The Mercury Environment: A modeling tool for performance and dependability evaluation

Paulo Maciel [a,b,1], Eduardo Tavares [a,b], Jamilson Dantas [a,b], Bruno Silva [b], Carlos Melo [a,b], Danilo Oliveira [b,c], Paulo Pereira [a,b], Rubens Matos [b,d], and Thiago Pinheiro [a,b]

[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. Decision making on both systems and processes at organizational level requires means to obtain indicators, whether these are performance or dependability associated. Measurement can have a high temporal and/or financial cost, which may be unfeasible for some systems. In some situations, the system has not been implemented. As an alternative, mathematical models provide estimates about system behavior even before its actual creation or operation. In this sense, computational tools are essential for the construction and evaluation of such models. Mercury is a framework which allows the creation and evaluation of performance and dependability models such as Reliability Block Diagrams, Fault Trees, Stochastic Petri Nets, Continuous and Discrete time Markov Chains, as well as Energy Flow. Mercury has a graphical interface and a scripting language that enables integration with external applications. This paper presents the new functionalities provided by recent versions of Mercury tool.

Keywords. Models, Fault Tree, Discrete time Markov Chains

1. Introduction

A System is defined as a set of connected pieces that work togheter[1], which can also be applied to both software and hardware components that compose a computational system. Many tools were created to provide us valuable information about the performance and dependability attributes of computer systems. As a consequence Mercury tool has been developed and maintained by the MODCS research group, and our tool allows creation and evaluation of performance and dependability models, such as: Continuous and Discrete-Time Markov Chains (CTMC and DTMC) [2], Reliability Block Diagrams and Dynamical Reliability Block Diagrams (RBD and DRBD) [3,4], Energy Flow Models (EFM) [5], Stochastic Petri Nets (SPN) [6], and Fault Trees (FT) [7]. Mercury also

[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

offers other features, which include more than 25 probability distributions supported in SPN simulation, sensitivity analysis evaluation of CTMC, SPN, and RBD models, computation of reliability importance indices, as well as moment matching [8] of empirical data. The proposed environment has been adopted by MODCS group and the tool's web page[2] presents remarkable results obtained using Mercury. This paper presents the new functionalities provided by recent versions of Mercury tool. This work is organized as follows. Section 2 presents a comparison between Marcury and similar tools. Section 3 provides a view of Mercury main functionalities. Section 4 presents the new functionalities, and, finally Section 6 presents limitations and the future of the Mercury modeling environment.

2. Related Tools

There is a non-exhaustive list of 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. This Section presents a small set of these tools and compares them with the current version of the Mercury framework. The ReliaSoft BlockSim[3] is one of these tools and provides the way to evaluate some dependability attributes (reliability, availability, and maintainability) through RBDs and FTs models. The main limitation of the BlockSim is that it cannot evaluate components dependencies with its current modeling formalisms. In order to evaluate more complex scenarios we recommend tools that are capable to model state-space models, which is the case of Relex[4] and SHARPE [9]. Other tools like TimeNET [10] and Snoopy [11] provides powerful Petri Net (PN) modeling and evaluation mechanisms, but are limited to PN extensions. Table 1 presents a general comparison of Mercury and the others present tools.

Table 1. Modeling Tools comparison

Formalism / Tool	RBD	DRBD[5]	FT	CTMC	DTMC	SPN	EFM
BlockSim	✓		✓				
Relex	✓		✓	✓		✓	
SHARPE	✓		✓	✓		✓	
TimeNet						✓	
Snoopy						✓	
Mercury	✓	✓	✓	✓	✓	✓	✓

3. Overview and Software Architecture

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

[2]Publications: `http://www.modcs.org/?page_id=11`
[3]BlockSim: https://www.reliasoft.com/BlockSim
[4]Relex: http://www.relex.com

