Intelligent Environments 2021
E. Bashir and M. Luštrek (Eds.)

doi:10.3233/AISE210075

The Mercury Environment: A Modeling Tool for Performance and Dependability Evaluation

Thiago PINHEIRO [a,b], Danilo OLIVEIRA [b,c], Rubens MATOS [b,d], Bruno SILVA [b], Paulo PEREIRA [a,b], Carlos MELO [a,b], Felipe OLIVEIRA [a,b], Eduardo TAVARES [a,b], Jamilson DANTAS [a,b], and Paulo MACIEL [a,b,1]

[a] *Centro de Informática, Universidade Federal de Pernambuco, Brazil*
[b] *MoDCS research group, Brazil*
[c] *HighQSoft GmbH, Germany*
[d] *Instituto Federal de Educação, Ciência e Tecnologia de Sergipe, Brazil*

Abstract. It is important to be able to judge the performance or dependability metrics of a system and often we do so by using abstract models even when the system is in the conceptual phase. Evaluating a system by performing measurements can have a high temporal and/or financial cost, which may not be feasible. Mathematical models can provide estimates about system behavior and we need tools supporting different types of formalisms in order to compute desired metrics. The Mercury tool enables a range of models to be created and evaluated for supporting performance and dependability evaluations, such as reliability block diagrams (RBDs), dynamic RBDs (DRBDs), fault trees (FTs), stochastic Petri nets (SPNs), continuous and discrete-time Markov chains (CTMCs and DTMCs), as well as energy flow models (EFMs). In this paper, we introduce recent enhancements to Mercury, namely new SPN simulators, support to prioritized timed transitions, sensitivity analysis evaluation, several improvements to the usability of the tool, and support to DTMC and FT formalisms.

Keywords. models, Mercury, DTMC, fault tree, simulation

1. Introduction

There are many tools for supporting performance and/or dependability analysis of computer systems. Some tools have limitations regarding the evaluation of non-exponential models and they support only one formalism or a small set of them, or even a limited number of evaluations. Considering this, the MoDCS[2] research group[3] started the development of the Mercury tool in 2008, having a vision to deliver to the community a tool for supporting performance and dependability evaluations overcoming the limitations found in other tools.

[1]Corresponding Author: Head of MoDCS Research Group and Full Professor at Centro de Informática, Av. Jorn. Aníbal Fernandes, s/n - Cidade Universitária, Recife - PE, 50740-560, Brazil; E-mail: prmm@cin.ufpe.br.
[2]Modeling of Distributed and Concurrent Systems.
[3]http://www.modcs.org/.

The Mercury tool has been developed to enable the creation and evaluation of stochastic models, such as: stochastic Petri nets (SPNs) [1], continuous and discrete-time Markov chains (CTMCs and DTMCs) [2], reliability block diagrams (RBDs) [3] and dynamic RBDs (DRBDs) [4], fault trees (FTs) [5], and energy flow models (EFMs) [6]. The tool has been widely adopted in numerous research projects, which had the results published in peer-reviewed journals[4] and conferences[5]. Mercury supports a considerable number of formalisms and empowers a large range of evaluations for each of them, so it can help the academy and industry to make predictions in several application fields. Among other distinguished features of this tool are: the possibility of evaluating non-exponential models through SPN and RBD simulations, supporting more than twenty-five probability distributions; a scripting language; a random variate generator (RVG); computation of reliability importance indices; moment matching [7] of empirical data, as well as sensitivity analysis evaluation of CTMC, SPN, and RBD models.

The MoDCS team is frequently improving the tool by adding new features, updating existing functionalities, and fixing bugs, and new versions are usually released every six months. In this paper, we introduce recent enhancements to Mercury up to version 5.0.2 that were not covered by the previous papers presenting the tool [8,9,10], namely: support to DTMC and FT formalisms; new SPN simulators; prioritized timed transitions; sensitivity analysis evaluation of RBDs; and several improvements to the usability of the tool.

This work is organized as follows. Section 2 presents a comparison between Mercury and similar tools. Section 3 provides an overview of the tool. Section 4 presents the recent enhancements. Section 5 presents a case study as an example to demonstrate the feasibility of using Mercury for supporting infrastructure planning, and finally Section 6 draws the final remarks.

