An Evaluation Framework for Comparative Analysis of Generalized Stochastic Petri Net Simulation Techniques

Ricardo J. Rodríguez, *Member, IEEE*, Simona Bernardi, and Armin Zimmermann, *Member, IEEE*

***Abstract*—Availability of a common, shared benchmark to provide repeatable, quantifiable, and comparable results is an added value for any scientific community. International consortia provide benchmarks in a wide range of domains, being normally used by industry, vendors, and researchers for evaluating their software products. In this regard, a benchmark of untimed Petri net models was developed to be used in a yearly software competition driven by the Petri net community. However, to the best of our knowledge there is not a similar benchmark to evaluate solution techniques for Petri nets with timing extensions. In this paper, we propose an evaluation framework for the comparative analysis of generalized stochastic Petri nets (GSPNs) simulation techniques. Although we focus on simulation techniques, our framework provides a baseline for a comparative analysis of different GSPN solvers (e.g., simulators, numerical solvers, or other techniques). The evaluation framework encompasses a set of 50 GSPN models including test cases and case studies from the literature, and a set of evaluation guidelines for the comparative analysis. In order to show the applicability of the proposed framework, we carry out a comparative analysis of steady-state simulators implemented in three academic software tools, namely, `GreatSPN`, `PeabraiN`, and `TimeNET`. The results allow us to validate the trustfulness of these academic software tools, as well as to point out potential problems and algorithmic optimization opportunities.**

***Index Terms*—Benchmarking, generalized stochastic Petri nets (GSPNs), performance, simulation software.**

I. Introduction

BENCHMARKS have been recognized as an effective tool for conducting experiments in computer science since they provide repeatable, quantifiable and, thus, comparable results [1]. Nowadays, there is a strong need for common benchmarks in different scientific communities supporting a common, shared way to provide reproducible and comparable data. For instance, they play an important role in the measurement-based testing within performance software engineering [2].

International consortia provide benchmarks in a wide range of domains, e.g., high-performance computing [3], transaction processing and databases [4], and image-processing and cyber security [5]. Such standardized benchmarks are used by industry and vendors for assessing the performance of their hardware/software products, as well as by researchers to compare their software techniques and tools.

The development of a new benchmark (*proto-benchmark*) can be initiated by a small group of researchers as an offer to a larger scientific community to initialize a consensus process [6]. In the model checking research field, for instance, a benchmark including a repository of untimed Petri net models was developed and used in a yearly software competition by the Petri net community [7] to compare the efficiency of new solution techniques. Such benchmark was aimed at checking the qualitative properties of the Petri Net models, such as the reachability of deadlocks or liveness, to name a few.

Petri nets [8] are a graphical and mathematical formalism that easily represent common characteristics of computer systems such as concurrency, synchronization, conditional branches, looping, and sequencing. In particular, Petri nets are a suitable model for the design and verification of real-life processes [9], being a common formalism used in automated manufacturing systems [10], [11]. Roughly speaking, a Petri net (PN) is a bipartite graph of places and transitions connected with arcs, describing the system behavior with concurrency and synchronous capabilities. Tokens are assigned to places, and a distribution of tokens in the places represents a state of the system, whereas transitions model events or activities. Stochastic Petri nets (SPN) are a timed extension of Petri nets, where each transition has associated an exponentially distributed firing time delay (e.g., modeling the duration of an activity).

To the best of our knowledge, there is not a similar benchmark to evaluate solution techniques for SPN models. In this paper, we aim at providing a proto-benchmark to fill this gap. This proto-benchmark considers a class of SPN, namely generalized SPNs [12] (GSPN), that are used for systems

Manuscript received November 4, 2017; accepted April 30, 2018. The work of R. J. Rodríguez and S. Bernardi was supported in part by the EU Horizon 2020 Research and Innovation Programme under Grant 644869 (DICE), in part by the Spanish Ministry of Economy, Industry, and Competitiveness Project CyCriSec under Grant TIN2014-58457-R, and in part by the Aragon Government Ref. T27—DisCo Research Group. This paper was recommended by Associate Editor M. P. Fanti. *(Corresponding author: Ricardo J. Rodríguez.)*

R. J. Rodríguez is with the Centro Universitario de la Defensa, General Military Academy, 50090 Zaragoza, Spain (e-mail: rjrodriguez@unizar.es).

S. Bernardi is with the Departamento de Informática e Ingeniería de Sistemas, University of Zaragoza, 50018 Zaragoza, Spain (e-mail: simonab@unizar.es).

A. Zimmermann is with the Department of System and Software Engineering, Technische Universität Ilmenau, 98684 Ilmenau, Germany (e-mail: armin.zimmermann@tu-ilmenau.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMC.2018.2837643

performance and reliability analysis. In addition to timed transitions, GSPN include immediate transitions that are used to model logical actions occurring in zero time.

In particular, we propose an evaluation framework that consists of a repository of GSPN models and a guideline for the solvers evaluation. The models of the repository are available in different formats, two of them compliant to the ISO standard place/transition nets with XML (PNML) [13]. Furthermore, we also provide a tool to interchange the models between different tools and the shell scripts used to launch experimentation as a way to make easier the reproducibility of the experiments carried out in this paper. As an example analysis, this paper furthermore applies the proposed evaluation framework to three selected event-driven simulators, namely `GreatSPN` [14], `PeabraiN` [15], and `TimeNET` [16]. A comparative analysis among these three tools is carried out and results are reported from the user perspective. This mainly includes run times and achieved result accuracy which may greatly differ as the selected tools implement various variants of Monte Carlo simulations. A comparison of the algorithms themselves is outside the scope of this paper.

Let us remark that although the evaluation framework is applicable for any GSPN solution method, in this paper we focus on GSPN simulation solvers to illustrate its usage.

Along this paper, we will use the notions for the description of benchmarking proposed in [17] to present and discuss the different aspects of our evaluation framework. A benchmark consists of the following data.

1) I, an input set of data to be processed.
2) E, an examination to be passed by a program.
3) P, the program to be benchmarked.
4) $\langle I, E, P\rangle$, a run corresponding to the execution of a P for an E with a given I.

This paper is organized as follows. Section II describes the evaluation framework, in particular by providing a general guideline for defining I and E. Section III briefly covers GSPN simulation and the selected tools (P) that are used in the comparative analysis. Section IV describes the application of the guideline for the comparative analysis of the GSPN simulator and details the experiments ($\langle I, E, P\rangle$). Related work is covered in Section V. Finally, Section VI sets out the conclusions of this paper.

II. Evaluation Framework

In this section, we present the evaluation framework that enables us to specify the input set of data I to be processed by a GSPN solver P and the examination E to be passed by a solver. We first describe the repository of GSPN models and, then, we introduce the guideline for the evaluation of solvers. The reader interested in examining in depth the GSPN formalism can refer to [12].

A. Generalized Stochastic Petri Net Repository

The repository consists of a collection of 50 GSPN models, listed in Table I. For each model, we indicate its identifier, its Petri net subclass, its size (in terms of places and transitions), its qualitative properties, and if it is a parametric model. Furthermore, we indicate the modeled system: most of the models come from case studies and test cases from the literature (see *reference* columns in the table).

Each model of the repository is characterized by an identifier (*Net_id*) and labeled (*PN subclass*) according to the structural Petri Net subclass it belongs. In particular, most the models belong to the well-known PN-subclasses of monoT-semiflows [37], Free-Choice nets (FC) [38], freely related T-semiflows [39], and deterministic systems of sequential processes (DSSP) [40]. The main reason is that the repository has been built on a collection of models that was used in [31] to assess bounding techniques on interval-based time Petri net models, which are known to produce tight bounds for these PN-subclasses. Nevertheless, we consider the repository as an open proposal, subject to future extensions to encompass also other PN-subclasses. Finally, we classified as *general* nets the rest of the GSPN models that do not belong to a specific PN-subclass.

