Quantitative Analysis of Resource-Constrained Business Processes

C´esar A. L. Oliveira, Ricardo M. F. Lima, Hajo A. Reijers and Joel T. S. Ribeiro

Abstract—To address the need for evaluation techniques for complex business processes, also known as *workflows*, this paper proposes an approach based on generalized stochastic Petri nets (GSPNs). We review ten related approaches published in the last fifteen years and compare them to our approach using a wide range of criteria. On the basis of this evaluation, we observe that the newly proposed approach provides results that are at least as good as those from the most accepted alternatives and holds a number of additional advantages, such as modeling simplicity, improved precision, and model reuse for qualitative analyses. The overall approach is formally defined in this paper, along with the definition of several performance metrics. Part of these metrics can be computed analytically, while the remainder can be obtained by simulating the GSPN. Furthermore, a tool has been developed to translate automatically BPEL processes into GSPNs. Finally, we present a case study in which we applied the proposed approach, CPN tools, and an industrial tool to obtain performance insights into a realistic workflow. The results were highly similar, demonstrating the feasibility and the accuracy of our approach.

Index Terms—business process, performance evaluation, resource constraints, generalized stochastic Petri nets

I. INTRODUCTION

The pursuit for competitive advantage has been the main driver for developing new technologies and for improving businesses processes. Since Porter [30] published his breakthrough work on this topic, companies have struggled in the direction of improving their operations and managerial capabilities. A strong competitive edge can be gained by consistently providing superior customer value. In this context, *Business Process Management* (BPM) [17] established itself as the standard framework for managing and optimizing the performance of modern enterprises. BPM can be characterized as the achievement of organizational goals through the improvement, management, and control of essential business processes [17], also known as *workflows*. The term *workflow* refers to the partial or the total automation of a business process through the use of information systems [6]. The term is also employed to refer to the automated process itself.

Business processes can be viewed as dynamical systems that are driven by *discrete business events*. In such systems, the output is dependent on a sequence of desirable actions taking place. The activation of events depends on logical conditions, which are an important part of the system and their mathematical model. Hence, business processes are part of a class of systems

Cesar Oliveira and Ricardo Lima are with Center for Informatics, Federal University of Pernambuco, Recife, Brazil. {calo,rmfl}@cin.ufpe.br

H.A. Reijers and J.T.S Ribeiro are with Eindhoven University of Technology, Eindhoven, The Netherlands. {h.a.reijers,j.t.s.ribeiro@tue.nl}@tue.nl

Manuscript received XXXXX, 2010; revised xxxxx, 2010.

called *discrete event dynamic systems* or DEDS for short [12] [13]. Every employee, machine, and computer system involved in enterprise operations generates events within the system. A similar behavior is observed for customers, partners, and suppliers. These entities can possibly interact in many complex ways, depending on a company's size, market, production strategies, policies, infrastructure, normative rules, and so on. Several activities are executed on a daily basis, involving many resources and presenting different flow and data dependencies.

Both modeling such kind of a discrete event dynamic system and predicting its performance are challenging tasks. For instance, the traditional *queueing network* [3] may turn out to be inadequate for capturing precedence constraints or complex synchronization behavior found in enterprise processes. Such intricacies, however, can be modeled through formal modeling languages such as Petri nets [29] [23] [15]. This is a well-known formalism, widely used for modeling concurrent and distributed systems, including business processes [1] [25] [36] [18] [21] [20].

The concept of *Stochastic Petri nets* (SPN) was first introduced by Molloy in 1982 [24]. It is an extension to Petri nets that associate independent continuous random variables with state transitions to specify their firing delays. In 1995, a group at the University of Torino extended SPN to introduce immediate transitions, which is useful to model instantaneous actions (typically choices) and logical actions (e.g., emptying a place). This new Petri net extension has been labeled *generalized stochastic Petri nets* (GSPN) [23]. It has proven to be a powerful technique for the modeling and performance analysis of complex stochastic dynamical systems in several application areas. Nevertheless, its use for modeling business processes has not been fully explored, being limited to trivial applications [8] [20].

In this paper, we propose a GSPN-based approach for both correctness verification and performance evaluation of business processes. The contributions of this work are manifold. We can *analytically* assess a wide range of performance metrics, such as throughput and utilization - a feature not found in related works. Also, we support the evaluation of processes with *multiple* customers and a limited number of possibly *shared* resources, which corresponds to the most general class of processes. Several related works are limited in this context to single customers and/or the assumption of infinite resources being available. Moreover, the vast majority of existing approaches cannot handle shared resources. Another unique feature is that we use the same model both for *performance evaluation* and analyses of *qualitative properties* of processes, such as soundness and liveness. Most of the related works are not intended to analyze qualitative properties at all. Finally, in our approach, performance metrics that cannot be analytically calculated may be alternatively assessed through simulation, also without changes in the proposed model. This unique combination of features distinguishes our approach among all others approaches available today.

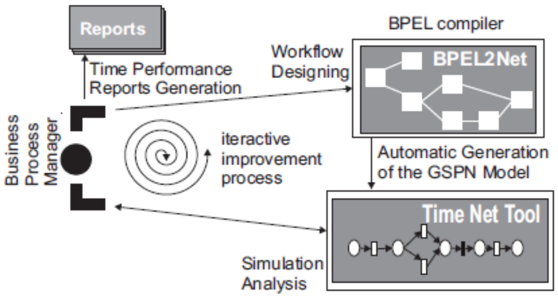


Fig. 1. General view of the modeling and analysis method

In addition to this main contribution, we formally define a set of building blocks and composition operations that may be employed for automatically constructing GSPN models on the basis of a given process definition. We also provide a computational tool, called *BPEL2Net*, to support automatic translation of executable processes into the GSPN models. *BPEL2Net* accepts processes described in the widely used *business process execution language* (BPEL) [27]. These additional contributions provide the means to apply the proposed verification/performance evaluation approach in practice, as is demonstrated in a case study.

Figure 1 shows a general view of the modeling and analysis methodology proposed in this paper.

The paper is structured as follows: Section II summarizes the GSPN formalism, as well as the colored Petri net extension, which is employed in many related works. Section III reviews the literature over the last fifteen years and provides a comprehensive comparison of each of the approaches by a rich set of criteria. Section IV discusses the benefits of employing GSPN as the modeling formalism. Section V proposes a set of reference building blocks and composition operations that may be used to construct GSPN models of business processes; both the blocks and the operations are formally defined. Section VI describes a case study involving the analysis of a real business process. It illustrates the feasibility and the accuracy of the approach we propose in this paper. Finally, our conclusions are presented in Section VII.

II. MATHEMATICAL BACKGROUND

This section reviews the fundamentals of the *generalized stochastic Petri nets* (GSPN) and the *colored Petri net* (CPN) formalisms.

A. Generalized Stochastic Petri Nets

Petri nets [29] (also known as *Place/Transition nets* or *P/Tnets*) are a well-known formalism for describing concurrent discrete event dynamic systems. The *generalized stochastic Petri net* (GSPN) [23] is an extension to this formalism, where time can be represented by means of random delays associated with state transitions to model their firing delays.

Transitions with delays assigned are called *timed transitions*. Transitions without delay, i.e., have a null delay, are called *immediate transitions*.

The following definition for GSPN is given by Balbo et al. [10]:

Definition 1 (Generalized Stochastic Petri Nets). A generalized stochastic Petri net (GSPN) is an 8-tuple defined as $GSPN = \{P, T, \Pi, I, O, H, M_0, W\}$, where:

- $P = \{p_1, p_2, \ldots, p_n\}$ is a finite set of places;
- $T = \{t_1, t_2, \ldots, t_n\}$ is a finite set of immediate and timed transitions, $P \cup T \neq \emptyset$ and $P \cap T = \emptyset$;
- $\Pi : T \rightarrow \mathbb{N}$ is the priority function, where:

$$\Pi(t) = \begin{cases} \geq 1, & \text{if } t \in T \text{ and it is an immediate transition;} \\ 0, & \text{if } t \in T \text{ and it is not an immediate transition.} \end{cases}$$

smaller value means lower priority.

- $I : (T \times P) \rightarrow \mathbb{N}$ is the marking-dependent input function that defines the multiplicities of directed arcs from places to transitions;
- $O : (T \times P) \rightarrow \mathbb{N}$ is the marking-dependent output function that defines the multiplicities of directed arcs from transitions to places;

- $H : (T \times P) \rightarrow \mathbb{N}$ is the marking-dependent inhibition function that defines the multiplicities of inhibitor arcs from places to transitions;
- $M_0 : P \rightarrow \mathbb{N}$ is the initial marking function;
- $W : T \rightarrow \mathbb{R}_+$ is the weight function that represents either the immediate transitions weights (w_t) and the stochastic transitions delay (d_t), where:

$$W(t) = \begin{cases} w_t \geq 0, & \text{if } t \in T \text{ and it is an immediate transition;} \\ d_t > 0, & \text{if } t \in T \text{ and it is not an immediate transition.} \end{cases}$$

A GSPN [illegible]nds to a bipartite directed graph where the nodes are places and transitions, and the edges are directed arcs connecting nodes of different types. The inhibitor arc is a special type of directed arc that connects an input place to a transition, and is pictorially represented by an arc terminated with a circle. The input, output, and inhibitor functions define the arcs multiplicity. The semantics of these arcs are defined by the GSPN's enabling and firing rules, which will be defined later in this section. It is often necessary to refer to the set of all places that are related to a transition. For this purpose, the concepts of *precondition*, *postcondition*, and *inhibitor set* are defined [23].

Definition 2 (Precondition). The set of all places p such that $I(t, p) > 0$, denoted by $I(t)$ or ${}^{\bullet}t$ is called the *precondition* of t.

Definition 3 (Postcondition). The set of all places p such that $O(t, p) > 0$, denoted by $O(t)$ or $t^{\bullet}$ is called the *postcondition* of t.