2. Related Tools

This section presents a general comparison of formalisms supported by Mercury and other similar tools. There are many modeling tools available worldwide, each one with its respective pros and cons, as well as with a set of modeling formalisms associated, which varies according to their representation power and complexity. Among the most cited tools, we can mention ReliaSoft BlockSim[6], Relex[7], SHARPE [11], TimeNet [12], Snoopy [13], SPNP[8], and GreatSPN[9]. BlockSim provides the way to evaluate some dependability attributes (reliability, availability, and maintainability) with RBDs and FTs models. The main limitation of BlockSim is that it cannot evaluate components dependencies with its current modeling formalisms. To evaluate more complex scenarios, it is necessary to employ tools that can handle space-state models, such as Relex and SHARPE. Other tools like TimeNET and Snoopy provide powerful Petri Net (PN) modeling and evaluation mechanisms but are limited to PN extensions. On the other hand,

[4]Publications in journals: https://www.modcs.org/?page_id=521.
[5]Publications in conferences: https://www.modcs.org/?page_id=525.
[6]BlockSim: https://www.reliasoft.com/BlockSim.
[7]Relex: http://www.relex.com.
[8]SPNP: https://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/db/spnp.html.
[9]GreatSPN: http://www.di.unito.it/~greatspn/index.html.

SPNP and GreatSPN support generalized stochastic Petri nets (GSPNs). Table 1 presents a general comparison of Mercury and these tools.

Table 1. Modeling tools comparison (DRBD only through Script Language).

Tool \ Formalism	RBD	DRBD	FT	CTMC	DTMC	SPN	EFM
BlockSim	✓		✓				
Relex	✓		✓	✓		✓	
SHARPE	✓		✓	✓		✓	
TimeNet						✓	
Snoopy						✓	
SPNP						✓	
GreatSPN						✓	
Mercury	✓	✓	✓	✓	✓	✓	✓

3. Overview and Software Architecture

This section presents the main features available on Mercury for all supported modeling formalisms. Currently, Mercury supports six formalisms via its GUI editor (see Figure 1) and an overview is depicted in Figure 2.

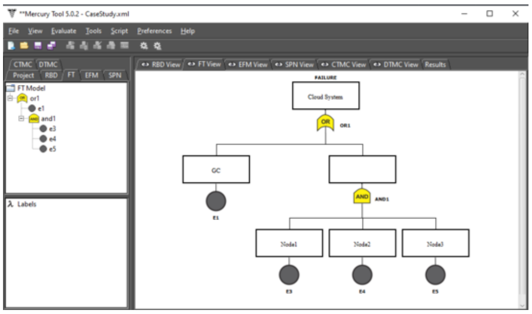
*Mercury Tool 5.0.2 - CaseStudy.xml
File View Evaluate Tools Script Preferences Help
CTMC DTMC
Project RBD FT EFM SPN
RBD View FT View EFM View SPN View CTMC View DTMC View Results
FT Model
or1
e1
and1
e3
e4
e5
Labels
FAILURE
Cloud System
OR
OR1
GC
E1
AND
AND1
Node1
Node2
Node3
E3
E4
E5

Figure 1. Mercury GUI.

The SPN editor and evaluator allow modeling and evaluating GSPNs. Mercury implements numerical analysis and simulation techniques to support performance and dependability evaluations for SPN models including steady-state and time-dependent metrics. Steady-state metrics are obtained by performing stationary evaluations and time-dependent metrics are obtained by performing transient evaluations. Regarding transient evaluation, the tool provides a simulator to evaluate the mean time to absorption (MTTA) of absorbing models with non-exponential transitions. For stationary analysis, two solution methods are available: Grassmann-Taksar-Heyman (GTH) [14] and Gauss-Seidel [2]. For transient analysis, two solution methods are available: Uniformization (also known as Jensen's method) and Runge-Kutta (4th order) [2]. Additionally, the SPN evaluator allows the execution of experiments, which evaluate the impact of varying a parameter of the model on a chosen metric. The SPN editor has also a feature called *token*