The repository is freely available at [41]. A wiki Web page was created to provide further detailed information for each model, such as the number of places/transitions (*model size* column, in Table I), the satisfied logical properties (*logical properties* column)—structural boundedness (*B*), deadlock-freeness (*DF*), liveness (*L*), and reversibility (*R*) [42]—, and the parameters, in case of parametric model (*parametric* column). In particular, for parametric models, three different variants of initial marking setting are provided which correspond to a coarse classification of the model workload, namely *low*, *medium*, and *high*. The *low* category includes the models where the initial marking parameters have been set to the reference values, that is the values in the original models.[1] The *medium* and *high* categories include the models where the initial marking parameters have been set to, respectively, one and two orders of magnitude greater than the reference values.

The repository includes the nets in the file formats required for the three tools used in this paper, i.e., `GreatSPN` [14], `PeabraiN` [15], and `TimeNET` [16]. The file format of `GreatSPN` does not follow any particular standard, while the file formats supported by `PeabraiN` and `TimeNET` are compliant to the adopted ISO standard for description of PNML [13] or have a corresponding import/export converter. It is worth mentioning that the last published version of `GreatSPN` incorporates a translator module from PNML-compliant file format to its own file format [43]. Since there does not exist at the moment of this writing any adopted ISO standard for timing specification in PNML, as the one proposed in [44], the file formats of both tools differ in this part. A tool for converting the file formats between some of these tools was developed as a side product of the proposed framework to overcome this issue. This tool is also available at the GSPN website repository and its source code has been released under the GNU/GPLv3 license.

B. Guideline for Evaluation of Simulation Solvers

The guideline for the solvers evaluation is aimed at providing a support for the definition of the examination E, to be

[1]The nonparametric models have been also included in this category.

TABLE I
GSPN REPOSITORY

Net_id	PN subclass	Model size (nP,nT)	Logical properties B-DF-L-R	Parametric	Modeled system	Reference
1	general	(49,41)	DF-L		robot system I	[18]
2	general	(42,35)	DF-L		robot system II	[18]
3	FRT	(37,34)	B-DF-L-R	✓	flexible manufacturing system	[18]
4	FRT	(12,11)	B-DF-L-R	✓	communication protocol	[19]
5	monoT-semiflow	(22,13)	B-DF-L-R	✓	job shop	[20]
6	monoT-semiflow	(16,12)	B-DF-L-R	✓	alternating bit protocol	[20]
7	FC	(25,22)	B-DF-L-R	✓	flexible manufacturing system	[21]
8	FRT	(167,109)	B-DF-L-R	✓	complex system scenario	*
9	monoT-semiflow	(72,53)	B-DF-L-R		computer assisted braking system	[22]
10	monoT-semiflow	(36,26)	B-DF-L-R	✓	simple system scenario	[22]
11	FRT	(18,19)	DF-L	✓	example	[23]
12	FC	(6,6)	B-DF-L-R	✓	example	[20]
13	FC	(9,9)	B-DF-L-R	✓	example	[20]
14	general	(29,26)	B-DF	✓	flexible manufacturing system	[24]
15	FRT	(89,88)	B-DF-L-R	✓	e-health system	[25]
16	general	(72,65)	B-DF		gas-pump system	[26]
17	general	(67,75)	DF-R		backbone distributed FT algorithm	[27]
18	FRT	(21,24)	DF	✓	message redundancy system	[28]
19	FC	(11,10)	B-DF-L-R	✓	example	[20]
20	FC	(26,28)	B-DF-L-R	✓	dataflow graph	[20]
21	FC	(13,12)	B-DF-L-R	✓	example	[20]
22	FC	(36,29)	B-DF-L-R	✓	example	[20]
23	FC	(53,33)	B-DF-L-R	✓	example	[20]
24	DSSP	(9,7)	DF-L	✓	example	[20]
25	DSSP	(28,28)	B-DF-L-R	✓	example	[20]
26	DSSP	(39,29)	B-DF-L-R	✓	example	[20]
27	DSSP	(16,15)	B-DF-L-R	✓	example	[20]
28	FC (EQ)	(21,14)	B-DF-L-R	✓	example	[20]
29	monoT-semiflow	(13,19)	B-DF-L-R	✓	Ada tasking system	[20]
30	monoT-semiflow	(11,8)	B-DF-L-R	✓	producer-consumer model	[20]
31	monoT-semiflow	(16,13)	B-DF-L-R	✓	example	[20]
32	FRT	(13,12)	B-DF-L-R	✓	example	[20]
33	FRT	(14,14)	B-DF-L-R	✓	example	[20]
34	FRT	(48,40)	B-DF-L-R	✓	software retrieval system (e-commerce)	[29]
35	monoT-semiflow	(12,9)	B-DF-L-R	✓	example	[20]
36	FRT	(30,30)	B-DF-L-R	✓	example	[20]
37	FRT	(13,13)	B-DF-L-R	✓	example	[20]
38	FC	(10,10)	B-DF-L-R	✓	example	[20]
39	FRT	(14,16)	B-DF-L-R	✓	example	[20]
40	monoT-semiflow	(37,20)	B-DF-L-R	✓	assembly line push strategy	[30]
41	monoT-semiflow	(35,19)	B-DF-L-R	✓	assembly line on-demand strategy	[30]
42	monoT-semiflow	(31,20)	B-DF-L-R	✓	assembly line Kanban strategy	[30]
43	FRT	(86,74)	B-DF-L-R	✓	flexible manufacturing cell	[30]
44	FRT	(140,110)	B-DF-L-R	✓	oil pipeline network	[31]
45	FRT	(150,117)	B-DF-L-R		oil pipeline network under attack	[31]
46	monoT-semiflow	(21,16)	B-DF-L-R	✓	Universal Control Hub	[32]
47	monoT-semiflow	(82,63)	B-DF-L-R	✓	web-service	[33]
48	FRT	(63,55)	B-DF-L-R	✓	mobile agent application	[34]
49	FC	(27,33)	B-DF-L-R	✓	Biomart Emboss workflow	[35]
50	FRT	(46,48)	B-DF-L-R	✓	clinical guideline	[36]

*Constructed following the approach given in [22]

passed by the GSPN solvers under analysis, and the selection of the input set of GSPN models *I* from the repository. It can be summarized as a list of steps, that are discussed in detail in this section.

1) Define the examination *E*.
   a) Decide the type of comparative analysis.
   b) Define the set of indexes for the assessment.
   c) Establish the conditions for passing the examination.
2) Select the input set *I*.
   a) Check the prerequisites for GSPN solvers.
   b) Define a selection criterion considering the features of the GSPN repository (Table II).

1) Define the Examination: The examination *E* should be defined first, since it determines the goals of the analysis. It implies the selection of the set of indexes for the assessment and the specification of the conditions to be passed by the GSPN solvers. In particular, we can distinguish two types of analysis: 1) correctness and 2) efficiency.

TABLE II
FEATURES OF THE GSPN REPOSITORY

Feature	Description
PN subclass	Structural PN characterization (i.e., monoT, FC, FRT, DSSP, general)
Model size	Number of places and transitions
Workload size	Parametrization of the initial marking

The first type of analysis is aimed at assessing the correctness of the GSPN solvers and/or accuracy of the performance results computed by the GSPN solvers. Therefore, at least an index needs to be defined for the assessment that is based on performance properties. The basic performance properties of a GSPN model are the transitions throughput (i.e., number of firings per unit of time) and the mean number of tokens in places. We have considered only the throughput of a transition as the reference property for the GSPN models of the repository. Let us remark that this decision is mainly motivated because, for those GSPN models that represent case studies, the reference

property has a precise meaning in the context of the modeled system (e.g., finished products per time unit for those models representing flexible manufacturing systems, such as the nets 3, 7, 14, and 40–43). Based on such a reference property, an example of index to be used for the tool assessment is the relative error of the value estimated by a GSPN solver under analysis with respect to a (known) value.

The second type of analysis is aimed at assessing the efficiency of the GSPN solvers and, similarly to the correctness analysis, performance indexes need to be defined for the assessment. Several performance indexes of interest are proposed in [45] for benchmarking, being the most common ones the CPU time and wall time (i.e., the elapsed time between start and end of a task) for measuring time efficiency, and the peak memory consumption for measuring space efficiency.