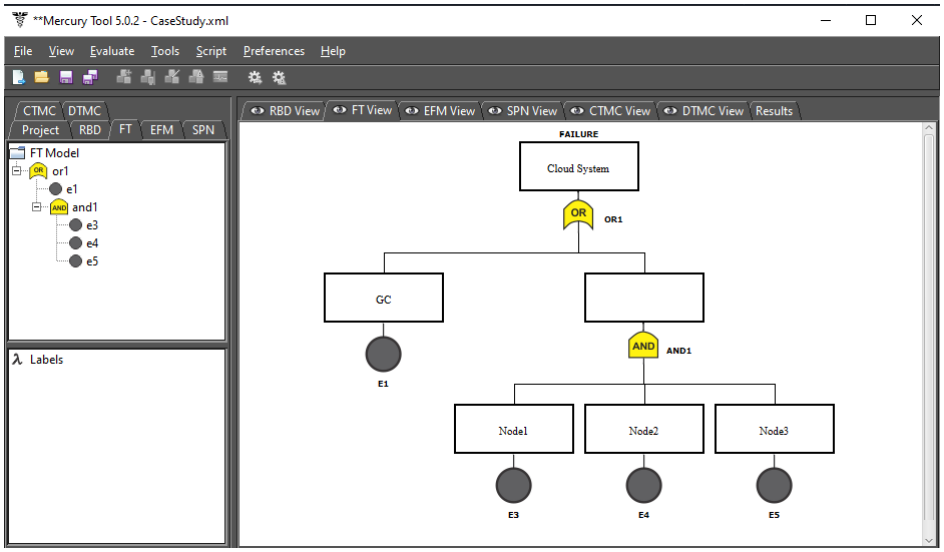

Figure 1. Mercury GUI

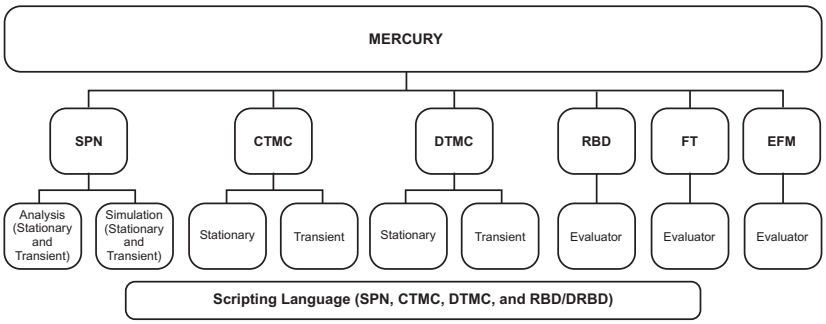

Figure 2. Mercury - Formalisms

SPN editor and evaluator allows modeling and evaluating generalized stochastic Petri nets (GSPNs). Mercury has implemented 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 evaluation and time-dependent metrics are obtained by performing transient evaluation. 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) [12] 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. Also, the SPN editor has a feature named token game to assist the validation of models. By using this feature, users can simulate graphically the firing of transitions, step by step. Thus, changing 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 behaviour 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. By performing this analysis, it is possible to evaluate structural properties of the model through analysis of place invariants, siphons and traps [13].

CTMC editor and evaluator allows modeling and evaluating CTMCs. Regarding stationary analysis, the following numerical techniques are supported: GTH and Gauss-Seidel. Regarding transient analysis, the following numerical techniques are supported: Uniformization and Runge-Kutta. Probabilities of absorption and MTTA may be computed when evaluationg absorbing CTMCs. Experiments are also supported. When defining custom metrics, users can use expression including state probabilities, making it possible to compose complex metrics. 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. Besides states and transitions, users can define reward rates assigned to states. In such a case, CTMCs become Markov reward models. Mercury also supports sensitivity analysis on CTMCs, making it possible to evaluate the sensitivity of state probabilities considering each input parameter entered.

DTMC editor and evaluator allows 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. Experiments are also supported. Expressions including state probabilities may compose metrics. Opposite to CTMCs, transitions between states are based on probability in DTMCs. Transition probabilities can be defined by means of algebraic expressions using symbolic parameters. Name of parameters can include greek letters.

RBD editor and evaluator allows modeling and evaluating availability and reliability using block diagrams. The tool supports three types of blocks configurations: series, parallel, and K-out-of-N (KooN). The RBD model is composed only by one input and one output node. 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 (method based on the structural function) and SDP (sum of disjoint products). 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. Also, 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 indicates 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 on reliability or availability. Evaluation of de-

pendability bounds [14] is a method to calculate dependability metrics when the model is huge. By applying such a method, approximations to the exact value 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.

Fault Tree editor and evaluator allows 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. By using the Mercury tool, it is possible to handle two types of nodes: basic events and gates (logic ports). Leaf nodes represent basic events. On the other hand, each supported gate has its own graphical representation. 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. All features and evaluations available for RBDs are also available for FTs. Thus, for sake of conciseness, they will not be described. It is important to highlight that the tool provides a feature that convert the FT model to RBD.

EFM editor and evaluator evaluates availability, sustainability, and cost of cooling infrastructures and datacenter/clouds power, considering the power constrains 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 [5]. *Combined Evaluation* provides an integrated evaluation of dependability, sustainability impact, and cost of cooling infrastructures and data center/cloud power [5]. *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) [15], power load distribution algorithm depth (PLDAD), and GRASP based algorithm.

