييم **(MOSEL)**، كأداة لنمذجة أداء واعتمادية نظم الاتصالات، الحواسيب، ونظم الإنتاج، عندما يحدد النظام باستخدام هذه اللغة. لغة النمذجة هذه هي جزء من تقييم البيئة. عندما يتم تحديد / توصيف النظام باستخدام هذه اللغة، يتم التقييم بتنفيذ تحليل الأداء للنموذج، وحساب احتمالات حالة الاستقرار. بعد هذه المرحلة، يمكن تجميع النتائج في ملف نتائج أو على هيئة بيانات / رسومات باستخدام لغة البيانات / الرسومات الوسيطة**(IGL)**، حيث تهدف هذه الورقة إلى إلقاء نظرة عامة على لغة **(MOSEL)** مع عرض مثال تطبيقي لهذه اللغة تحت مظلة برنامج ويندوز.

الكلمات المفتاحية: تقييم النمذجة ، MOSEL-2 .

1. Introduction

The structure of this paper started with a brief definition of mosel, then we installed the language, after compiling it under windows, after that, we gave a real case of M/M/N/K, showing the delay and blocking probability as a performance measure. Currently, there already exist many different performance evaluation tools, and all have different input languages. The input languages for these tools are not only different, but they are also hard to use, since they all have a syntax which is oriented on the characteristics of the tool and not on the system which the user wants to describe. The basic idea of the new language MOSEL is to have only one language which is easy to use and reflects the real structure of the system, where the user wants to describe. MOSEL can be compared with a higher programming language which is problem and no longer machine orientated. MOSEL will be a common frontend for many performance evaluation tools. The user has only to become familiar with MOSEL and has not to learn all the different tool specific languages to use these tools. After the user has specified his/her system in MOSEL, the MOSEL- Compiler will produce the appropriate input description for the specific tools. By giving options the user may specify which tools he/she wants to use. Currently MOSEL can produce input descriptions for the tools: MOSES [1], which is a Markov analyzer, and the tool SPNP [2], [3], which is a Stochastic Petri Net Package. It is planned to use MOSEL as a common description language for other already implemented performance evaluation tools, like PEPSY [4] or SHARPE [5]. The MOSEL-Compiler is implemented in the programming language C by using the scanner and parser tools LEX and YACC [6]. MOSEL is a high level programming language used to model complex systems, like communication networks, production lines, computer systems, and many more. The models that are described in MOSEL will be evaluated by

a modeling environment that is also called "MOSEL", using numeric analysis methods.
The basic modeling primitives of a MOSEL model are nodes and rules.
MOSEL model is determined by the values of all its nodes; the rules describe the possible transitions between these nodes.

1.1 History of MOSEL

The model description language MOSEL was developed by Helmut Herold at University *Erlangen-Nuremberg,* Germany in 1994[7]. To evaluate the performance of systems, like computer systems, networks, production lines and Communication systems. Buetel revised the MOSEL language and called the revised form MOSEL-2 appeared in 2003[8] which looks different from the previous one, and it is under Solaris & Linux, where the source code under Linux can be downloaded from [11].

MOSEL2 features:

-Markov and DSPN based solution (immediate, exponential and deterministic transitions)

- Simulation

- Moved away from C
- Support for SPNP and MOSES tools

1.2 Installing MOSEL-2 in Windows

1. Download and Compile the source code under windows[11]
2. Download and Install the Tool Command Language (TCL) from [12]
3. Download and Install Stochastic Petri Net Package (SPNP) tool from [13].
4. Set the variables, Path and Editing the batch file in [13] after installation.
5. Install Intermediate Graphics Language (IGL) from [11].
6. Install TimeNET from [14]
7. Then, run the command line Wish86 igl.tcl
8. Install Java jdk 7 from [15]
9. Run the batch file Init_spnp.bat
10. MOSEL2, this is a Windows compiled version which is easy to use in modelling.
11. Install Mingw from [16]
12. Run \TimeNET\bin\startGUI.bat if you want to use Petri Net tool TimeNet (Analysis & Simulation).