Once the indexes for the assessment are defined, the last task is to establish the conditions for passing the examination. A naïve condition is to define, for example, a maximum threshold for the relative error used as index for the correctness analysis. More complex conditions may consider the sensitivity of the GSPN solvers to one or more features of the repository.

2) Select the Input Set: A different input set of GSPN models can be selected from the repository, according to the GSPN solvers constraints (such as the fulfillment of logical properties) and the features of the repository that are of interest for the assessment.

Therefore, a first task is the selection of those GSPN models that fulfill the *logical properties* which are a prerequisite for the use of the GSPN solver under analysis. As shown in Table I, the considered logical properties are satisfied by the majority of the models; however, there are also some nets that only satisfy a subset of properties. Consider for instance a state-based GSPN solver. Hence, in this case only bounded models shall be selected for analysis.

The second selection task is carried out with the help of the features of the repository summarized in Table II. Since the features can be also used for the sensitivity analysis of the GSPN solvers, the choice of using a feature as selection or sensitivity criterion depends on the goals of the assessment.

III. GSPN Simulation

This section informally introduces simulation techniques for GSPN models, and the selected tools to be benchmarked with the evaluation framework. Three GSPN tools are considered in this paper: 1) `GreatSPN` (batch simulator); 2) `PeabraiN` (replication simulator with discrete stochastic simulation algorithm); and 3) `TimeNET` (batch simulator with spectral variance analysis), mainly because of their accessibility to the authors as well as their distribution in the research community. The tools implement different variants of steady-state simulation algorithms, for which the individual optimization details are outside the scope of this paper. Moreover, the tools usually contain several variants such as rare-event simulation and transient as well as stationary simulations, out of which we only consider the (standard) steady-state simulation here.

Monte Carlo simulation [46] can be used as a solution technique for (G)SPN models and is often the alternative choice to state-based techniques when the state space of the model is too large to be analyzed as a whole. Simulation enables to estimate performance and reliability measures by generating and analyzing randomly chosen paths through the state space. For each measure of interest, the simulation provides the estimated value and the precision error at a confidence interval, i.e., the real (unknown) value falls into this interval with a certain probability (i.e., confidence level) and with a precision error [46]. The width of the confidence interval is a measure of the accuracy of the estimated value.

Different simulation approaches exist. A common approach is *replication*, where N statistically independent simulation runs are executed and the measure is estimated considering the N independent measurements collected during different runs. A parallel variant simulates N models concurrently, and samples from the resulting independent paths. Other approaches can be used when the aim of the analysis is the estimation of steady state measures. Indeed, in such cases just a single simulation run can be executed, where the simulation time is "long enough" to bring the modeled system in steady state and measurements are taken across time (rather than measurements taken across replications). For example, the *batch method* [47] partitions the simulation interval into successive epochs and for each epoch a measurement is collected to avoid common pitfalls of correlated subsequent samples. The computational cost of the simulation depends on various factors.

1) The length of transient period, i.e., the simulation time until the steady state is reached.
2) The length of the simulation time needed to collect a representative sample.
3) The length of the epochs needed to ensure statistical independence of successive measurements.

It is worth to observe that such factors depend on the model under analysis, so the choice of simulation input parameters is a critical point since both statistical quality of the simulation results and computer time that is required to simulate a desired amount of system time rely on them. Concerning the choice of the length of the epochs, a possible approach is the regenerative points method [48], where a regeneration point is a time instant at which the system enters a given state (e.g., in an SPN model, the initial marking).

Statistical issues and very long simulation times may arise in case of rare events, for example when the measure to be estimated is a very small probability such as the system failure probability. So-called variance reduction techniques have been developed to overcome these issues [49], as well as specific methods such as RESTART [50] that allows for estimating probabilities as low as 10^{-100} within a reasonable time. Other possible approaches propose an approximate accelerated stochastic simulation approaches to reduce computation time [51].

IV. Comparative Analysis

In this section, we use the proposed GSPN benchmark for a comparative analysis of the three GSPN simulator tools

considered in this paper (namely, GreatSPN, PeabraiN, and TimeNET). The details of each simulation algorithm have been described in Section III.

We compare the tools under correctness and efficiency criteria. Regarding the correctness criterion, we are interested in *how good* the results provided by the simulation tools are. Hence, we compute the analytic results for each net in the benchmark (when feasible) and use these values as a reference for comparison. Regarding the efficiency criterion, we measure the execution time and the maximum memory consumption of the simulation tools for the benchmark models.

A. Correctness and Accuracy of Simulation Results

To evaluate the accuracy of the simulation results, we first need to compute analytic results of the nets and validate them. For this purpose we analytically solve the underlying CTMC of the nets (when feasible) using the analytic solvers of GreatSPN and TimeNET.

Table III summarizes the comparison of analytic results computed by GreatSPN and TimeNET solvers, considering the benchmark nets with low marking. As indicated, for some models the analytic results could not be obtained because these nets are unbounded or the state-space explosion problem applies (i.e., huge state space and P-semiflows explosion). These results are highlighted (in light blue) in Table III. For instance, *net43*, *net44*, and *net45* have a huge state-space such that the analytic solvers exhaust the available memory when solving the underlying CTMC. For two particular models, the TimeNET solver was unable to compute analytical results: *net26* has a huge state space that TimeNET is unable to handle properly and the performance measurement in *net17* and *net48* is defined over an immediate transition. However, TimeNET does not compute throughput values of immediate transitions directly. The last column in Table III shows the relative error between both solvers. As it is shown, both tools achieve a very similar value for the reference transition throughput considered in each model. Hence, we conclude that these analytic results are valid for comparison.

TABLE III
COMPARISON OF ANALYTIC RESULTS

Net name	GreatSPN results	TimeNET results	Relative error
net01	Unbounded net		
net02	Unbounded net		
net03	0.155084	0.155085	0.00%
net04	0.201510	0.201510	0.00%
net05	0.096975	0.096976	0.00%
net06	3.861004	3.861004	0.00%
net07	0.176757	0.176757	0.00%
net08	Place invariants explosion		
net09	0.241935	0.241935	0.00%
net10	0.036225	0.036225	0.00%
net11	Unbounded net		
net12	0.221345	0.221345	0.00%
net13	0.086957	0.086956	0.00%
net14	0.074986	0.075072	0.11%
net15	0.000955	0.000955	0.04%
net16	0.006174	0.006168	-0.10%
net17	0.033642	–	(n/a)
net18	Unbounded net		
net19	0.864793	0.864801	0.00%
net20	0.052561	0.052560	0.00%
net21	0.500105	0.500105	0.00%
net22	0.157043	0.157042	0.00%
net23	0.149841	0.149837	0.00%
net24	Unbounded net		
net25	0.581108	0.581107	0.00%
net26	0.157423	–	(n/a)
net27	0.215171	0.215171	0.00%
net28	0.479455	0.479451	0.00%
net29	0.250000	0.250000	0.00%
net30	0.122384	0.122384	0.00%
net31	0.227315	0.225658	-0.73%
net32	0.301352	0.301349	0.00%
net33	0.251126	0.251132	0.00%
net34	0.001204	0.001204	-0.01%
net35	0.065193	0.065193	0.00%
net36	0.083770	0.083770	0.00%
net37	0.205128	0.205128	0.00%
net38	0.200000	0.200000	0.00%
net39	0.242236	0.242236	0.00%
net40	0.273337	0.272827	-0.19%
net41	0.308482	0.308120	-0.12%
net42	0.266471	0.266362	-0.04%
net43	Huge state-space		
net44	Huge state-space		
net45	Huge state-space		
net46	0.030303	0.030303	0.00%
net47	0.062500	0.062500	0.00%
net48	0.187958	–	(n/a)
net49	0.000317	0.000317	0.03%
net50	3.241418	3.241418	0.00%

Fig. 1 shows the comparison of simulation results with regard to analytic ones. We simulated 100 times each net considering a low marking (see Section II-A) with two different precision errors and confidence levels. In particular, we set a precision of 5% at the 95% confidence level and of 1% at the 99%. Both results are plotted in Fig. 1(a) and (b), respectively. The mark in each error bar represents the average of the simulation results, while the lower and upper vertical error bars represent the quantile of 95% (99%) and 5% (1%) of the collected results, respectively. Furthermore, we also plotted an horizontal line indicating the predefined precision error of 5% and 1% in each graph.