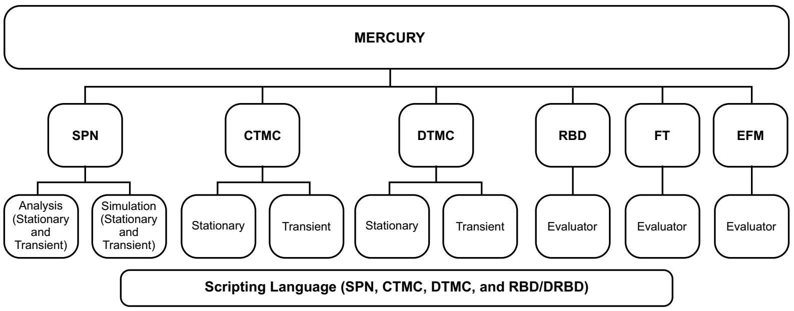
MERCURY
SPN
CTMC
DTMC
RBD
FT
EFM
Analysis (Stationary and Transient)
Simulation (Stationary and Transient)
Stationary
Transient
Stationary
Transient
Evaluator
Evaluator
Evaluator
Scripting Language (SPN, CTMC, DTMC, and RBD/DRBD)

Figure 2. Mercury - Formalisms.

game to assist the validation of models. By using this feature, users can simulate graphically the firing of transitions, step by step. This mechanism allows one to change the state of the model under evaluation considering the current marking state. For example, users can simulate equipment failures as well as the corresponding consequences on the availability of a system. This feature can be useful to test the behavior of transitions that have guard expressions or priorities assigned to them, making it possible to test whether the logical rules applied to the model are properly implemented. Another important feature regarding SPN evaluation is the structural analysis. This allows the user to evaluate the structural properties of the model through analysis of place invariants, siphons, and traps [15].

The CTMC editor and evaluator allow modeling and evaluating CTMCs. Regarding stationary analysis, the GTH and Gauss-Siedel numerical techniques are supported. Regarding transient analysis, the Uniformization and Runge-Kutta numerical techniques are supported. Probabilities of absorption and MTTA may be computed when evaluating absorbing CTMCs. Experiments are also supported. Algebraic expressions using symbolic parameters can be used to define transition rates between states. Also, greek letters can be used to compose the name of parameters. In addition to states and transitions, users can define reward rates assigned to states, which enables the modeling of *Markov reward models*. Mercury also supports sensitivity analysis on CTMCs, making it possible to evaluate the sensitivity of state probabilities with respect to each input parameter entered.

The DTMC editor and evaluator allow modeling and evaluating DTMCs. The same numerical techniques available for CTMCs are also available for DTMCs. When performing stationary analysis, it is possible to compute sojourn times and recurrence time [2]. For absorbing models, Mercury also enables computing mean time to absorption and probabilities of absorption. Expressions including state probabilities may compose metrics. Unlike CTMCs, transitions between states are based on probability in DTMCs. Transition probabilities can be defined by means of algebraic expressions using symbolic parameters. Using greek letters for parameter labels and experiments are also supported.

The RBD editor and evaluator allow modeling and evaluating availability and reliability using block diagrams. The tool supports three types of block configurations: series, parallel, and K-out-of-N (KooN). RBDs provide closed-form equations making it possible to obtain the results more quickly than using other methods, such as SPN simulation. However, there are many situations (e.g., dependency among components) in which modeling using RBD is more difficult than adopting SPN. Mercury provides two methods for computing dependability metrics: SFM (a method based on the structural function) and SDP (sum of disjoint products). The SDP method, based on Boolean algebra, computes metrics considering minimal cuts and minimum paths. Mercury supports the evaluation of the following metrics: mean time to failure (MTTF), mean time to repair (MTTR), steady-state availability, instantaneous availability, reliability, unreliability, uptime, and downtime. In addition, when evaluating time-dependent metrics, multiple points on time may be computed. Experiments are also supported. Additionally, the RBD evaluator allows the calculation of component importance measures, bounds for dependability analysis, structural and logical functions as well as sensitivity analysis. Component importance measures indicate the impact of a particular component with respect to the overall reliability or availability. Thus, the most important component (i.e., that one with the highest importance) should be improved in order to obtain an increase in reliability or availability. Evaluation of dependability bounds [16] is a method to calculate dependability metrics when the model is very large. By applying such a method, approximations of the chosen metric can be obtained more quickly than solving all the closed equations involved. Structural and logical functions are alternative ways of representing the system mathematically, in which the former adopts algebraic expressions and the latter adopts boolean expressions. Mercury also provides a feature to reduce the complexity of RBD models.