2 The MOSEL-2 Model

Description Language

In MOSEL-2, indentation and line breaks have no special meaning. Any sequence of spaces, tabs and new-line characters is considered as white space and is ignored. There are three exceptions:

(1) In strings, spaces and tabs are copied literally.

(2) A "//" comment must be ended by a new- line character.

(3) Two consecutive names have to be separated by white space.

A comment in a MOSEL-2 description is ignored. There are two forms of comments, both known from C/C++, namely a one-line comment that starts with "//" and ends at the end of the current line, and a free form that starts with "/*", may contain any text including line breaks, and ends with "*/". MOSEL-2 does not distinguish integer values and floating point values; integer values are just a subset of the floating point values.
A variable (identifier) may be used as the name of a node, a result, or a constant, an enumeration, a function, or a named condition.
Each identifier can only denote one of those types. It must have a single definition in the source code.
The following names are reserved words with special meanings; they may not be used as variables:

Table1:Reserved words in MOSEL-2

AFTER	AND	ASSERT	AVG	COND
CONST	CUM	CURVE	DIST	ELIF
ELSE	ENUM	EXTERN	FIXED	FLOOR
FROM	FUNC	IF	MEAN	NODE
NOT	OR	PARAMETER	PICTURE	PRD
PRINT	PRIO	PROB	PRS	RATE
RESULT	SIN	SQRT	STEP	THEN
TIME	TO	UTIL	WEIGHT	WITH
XLABEL	YLABEL			

MOSEL-2 is case sensitive, so capital letters are different from small letters, so "parameter", "Parameter" and "PARAMETER" are all different names.

Expressions

A MOSEL-2 expression yields a floating point value. "IF" *condition* "THEN" *expr* "ELIF" *condition* "THEN" *expr* "ELSE" *expr* .
A conditional expression starts with an "IF", and yields the value of the first *expr* for which the associated *condition* holds. If no *condition* holds, it yields the value of the final *expr*.

The arithmetic operators "+", "-", "*" and "/" are supported. Division by zero is forbidden and yields an error. The arithmetic operators are left-associative. The operators "*" and "/" have higher priority, unless the evaluation order is changed by parentheses.
The operator "^" is right-associative and used for exponentiation. The exponentiation base (the left operand of an exponentiation) must be positive.

Parentheses can be used to express and/or change the evaluation order.

"SIN(*expr*)" yields the sine of *expr*, where *expr* is measured in radians.
"SQRT(*expr*)" yields the non-negative square root of *expr*, which must be non- negative.
"FLOOR(*expr*)" yields the largest integer value that is not greater than *expr*.

A PROB construct yields the overall probability of all states where *condition* holds. A MEAN construct yields the expectation value of *state-expr*; i.e., the sum of *state-expr* evaluated for all states, each term weighed by the probability of its state. Conditions and expressions in a PROB or MEAN construct are evaluated for each state. Anode name in such a construct evaluates to the node's value in that state[17].
The keywords PROB and MEAN may be prefixed by AVG if the analysis is transient, which computes the time-averaged probability or expectation value; i.e., the values are evaluated and integrated in the time span from $t = 0$ up to the evaluation time point and divided by the length of the time span. The keyword MEAN may be prefixed by CUM if the analysis is transient, computing the cumulated expectation value; i.e., the expectation value is evaluated and integrated in the time span from $t = 0$ up to the evaluation time point.

2.1 MOSEL-2 specification can be divided into six main parts:

1. Definition
This is optional part, which consists of constant definition, parameter definition and enumeration definition. In a "CONST" definition, the variable is given a floating point constant. In a "PARAMETER" definition, the variable is given a set of values, and themodel is evaluated at each value. In "ENUM" definition, the variable is a given set of constants between two brackets "{}".

2. Node
The node definition part contains the definition of the nodes. Nodes are used to describe the model's state. Each node has a certain value ranges from 0 to a maximum value called the capacity of the node.