The plots show interesting results. Regarding the precision of 5% at the 95% confidence level [Fig. 1(a)], for all simulation tools the average of relative errors are close to 0%. Namely, only for the *net10* and *net17* models the relative errors reach almost 3% and 5%, respectively. The results of the PeabraiN simulation tool show, however, some models for which the average of error results are above/below the precision boundaries (see *net10*, *net14*, or *net16*). In any event, the average error results are always in the 10% boundaries. Considering the quantiles, the PeabraiN simulation tool is the one showing more models whose quantiles exceed the precision boundaries.

Furthermore, the average results of PeabraiN simulation tool for the *net49* model are outside the precision boundaries [reaching almost 20% of precision error, which was left outside the axes boundaries for the sake of visibility in both Fig. 1(a) and (b)]. This indicates that the simulation is not correct. In summary, the results show that in general the stopping criteria of all simulation tools are accurate, but the PeabraiN simulation tool needs further analysis.

Regarding the precision of 1% at the 99% confidence level [Fig. 1(b)], the results show that the GreatSPN simulation tool has the most accurate intervals, since for almost

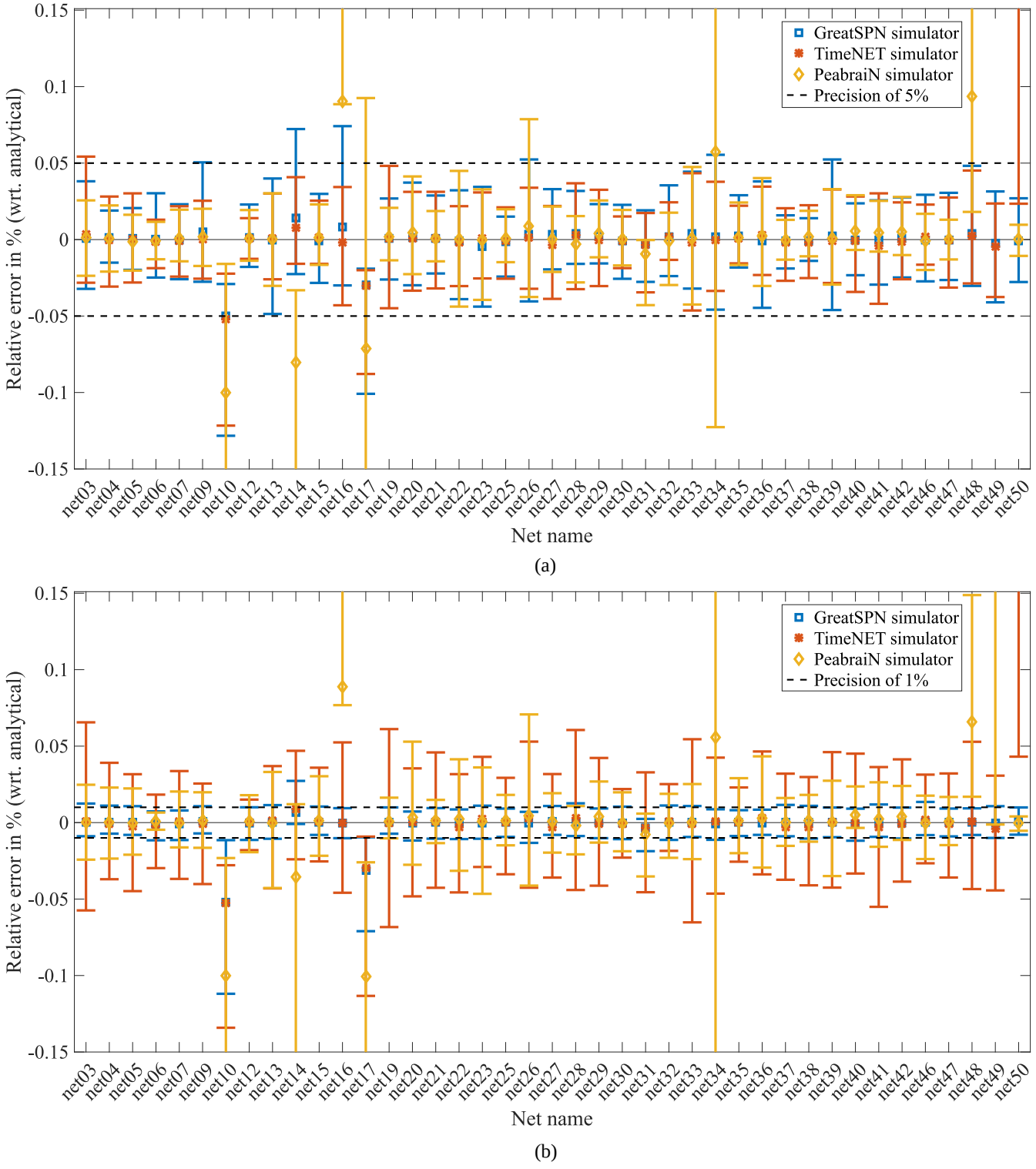

Fig. 1. Comparative simulation results for `GreatSPN`, `TimeNET`, and `PeabraiN` with regard to analytic results. Precision of (a) 0.05 at the 0.95 confidence level and (b) 0.01 at the 0.99 confidence level.

all the models the lower and upper error bars are in the precision bound of $\pm 1\%$. As before, the average error results are close to 0% for all models, but for *net10* and *net17* models. The `PeabraiN` simulation tool presents similar errors as in the previous settings, showing in all the cases quantile intervals outside the precision boundaries. Thus, this confirms that the simulation method and stopping criterion of the `PeabraiN` simulation tool needs further development. Surprisingly, the quantile intervals of the `TimeNET` simulation tool exceed the precision error bounds for all the models. In summary, the `GreatSPN` tool shows the better behavior, while the stopping criterion of both `TimeNET` and `PeabraiN`