Definition 4 (Inhibitor Set). The set of all places p such that $H(t, p) > 0$, denoted by $H(t)$ or ${}^{\circ}t$ is called the *inhibitor set* of t.

The state of a Petri net is defined by its *marking*. A marking is a function $M : P \rightarrow \mathbb{N}$ that indicates the number of tokens present on each place of the net. Tokens are represented by small filled circles inside a place. A transition is *enabled* at its current marking according to the number of tokens present on its precondition and inhibitor set, according to the following enabling rule.

Definition 5 (Enabling Rule). A transition $t \in T$ is said to be enabled in a marking M iff:

- $\forall p \in {}^{\bullet}t, M(p) \geq I(t, p)$, and
- $\forall p \in {}^{\circ}t, M(p) < H(t, p)$ or $M(p) = 0$.

The dynamic behavior of a Petri net is governed by the *firing rule*. Only enabled transitions can fire. The firing of an enabled transition removes tokens from all of its input places and inserts tokens in its output places. Because the state of a Petri net is given by the distribution of tokens in its places (marking function), a transition firing may change its state, generating a new marking function.

Definition 6 (Firing Rule). The firing of transition t enabled in the marking M leads to a new marking M' such that

$$\forall p \in ({}^{\bullet}t \cup t^{\bullet}), \quad M'(p) = M(p) - I(t, p) + O(t, p) \, . \ (1)$$

The notation $M_i[t)M_j$ is commonly used to indicate that a certain marking M_j is *directly reachable* from M_i, by firing transition t.

Definition 7 (Reachability Set). The set of all markings that can be reached from the marking M_0 after the firing of one or more transitions is called the *reachability set* and is denoted by $RS(M_0)$.

Definition 8 (Boundness). A Petri net is said to be *k-bounded* if the number of tokens in any place is never greater than k, $k > 0$. If any place can have an infinite number of tokens, the net is said to be *unbounded*.

As long as the firing of timed transitions in a GSPN is an event in a continuous-time stochastic process, the probability of two firings of these transitions to occur at the same time is considered to be equal to zero.

Another characteristic of a GSPN is related to its behavior when multiple tokens are enabling a transition,

When the number of tokens is N times the minimum necessary to enable a transition, allowing it to fire more than one time, this transition is said to be enabled with a degree $N > 0$. With respect to this, a transition can behave according to one of three semantics:

- single-server semantics - the transition needs to fire before being enabled again; thus, it fires N times sequentially;
- infinite-server semantics - the transition is enabled N times in parallel;
- k-server semantics - the transition is enabled up to k times in parallel; tokens that enable the transition to a degree higher than k are handled after the first k firings.

A GSPN is isomorphic to a continuous-time Markov chain (CTMC). The CTMC can be obtained as follows:

1) The set of states $S = \{s_1, s_2, \ldots\}$ of the CTMC corresponds to the reachability set of the GSPN $RS(M_0)$, such that $s_i \in S \Leftrightarrow M_i \in RS(M_0)$ $i = 1, 2, \ldots$.

1) 2) The transition rate q_{ij} from state s_i to s_j is the sum of the firing rates of all transitions that lead from marking M_i to marking M_j, expressed as:

$$q_{ij} = \sum_{t \in E_j(M_i)} \lambda_t \ , \qquad (2)$$

where $\lambda_t = 1/d_t$ and $E_j(M_i) = \{ \ t \ \mid \ M_i[t\rangle M_j \}$.

B. Colored Petri Nets

The so-called colored Petri net (CPN or CP net) [14] is a Petri net extension that introduces the notion of token types. A token stores a value (color) of its corresponding type. Each place is associated to a *color set*, usually described by a type in the *ML* (acronym for *Meta-Language*) functional programming language [33] [14]. ML functions and expressions may be embedded in arcs and transitions of a CPN to manipulate tokens values, thus providing full computational power to the formalism.

Colored Petri nets enable the modeler to implement algorithms that manipulate token data while transitions are fired along the simulation of the net. By this way, it is not only a formal specification of the system but also an executable implementation. For this reason, the formalization of colored Petri nets is complex. Moreover, due to the use of complex data types, the number of states of a CPN model is usually infinite. This is a serious limitation for the development of efficient analysis methods.

III. LITERATURE REVIEW

In this section, we review some works on performance evaluation of workflows. After a brief review of each one, we classified them according to their resemblances and particularities. This classification is based on some qualitative criteria and is summarized in Table I, presented in the end of this section.

Rud et al. [35] propose a model based on operational research techniques to estimate the performance of BPEL processes and the workload of web services. They collect statistical information by monitoring the network and service operations. Their model supports multiple customers and multiple processes concurring for limited resources (server capacity). The authors provide equations based on mean values to compute response time and to estimate resource utilization.

Reijers [31] proposes a Petri net-based model, called *stochastic workflow net* (SWN), which is able to compute numerically the distribution of workflow execution time. The system processes a single customer in this model and resources are unlimited. The model evaluation mechanism takes into account a single process. The time representation is discrete, which allows for an easier computation of time distributions. Independently, Hao [11] presents a model that has the same characteristics proposed earlier by Reijers, without presenting relevant differences or advantages.

Van der Aalst et al. [2] [1] show the application of queueing theory for the performance evaluation of workflow net (WF-Net) models. WF-Nets are a widely known Petri net representation for workflows used for qualitative analysis, e.g., correctness checking. However, queueing networks does not support parallelism and synchronization, which limits its application to workflows with very simple structures. For workflows with more complex structures, WF-Nets allow for an alternative analysis method using a colored Petri net (CPNs) model. Token colors represent different customer orders and simulation of the CPN model is employed to retrieve approximated performance measures with certain confidence levels.

As far as we know, Ferscha [8] was the first to propose the use of GSPN for evaluating the performance of business processes. His model represents a set of agents concurring for resources required to execute the processes for which they are responsible. The interactions and the dependencies between processes are taken into consideration (e.g., producer-consumer relations). The model has no clear notion of the customer, as the agents are working continuously, independent of any customer demand. Also, it does not express how a single agent executes parallel activities, which makes the model confusing when trying to compare it to today’s workflow concepts and practices. The work mentions a single performance metrics: the system throughput.

Schomig & Rau [36] propose the use of a colored GSPN for performance evaluation of workflows that is aligned with concepts recognized by the workflow management coalition (WfMC). They argue that it is important to distinguish one token from another, as decisions taken at one point of the workflow can affect those at another point in the future. Four basic branch structures are modeled: *AND-Split* (fork), *AND-Join* (synchronization), *OR-Split* (exclusive decision), and *OR-Join* (path merging). Similar to Ferscha’s model and Reijer’s SWN, this approach does not take into account customer demands. A process executes continuously and a single customer is served in each execution cycle. Resource constraints are considered, but once a single customer is being served, these constraints affect only the execution of parallel activities. The authors also show that state-space explosion seriously limits the application of the technique.

Shuxia Li & Zhu [20] also present a GSPN model for the analysis of workflow performance and give the name *generalized stochastic workflow net* (GSWN) to their approach. The model assumes a single customer and infinite resources. They argue that it is reasonable to assume infinite resources, as human resources can deal with several tasks in parallel. They consider the same four routing structures as Schomig & Rau. They also recognize that state-space explosion impairs the application of their approach for complex workflows.

JianQiang Li et al. [19] present a hybrid approach called *multidimension workflow net* (MWF-net). Their work represents a set of independent processes that are executed by a set of shared resources. Each process is represented by a time-extended WF-net. These processes are linked together by mapping them to a common set of organizational roles. Each timed transition is associated to a role in the organizational structure. In a third layer, these roles are mapped to resource pools, which represent the workforce available in each role. By applying decomposition and combination algorithms, the authors show how to obtain information about resource utilization and a lower bound for the process performance. These algorithms employ both Petri net analysis and complementary analytical formulae based on queueing theory. Multiple customers are considered to arrive independently at each workflow.

The simulation of workflows is a common practice in industry. Many industrial workflow systems provide simulation features. These applications employ different discrete event simulation (DES) algorithms. Due to this distinction, results obtained from one tool can significantly differ from another. Scientific works that employ simulation of workflow mostly use Petri net–based simulations, due to its formal semantics.

As previously mentioned, van der Aalst et al. extend their WF-net models with color to create a colored Petri net model that can be used for performance evaluation [2]. This approach relies on simulation of the colored Petri net models, which are executable specifications of the workflow.

Reijers presents a resource-extended SWN [31], which adds resource constraints to the original SWN model and employs colored tokens for representing multiple different customers in the system.

However, the algorithms adopted in the SWN model are not valid for the resource-extended version. The results in this new model are assessed by simulating the colored Petri net.

Netjes et al. [25] provide a model for evaluating resource allocation alternatives for optimizing workflow performance. Again, colored Petri nets are employed and results are obtained by simulation.

Dehnert et al. [7] present a model that employs a colored GSPN to evaluate workflow performance. The model is divided into two parts: the resources model and the workflow model. The former represents every communication and documents transport between the departments and the employees. It also takes into account employee vacancies or holidays. The workflow model represents the activities and the dependencies between activities. These two models are merged for analysis purposes. Multiple processes can be evaluated in the same model, sharing the resources. Customer demand is not represented. Resource utilization and execution time can be estimated from this model through an analytical solution based on state-space generation or by simulation.