3. Function
This is optional part. MOSEL-2 offers two types of functions: either the "FUNC", which yields to a numeric value, or the "COND",
which is a placeholder of logical expression.

4. Rule
The rule part contains MOSEL-2 rules. Rules are used to describe how the system may change from one state to another.
A rule is composed of the following parts:
• A precondition, which describes the subspace of states in which the rule is enabled.
• One or more actions, which describe the changes of the current state that take place when the rule fires.
• A firing distribution. When the rule gets enabled, it may fire immediately, after a fixed time interval, with exponentially distributed probability, or with (discrete) uniform distribution.
• A re-enabling policy. When a rule gets disabled, it may remember or forget the time
that has elapsed while the rule was enabled. This has impact on the time until the remaining firing delay when the rule gets re- enabled.
• A priority and a weight. If several rules are enabled and may fire at the same time, only one of the rules with maximum priority will do so.

5. Result
Result part is optional. This part contains the computation of the performance measures results. There are two types of result definitions:
a) Proper results:
Such a result definition defines *result* and assigns the value of *expr* to it. The value of *result* can be used in subsequent result definitions. If the keyword PRINT is used, the value of the result will be written into the result file and can be used in picture definitions that may follow.
b) Durations:
Such a result definition prints a *duration*; i.e. the expected time span until *condition* holds for the first time. The *duration* can be used in picture definitions. It must not be used in subsequent result definitions.

6. Picture
This is an optional part. The graphs of the computed performance measures are created
in this part via Intermediate Graphical Language (IGL) which is a user-friendly tool associated with MOSEL-2 package to generate the graphical representation of the performance measures in a very nice way and compatible with any operating system.

3 The MOSEL-2 Evaluation Environment

MOSEL-2 model will be evaluated and analyzed by an external analysis tool. The model description will be translated into the format that is used by the respective tool. This tool is invoked automatically by the MOSEL-2 environment, its results are read in and written to the MOSEL-2 result file or may be displayed as graphs. The graph descriptions are written to a separate File in the proprietary IGL format. The graphs can be displayed, printed or changed using the IGL interpreter, which is part of the MOSEL-
2 program package. The following stochastic analysis tools are supported:
-The stochastic Markov analysis tool MOSES, developed at *Universit¨at Erlangen-N¨urnberg*, Germany.
- The Petri Net analysis tool SPNP [SPNP], developed at Duke University, USA.
- The Petri Net Tool TimeNET, developed at TU Berlin since 1991, is a collection of tools that

support the creation, testing, and evaluation (analysis and simulation) of stochastic Petri nets.
Other stochastic analysis or simulation tools can be integrated with the MOSEL environment due to its modular structure.
The semantics of a MOSEL-2 description can be defined by describing how it can be converted into a stochastic process with finite state space which is reached in three steps:
1. The MOSEL-2 description is translated into a mathematical model called the *Explicit State Model* (ESM).
2. ESM is converted to a process with mixed continuous/discrete state space and continuous time base. The state space of the Markov process consists of the states of the ESM, but supplementary real numbers have been added to each state, which indicate the remaining firing times for each rule. Since the remaining firing times are part of the process' state space, the probabilistic behavior can be predicted in the future from the current behavior and do not need to examine past states.
3. The stochastic chain with finite state space can be derived by dropping the remaining firing times in the states.