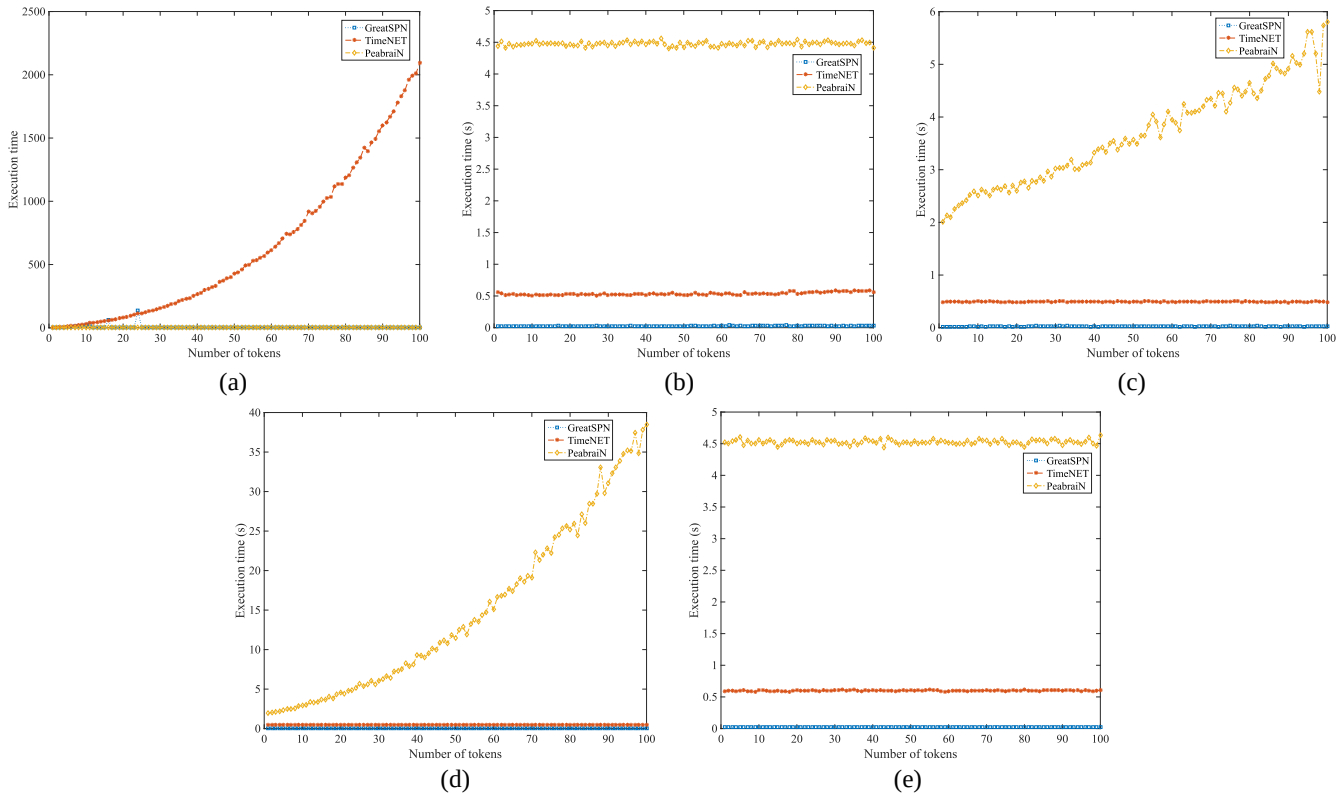

Fig. 2. Results of experiments: execution time. (a) *net14* (general net). (b) *net15* (FRT net). (c) *net22* (FC net). (d) *net25* (DSSP net). (e) *net47* (mono T-semiflow net).

shall be revised. In the beta version of `TimeNET`, enhanced stopping criteria [52] are being evaluated.

B. Execution Time and Memory Peak Consumption

Regarding the efficiency criterion, we are interested in measuring both execution time and memory peak consumption of the simulation tools. Since the GSPN benchmark is composed of 50 nets and we do not want to overwhelm the reader with tens of graphs, only a subset of models is considered for performance evaluation. In particular, we selected one net of each Petri net subclass available in the GSPN benchmark. These subclasses are mono T-semiflow, FC, freely related T-semiflow, DSSP, and general nets. The subset of the GSPN benchmark used for the efficiency comparison is thus composed of five nets. The selection criterion of the each net was as follows: first, to have analytic results; and then, a net with larger number of places and transitions. Considering those criteria, the selected nets are: *net47* (monoT), *net22* (FC), *net15* (FRT), *net25* (DSSP), and *net14* (general).

For the experiment configuration, we varied the initial marking from 1 to 100 in all nets to verify whether the simulators are in some way affected by the number of tokens. Any other aspect of the net was left as in the original models. Experiments were performed in a Debian GNU/Linux x86-64 with an Intel Core i7-4790 CPU @ 3.60 GHz and 4 GB of RAM memory, running as a virtual machine on top of a Proxmox Virtual Environment version 4.1-5/f910ef5c. The versions of the tools evaluated were, namely, `GreatSPN` version 2016, `TimeNET` version 4.3, and `PeabraiN` version 1.2 (plus OpenJDK Runtime Environment IcedTea 2.6.11). A shell script was prepared to execute batch simulations with a precision of 0.05 at the 0.95 confidence level and collect simulation results. In particular, we performed 20 iterations and then compute the average of the obtained results. Execution times and memory peak consumption data were collected through GNU's `time` command.

Fig. 2 plots the execution time (in seconds) for each net considered in experimentation. Results show that for the FRT and the mono T-semiflow nets, the number of tokens does not affect the simulation time of any of the simulation tools under study. It is worth noting that for both types of nets `GreatSPN` clearly outperforms the others. Both `GreatSPN` and `TimeNET` have times lower than 1 s for those nets, unlike `PeabraiN`, which has an execution time near to nine times the one of `TimeNET`. For the DSSP and FC nets, our results illustrate that the convergence of the `PeabraiN` simulation method is linearly and exponentially affected, respectively, by the number of tokens. Surprisingly, the convergence of the `TimeNET` simulation tool is (exponentially) affected by the number of tokens for general nets. In this case, the execution time of `PeabraiN` simulation tool is near to two orders of magnitude with regard to the `GreatSPN` tool, with values ranging from 1.96 to 2.34 s. This overhead might be caused by the execution of Java Virtual Machine. On the contrary, the results of `GreatSPN` range from 0.02 to 138 s (indeed, the latter execution time occurs with 24 tokens). The difference between `GreatSPN` and `TimeNET` is small in absolute values

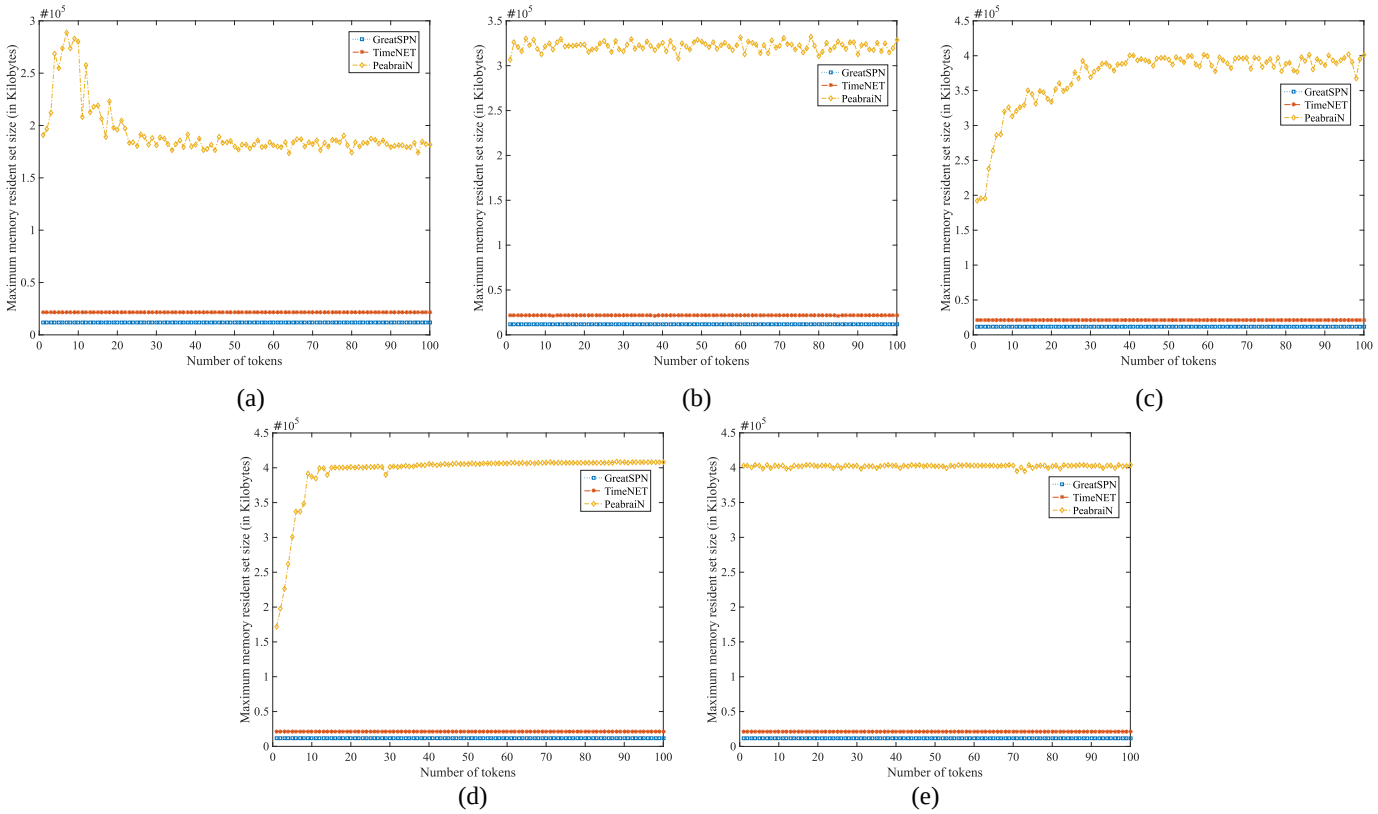

Fig. 3. Results of experiments: maximum peak of memory consumption. (a) *net14* (general net). (b) *net15* (FRT net). (c) *net22* (FC net). (d) *net25* (DSSP net). (e) *net47* (mono T-semiflow net).

but systematic, which may be based on a different implementation architecture - TimeNET compiles some model details into C code to be used in an efficient simulation for longer run times. As our models did not cover this case explicitly, there is no conclusion on relative run times for "harder" simulation problems.