Mercury scripting language was designed to facilitate the creation and evaluation of complex models. It allows greater flexibility in model evaluations by 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 which is available inside the Mercury tool. The tool has a feature that automatically generate 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 struc-

ture in a model. It is important to highlight that for some tools [10,16] 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 parameters values on certain metrics.
- **Support for loop and conditional structures.** It allows the creation of nets with variable structure. The variables for controlling those structures can be treated as parameters of the model.
- **Support for phase-type distributions [17].** 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 is based on matching the moments of continuous time phase-type distributions. Models having non-Markovian properties may only be evaluate numerically through the adoption of phase-type approximation technique [18].

A script has the following structure: (i) models, each one with its metrics and parameters; (ii) the main section where values for input parameters are defined. Listing 1 presents a CTMC extracted from [19].

```
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;
    metric p1 = stationaryProbability( st = fw );
}
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;
    a1 = solve( model = RedundantGC, metric = aval );
    println(a1);
}
```

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 the support for two new formalisms (DTMC and fault tree). Also, the stationary and transient simulators were reimplemented. Several improvements were also performed on the usability of the tool, such as the possibility to assign a description to each component of a model. It represents additional information about the component for the comprehension about 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 functionalites and improvements are presented.

DTMC. An editor for DTMCs is among the main features that was included in Mercury. The analysis of DTMC models comprehend the computation of holding time and recurrence time [2], besides the usual state probabilities that are already obtained for CTMC models. Section 3 provides more information about the types of evaluations available for DTMCs.

Fault Tree. An editor for fault trees is another main feature that was included in Mercury. FT is a top-down logical diagram and it makes 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*. Section 3 provides more information about the types of evaluations available for FTs.

RBD. 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.

SPN. Several improvements were performed on the SPN editor and evaluator. Regarding the drawing area, Mercury now supports two types of arcs 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, an 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 simulator were reimplemented. Besides several improvements, such as on the GUI and possibility to export detailed information of the simulation. Besides, stationary and transient simulators can detect occurent of rare events, which occur when the difference between the delays assigned to the transitions is huge.

- **Stationary Simulator.** Stationary simulation can be used when evaluating steady-state metrics of non-Markovian SPN models. Mercury implements the method of batch means [20]. 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 is 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 by a set of replications and each replication is composed by a set of runs [20]. Each run runs 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[6] + Linear Regression 1** computes the value of each sampling point at the end of each run by adopting linear interpolation between two known points. When the number of runs of a replication has been executed, the values of each point are collected in the current set of runs, and its mean value is assigned to that same 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. Oposite to the first method, this one computes the value of each point of the current replication considering its entire set of runs. This method adopts linear regression between multiple known points.

  MTTA Simulator. Mercury provides a type of evaluation in the transient simulator to evaluate the behavior of non-Markovian absorbing models, and from there generating a large number of statistics.

4.1. Supplementary Tools

Mercury environment has a supplementary tool, namely, random variate generator, which is capable of generating random numbers contemplating several probability distributions. This tool provides statistical summaries for random numbers (e.g., standard deviation); it can also export the generated data to files so external applications might use them. Furthermore, the SPN simulation uses this functionality, in which the user might choose distinct probability distributions associated with transitions. Another important supplementary tool offered in the Mercury environment is moment matching, in which users may estimate what exponential-based probability distribution best fits the mean (first moment) and standard deviation (second moment) for a data sample.

[6] Discret Event Simulation

5. Case Study

This section presents a case study to demonstrate the feasibility of adopting the Mercury tool to create and evaluate models for some systems. We evaluate a cloud computing architecture presented in [19] which implements a redundant mechanism in the main component using a warm-standby strategy. A FT and a CTMC are adopted to represent the hierarchical heterogeneous models. The FT describes the high-level components, whereas the CTMC represents the components involved in the redundancy mechanism. This case study considers an architecture with three nodes, where at least one node must be available for the cloud to work properly.

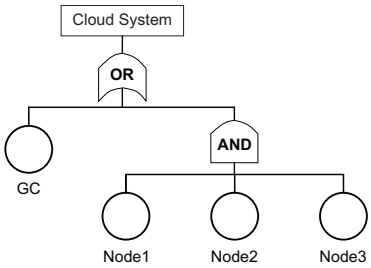

Figure 3. FT for the non-redundant cloud system

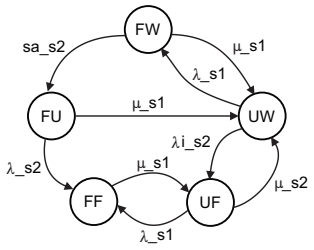

Figure 4. CTMC to redundant system (two hosts [19])

Figure 3 shows the complete 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 architecture, with one GC active and one replicated GC host configured in warm-standby [19] (see Figure 4). Table 2 presents the measures obtained with Mercury.