3.1 The MOSEL Modelling- Environment

The process of performance and reliability analysis in the MOSEL modeling environment is divided into the following steps as shown in Figure 1:

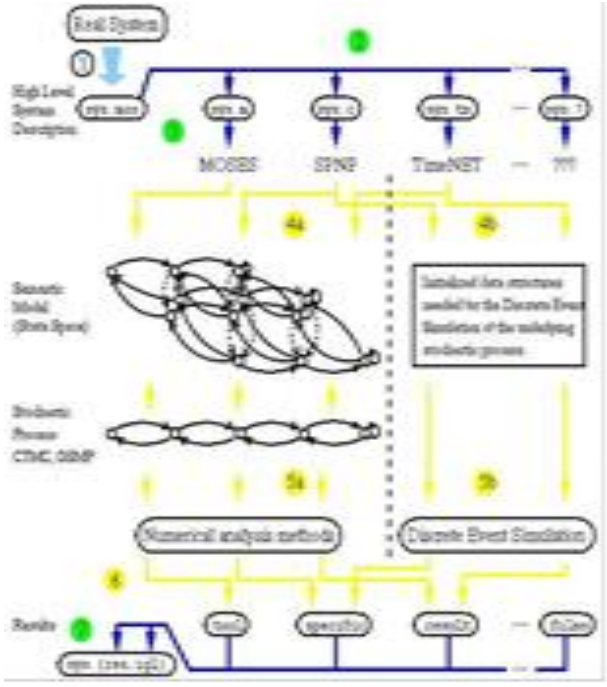

Fig. 1 The modelling and analysis process in the MOSEL-2 environment [10].

1. The modeller inspects the real-world system and generates a high-level system description using the MOSEL specification language. The result part ends up with the desired performance and reliability measures; it passes the model to the environment which then performs all following steps without user interaction.

2. The MOSEL environment automatically translates the MOSEL model into a tool- specific system description, for example a [C based Stochastic Petri net Language] (CSPL) file suitable to serve as input for SPNP.

3. The appropriate tool (i.e., SPNP) is invoked by the MOSEL environment.

4. Depending on the structure of the model, particularly with regard to the types of distributions used, the analysis continues along one of the following paths:

4a) If a numerical solution of the subordinated stochastic process is feasible, the state space of the model is generated out of the static description according to the semantic rules favored by the tool.

4b) If none of the tool's numerical solution methods is applicable, a structural analysis of the model is performed. All information needed for a subsequent Discrete Event Simulation (DES) is extracted and preprocessed during this stage. DES analysis is available in TimeNET and SPNP.

5. The computation of the stationary or transient state distributions of the underlying stochastic process of the model is prepared.

5a) For state-space based numerical analysis methods, all the information needed by the numerical solution algorithm is extracted from the semantic model.
5b) If the model is analyzed via discrete event simulation, the simulation engine is ended by the command-line options given by the modeller.

6. The stochastic process is solved by the selected solution method. From the computed stationary or transient state probabilities the tool derives the result measures as specified in the tool-specific high-level description and stores them in a file with a tool-specific structure.

7. The MOSEL-environment parses the tool-specific output and generates a result file containing the performance and reliability measures which the user specified in the MOSEL system description. If the modeller requested graphical representation of the results, an IGL-file is generated by the MOSEL environment.

4 A Case Study System

In this paper, we would like to introduce the open queuing system M/M/N/K as our case study with a single station considering N servers with the following assumptions:

1) We assume that the arrival stream of jobs forms a Poisson process with rate λ.

2) Service times of jobs are independent and

identically distributed exponential random variables with service rate μ.

3) Assuming maximum queue length is K.

4) If there are K jobs in the system and a job arrives at the station, the new job is turned away.