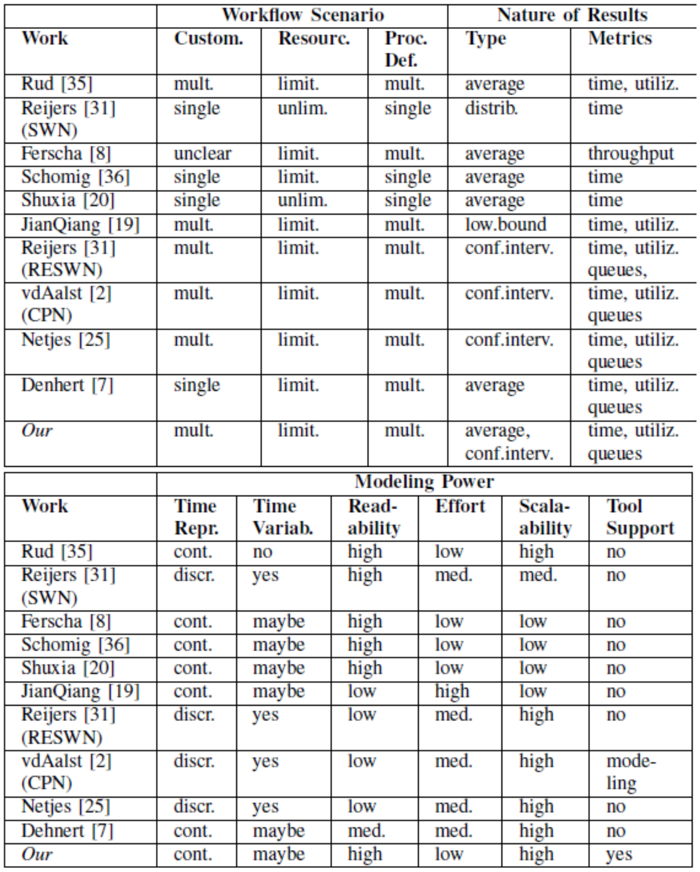
The ten works found in the literature approach the performance evaluation of workflows with different points of view. They differ with respect to how customers and resources are represented, which metrics can be computed, how these metrics are evaluated, and which type of results can be obtained. To classify them, we propose three groups of criteria: workflow scenario; nature of results; and modeling power.

The criterion **workflow scenario** evaluates whether the considered modeling approach represents relevant elements of the real workflow environment. In particular, we classified the approaches according to the following parameters 1) *number of customers*; 2) *number of resources*, and; 3) *number of process definitions*. All these factors are of great importance. For instance, a model that only represents a single customer is not useful for estimating queues and resource utilization.

The criterion **nature of results** refers to the characteristics

TABLE I

COMPARATIVE STUDY. CRITERIA: WORKFLOW SCENARIO, NATURE OF RESULTS, AND MODELING POWER



Work	Workflow Scenario			Nature of Results	
	Custom.	Resourc.	Proc. Def.	Type	Metrics
Rud [35]	mult.	limit.	mult.	average	time, utiliz.
Reijers [31] (SWN)	single	unlim.	single	distrib.	time
Ferscha [8]	unclear	limit.	mult.	average	throughput
Schomig [36]	single	limit.	single	average	time
Shuxia [20]	single	unlim.	single	average	time
JianQiang [19]	mult.	limit.	mult.	low.bound	time, utiliz.
Reijers [31] (RESWN)	mult.	limit.	mult.	conf.interv.	time, utiliz. queues,
vdAalst [2] (CPN)	mult.	limit.	mult.	conf.interv.	time, utiliz. queues
Netjes [25]	mult.	limit.	mult.	conf.interv.	time, utiliz. queues
Denhert [7]	single	limit.	mult.	average	time, utiliz. queues
Our	mult.	limit.	mult.	average, conf.interv.	time, utiliz. queues

Work	Modeling Power					
	Time Repr.	Time Variab.	Read-ability	Effort	Scala-ability	Tool Support
Rud [35]	cont.	no	high	low	high	no
Reijers [31] (SWN)	discr.	yes	high	med.	med.	no
Ferscha [8]	cont.	maybe	high	low	low	no
Schomig [36]	cont.	maybe	high	low	low	no
Shuxia [20]	cont.	maybe	high	low	low	no
JianQiang [19]	cont.	maybe	low	high	low	no
Reijers [31] (RESWN)	discr.	yes	low	med.	high	no
vdAalst [2] (CPN)	discr.	yes	low	med.	high	mode-ling
Netjes [25]	discr.	yes	low	med.	high	no
Dehnert [7]	cont.	maybe	med.	med.	high	no
Our	cont.	maybe	high	low	high	yes

of the results obtained through the modeling approach. We adopt two parameters for this element: 1) *type*: the mathematical nature of the results computed by the approach (simple average, probability distribution, lower/upper bounds); 2) *metrics*: a list of metrics that are directly calculated through the model.

We also define a set of parameters to evaluate the criterion **modeling power**. The intention here is to evaluate the modeling strategy employed and to understand both how accurately the model can express system's characteristics and how much effort is required to design a model of the system. Here, six parameters are explored: 1) *time representation*: determines whether the approach deals with discrete or continuous time; 2) *time variability*: captures whether the approach allows for the direct representation of time with different probability distributions - if so, it is marked as *yes*; otherwise, it is marked as *no*; the cases where the variability can be achieved with extra effort are marked as *maybe*; 3) *readability*: indicates how easy it is to understand, read, and maintain the model. We make this classification following the principles by which researchers classify different programming languages according to their maintainability; 4) *effort*: evaluates the abstraction level of the modeling language, as well as the effort required to calculate the desired metrics - we assume three levels of modeling effort: *high*, *medium*, and *low*; 5) *tool*: indicates an approach that is supported by a computational tool (we take into consideration only those tools created specifically for the approach); 6) *scalability*: indicates how the approach scales with the size of the system. All works that rely on state-space generation were classified as having low scalability, while works that use simulation were assumed to be highly scalable.

The work proposed in the current paper used GSPN as a technique for enabling the modeling of scenarios where multiple customers compete for a limited number of resources in the execution of a workflow. Each resource is assigned a role and a single role can be responsible for executing multiple activities. In turn, each activity can be executed by more than one role. We provide a number of analytical formulae for computing the average value of performance metrics such as *utilization* and *throughput*. Also, simulation is employed for providing other important metrics, such as *queue* sizes, *synchronization times*, and overall *response time*. Time is represented by continuous random variables. Our approach is described in detail in Section V.

Table I shows the result of this classification methodology.

IV. THE CHOICE FOR GSPN

Based on the analysis of related works conducted in Section III, we defined a set of requirements to guide the development of new methodologies for performance evaluation of workflow systems.

- support for multiples customers;
- support the definition of resource constraints;
- support for multiple concurrent processes;
- provide analytical formulae;
- measure response times, queue size, resource utilization, and throughput;
- support for continuous time;
- support a variety of distribution functions for representing time;
- be scalable;
- be easy to write, read, and maintain.

A careful analysis of Table I reveals that the set of works in perspective only partially fulfill these requirements. Moreover, one can notice that works based on colored Petri nets (CPN) are more complete in terms of these requirements. In this work, we demonstrate that an approach based on generalized stochastic Petri nets (GSPN) can provide the same benefits found in works employing CPN. Furthermore, in the context of performance evaluation of workflow, we enumerate some advantages that GSPN has over CPN.

In this section, we explain the reasons of our choice for a GSPN-based approach. We highlight key advantages of GSPN over CPN for the purpose of performance evaluation. In this comparison, we assume the implementation of Jensen et al., called *CPN Tools* [16], as a reference. This is the most widely used implementation of the CPN formalism.

We summarize some important drawbacks found in CPN:

1) time is not a natural concept in CPN. Designers are responsible for keeping control of time stamps during system simulations. They must include arc or transition expressions to calculate the time stamp at each point of the CPN model. This makes the model more susceptible to modeling errors not verifiable through analysis;

2.) CPN uses an integer global variable to represent time. When a new time stamp is calculated through arc or transition expressions, the result is rounded to an integer value. As the simulation evolves, the number of rounding executed increases. The resulting loss of accuracy may cause undesirable effects in complex models. For instance, rounding an exponentially distributed random variable to the next integer will lead to a geometric distribution instead of an exponential.

3) defining a stochastic process corresponding to the CPN model is a complex task. This makes it difficult to investigate the CPN model analytically to find mathematical relations between parameters and metrics. Thus, if someone intends to study properties of the system without relying on model simulation, CPN will give little support for that task. For example, one might be interested in the impact of different arrival distributions to the overall response time. The only option is to simulate the model using different distributions and, then, measure the impact of each simulation round.\

It should be mentioned that CPN has some attractive characteristics for the sake of performance evaluation. Firstly, it provides a great flexibility to express time behavior, which is controlled by model designers. For instance, this flexibility would allow to go back in time, if wanted. It also allows for the creation of arbitrary discrete time distributions, being only necessary to implement a random number generator for that distribution. As an illustrating example, Fig. 2 shows a single server queue modeled in CPN, extracted from the CPN Tools user manual [38]. Without explaining in much detail, it can be observed the use of the "@+" operator for incrementing the timestamps associated to tokens.

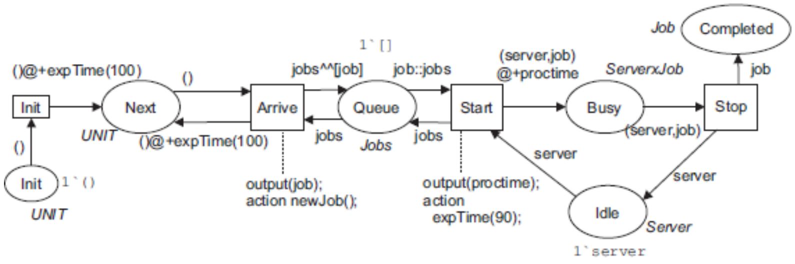


Fig. 2. Queue modeled in CPN

Generalized stochastic Petri nets (GSPNs), in contrast to CPNs, is a formalism designed specifically to represent stochastic systems. It is isomorphic to continuous-time Markov chains (CTMC), which have been used for stochastic studies for decades. Time is a natural concept in GSPN models and is associated with *timed* transitions. Therefore, no manipulation of timing variables is necessary. Moreover, as the CTMC associated to a GSPN is clearly defined, it is possible to study the properties of the system on a mathematical basis.