The Fault Tree editor and evaluator allow modeling and evaluating availability and reliability using fault trees. FTs and RBDs differ from each other in their purposes. RBDs are success-oriented while FTs are a failure-oriented modeling approach. Using the Mercury tool, it is possible to handle two types of nodes: basic events and gates (logic ports). Leaf nodes represent basic events. Mercury supports three types of gates: *and*, *or*, and *KooN*. The events leading to the top-event *failure* must be directly linked to a *gate*, making it possible to evaluate the probability of an event happening based on the probability obtained by joining basic events and child gates. Converting FT to RBD models is supported by the tool.

The EFM editor and evaluator enable the evaluation of availability, sustainability, and cost of cooling infrastructures and datacenter/clouds power, considering the power constraints of each component. Mercury supports five types of evaluations for EFM models. *Cost Evaluation* evaluates operational, acquisition, and total costs. *Exergy Evaluation* computes the sustainability through the exergy metric. Exergy estimates energetic efficiency. *Energy Flow Evaluation* evaluates the energy that flows through each device considering the power constraints of each one. *Combined Evaluation* provides an integrated evaluation of dependability, sustainability impact, and cost of cooling infrastructures and data center/cloud power [6]. *Combined Evaluation* provides an integrated evaluation of dependability, sustainability impact, and cost of cooling infrastructures and data center/cloud power [6]. *Flow Optimization (PLDA, PLDAD, and GRASP)* evaluates SPN, CTMC, RBD, and EFM models. Three optimization techniques were implemented for supporting this evaluation: power load distribution algorithm (PLDA) [17], power load distribution algorithm depth (PLDAD), and GRASP-based algorithm.

The Mercury scripting language was designed to facilitate the creation and evaluation of complex models. It allows greater flexibility in model evaluations using the Mercury engine. The language supports SPN, CTMC, DTMC, and RBD/DRBD formalisms.

Scripts can be executed by a command-line interface (CLI) or via an editor available inside the Mercury tool. The tool has a feature that automatically generates the script representing the selected model in the GUI. The advantage of using this language in conjunction with the CLI tool, using shell scripts, for example, is the possibility to automate an evaluation workflow. In addition, there are other advantages offered by the language which are not supported by modeling through the graphical interface:

- **Support for hierarchical modeling.** The resulting metric of a model can be used as input parameters for any other model, independent of the modeling formalism being adopted;
- **Support for hierarchical transitions on GSPNs.** This type of transition can be used as a way to reduce the complexity of models or to express a recurring structure in a model. It is important to highlight that for some tools [12] the support for hierarchical SPN models is only for coloured Petri nets;
- **Support for symbolic evaluations and experiments.** Parameters of a model can be defined as variables left open. Thus, these variables can be changed at the time of evaluation in order to measure the impact of these new parameter values on certain metrics;
- **Support for loop and conditional structures.** It allows the creation of nets with variable structures. The variables for controlling those structures can be treated as parameters of the model;
- **Support for phase-type distributions [18].** The family of phase-type distributions can be used to approximate any distribution that does not fit an exponential distribution. A number of approximate analysis techniques are based on matching the moments of continuous-time phase-type distributions. Models having non-Markovian properties may only be evaluated numerically through the adoption of phase-type approximation technique [19].