However, a reason for the higher values of execution time of `TimeNET` is that some of the results of `GreatSPN` and `PeabraiN` are invalid and the simulation stops too early. A careful inspection of the results showed that most of the execution times of both simulators converge rapidly and provide a throughput value of zero. We empirically analyzed *net14* and found out that it becomes saturated very early with regard to the variation of number of tokens. Furthermore, the transition rates in *net14* differ in several order of magnitude, i.e., the transition rates setting is similar to a rare-event system. We have also verified that the stopping criteria of `GreatSPN` and `PeabraiN` simulation algorithms do not consider a minimum firing time of the transitions of interest. Based on these findings, we conclude that the stopping criterion used in both `GreatSPN` and `PeabraiN` simulation algorithms does not work properly in rare-event (or similar) setting and deserves further improvements, while `TimeNET` handles these cases correctly. This is a starting point to improving the stopping criterion of both algorithms, considering the current state-of-the-art.

Based on our experimentation results, we conclude that `PeabraiN` is the simulator tool that needs more time to converge at the precision and confidence level given. Furthermore, its simulation method is the one most affected by the number of tokens, especially for DSSP and FC nets.

Fig. 3 plots the results of the maximum memory resident size (in kilobytes) for each net considered in the experimentation. Results clearly show that for all type of nets but FC nets the maximum peak of memory used remains constant, regardless the number of tokens. Regarding FC net, the behavior of `PeabraiN` simulator is close to an exponential line. In all cases, `GreatSPN` outperforms the other two simulation tools. Although the difference between `GreatSPN` and `TimeNET` is very narrow, it is totally the opposite for `PeabraiN`: the results clearly show a huge consumption of memory. We believe that, again, this might be mainly motivated because of the execution of the Java Virtual Machine.

As conclusions, based on our results `GreatSPN` would be our first choice as simulation tool for short simulation problems, immediately followed by `TimeNET`. These results also indicate that the batch simulator method outperforms the replication method. Finally, we would like to remark that although the `PeabraiN` simulation tool is not as good as expected regarding GSPN simulation features, `PeabraiN` provides other interesting features that enriches the analysis of GSPN model, particularly the structural analysis. Furthermore, `PeabraiN` was designed to provide an easily extendable framework for fast prototyping of techniques of the realm of linear programming problems applied to Petri net analysis.

V. Related Work

This section revises the related works in the literature. We divided this section into two parts: we first review the SPN tools—with focus on the status of the standard interchange file format PNML—and, then, the availability of model benchmarks.

A. On SPN Tools

There is a large variety of modeling and evaluation tools for SPN. A comprehensive list of these tools is found in the PN database maintained by the University of Hamburg [53]. A recent survey in [54] drew up a ranking of SPN tools according to a set of criteria including: multiplatform support, open source, embedded graphical animation, structural analysis, and stochastic and colored Petri Net support. In this paper, we consider three Petri net tools (namely, `GreatSPN`, `PeabraiN`, and `TimeNET`) that provide support to performance evaluation of GSPN [12] through different solution techniques, including simulation.

Concerning PNML import/export features, all the aforementioned tools support them. However, since the PN timing and stochastic PNML extensions is not standardized so far, each tool provides its own solution. The compliance of the PNML specification is thus limited to the untimed PN (i.e., the net structure and initial marking).

B. On Model Benchmarks

Model benchmarks can be found mainly in the model checking community for testing algorithm implementations that support verification of qualitative properties [17], [55]–[57]. The BEEM benchmark [55] contains a predefined set of parametric models expressed in a low-level modeling language based on communicating extended finite state machines, and supported by the tool DiVinE. The VLTS benchmark suite [56] collects labeled transition systems modeling communication protocols and concurrent systems. The MIST toolset [57] includes a repository of models, some of them place/transitions (P/Ts) net models that can be used for verifying coverability and reachability properties. The BenchKit [17] is the reference benchmark tool for the model-checking context yearly events, organized within the Petri net conference, and provides a repository of colored Petri nets and P/T nets. In particular, the repository is yearly updated with new parameterized PN models [58]; most of them refer to well-known problems or their variants (e.g., Kanban model and Lamport mutex algorithm), instead of real scenarios as our GSPN benchmark does. For each PN model, a description is provided together with the set of satisfied properties (e.g., reversibility and DF). Other benchmark commonly used is the ISCAS benchmark [59], which contains a set of combinational and sequential circuits. Although this benchmark was initially designed for circuit testing, there are works in the literature of Petri net community using it to validate performance estimation approaches.

Automatic generation of PN benchmarks have been also proposed [60]–[62]. The approach in [60] enables to generate random workflow nets (i.e., a structural class of PNs used to model workflow processes), according to a set of properties given as input such as the length of the shortest path from the source to the sink place, number of nodes, soundness, etc. The generation algorithm has been implemented as a plugin of the process-mining tool ProM. In [61], the PN benchmark is automatically generated for evaluating the efficiency of pattern matching in graph transformation algorithms. In [62], a tool is developed to automatically generate S^4PR, a particular subclass of Petri nets useful for modeling concurrent sequential processes with shared resources.

None of the aforementioned benchmarks, but the one generated with the tool introduced in [62], consider Petri nets with timing and stochastic specifications. Hence, these benchmarks based on untimed Petri net classes are unsuitable for comparing performance solvers. The automatic benchmark generated with the tool introduced in [62] could be useful for this purpose, although the generated models are limited to a single class of Petri net and they do not represent any real scenario, unlike the GSPN benchmark introduced in this paper.

VI. Conclusion

This paper has introduced a framework for the comparison of software tools supporting the performance evaluation of GSPNs. Apart from the technical design of the framework itself, a repository of 50 models from the literature has been collected and made available to the public. This benchmark suite covers a wide area of applications and thus serves as a comprehensive validation base line for algorithms and tools now and in the future.

Three tools have been selected and evaluated, with special emphasis on their simulation components: `GreatSPN`, `PeabraiN`, and `TimeNET`. The numerical results of the presented comprehensive experiments show that the results are in most cases in the expected range of accuracy, thus validating the trustfulness of academic software tools. As a side result, characteristic run time and memory consumption profiles have been derived.

The experimental results allow us to analyze how the tools perform, pointing out possible problems and algorithmic optimization opportunities. Future algorithm development and tool implementation can benefit from this approach by checking if the results are correct, and by allowing a comparative run time and memory consumption evaluation. This will increase the quantitative rigor of computer algorithm engineering efforts in this area.

References

[1] W. F. Tichy, "Where's the science in software engineering?: Ubiquity symposium: The science in computer science," *Ubiquity*, vol. 2014, pp. 1–6, Mar. 2014.

[2] C. U. Smith, "Introduction to software performance engineering: Origins and outstanding problems," in *Proc. 7th Int. Conf. Formal Methods Perform. Eval.*, Berlin, Germany: Springer-Verlag, 2007, pp. 395–428.