There are many algorithms for the solution and simulation of CTMCs, and several known properties that can be analytically obtained. Furthermore, GSPNs have a graphical representation that clearly expresses concurrence, synchronization and both states and actions, properties that are not present in CTMCs. For these and other Petri nets characteristics, it seems rather natural to use GSPN as a formal representation of workflows for performance evaluation purposes. For a comparison with the CPN queue model, Fig. 3 illustrates the same queue modeled in GSPN. The time information is annotated to the timed transitions.

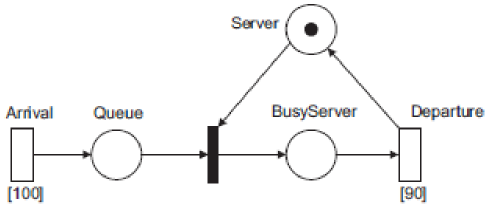


Fig. 3. Queue modeled in GSPN

The noteworthy drawback is that GSPN transitions can only be associated to exponentially distributed times. However, methods for approximating other distributions do exist and are widely used [4] [23]. These methods only require the addition of auxiliary structures to the model to obtain the desired distribution. Thus, this drawback is not a practical limitation. Therefore, the use of GSPN by no means is restricted to systems in which all variables are exponentially distributed.

V. MODELS DESCRIPTION

In this section, we propose a collection of building blocks modeled in generalized stochastic Petri nets (GSPN) and composition operations. By employing such composition operations over the building blocks, one can create models for the analysis and evaluation of a large number of workflows.

To present a mathematically sound composition algebra, we formally define every element of the GSPN that is built up from the operations. This possibly makes the text hard to follow at some parts or present some repetitions, but it is a necessary cost in favor of mathematical rigor. Nevertheless, explanatory text and pictures provide the informal description of the models, which give an intuition on the mathematical definitions. Yet, some formalization has been subtracted to simplify the text. A more detailed description and formalization is available in a technical report [26]

Our approach assumes a *limited* number of resources assigned to a subset of business process roles. Each role can be responsible for several activities. This feature may be overlooked at first glance, but it is exactly the sharing of limited resources that impairs the application of queue theory and other related techniques available today.

A *workflow* is composed of atomic *activities*, order relations between these activities, and the definition of *roles* that are responsible for executing them [5] [6]. In correspondence, the building blocks proposed in this section represent activities and roles, while the composition operations model the order relations between activities. We provide a number of composition rules that allow for the construction of complex workflows containing concurrence, synchronization, loops, and so on. These structures are found in several process notations, but there is a lack of uniformity in their terminology. For this reason, we adopt the terminology provided by the *Workflow Management Coalition* (WfMC). One can refer to the

WfMC Glossary [6] and the *WfMC Reference Model* [5] to find synonymous and related terms for a specific notation.

A workflow is executed in a run-time environment. We assume this environment to be defined by the arrival process, which correspond to the creation of new process instances, and the number of resources available in each role for performing the activities.

Regarding expressiveness, the models present the following main characteristics:

- represent simultaneous execution of *multiple process instances*;
- represent resources grouped in roles that can be responsible for executing several activities (*shared resources*);
- distinguish *work items* (works to be processed in an activity) and ongoing *activity instances* (work being processed in an activity);
- assume that *case arrival* is a Poisson process or that it can be approximated by a mixture of exponential interarrival times [4];
- assume that *service times* are exponentially distributed random variables or that they can be approximated by a mixture of exponential variables [4].

The use of mixtures of exponentials is ubiquitous in performance analysis. Despite the standard way of using exponential times for delays in GSPNs, it is possible to approximate any random distribution with rational *Laplace* transform by combining exponential variables. Methods for modeling these distributions with GSPN are well known [4] [23] [22] and can be applied in the proposed model to improve its representativeness. For the sake of simplicity, all models will be presented assuming exponential delays. It must be observed, however, that such exponential transitions can be readily replaced by mixtures of exponentials by employing the proper procedures, as described by the literature [23]

The following metrics can be assessed through the proposed model:

- *soundness* verification;
- minimum number of resources demanded by each role;
- number of ongoing activity instances;
- number of work items in each worklist;
- number of available resources in each role;

- mean time of case processing (*response time*).

A. Basic Blocks

In this section, we describe the basic structures for modeling a *process definition* (or *workflow model*) [6] and formulae for calculating metrics from them.

A *pool* [6] is a structure that groups the roles that participate in the process. In most graphical notations, each role is represented by a *swimlane* [6] [40] in the pool. When an activity is placed on that swimlane, it means that the respective role is responsible for the execution of that activity.

We use the concept of pool to represent the set of roles present in the workflow.

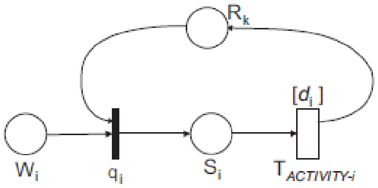


Fig. 4. Activity Model in GSPN

Definition 9 (Structure - Pool). The Pool, denoted by $\mathcal{P}$, contains the roles that participate in the process. It is defined as a set $\mathcal{P} = \{X, Y, ..., Z\}$, where each element in the set is a role identifier.

Each time the company starts the execution of a process, it creates a new *process instance*, also called a *(business) case* [6].

Definition 10 (Structure - Process Instance (Case)). Ongoing *process instances* or *cases* are represented by *tokens* in the GSPN model.

The fundamental structure in the workflow model is the *activity model*. This model represents the execution of an activity by a resource. Notice that activities are the atomic unit of work in a workflow [5].

Definition 11 (Structure - Activity Model). An *activity model* is denoted by $A_i(k, d_i)$, where:

1) $k \in \mathcal{P}$ is the role responsible for the execution of the activity;

1) 2) $d_i \in \mathbb{R}$[?]is the mean time delay for the activity execution.

It corresponds to a GSPN, $A_i(k, d_i) = (P_i, T_i, \Pi_i, I_i, O_i, H_i, M_{[?]}, \omega_i)$, which is defined as follows:

1) $P_i = \{R_k, W_i, S_i\}$, where:

    - R_k is a place that represents the role k;

- W_i is a place for holding the activity's work wtems, therefore called *worklist place*;
- S_i is a place for containing the activity instances, therefore called *service place*.

2) $T_i = \{q_i, T_{ACTIVITY-i}\}$, where:

- q_i is an immediate transition, with $\omega_i(q_i) = 1$ and $\Pi_i(q_i) = 1$;
- $T_{ACTIVITY-i}$ is a timed transition with mean delay d_i and infinite server semantics [23].

3) and I_i, O_i, are such that:

a) the precondition $\cdot q_i = \{W_i, R_k\}$ and the postcondition $q_i \cdot = \{S_i\}$;

b) the precondition $\cdot T_{ACTIVITY-i} = \{S_i\}$ and the postcondition $T_{ACTIVITY-i} \cdot = \{R_k\}$

For the sake of simplicity, when the parameters k and d_i are not relevant for the discussion, we use the simplified notation "A_i" in substitution to $A_i(k, d_i)$.

Fig. 4 presents the GSPN basic model for *activity*.

Notice that, according to the Definition 11, although one might assign a *role* to many different *activities*, an activity can be assigned only to a single *role*. However, this assumption is too restrictive, as many real workflow have resources in more than one *role* handling the same *activity*. For example, a supervisor can decide that he/she will execute an *activity* that is usually performed by a subordinate. This means that work items for that activity are shared between them. For this situation, we provide the structure named *Multiple-Role Activity Model*, which is pictured in Fig. 5.

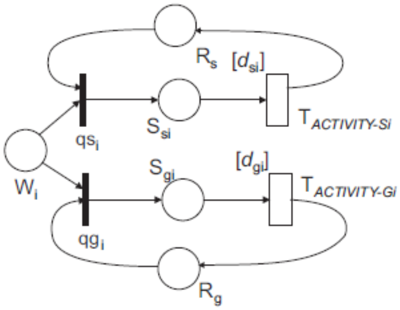


Fig. 5. Example of Two-Role Activity Model in GSPN

Definition 12 (Structure - Multiple-Role Activity Model). A Multiple-Role Activity Model is denoted by A[?] ($\mathcal{P}_i, D_i$), where:

1) $\mathcal{P}_i \subseteq \mathcal{P}$ is the set of roles that can execute the activity;

2) $D_i : \mathcal{P}_i \rightarrow \mathbb{R}$[?] is a function that relates each role to a time delay, that is the mean time for the activity execution by that role;

3) m is the number of different roles that can perform the activity (cardinality of set $\mathcal{P}_i$).

and corresponds to a GSPN, A[?] $(\mathcal{P}_i, D_i) = (P_i,\ T_i, \Pi_i,\ I_i, O_i, H_i, M_{i0},\ \omega_i)$, which is defined as follows:

1.) $P_i = \{W_i\} \cup P_r \cup P_s$,

$P_r = \{R_k \mid k \in \mathcal{P}_i\}$,

$P_s = \{S$[?] $\mid k \in \mathcal{P}_i\}$, where:

- W_i is a place for holding the activity's work items (*worklist place*);
- Each place R_k is a place that represents a role k;
- Each place S[?] is a place for containing the *activity instances* that are being executed by role k (*Service places*).

2.) $T_i = T_q \cup T_a$,

$T_q = \{q$[?] $\mid k \in \mathcal{P}_i\}$,

$T_a = \{T^k{}_{ACTIVITY-i} \mid k \in \mathcal{P}_i\}$, where:

- Each q[?] is an immediate transition, with $\omega_i(q$[?]$) = 1$ and $\Pi_i(q$[?]$) = 1$;
- Each $T^k{}_{ACTIVITY-i}$ is a timed transition with mean delay $D_i(k)$ and infinite server semantics [23].