5) M/M/ stand for the type of arrival and service process, where it is memory less, meaning for inter-arrival and service times are independent, exponentially distributed random variables. Using these assumptions in the code shown below:

```
//M/M/N/K
/*PARAMETER AND CONSTANTS
***********/

 PARAMETER K:=1 .. 100;     //CONST K =
 100; //Queue
Size
CONST N=7;            // NUMBER OF SERVERS
CONST λ = 20.2;     //ARRIVAL RATE /min
CONST μ = 2.4;       //SERVICE RATE /min
NODE n[K]=0;

/*TRANSITIONS **********************/

 FROM EXTERN TO n RATE λ;
 IF n > N FROM n TO EXTERN RATE N*μ;
 //NORMAL
DEPARTURE
 IF n<= N FROM n TO EXTERN RATE n * μ;

/*RESULTS ***************************/
 // TEXTUAL (.res)

 PRINT Buffer :=K;
 PRINT SERVERS
 :=N;
 PRINT prob_full:= (PROB(n = K));

 //fraction of Packets turned away /min
 PRINT rate_reject:= λ*prob_full;

// RESULT rate_reject= λ*prob_full;
 PRINT mean_number_of_Packets :=
 MEAN (n); PRINT RHO := λ/μ ;
 PRINT UT := UTIL (n);

//average time a Packets has to spend for
check-out Min
// response time
PRINT mean_number_of_Packets/(λ -
rate_reject);

// GRAPHICAL (.igl)

 PICTURE "average time a Packets has to
spend for check- out (Min)"
 PARAMETER K
 XLABEL "Queue Size"
 YLABEL "average time a Packets has to
 spend(Min)" CURVE Tq;

 PICTURE "fraction of Packets turned
 away /min" PARAMETER K
 XLABEL "Queue Size"
 YLABEL "fraction of Packets
 turned away" CURVE prob_full;

 PICTURE
 "Utilization"
 PARAMETER K
 XLABEL "Queue
 Size" YLABEL
 "Utilization"
 CURVE UT;

 PICTURE
 "mean_number_of_Packets"
 PARAMETER K
 XLABEL "Queue Size"
 YLABEL
 "mean_number_of_Packets"
 CURVE
 mean_number_of_Packets;
End
```

4.1 Analyzing the open queuing system

M/M/N/K using the MOSEL-2 Environment.

Suppose that we have saved our MOSEL-2 model in a file named MMNK.mos. We are going to invoke the MOSEL-2 Environment on the command line via:

```
mosel2 -cs MMNK.mos
```

The option *-c* indicates that we request that our MOSEL-2 specification should be translated into a set of CSPL files (C based Stochastic Petri net Language), which serve as input for the tool SPNP. We also use the *s* option to start the appropriate tool automatically. The MOSEL-2 environment will now perform the rest of the analysis and result post-processing steps automatically using the SPNP package as shown in Figure
1.

Upon completion of the analysis, the MOSEL-2 environment creates two files named MMNK.res and MMNK.igl which contain the requested performance measures of the system in text and graphic form. The text file is very useful if we are interested in the particular values of the performance measures for a specific value of the variable system parameter which was given in the model specification. The graphical demonstration of the results shows how the performance measures depend on a variable system parameter over an interval. The graphical result file will be processed and viewed with the IGL-utility, which is part of the MOSEL-2 environment.

This is a sample of Stationary analysis of "MMNK.mos" by

SPNP.

```
Parameters:
   K = 1

Results:
   Buffer = 1
   SERVERS = 7 prob_full
   = 0.89381 rate_reject =
   18.0549
   mean_number_of_Packets = 0.89381
   RHO = 8.41667
   UT = 0.89381
   Tq = 0.41667
================================
                  ===

Parameters:
   K = 2

Results:
   Buffer = 2
   SERVERS = 7 prob_full
   = 0.78998 rate_reject =
   15.9576
   mean_number_of_Packets = 1.76768
   RHO = 8.41667
   UT = 0.9777
   Tq = 0.41667
================================
                  ===

Parameters:
   K = 3

Results:
   Buffer = 3
   SERVERS = 7
   prob_full = 0.68909 rate_reject =
   13.9196 mean_number_of_Packets
   = 2.61685
   RHO = 8.41667
   UT = 0.99307
   Tq = 0.41667
================================
                  ===

Parameters:
   K = 4

Results:
   Buffer = 4
   SERVERS = 7
   prob_full = 0.59183 rate_reject =
   11.9549 mean_number_of_Packets
   = 3.43544
   RHO = 8.41667
   UT = 0.99717
   Tq = 0.41667

Parameters:
   K = 5

Results:
   Buffer = 5
   SERVERS = 7
   prob_full = 0.49906 rate_reject =
   10.081 mean_number_of_Packets =
   4.21625
   RHO = 8.41667
   UT = 0.99858
   Tq = 0.41667
```

The results could be displayed in graphics form as shown in Figure2 and 3.