[3] *Standard Performance Evaluation Corporation*. Accessed: Oct. 20, 2017. [Online]. Available: https://www.spec.org/

[4] *Transaction Processing Corporation*. Accessed: Oct. 20, 2017. [Online]. Available: http://www.tpc.org

[5] *Defence Advanced Research Projects Agency*. Accessed: Oct. 20, 2017. [Online]. Available: http://www.darpa.mil

[6] J. Waller, "Performance benchmarking of application monitoring frameworks," Ph.D. dissertation, Kiel Comput. Sci. Series, Dept. Comput. Sci., Kiel Univ., Kiel, Germany, Dec. 2014.

[7] *Model Checking Contest at Petri Nets*. Accessed: Oct. 20, 2017. [Online]. Available: https://mcc.lip6.fr

[8] T. Murata, "Petri nets: Properties, analysis and applications," *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, Apr. 1989.

[9] I. Grobelna, R. Wiśniewski, M. Grobelny, and M. Wiśniewska, "Design and verification of real-life processes with application of petri nets," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 11, pp. 2856–2869, Nov. 2017.

[10] X. Han, Z. Chen, Z. Liu, and Q. Zhang, "Calculation of siphons and minimal siphons in Petri nets based on semi-tensor product of matrices," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 3, pp. 531–536, Mar. 2017.

[11] G. Liu, P. Li, Z. Li, and N. Wu, "Robust deadlock control for automated manufacturing systems with unreliable resources based on Petri net reachability graphs," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published, doi: 10.1109/TSMC.2018.2815618.

[12] M. A. Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, *Modelling With Generalized Stochastic Petri Nets* (Wiley Series in Parallel Computing). Chichester, U.K.: Wiley, 1995.

[13] *Systems and Software Engineering—High-Level Petri Nets—Part 2: Transfer Format*, Int. Org. Stand., Geneva, Switzerland, 2008.

[14] E. G. Amparore, G. Balbo, M. Beccuti, S. Donatelli, and G. Franceschinis, "30 years of GreatSPN," in *Principles of Performance and Reliability Modeling and Evaluation* (Springer Series in Reliability Engineering), L. Fiondella and A. Puliafito, Eds. Cham, Switzerland: Springer, 2016, pp. 227–254.

[15] R. J. Rodríguez, "A Petri net tool for software performance estimation based on upper throughput bounds," *Autom. Softw. Eng.*, vol. 24, no. 1, pp. 73–99, Jan. 2017.

[16] A. Zimmermann, "Modelling and performance evaluation with TimeNET 4.4," in *Proc. 14th Int. Conf. Quant. Eval. Syst. (QEST)*, Berlin, Germany, Sep. 2017, pp. 300–303.

[17] F. Kordon and F. Hulin-Hubard, "BenchKit, a tool for massive concurrent benchmarking," in *Proc. 14th Int. Conf. Appl. Concurrency Syst. Design (ACSD)*, La Marsa, Tunisia, 2014, pp. 159–165.

[18] S. Bernardi and J. Campos, "Computation of performance bounds for real-time systems using time Petri nets," *IEEE Trans. Ind. Informat.*, vol. 5, no. 2, pp. 168–180, May 2009.

[19] S. Bernardi and J. Campos, "On performance bounds for interval time Petri nets," in *Proc. 1st Int. Conf. Quant. Eval. Syst. (QEST)*, Enschede, The Netherlands, Sep. 2004, pp. 50–59.

[20] J. Campos, "Performance bounds for synchronized queueing networks," Ph.D. dissertation, Departamento de Ingeniería Eléctrica e Informática, Universidad de Zaragoza, Zaragoza, Spain, Oct. 1990.

[21] E. P. Naumovich and S. Bernardi, "Modelado de redes de Petri con intervalos de tiempo mediante la herramienta ITPN-PerfBound," in *Proc. V Encuentro De Investigadores Y Docentes De Ingeniera (EnIDI)*, Mendoza, Argentina, 2009, pp. 154–167.

[22] S. Bernardi, J. Campos, and J. Merseguer, "Timing-failure risk assessment of UML design using time Petri net bound techniques," *IEEE Trans. Ind. Informat.*, vol. 7, no. 1, pp. 90–104, Feb. 2011.

[23] S. Bernardi and G. Balbo, "Concurrent generalized Petri nets: Regenerative conditions," in *Proc. IEEE 9th Int. Workshop Petri Nets Perform. Models*, Aachen, Germany, 2001, pp. 125–134.

[24] S. Bernardi, S. Marrone, J. Merseguer, R. Nardone, and V. Vittorini, "Towards an MDE approach for NFPs assessment using multiformalism: An application to performability," Group Discr. Event Syst. Eng., Univ. Zaragoza, Zaragoza, Spain, Rep., 2015.

[25] L. Berardinelli, S. Bernardi, V. Cortellessa, and J. Merseguer, "The fault-error-failure chain: A challenge for modeling and analyzing performability in UML-based software architectures," Group Discr. Event Syst. Eng., Univ. Zaragoza, Zaragoza, Spain, Rep., 2010.

[26] S. Bernardi and J. Merseguer, "Performance evaluation of UML design with stochastic well-formed nets," *J. Syst. Softw.*, vol. 80, no. 11, pp. 1843–1865, 2007.

[27] S. Bernardi and J. Merseguer, "QoS assessment via stochastic analysis," *IEEE Internet Comput.*, vol. 10, no. 3, pp. 32–42, May 2006.

[28] L. Berardinelli, S. Bernardi, V. Cortellessa, and J. Merseguer, "UML profiles for non-functional properties at work: Analyzing reliability, availability and performance," in *Proc. 2nd Int. Workshop Non Funct. Syst. Properties Domain Specific Model. Lang. Affiliated MoDELS (NFPinDSML)*, vol. 553. Denver, CO, USA, Oct. 2009, pp. 1–15. [Online]. Available: http://www.CEUR-WS.org

[29] J. Merseguer, J. Campos, and E. Mena, "Analysing Internet software retrieval systems: Modeling and performance comparison," *Wireless Netw.*, vol. 9, no. 3, pp. 223–238, 2003.

[30] A. Zimmermann, *Stochastic Discrete Event Systems—Modeling, Evaluation, Applications*. Berlin, Germany: Springer, 2008.

[31] S. Bernardi and J. Campos, "A min-max problem for the computation of the cycle time lower bound in interval-based time Petri nets," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 5, pp. 1167–1181, Sep. 2013.

[32] E. Gómez-Martínez and J. Merseguer, "Performance modeling and analysis of the universal control hub," in *Proc. 7th Eur. Perform. Eng. Workshop Comput. Perform. Eng. (EPEW)*, vol. 6342. Berlin, Germany: Springer, Sep. 2010, pp. 160–174.

[33] E. Gómez-Martínez and J. Merseguer, "Impact of SOAP implementations in the performance of a Web service-based application," in *Proc. Int. Workshops Front. High Perform. Comput. Netw. (ISPA)*, vol. 4331. Berlin, Germany: Springer, Dec. 2006, pp. 884–896.

[34] E. Gómez-Martínez, S. Ilarri, and J. Merseguer, "Performance analysis of mobile agents tracking," in *Proc. 6th Int. Workshop Softw. Perform. (WOSP)*, Buenos Aires, Argentina, Feb. 2007, pp. 181–188.

[35] D. Perez-Palacin, J. Merseguer, and S. Bernardi, "Performance aware self-managed software: Evaluation using Petri nets," Univ. Zaragoza, Zaragoza, Spain, Rep., 2013.

[36] S. Bernardi, J.-M. Colom, J. Albareda, and C. Mahulea, "A model-based approach for the specification and verification of clinical guidelines," in *Proc. IEEE Emerg. Technol. Factory Autom. (ETFA)*, Barcelona, Spain, Sep. 2014, pp. 1–8.

[37] J. Campos, G. Chiola, and M. Silva, "Ergodicity and throughput bounds of Petri nets with unique consistent firing count vector," *IEEE Trans. Softw. Eng.*, vol. 17, no. 2, pp. 117–125, Feb. 1991.

[38] J. Desel and J. Esparza, *Free Choice Petri Nets*. New York, NY, USA: Cambridge Univ., 1995.