3.) and I_i, O_i, are such that:

a) the precondition $\cdot q$[?] $= \{W_i, R_k\}$ and the postcondition q[?]$\cdot = \{S$[?]$\}$, for all $k \in \mathcal{P}_i$;

b.) the precondition $\cdot T^k{}_{ACTIVITY-i} = \{S$[?]$\}$ and the postcondition $T^k{}_{ACTIVITY-i}\cdot = \{R_k\}$, for all $k \in \mathcal{P}_i$.

This model corresponds to a replication of multiple copies of the activity, where there are different roles and different delays for each one, but with a shared worklist (all worklist places were merged into a single place). The replication is necessary to maintain different instances of the activity being executed by different types of resources. Also, each type of resource may provide its own quality of service, which affects time delay. Thus, the need for copies of the timed transition with different delays. Notice that this affects only the structure of the activity. The rest of the workflow model remains the same.

B. Metrics for the Basic Blocks

The following metrics can be evaluated for these basic blocks. These metrics assume that the workflow model executes in an environment characterized by the case arrivals distribution and resources available for processing customer requests. We call the combination of a workflow model and an environment model a *workflow system*, which will be formally defined in Sec. V-I.

Definition 13 (Measure - Minimum Number of Resources for a Role). Let k be a role with K resources that perform a set of activities

$A_1(k, d_1), A_2(k, d_2), \ldots, A_N(k, d_N)$ in a workflow system, a stationary solution for that system exists only if:

$$K > \sum_{i=1}^{N} \lambda_i d_i \ , \qquad (3)$$

where λ_i is the rate at which *cases* arrive at activity A_i.

Notice that the arrival rate for each activity can be different, due to the characteristics of the case flow inside the process. Formulae for computing this flow are provided in Sec. V-C.

Definition 14 (Measure - Expected Number of Activity Instances). For an activity A_i with mean delay d_i and arrival rate λ_i, *provided with sufficient resources*, the expected number of *activity instances* during the workflow system execution is given by:

$$E(S_i) = \lambda_i d_i \ . \qquad (4)$$

Definition 15 (Measure - Expected Number of Available Resources). For a role k with K resources and performing activities $A_1, \ldots, A_N$ in a workflow system, the mean number of available resources is given by:

$$E(R_k) = K - \sum_{i=1}^{N} E(S_i) \ , \qquad (5)$$

where $E(S_i)$ is the expected number of instances of A_i.

Definition 16 (Measure - Expected Number of Work Items). For an activity A_i, the mean number of *work items* of this activity during the workflow system's execution is equal to the expected marking of place W_i.

Definition 17 (Measure - Expected Number of Cases). For an activity A_i, the mean number of cases being processed by this activity during the workflow system's execution is given by:

$$E(n_i) = E(W_i) + E(S_i) \ . \qquad (6)$$

TABLE II

METRICS FOR THE BASIC MODELS

Metric	Expression
Expected Number of Activity Instances	$E(S) = \lambda d$
Expected Number of Work Items	$E(W) =$ expectation of W
Expected Number of Cases	$E(n) = E(W) + E(S)$
Mean Response Time	$E(\tau) = E(n)/\lambda$

Definition 18 (Measure - Expected Activity Response Time). Let A_i be an activity with case arrival rate λ_i and mean service time d_i, the mean activity's response time is given by:

$$E(\tau_i) = \frac{E(n_i)}{\lambda_i} \ , \qquad (7)$$

where $E(n_i)$ is the mean number of Cases in A_i.

Table II summarizes the metrics defined for the basic models.

C. Composition Operations

The composition operations are uniformly defined such that every composed structure contains a single *starting place* and a set of *departing transitions*. Every token that arrives at that *starting place* must eventually depart through one of the *departing transitions*.

We call such structures *subprocesses*. An activity model is the most simple subprocess structure. Every composition operation is defined as a function that maps one or more subprocess operands to a resulting subprocess.

A subprocess is a GSPN that attends to the restrictions presented by Def. 19. We denote by *SProc* the set of all GSPNs that form a valid subprocess.

Definition 19 (Structure - Subprocess). A subprocess is a GSPN

$U = (P_U, T_U, \Pi_U, I_U, O_U, H_U, M_{U0}, \omega_U)$, such that

1) there exists a unique place $Sp \in P_U$, such that $\cdot Sp = \varnothing$, called *starting place*;

2) there exists a nonempty set of transitions $Dt \subseteq T_U$, such that $\forall t \in Dt : t\cdot = \varnothing$, called *departing transitions*;

3) for each token arriving at starting place Sp, exactly one token departs from the subprocess through any one of the transitions in the set Dt.

In what follows, we define some auxiliary functions.

Definition 20 (Utility - Starting Place Function). For a subprocess

$U = (P_U, T_U, \Pi_U, I_U, O_U, H_U, M_{U0}, \omega_U)$, we denote by *Start*($U$) the unique place $Sp \in P_U$ such that $\cdot Sp = \varnothing$, called the starting place of U.

Definition 21 (Utility - Departing Transitions Function). For a subprocess

$U = (P_U, T_U, \Pi_U, I_U, O_U, H_U, M_{U0}, \omega_U)$, we denote by *End*($U$) the set of transitions $Dt \subseteq T_U$ such that $\forall t \in Dt : t\cdot = \varnothing$, which correspond to the departing transitions set of U.

A subprocess model represents the *process definition* to be evaluated.

When subprocesses are composed, their respective GSPNs are united. Def. 22 presents a definition for GSPN union operation.

Definition 22 (Utility - GSPN Union). Let *GSPNSet* be the set of all existing GSPNs, the operation of uniting two GSPNs can be defined as follows:

$$\cup : \mathit{GSPNSet} \times \mathit{GSPNSet} \rightarrow \mathit{GSPNSet}$$

$G_3 = G_1 \cup G_2$, where:

1) $G_1 = (P_1, T_1, \Pi_1, I_1, O_1, H_1, M_0^1, \omega_1)$;
2) $G_2 = (P_2, T_2, \Pi_2, I_2, O_2, H_2, M_0^2, \omega_2)$;
3) $G_3 = (P_3, T_3, \Pi_3, I_3, O_3, H_3, M_0^3, \omega_3)$;
4) $P_3 = P_1 \cup P_2$;
5) $T_3 = T_1 \cup T_2$;
6)
$$I_3(p,t) = \begin{cases} I_1(p,t) & \text{if} \quad p \in P_1 \quad \text{and} \quad t \in T_1 \\ I_2(p,t) & \text{if} \quad p \in P_2 \quad \text{and} \quad t \in T_2 \\ 0 & \text{otherwise} \end{cases}$$
7)
$$O_3(p,t) = \begin{cases} O_1(p,t) & \text{if} \quad p \in P_1 \quad \text{and} \quad t \in T_1 \\ O_2(p,t) & \text{if} \quad p \in P_2 \quad \text{and} \quad t \in T_2 \\ 0 & \text{otherwise} \end{cases}$$
8)
$$H_3(p,t) = \begin{cases} H_1(p,t) & \text{if} \quad p \in P_1 \quad \text{and} \quad t \in T_1 \\ H_2(p,t) & \text{if} \quad p \in P_2 \quad \text{and} \quad t \in T_2 \\ 0 & \text{otherwise} \end{cases}$$
9)
$$\omega_3(t) = \begin{cases} \omega_1(t) & \text{if} \quad t \in T_1 \\ \omega_2(t) & \text{if} \quad t \in T_2 \end{cases}$$
10)
$$\Pi_3(t) = \begin{cases} \Pi_1(t) & \text{if} \quad t \in T_1 \\ \Pi_2(t) & \text{if} \quad t \in T_2 \end{cases}$$
11)
$$M_0^3(p) = \begin{cases} M_0^1(p) & \text{if} \quad p \in P_1 \\ M_0^2(p) & \text{if} \quad p \in P_2 \end{cases}$$

Next, we formally define the composition operators on the basis of the building blocks and functions we have presented so far. The composition operations are:

- Sequence (SEQ) - two or more subprocesses are executed in sequence;
- Alternative Path (XOR) - a selection is made to perform one from a set of subprocesses that can be executed;
- Parallelism (AND) - a set of subprocesses are executed in parallel and synchronized at the end;

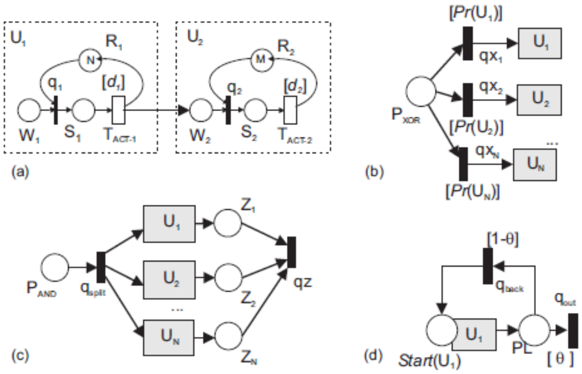


Fig. 6. Composition operations: a) SEQ; b) XOR; c) AND; d) LOOP

- Simple Iteration (LOOP) - one subprocess is executed several times;
- Grid-form Iteration (GRID-LOOP) - a set of subprocesses is executed several times, but there is an exit point after each subprocess that allows the iteration to finish after the execution of that subprocess, without completing the whole cycle;
- Multiple Path (OR) - there are two subprocesses that can be executed in a nonexclusive way. If both are executed, they must be synchronized at the end of the structure;
- Interleaving (INTER) - a set of subprocesses can be executed in any order, but two subprocesses from this set cannot be executed in parallel for the same process instance.

The structures for these composition operations are represented in Figures 6 and 7. The subprocesses are represented by gray-filled rectangles and denoted by letter U. These subprocesses are given as operands. The composition operation, then, creates the auxiliary structures that can be seen in the pictures – places, transitions, and arcs – that model the composition behavior.