A script contains models, each one with its metrics and parameters, and a main section where values for input parameters are defined. Listing 1 presents a CTMC extracted from [20].

```
markov RedundantGC{
    state fu up; state fw; state ff; state uf up; state uw up;
    transition fw -> fu(rate = sa_s2);
    transition fu -> ff(rate = lambda_s2);
    transition ff -> uf(rate = mu_s1);
    transition uf -> uw(rate = mu_s2);
    transition uw -> fw(rate = lambda_s1);
    transition fw -> uw(rate = mu_s1);
    transition uw -> uf(rate = lambdai_s2);
    transition uf -> ff(rate = lambda_s1);
    transition fw -> ff(rate = lambdai_s2);
    transition fu -> uw(rate = mu_s1);
    metric aval = availability;
} main {
    lambda_s1 =  1/180.72; mu_s1 = 1/0.966902;  mu_s2 = 1/0.966902;
    lambdai_s2 = 1/216.865; lambda_s2 = 1/180.721; sa_s2 = 1/0.005555555;
    print("Availability: " .. solve(model=RedundantGC, metric=aval));
}
```

Listing 1. Mercury script for a CTMC model.

4. New Functionalities and Updates

Several new features, updates, and bug fixes have been included in Mercury. These features include support for two new formalisms, namely DTMC and FT. In addition, the stationary and transient simulators were reimplemented. Improvements were also made to the usability of the tool, such as the possibility to assign a description to each component of a model. A description represents additional information about the component for the comprehension of the model under construction. Another improvement increases the readability of models. Once a component has been inserted, it is possible to read its properties in the drawing area by positioning the mouse cursor on it. A tooltip appears showing all properties of the component. Mercury provides this feature for all components of all supported formalism. As follows, new functionalities and improvements are presented.

The DTMC editor is one of the main features that have been added to Mercury. The analysis of DTMC models comprehends the computation of holding time and recurrence time [2], besides the usual state probabilities that are already obtained for CTMC models. The fault tree editor is another main feature that was included in the latest Mercury release. FT is a top-down logical diagram and it makes it possible to create a visual representation of a system showing the logical relationships between associated events and causes lead that may lead the evaluated system to a failure state. In the current version, Mercury supports *and*, *or* and *koon* logic gates. In the future, we intend to add suport to other logical operations such as *xor* and *priority and*.

The two main updates on the RBD editor were the implementation of sensitivity analysis and the change in the way the nodes are represented. Mercury computes partial derivative sensitivity indices for RBDs, which indicate the impact that every input parameter has on availability. Sensitivity analysis can only be performed when the model under evaluation has only exponential blocks.

Several improvements were performed on the SPN editor and evaluator. Regarding the drawing area, Mercury now supports two types of arc styles: rectangular and curved. An expression editor was implemented to make it easy to create large and complex guard expressions and metrics. The editor highlights parentheses, brackets, and braces as well as some keywords. Also, a reference updater was implemented to update all properties marking references to a definition when it is updated or removed. A definition is a variable that stores a numeric value. It may be attached to some properties of others SPN components. More than one property or expression may refer to the same definition. Definitions are useful for supporting experiments. In this case, by changing the value of a definition, it is possible to evaluate the impact of that change on an evaluated metric. Regarding the SPN evaluator, similar to immediate transitions, Mercury now supports prioritized timed transitions. It is important to highlight that in that case immediate transitions always take precedence to fire over timed transitions. Both SPN simulators were reimplemented. Besides several improvements, such as on the GUI and the possibility to export detailed information of the simulation, the stationary and transient simulators can detect the occurrence of rare events, which occur when the difference between the delays assigned to the transitions is very large.

- **Stationary Simulator.** Stationary simulation can be used when evaluating steady-state metrics of non-Markovian SPN models. Mercury implements the method of batch means [21]. This method comprises three steps: running a long simulation run, discarding the initial transient phase, and dividing the remaining events run into batches. In this new simulator, it is possible to follow the simulation step by step on the GUI. A large number of statistics are computed and displayed at the end of the simulation;

- **Transient Simulator.** Transient simulation may be adopted when evaluating metrics of non-Markovian SPN models considering a specific point in time. A transient simulation is composed of a set of replications where each replication is composed by a set of runs [21]. Each run executes from time 0 until the evaluated time t' is reached. A set of sampling points may be evaluated considering this time interval. When the current set of runs is finished, the value of each sampling point of the current replication is computed. A replication represents the mean values of the points in its set of runs. Mercury supports two methods for computing the value of each point:

  $*$ **DES[10] + Linear Regression 1** computes the value of each sampling point at the end of each run using linear interpolation between two known points. When the required number of runs is executed, the obtained values for each point are stored. Also, the mean of the obtained values is assigned to the corresponding point in the current replication;

  $*$ **DES + Linear Regression 2** calculates the value of each sampling point of the current replication when its set of runs has been executed. Unlike the first method, this technique involves the computation of each point of the current replication considering its entire set of runs. Linear regression is applied between multiple known points.

MTTA Simulator. The Mercury tool also provides a transient simulator, which evaluates the behavior of non-Markovian absorbing models. This simulator generates a large number of related statistics.

4.1. Supplementary Tools

An RVG is available in the Mercury tool, which is capable of generating random numbers from a range of probability distributions. The RVG module provides descriptive statistics from the generated data, and the sample data can be exported in order to be used in other applications. *Moment Matching* is another module from which it is possible to estimate what exponential-based probability distribution best fits the mean (first moment) and standard deviation (second moment) for a data sample.

5. Case Study

This section presents a case study to demonstrate the feasibility of adopting the Mercury tool for supporting the deployment of a cloud system. We investigated the gain in the availability by implementing a redundancy mechanism using a warm-standby strategy in the main component of a cloud architecture. Detailed information can be found in [20]. This case study considers an architecture with three nodes, where at least one node must be available for the cloud to work properly. An FT and a CTMC are adopted to represent

[10]Discret Event Simulation.

the hierarchical heterogeneous models. The FT describes the high-level components and is used to compute the total availability of the system, whereas the CTMC represents the components involved in the redundancy mechanism. Figure 3 shows the FT model with one non-redundant General Controller (GC) and its three nodes. It is important to stress that a warm-standby replication strategy cannot be properly represented by FT models, due to dependency between component states. In this case, a CTMC needs to be adopted to represent the redundant mechanism, with one active GC and one replicated GC host configured in warm-standby (see Figure 4). Through the computation of the availability of the redundant mechanism using the CTMC model, the availability of the cloud system can be known. Table 2 presents the measures obtained with Mercury. As we can see, there is a difference of 24 hours less in downtime when using redundancy.

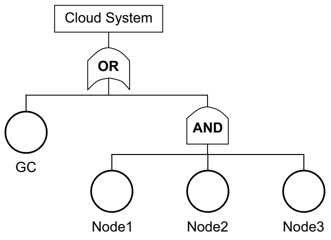

Figure 3. FT for the non-redundant cloud system.

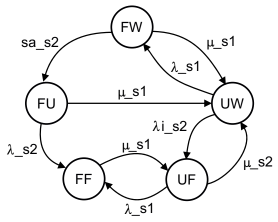

Figure 4. CTMC to redundant system (two hosts [20]).

Table 2. Availability Measures of the System with and without Redundancy.

Metric	**GC without redundancy**	**GC with redundancy**
Steady-state availability	0.997192102190714	0.9999731974218314
Number of 9's	2.5516187019805443	4.571823428711702
Annual downtime	24.61 h	0.23 h

6. Conclusions

In this paper, we introduce the recent enhancements to Mercury. We evolved the Mercury tool aiming to support discrete-time Markov chain (DTMC) and fault tree (FT) formalisms. In addition, among other updates, both SPN simulators were reimplemented, and now stationary and transient simulations can detect the occurrence of rare events. One case study demonstrates the feasibility of applying Mercury for supporting infrastructure planning. Many research projects have been supported by Mercury and have been published in peer-reviewed journals and conferences. Mercury has proven to be a useful tool for what it has been designed. As future works, we intend to implement support for project creation where multiple models of the same formalism can be created in a single project file.

References

[1] Molloy MK. Performance analysis using stochastic Petri nets. IEEE Computer Architecture Letters. 1982;31(09):913–917.

[2] Trivedi K. Probability and Statistics with Reliability, Queueing, and Computer Science Applications, ed.: JHON WILEY &SONS. INC; 2002.