[39] J. Campos and M. Silva, "Structural techniques and performance bounds of stochastic Petri net models," in *Advances in Petri Nets* (Lecture Notes in Computer Science), vol. 609. Berlin, Germany: Springer-Verlag, 1992, pp. 352–391.

[40] Y. Souissi, "Deterministic systems of sequential processes: A class of structured Petri nets," in *Advances in Petri Nets* (Lecture Notes in Computer Science), vol. 674, G. Rozenberg, Ed. Berlin, Germany: Springer, 1993, pp. 406–426.

[41] *GSPN Repository*. Accessed: Nov. 4, 2017. [Online]. Available: https://bitbucket.org/simbern/gspn-benchmark/wiki/Home

[42] J. M. Colom, E. Teruel, and M. Silva, "Logical properties of P/T systems and their analysis," in *Performance Models for Discrete Event Systems With Synchronization: Formalisms and Analysis Techniques*, G. Balbo and M. Silva, Eds. Zaragoza, Spain: Kronos, 1998, pp. 185–232.

[43] E. G. Amparore, "Reengineering the editor of the GreatSPN framework," in *Proc. Int. Workshop Petri Nets Softw. Eng. (PNSE)*, vol. 1372, Jun. 2015, pp. 153–170.

[44] L.-M. Hillah, F. Kordon, C. Lakos, and L. Petrucci, *Extending PNML Scope: A Framework to Combine Petri Nets Types*. Berlin, Germany: Springer, 2012, pp. 46–70.

[45] D. Beyer, S. Löwe, and P. Wendler, "Benchmarking and resource measurement," in *Proc. 22nd Int. Symp. Model Checking Softw. (SPIN)*, vol. 9232, Springer, Aug. 2015, pp. 160–178.

[46] R. Rubinstein and D. Kroese, *Simulation and the Monte Carlo Method*. Hoboken, NJ, USA: Wiley, 2008.

[47] J. Banks, J. Carson, B. Nelson, and D. Nicol, *Discrete Event System Simulation*. Upper Saddle River, NJ, USA: Prentice-Hall, 2000.

[48] G. S. Shedler, *Regenerative Stochastic Simulation*. Boston, MA, USA: Prentice-Hall, 1992.

[49] D. Gillespie, "Approximate accelerated stochastic simulation of chemically reacting systems," *J. Chem. Phys.*, vol. 115, no. 4, p. 1716, 2001.

[50] M. Villen-Altamirano and J. Villen-Altamirano, "RESTART: A straightforward method for fast simulation of rare events," in *Proc. Win. Simulat. Conf.*, Dec. 1994, pp. 282–289.

[51] P. Heidelberger, "Fast simulation of rare events in queueing and reliability models," *ACM Trans. Model. Comput.*, vol. 5, no. 1, pp. 43–85, 1995.

[52] A. Zaliaeva, "Better stopping criterion for SCPN simulation," M.S. thesis, Dept. Comput. Sci. Autom., Technische Universität Ilmenau, Ilmenau, Germany, Oct. 2017.

[53] University of Hamburg. *Petri Net Tool Database*. Accessed: Oct. 1, 2015. [Online]. Available: http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/db.html

[54] A. Charalambous, "Extension of PIPE2 to support coloured generalised stochastic Petri nets," Ph.D. dissertation, Dept. Comput., Imperial College London, London, U.K., 2010.

[55] R. Pelánek, "BEEM: Benchmarks for explicit model checkers," in *Model Checking Software* (Lecture Notes in Computer Science), vol. 4595, D. Bošnacki and S. Edelkamp, Eds. Berlin, Germany: Springer, 2007, pp. 263–267.
[56] S. Blom and H. Garavel. *Very Large Transition Systems (VLTS) Benchmark Suite*. Accessed: Oct. 20, 2015. [Online]. Available: http://cadp.inria.fr/resources/vlts/
[57] P. Ganty. *Coverability Checkers Included in Mist*. Accessed: Oct. 1, 2015. [Online]. Available: https://github.com/pierreganty/mist/wiki
[58] L. M. Hillah and F. Kordon, "Petri nets repository: A tool to benchmark and debug Petri net tools," in *Proc. 38th Int. Conf. Appl. Theory Petri Nets Concurrency (PETRI NETS)*. Cham, Switzerland: Springer Int., Jun. 2017, pp. 125–135.
[59] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 3. Portland, OR, USA, May 1989, pp. 1929–1934.
[60] K. M. van Hee and Z. Liu, "Generating benchmarks by random stepwise refinement of Petri nets," in *Proc. Workshops 31st Int. Conf. Appl. Theory Petri Nets Models Concurrency (PETRI NETS) 10th Int. Conf. Appl. Concurrency Syst. Design (ACSD)*, vol. 827. Braga, Portugal, Jun. 2010, pp. 403–417. [Online]. Available: http://www.CEUR-WS.org
[61] G. Bergmann, A. Horváth, I. Ráth, and D. Varró, "A benchmark evaluation of incremental pattern matching in graph transformation," in *Graph Transformations* (Lecture Notes in Computer Science), vol. 5214, H. Ehrig, R. Heckel, G. Rozenberg, and G. Taentzer, Eds. Heidelberg, Germany: Springer, 2008, pp. 396–410.
[62] R. J. Rodríguez and J. Campos, "On throughput approximation of resource-allocation systems by bottleneck regrowing," *IEEE Trans. Control Syst. Technol.*, to be published, doi: 10.1109/TCST.2017.2768512.

Ricardo J. Rodríguez (M'13) received the M.S. and Ph.D. degrees in computer science from the University of Zaragoza, Zaragoza, Spain, in 2010 and 2013, respectively.

He is currently an Assistant Professor with the Centro Universitario de la Defensa, General Military Academy, Zaragoza. He was a Visiting Researcher with Cardiff University, Cardiff, U.K., in 2011 and 2012, and Mälardalen University, Västerås, Sweden, in 2014. He was also a Visiting Professor with the Second University of Naples, Caserta, Italy, during a three-month period in 2016, and Technische Universität Ilmenau, Ilmenau, Germany, in 2017. His current research interests include performability and dependability analysis, program binary analysis, and contactless card security.

Dr. Rodríguez has been involved in reviewing tasks for international conferences and journals. He is a member of the IEEE Computer Society and the IEEE IES Technical Committee on Factory Automation Subcommittee on Fault Tolerant and Dependable Systems.

Simona Bernardi received the M.S. degree in mathematics and the Ph.D. degree in computer science from the University of Turin, Turin, Italy, in 1997 and 2003, respectively.

She is an Assistant Professor with the Department of Computer Science and Systems Engineering, University of Zaragoza, Zaragoza, Spain. She has been a Visiting Researcher with Carleton University, Ottawa, ON, Canada, the Università degli Studi of L'Aquila, L'Aquila, Italy, and the Università Federico II of Napoli, Naples, Italy. She has co-authored the book entitled *Model-Driven Dependability Assessment of Software-Systems* (Springer). Her current research interests include software engineering and process mining, in particular model driven engineering, verification and validation of performance, dependability and survivability software requirements, and formal methods for the modeling and analysis of software systems.

Dr. Bernardi has been serving as a Referee for international journals, and a Program Committee Member for over 30 different international conferences and workshops.

Armin Zimmermann (M'06) received the Diploma, Ph.D., and Habilitation degrees in computer science from Technische Universität Berlin, Berlin, Germany, in 1993, 1997, and 2006, respectively.

He is a Professor of Systems and Software Engineering and the Director of the Institute for Computer and Systems Engineering, Technische Universität Ilmenau, Ilmenau, Germany. He was a Visiting Researcher with the University of Zaragoza, Zaragoza, Spain, in 1999. He is the Head of the TimeNET tool development since the late 1990's. He has authored the book entitled *Stochastic Discrete Event Systems* (Springer). His current research interests include discrete-event system modeling and performance evaluation/rare-event simulation as well as their tool support with applications in reliability and embedded systems.

Dr. Zimmermann has been serving as an Associate Editor for the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS since 2008. He is a member of the Industrial Automated Systems and Controls Subcommittee of the IEEE IES Technical Committee on Factory Automation.