Observe that, in all the pictures, each arc that enters a subprocess is considered to be connected to its *starting place*, according to the mathematical definition of the operators. Each arc going out from the subprocess is considered to be connected to *all* of its *departing transitions*. The SEQ operator, illustrated in (Fig. 6.a), just adds arcs connecting each departing transition of the first subprocess to the starting place of the second one. Notice that the SEQ operator can compose any two subprocesses in this way, although the illustration depicts a particular situation where two single-activity subprocesses are connected.

D. Sequence

This operator combines two subprocesses, U_1 and U_2, with a sequential relation such that U_2 is executed after U_1. The model is constructed by adding an arc connecting each departing transition of U_1 to the starting place of U_2. This is illustrated in Fig. 6.a.

Definition 23 (Composition - Sequence Operator – SEQ).

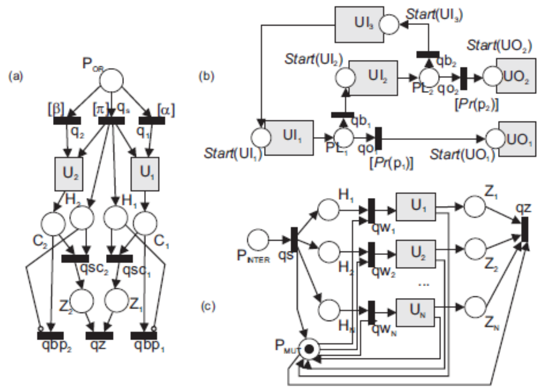


Fig. 7. Other composition operations: a) OR; b) GRID-LOOP; c) INTER

$\mathcal{SEQ}$: *SProc* × *SProc* → *SProc*

$\mathcal{SEQ}(U_1, U_2) = U_R$, where:

$U_R = U_1 \cup U_2$, with the addition of an arc such that:

1) *Start*(U_R) = *Start*(U_1);

2) *End*(U_R) = *End*(U_2);

3) *Start*(U_2) = *End*(U_1).

For notation simplification, it is possible to use a more general operator $\mathcal{SEQ}(U_1, U_2, \ldots, U_N)$ (multiple arguments), as an abbreviation to the composition $\mathcal{SEQ}(U_1, \mathcal{SEQ}(U_2, \ldots \mathcal{SEQ}(U_{N-1}, U_N)))$, without differences in the resulting model.

E. Alternative Path (XOR)

This operator combines a set of N subprocesses ($U_1, \ldots, U_N$) in a way that they are alternatively executed. Each case arriving is forwarded to one of these subprocesses (which we call *paths*), according to a probability distribution defined by a function P_r. For each subprocess U_i, a probability $P_r(U_i)$ for the case be routed to that subprocess is assigned.

The composition is modeled by the addition of a place P_{XOR}, which is the starting place of the subprocess, a set of immediate transitions $qx_1, \ldots, qx_N$, which removes a token from P_{XOR} and puts it in the starting place of subprocess $U_1, \ldots, U_N$, respectively. Each transition receive a weight $\omega(qx_i)$ equal to the probability $Pr(U_i)$ of the subprocess U_i be chosen

This model is shown in Fig. 6.b.

Definition 24 (Composition - Alternative Path Operator – XOR). Let $U_1, U_2, \ . \ . \ . \ , U_N$ be subprocesses and Pr a probability distribution function

XOR : *SProc* × . . . × *SProc* × (*SProc* → $\mathbb{R}[0; 1]$) → *SProc*

$U_R = XOR(U_1, U_2, \ldots, U_N, Pr)$, where:

1.) Let G_{XOR} be a GSPN containing a place P_{XOR} and immediate transitions $qx_1, \ldots, qx_N$, with $P_{XOR}{}^{\bullet} = \{qx_i\}$, $i = 1, \ldots, N$;

2.) $\omega(qx_i) = Pr(U_i)$, $i = 1, \ldots, N$;

3.) $U_R = U_1 \cup \ldots \cup U_N \cup G_{XOR}$, with the addition of arcs such that:

   a) *Start*(U_R) = P_{XOR};

   b) *End*(U_R) = *End*(U_1) $\cup \ldots \cup$ *End*(U_N);

   c) ${}^{\bullet}$*Start*(U_i) = $\{qx_i\}$, $i = 1, \ldots, N$.

F. Parallel Execution (AND)

This operator creates a subprocess that consists of the parallel execution of N other subprocesses that compose it. Each arriving case is sent to all of these subprocesses simultaneously to be processed by them. Synchronization occurs before the departure of the case, in a way that it leaves the subprocess only after every parallel process have been done.

This composition is modeled by the addition of an initial structure, responsible for splitting the tokens that arrive and another structure in the exit, responsible for the synchronization and for merging the tokens back. This model is presented in Fig. 6.c.

Definition 25 (Composition - Parallel Operator – AND).

AND : *SProc* × . . . × *SProc*× → *SProc*

$U_R = AND(U_1, U_2, \ldots, U_N)$, where:

1) Let G_{AND} be a GSPN containing place P_{AND}, immediate transition q_{split}, with $P_{AND}{}^{\bullet} = \{q_{split}\}$, a set of places $\{Z_1, \ldots, Z_N\}$ and another immediate transition q_Z, such that ${}^{\bullet}q_Z = \{Z_1, \ldots, Z_N\}$;

2) $U_R = U_1 \cup \ldots \cup U_N \cup G_{AND}$, with the addition of arcs in such a way that:

   a) $Start(U_R) = P_{AND}$;

   b) $End(U_R) = \{q_Z\}$;

   c) $Start(U_i) = q_{split}$, $i = 1, \ldots, N$;

   d) $Z_i = End(U_i)$, $i = 1, \ldots, N$.

G. Iterations

An *iteration* is a subprocess executed several times for processing the same case. In one or more points of the subprocess execution, a decision is made about whether the case must continue iterating or leave the structure.

The iterative structure in our model needs that a single entry point exist for the iteration, but several exit points are allowed. When there is only one exit point and no activity exists in the return path from the exit point to the entry point, we simplify the structure and call it *simple iteration*, created by the LOOP operator. Otherwise, we use the more general model, called *grid iteration*, constructed by the GRID-LOOP operator.

Fig. 6.d depicts the *simple iteration model* and Fig. 7.b shows a *grid iteration model* example with two exit points.

Here, we defined the simple iteration model.

Definition 26 (Composition - Simple Iteration Operator – LOOP). Let U_1 be a subprocess and θ the probability of

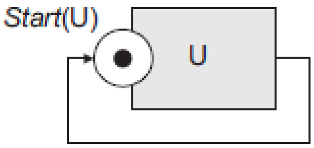


Fig. 8. Soundness model

leaving the iterative loop, the simple iteration operator can be defined as

$$LOOP: SProc \times \mathbb{R}[0;1] \rightarrow SProc$$

$LOOP(U_1, \theta) = U_R$, where:

1) Let G_{LOOP} be a GSPN consisting of a place PL and two immediate transitions q_{back} and q_{out}, such that $PL^{\bullet} = \{q_{back}, q_{out}\}$;

2) 2) $\omega(q_{out}) = \theta$;

3) 3) $\omega(q_{back}) = 1 - \theta$;

4) 4) $U_R = U_1 \cup G_{LOOP}$, with the addition of arcs in such a way that:

   a) $\forall t \in End(U_1), t^{\bullet} = \{PL\}$;

   b) $q_{back}^{\bullet} = \{Start(U_1)\}$;

   c) $Start(U_R) = Start(U_1)$;

   d) $End(U_R) = \{q_{out}\}$.

H. Other Models

We formally defined the main basic blocks and operators for the modeling of complex business processes. Some operators and structures were not formally defined to make the paper less exhaustive to the reader. Readers can refer to Oliveira & Lima [26] for the formalization of the remaining structures.

I. Metrics and Analysis

The proposed model can be used for both qualitative (correctness) and quantitative (performance) analyses. The correctness of a subprocess is analyzed using the *soundness model*, shown in Fig. 8. This model allows the verification of the *soundness* property, as stated by van der Aalst and van Hee [2]:

Soundness. *A process is sound if it contains no unnecessary tasks and every case submitted to the process is completed in full and with no references to it remaining in the process*.

All role places receive resource tokens according to the scenario under study. If the model is *live* and *bound*, then it is *sound*.

Once the workflow is verified to be sound, performance evaluation can be performed. To evaluate the performance of the workflow, one must insert this model in an environment, where customers and resources are present. We define this as *workflow system*, as seen in Def. 27.

Definition 27 (Structure - Workflow System). A *workflow system*, defined as a tuple *Wf* = $(\lambda, \mathcal{P}, U, Emp)$, where:

1) $\lambda \in \mathbb{R}$[?] is the arrival rate, which indicates the rate at which cases are produced to the system;
2) $\mathcal{P}$ is a pool;

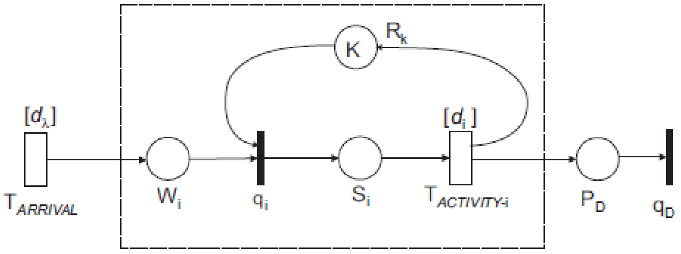


Fig. 9. Simplest Workflow System

3) $U \in$ *SProc* is the subprocess model that contains the *process definition*;
4) $Emp : \mathcal{P} \rightarrow \mathbb{N}_+$ is a *employing function*, which assigns a number of resources to each role.

is a GSPN composed of subprocess U with the following additional elements:

1) a timed transition $T_{ARRIVAL}$ with mean delay $d = 1/\lambda$, empty precondition and postcondition given by $T_{ARRIV AL}{}^{\bullet} = \{Start(U)\}$;

2) a place P_D with precondition ${}^{\bullet}P_D = End(U)$;

3) an immediate transition q_D with precondition ${}^{\bullet}q_D = \{P_D\}$ and empty postcondition;

4) an initial marking function M_0 such that $M_0(R_j) = Emp(r_j)$, $\forall r_j \in \mathcal{P}$, where $R_j \in P_U$ is the place representing role $r_j \in \mathcal{P}$.