[3] Maciel PR, Trivedi KS, Matias R, Kim DS. Dependability modeling. In: Performance and Dependability in Service Computing: Concepts, Techniques and Research Directions. IGI Global; 2012. p. 53–97.

[4] Distefano S, Xing L. A new approach to modeling the system reliability: dynamic reliability block diagrams. In: RAMS '06. Annual Reliability and Maintainability Symposium, 2006.; 2006. p. 189–195.

[5] Vesely WE, Goldberg FF, Roberts NH, Haasl DF. Fault tree handbook. Nuclear Regulatory Commission Washington DC; 1981.

[6] Callou G, Maciel P, Tutsch D, Ferreira J, Araújo J, Souza R. Estimating sustainability impact of high dependable data centers: a comparative study between brazilian and us energy mixes. Computing. 2013;95(12):1137–1170.

[7] Johnson MA, Taaffe MR. Matching moments to phase distributions: Mixtures of Erlang distributions of common order. Stochastic Models. 1989;5(4):711–743.

[8] Silva B, Matos R, Callou G, Figueiredo J, Oliveira D, Ferreira J, et al. Mercury: An integrated environment for performance and dependability evaluation of general systems. In: Proceedings of Industrial Track at 45th Dependable Systems and Networks Conference, DSN; 2015. .

[9] Maciel P, Matos R, Silva B, Figueiredo J, Oliveira D, Fé I, et al. Mercury: Performance and Dependability Evaluation of Systems with Exponential, Expolynomial, and General Distributions. In: 2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC); 2017. p. 50–57.

[10] Oliveira D, Matos R, Dantas J, Ferreira Ja, Silva B, Callou G, et al. Advanced Stochastic Petri Net Modeling with the Mercury Scripting Language. In: Proceedings of the 11th EAI International Conference on Performance Evaluation Methodologies and Tools. VALUETOOLS 2017. New York, NY, USA; 2017. p. 192–197. Available from: https://doi.org/10.1145/3150928.3150959.

[11] Sahner RA, Trivedi KS. Reliability modeling using SHARPE. IEEE Transactions on Reliability. 1987;36(2):186–193.

[12] German R, Kelling C, Zimmermann A, Hommel G. TimeNET: a toolkit for evaluating non-Markovian stochastic Petri nets. Performance Evaluation. 1995;24(1-2):69–87.

[13] Heiner M, Herajy M, Liu F, Rohr C, Schwarick M. Snoopy–a unifying Petri net tool. In: International Conference on Application and Theory of Petri Nets and Concurrency. Springer; 2012. p. 398–407.

[14] Grassmann WK, Taksar MI, Heyman DP. Regenerative Analysis and Steady State Distributions for Markov Chains. Oper Res. 1985 Oct;33(5):1107–1116. Available from: https://doi.org/10.1287/opre.33.5.1107.

[15] Marsan MA, Balbo G, Conte G, Donatelli S, Franceschinis G. Modelling with Generalized Stochastic Petri Nets. SIGMETRICS Perform Eval Rev. 1998 Aug;26(2):2. Available from: https://doi.org/10.1145/288197.581193.

[16] Kuo W, Zuo MJ. Optimal reliability modeling: principles and applications. John Wiley & Sons; 2003.

[17] Ferreira J, Callou G, Maciel P. A power load distribution algorithm to optimize data center electrical flow. Energies. 2013;6(7):3422–3443.

[18] Breuer L, Baum D. An introduction to queueing theory: and matrix-analytic methods. Springer Science & Business Media; 2005.

[19] Desrochers AA, Al-Jaar RY, Society ICS. Applications of petri nets in manufacturing systems: modeling, control, and performance analysis. IEEE Press; 1995. Available from: https://books.google.it/books?id=mL1TAAAAMAAJ.

[20] Dantas J, Matos R, Araujo J, Maciel P. Eucalyptus-based private clouds: availability modeling and comparison to the cost of a public cloud. Computing. 2015:1–20.

[21] Jain R. The art of computer systems performance analysis : techniques for experimental design, measurement, simulation, and modeling. New York: Wiley; 1991.