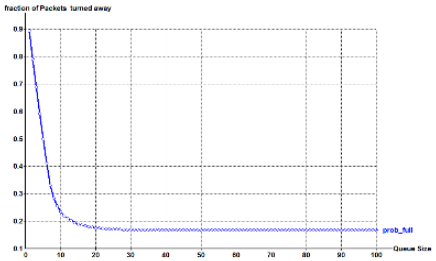

Fig. 2 Shows Tq or the Delay or the time that the packet spends in the queue

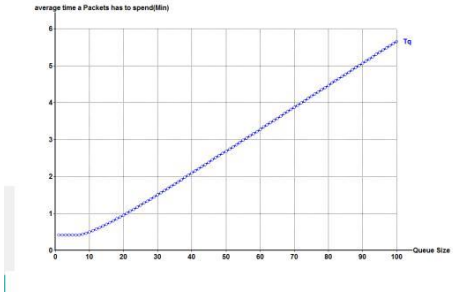

Fig.3 Probability of packets rejected as we increase the queue size.

5 Conclusion

This paper is good start for people trying to use this tool. MOSEL is growing and getting more interesting features in modeling of communication Networks. The case study used is an M/M/N/K where we have showed the results in text and in Graphics mode. Several concepts of MOSEL can help the user to avoid tedious tasks while modeling complex systems.

Referencs

[1]- Bolch, G.; Greiner, S; Jung, H.; Zimmer, R: The Markov Analyzer MOSES,

[2]- Ciardo, G; Muppala, J.K.: Manual for the SPNP Package Version 3.1,Duke University Durham, North Carolina,1991

[3]- Herold, H.: lex und yacc, Addison-Wesley, Bonn 1995

[4]- Kirschnick, M: PEPSY-QNS, Technical Report (TR-14-18-94), IMMD IV, Erlangen-Nürnberg, 1994

[5]- Sahner, R A.; Trivedi, K S.: SHARPE: Symbolic Hierarchical Automated Reliability andPerformance Evaluator, Duke University Durham, North Carolina, 1986

[6]- Trivedi, K S.; Ciardo, G.: A Decomposition Approach for Stochastic Reward Net

Models,Duke University Durham, North Carolina, 1991
[7]- [7] Technical Report (TR-14-10-94), IMMD IV, Erlangen-Nürnberg, 1994
[8]- Bj¨orn Beutel:Integration of the Petri Net Analysator TimeNET into the Model Analysis Environment MOSEL Diploma Thesis in Computer Science University of Erlangen-N¨urnberg
[9]- Practical Performance Modelling: Application of the MOSELKhalid Begain, Gunter Bolch, Helmut Herold,Laguage,Khuwer Academic Publisher.
[10]- Queueing Networks & Markov Chains,Gunter Bolch, Stefan Greiner, Hermann de Meer, Kishor S. Trivedi,Jhon Wiley & Sons, Inc., Publication,Second edition,2006
[11]- http://www4.informatik.uni-erlangen.de/DE/Projects/MOSEL/Download
[12]- http://www.tcl.tk/software/tcltk/download.html,1/1/2015
[13]- http://people.ee.duke.edu/~kst/software_packages.html,/6/2013

[14]- https://www.tuilmenau.de/sse/timenet/information-for-users/download-area/,1/12/2014

[15]- http://www.oracle.com/technetwork/java/javase/downloads/index.html,/12/2014
[16]- http://sourceforge.net/projects/mingw/files/,/12/2014
[17]- Tien V. Do, Patrick W¨uchner, Tam´as B´erczes and J´anos Sztrik, Hermann De Meer, A New Finite-Source Queueing Model For Mobile Cellular Networks Applying Spectrum Renting, Asia-Pacific Journal of Operational Research,2013.