Fig. 9 presents the simplest workflow system model. The subprocess contains just one activity and is highlighted by the dashed square. Notice that place R_k receives an initial marking K, corresponding to the value of $Emp(k)$.

It must be noticed that roles must be provided with the minimum number of resources, computed with the formula presented in Def. 13 to the system be able to reach an stationary state. Once the minimum number of resources is provided, we can retrieve stationary metrics.

An important metric that must be calculated for each activity or subprocess is the local arrival rate, i.e., the customer arrival rate at that specific point in the workflow. These rates can be obtained by the formulae below, on the basis of the composition operations applied.

Let U_R be a subprocess composed of a set of minor subprocesses U_i and λ be the customer arrival rate at the beginning of U_R, the local arrival rate λ_i at each subprocess U_i can be computed as follows.

- Sequence $U_R = SEQ(U_1, U_2)$:

  $$\lambda_1 = \lambda_2 = \lambda \ . \qquad (8)$$

- Alternative Path $U_R = XOR(U_1, \ . \ . \ . \ , U_N, \ Pr)$, where $Pr(U)$ is the probability of choosing the path U:

  $$\lambda_i = \lambda Pr(U_i), \quad i = 1, \ldots, N \ . \qquad (9)$$

- Parallelism $U_R = AND(U_1, \ . \ . \ . \ , U_N)$:

  $$\lambda_i = \lambda, \quad i = 1, \ldots, N \ . \qquad (10)$$

- Simple Iteration $U_R = LOOP(U_1, \ \theta)$, where θ is the probability of leaving the iteration:

  $$\lambda_1 = \frac{\lambda}{\theta} \ . \qquad (11)$$

- Grid-form Iteration $U_R = GRID - LOOP(\{UI_1, \ . \ . \ . \ , UI_{k+1}\}, \ \{UO_1, \ . \ . \ . \ , UO_k\}, \ Pr)$, where each UI_i is a subprocess in the iteration cycle, each UO_j is a subprocess executed after exiting from the exit point p_j and Pr maps a probability to each exit point to be taken

  $$\lambda^I{}_1 = \frac{\lambda}{\sum_{j=1}^{k} Pr(p_j)} \ ; \ \lambda^I{}_i = \lambda^I{}_1 \prod_{v=1}^{i-1} (1 - Pr(p_v)) \ ; \quad (12)$$

  $$\lambda^O{}_i = \lambda^I{}_1 \prod_{v=1}^{i} Pr(p_v) \ . \qquad (13)$$

- Multiple Path $U_R = OR(U_1, U_2, \ \alpha, \ \pi, \ \beta)$, where α is the probability of only U_1 be chosen, β is the probability of only U_2 be chosen, and π is the probability of both be chosen ($\alpha + \beta + \pi = 1$):

  $$\lambda_1 = (\alpha + \pi)\lambda \ ; \lambda_2 = (\beta + \pi)\lambda \ . \qquad (14)$$

- Interleaving $U_R = INTER(U_1, U_2, \ . \ . \ . \ , U_N)$:

$$\lambda_i = \lambda, \quad i = 1, \ldots, N \ . \qquad (15)$$

After computing the arrival rates, each metric from the activity model can be analytically obtained from the formulae presented in Table II, except for the worklist sizes. Also, the time spent at synchronization points cannot be obtained from these formulae. For obtaining these metrics, the complete GSPN must be evaluated. Notice that the *workflow system* is unbounded. Therefore, the GSPN must be evaluated through simulation. After obtaining the expected markings of the places of interest, the complete set of metrics can be computed.

For improving the precision of the results, we recommend that both theoretical results (obtained by formulae) and simulation results be combined for computing the final metrics. Every time a formula can be applied, its result should be used instead of the simulation results.

VI. CASE STUDY

With the purpose of validating the GSPN model, in this section we evaluate the performance of a real business process using three different approaches:

1) colored Petri nets (CPN) [2] - this technique has been widely used to evaluate the performance of several real processes, including the process that is analyzed in this case study [32];

2) Oracle BPM [28] - this is a complete set of tools for creating, executing, and optimizing business processes. The suite enables unparalleled collaboration between business and IT to automate and optimize business processes. The suite includes a simulator to evaluate the performance of business processes.

3) generalized stochastic Petri net (GSPN) - the approach proposed in this paper.

In this section, we apply these three techniques to evaluate the performance of the process used in the urban management service of a municipality located in north of Holland. This process is the focus of a process mining study as presented in Reijers et al. [32]. We use records of process executions collected in a period of six months. The workflow management system TIBCO Staffware [37] generated these records in the form of *event logs*.

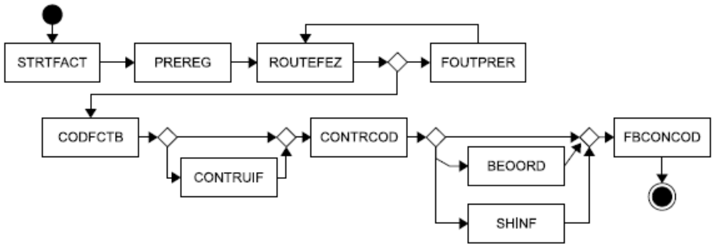


Fig. 10. Invoice Processing Workflow

TABLE III

ACTIVITY NAMES AND EXECUTION TIMES

Code	Label	Mean time (min.)
A1	STRTFACT	1.7977
A2	PREREG	0.4096
A3	ROUTEFEZ	0.1979
A4	CODFCTBF	0.2698
A5	FOUTPRER	0.5714
A6	CONTRUIF	0.5714
A8	CONTRCOD	0.3693
A9	BEOORDSR	0.5949
A10	SHINF	1.0
A12	FBCONCOD	0.3514

By employing process mining techniques available in the ProM (Process Miner) tool [34], we were able to discover semiautomatically the workflow model employed by the civil servants and find the statistics about its execution, including customer demand and process response time. ProM is also capable of generating a CPN model that may be used to simulate the process. During the simulation, the CPN model generates more *event logs*, which fit within the statistics of the real data. Using this technique, we assessed some important performance metrics of the process. For instance, we were able to extract the mean time of execution of each activity.

A. Context

The municipality has about 90,000 citizens and receives about 20,000 invoices per month [32]. The process involves almost every employee of the urban management service. Fig. 10 depicts the process structure.

Each invoice requires several checks that are made by different clerks, possibly in different geographical locations of the municipality (e.g., the mayor’s office, the fire brigade, etc.). It involves 110 participants, each performing multiple activities. In turn, each activity can be performed by different roles. The Dutch law states that governmental bodies need to pay their invoices within 30 days or risk financial penalties. For this reason, the performance of this process deserves special attention.

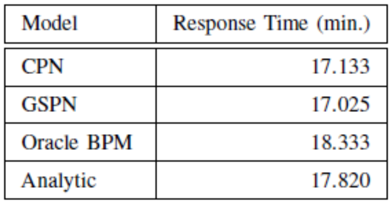
As stated before, we used ProM to measure the mean time of execution of each activity. Table III presents this information.

B. Experiments Conducted

We evaluated two scenarios: 1) the actual setting of the process (*as-is*) with the data retrieved from the execution logs; 2) simulating a stressed condition (*what-if* analysis) by multiplying the rate of invoice arrivals by a factor of 75 (this value was experimentally found to be high enough to generate queues in the system).

TABLE IV

RESPONSE TIMES FOR THE FIRST VERSION OF THE PROCESS (CONF. LEVEL. 95%)



Model	Response Time (min.)
CPN	17.133
GSPN	17.025
Oracle BPM	18.333
Analytic	17.820

The first scenario is used to validate the models against the real data mined from the event logs. The second scenario evaluates the capacity of each technique to identify bottlenecks (i.e., queues) that would appear when the process is executing under overloaded conditions.

We constructed three models:

- *CPN model*: discovered by the ProM tool from the actual event logs;
- *Oracle XPDL model*: designed using the Oracle BPM tool in the XML Process Definition Language (XPDL) [39];
- *GSPN model:* constructed in the TimeNet tool [9] using our approach.

In the first experiment, we simulated the three models and calculated the response time of the process (average execution time). It was computed as follows:

1) CPN model: by simulating the CPN model, synthetic logs were generated. These logs were used as input to ProM's performance analysis;

2) Oracle model: the XPDL model was simulated by the Oracle BPM's simulation feature;

3) GSPN model: this model was evaluated using TimeNet's stationary simulation feature.

The actual version of the process (with the original arrival rate) does not present queues. Notice that, when there are no queues in the system, our approach provides *analytical formulae* for computing the response time directly. Therefore, we also present the result of the analytical response time, calculated in this way. The results can be seen in Table IV. We applied the ANOVA test and concluded that the difference among the results is not statistically significant. GSPN and Oracle BPM results were calculated with a confidence level of 95% and an error of 10%. Analytical results are computed with exact formulae. CPN results fit with the data extracted from the event logs, meaning that they are statistically equal to the real process.

In the second set of experiments, with the purpose of generating a stress condition, we increased the arrival rate by 75 times the original rate and simulated each model again. We measured the resource demand and the queues formed on each process activity. Then, we computed the overall response time.