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 work, we presented the main features and updates of Mercury. Mercury environment has been widely adopted in numerous research projects, which had the results published in peer-reviewed journals and conferences. The MODCS research group is frequently improving the tool by adding new features and updating existing functionalities. New versions are usually released every six months, but this is a flexible policy adapted according to the user demands. This modeling tool can support the academy and industry struggles on capacity planning of many systems such as performance, reliability and availability evaluation, and data-center energy flow planning.

References

[1] System def. `https://dictionary.cambridge.org/pt/dicionario/ingles-portugues/system`. Accessed: 2021-02-26.

[2] K Trivedi. Probability and statistics with reliability, queueing, and computer science applications, ed.: Jhon wiley &sons, 2002.

[3] Paulo RM Maciel, Kishor S Trivedi, Rivalino Matias, and Dong Seong Kim. Dependability modeling. In *Performance and Dependability in Service Computing: Concepts, Techniques and Research Directions*, pages 53–97. IGI Global, 2012.

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

[5] Gustavo Callou, Paulo Maciel, Dietmar Tutsch, João Ferreira, Julian Araújo, and Rafael Souza. Estimating sustainability impact of high dependable data centers: a comparative study between brazilian and us energy mixes. *Computing*, 95(12):1137–1170, 2013.

[6] Michael K. Molloy. Performance analysis using stochastic petri nets. *IEEE Computer Architecture Letters*, 31(09):913–917, 1982.

[7] William E Vesely, Francine F Goldberg, Norman H Roberts, and David F Haasl. Fault tree handbook. Technical report, Nuclear Regulatory Commission Washington DC, 1981.

[8] Mary A Johnson and Michael R Taaffe. Matching moments to phase distributions: Mixtures of erlang distributions of common order. *Stochastic Models*, 5(4):711–743, 1989.

[9] Robin A Sahner and Kishor S Trivedi. Reliability modeling using sharpe. *IEEE Transactions on Reliability*, 36(2):186–193, 1987.

[10] Reinhard German, Christian Kelling, Armin Zimmermann, and Günter Hommel. Timenet: a toolkit for evaluating non-markovian stochastic petri nets. *Performance Evaluation*, 24(1-2):69–87, 1995.

[11] Monika Heiner, Mostafa Herajy, Fei Liu, Christian Rohr, and Martin Schwarick. Snoopy–a unifying petri net tool. In *International Conference on Application and Theory of Petri Nets and Concurrency*, pages 398–407. Springer, 2012.

[12] Winfried K. Grassmann, Michael I. Taksar, and Daniel P. Heyman. Regenerative analysis and steady state distributions for markov chains. *Oper. Res.*, 33(5):1107–1116, October 1985.

[13] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. Modelling with generalized stochastic petri nets. *SIGMETRICS Perform. Eval. Rev.*, 26(2):2, August 1998.

[14] Way Kuo and Ming J Zuo. *Optimal reliability modeling: principles and applications*. John Wiley & Sons, 2003.

[15] Joao Ferreira, Gustavo Callou, and Paulo Maciel. A power load distribution algorithm to optimize data center electrical flow. *Energies*, 6(7):3422–3443, 2013.

[16] Anne Vinter Ratzer, Lisa Wells, Henry Michael Lassen, Mads Laursen, Jacob Frank Qvortrup, Martin Stig Stissing, Michael Westergaard, Søren Christensen, and Kurt Jensen. Cpn tools for editing, simulating, and analysing coloured petri nets. In *Applications and Theory of Petri Nets 2003*, pages 450–462. Springer, 2003.

[17] Lothar Breuer and Dieter Baum. *An introduction to queueing theory: and matrix-analytic methods*. Springer Science & Business Media, 2005.

[18] A.A. Desrochers, R.Y. Al-Jaar, and IEEE Control Systems Society. *Applications of petri nets in manufacturing systems: modeling, control, and performance analysis*. IEEE Press, 1995.

[19] Jamilson Dantas, Rubens Matos, Jean Araujo, and Paulo Maciel. Eucalyptus-based private clouds: availability modeling and comparison to the cost of a public cloud. *Computing*, pages 1–20, 2015.

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