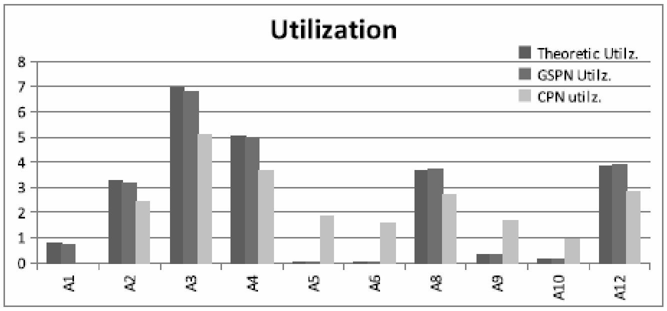


Fig. 11. Utilization of each activity in the second scenario (number of resources demanded)

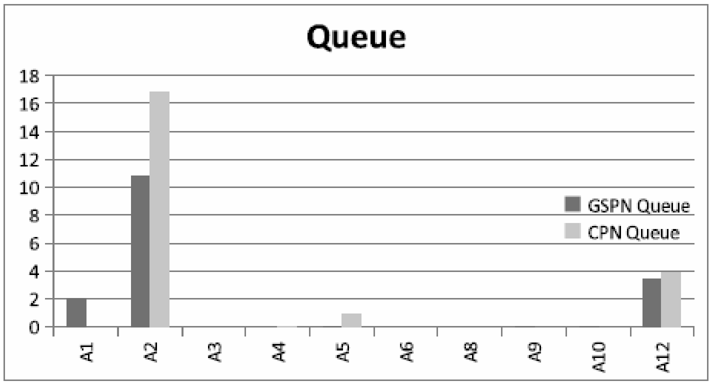


Fig. 12. Average queue sizes on each activity in the second scenario

The Oracle BPM tool could not reach a stationary state for this new configuration. The response time increases indefinitely. Therefore, we discarded its results.

We applied the paired t-test at the 95% confidence level to compare the *utilization* on each activity (i.e., the expected number of resources working on the activity). No statistical difference between GSPN and CPN models was found. We observed the same result when comparing the GSPN model against the analytical values for the utilization on each activity. Figure 11 shows the resource utilization for each activity (which corresponds to the mean number of resources demanded by each activity [3]). We also applied the paired t-test at the 95% confidence level to compare the average queue sizes on each activity during the simulation of the GSPN and the CPN models. Again, we found no statistical difference between the GSPN and the CPN models. Figure 12 presents the queue sizes as calculated by the GSPN and the CPN methods. Notice that activity $A2$ has proven to be the bottleneck of this system.

Table V presents the response times for the new scenario. The CPN model generated logs that were statistically analyzed using the ProM tool. The result labeled *GSPN* does not employ the analytical formulae proposed in this work, but computes all metrics from the results of simulation. The result labeled as *Analytic+GSPN* refers to the response time for the combination of the analytical formulae that our approach provides and the results of GSPN simulation. This combination provides more accurate results than simulation alone.

From the experiment, we can infer that the GSPN results are consistent with the findings of popular tools: colored Petri nets, on the academic side; and Oracle BPM, on the industrial side. But the added advantage of our approach is that we are able to determine specific results analytically instead of

TABLE V

RESPONSE TIMES FOR THE SECOND VERSION OF THE PROCESS (CONF. LEVEL. 95%)

Model	Response Time (min.)
CPN	32.89
GSPN	29.71
Oracle BPM	No results
Analytic+GSPN	29.86

on the basis of simulation, aside to other arguments to use GSPN instead of colored Petri nets for performance modeling and evaluation (see Section IV). It is also worth mentioning that, despite being widely used in industry, the Oracle BPM tool has proven to be ineffective to evaluate the process under overloaded conditions.

VII. CONCLUSIONS

We proposed the use of generalized stochastic Petri nets (GSPN) as a basis to support both the correctness verification and the performance evaluation of realistic business processes. We showed that GSPN provides several benefits in contrast to currently used techniques. To assure the correct mapping between workflow concepts and GSPN models, we designed a set of building blocks and composition operations that can be used to represent the key components present in most workflow languages. Such structures and operations enable the modeler to create GSPNs that provide a wide range of qualitative and quantitative information about a workflow.

We can enumerate the following main contributions that distinguish our work from current approaches: 1) we can *analytically* assess a wide range of performance metrics, such as throughput and utilization - a feature not found in related works; 2) we support the evaluation of processes with *multiple* customers and a limited number of possibly *shared* resources, while several related works are limited in this context; 3) we use the same model both for *performance evaluation* and analyses of *qualitative properties* of processes, such as soundness and liveness. Most of the related works are not intended to analyze qualitative properties at all; 4) Performance metrics that cannot be analytically calculated may be alternatively assessed through simulation, also without changes in the proposed model.

The list of criteria that we used for the comparison of the various existing methods to analyze the performance of business processes can be seen as an additional contribution of this work. Such criteria were employed to compare eleven different works, including the new approach proposed in this paper and can be used as a basis for future comparisons in other works.

From the comparative study performed, we observed that CPN-based approaches demonstrated to cover most of the desirable characteristics prescribed by our list of criteria. However, one noticeable drawback of CPN models is that they represent time as integer values. Rounding timestamp values to integer values can potentially cause a loss of precision. In contrast, GSPNs deal with continuous time, as such providing more accurate results. Furthermore, CPN traditionally requires the codification of process data and certain decision algorithms to an extent similar to that necessary for implementing the real workflow model. Our approach does not require such refinements and provide evidence that these data are not relevant for the results of the performance analysis.

Eventually, analyzing the set of criteria proposed in this paper, our approach achieves at least the same level of quality observed in those CPN-based approaches of higher quality. This is not the case for other works that also employ the GSPN formalism. Therefore, this analysis revealed that our work incorporates more desirable characteristics than those observed in other works, which adopt GSPN for modeling and analyzing the performance of business processes. Moreover, due to the use of continuous time, the choice for GSPN potentially provides more accurate results when compared with CPN. Overall, we believe that the proposed approach significantly extends the state of the art and should be considered as the preferred framework to assess both quantitative and qualitative aspects of complex business processes.

To validate our approach, we used the event logs from a real business process (the urban management service of a municipality situated in the northern part of the Netherlands). We employed the GSPN model, a CPN-based approach, and the commercial tool Oracle BPM for evaluating two scenarios: one with a low resource demand, in correspondence with the real system; and another with a hypothetical high demand.

The results of the evaluation were very satisfactory as they showed quite similar outcomes from our approach to those obtained through a well-known CPN-based approach for a realistic situation, even though we could establish these results *analytically* instead of on the basis of simulation results. We believe that the increased efficiency and modeling ease of using an analytical approach in combination with a satisfactory accuracy is one of the main advantages of our approach. It must be emphasized that the CPN model employed was built up on the basis of measured data, by the application of validated process mining techniques [32].

As a final contribution, we developed the *BPEL2Net* tool for translating BPEL workflow descriptions into a GSPN model. The compiler uses the basic models and composition rules as proposed in this paper. The tool can be obtained at http://www.cin.ufpe.br/˜calo/bpel2net.

As a future work, we will implement a framework with several plugins to support the design and the performance analysis of business processes using our proposal. We are currently working on a plugin to support the graphical modeling of workflows. Furthermore, we plan to develop plugins for communicating with the TimeNet tool from our graphical interface and importing workflows modeled using different tools and languages.

REFERENCES

[1] W.M.P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1):21– 66, 1998.

[2] W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT press, Cambridge, MA, 2002.

[3] Gunter Bolch, Stefan Greiner, Hermann de Meer, and Kishor Shridharbhai Trivedi. *Queueing Networks and Markov Chains*. Wiley-Interscience, 2005.

[4] Steven C. Bruell, Pozung Chen, and Gianfranco Balbo. Alternative methods for incorporating non-exponential distributions into stochastic timed Petri nets. In *PNPM*, pages 187–197. IEEE Computer Society, 1989.

[5] Workflow Management Coalition. WfMC standards: The workflow reference model, version 1.1., 1995.

[6] Workflow Management Coalition. Workflow management coalition terminology and glossary, version 3.0 (WFMC-TC-1011). Technical report, Workflow Management Coalition, Brussels, 1999.

[7] Juliane Dehnert, Jorn Freiheit, and Armin Zimmermann. Modeling and performance evaluation of workflow systems, 2000.

[8] A. Ferscha. Qualitative and quantitative analysis of business workflows using generalized stochastic Petri nets, 1994.

[9] Reinhard German, Christian Kelling, Armin Zimmermann, and G¨unter Hommel. TimeNET: A toolkit for evaluating non-markovian stochastic Petri nets. *Perform. Eval*, 24(1-2):69–87, 1995.

[10] Steven C. Bruell Gianfranco Balbo and Matteo Sereno. Product form solution for generalized stochastic Petri nets. *IEEE Transactions on Software Engineering*, 28(10):915–932, 2002.

[11] Jiang Hao and Wang Pei-an. An approach for workflow performance evaluation based on discrete stochastic Petri net. In *ICEBE '07: Proceedings of the IEEE International Conference on e-Business Engineering*, pages 327–330, Washington, DC, USA, 2007. IEEE Computer Society.

[12] Y. C. Ho. Dynamics of discrete-event systems. *Proceedings of the IEEE*, 77(1):3–6, 1989.

[13] Y. C. Ho. *Discrete Event Dynamical Systems: Analyzing Complexity and Performance in the Modern World*. IEEE Press, New York, 1991.

[14] Kurt Jensen. An introduction to the theoretical aspects of coloured Petri nets. In *A Decade of Concurrency, Reflections and Perspectives, REX School/Symposium*, pages 230–272, London, UK, 1994. Springer-Verlag.

[15] Kurt Jensen. *Coloured Petri nets: basic concepts, analysis methods and practical use, vol. 2*. Springer-Verlag, London, UK, 1995.

[16] Kurt Jensen. An introduction to the practical use of coloured Petri. In *Lectures on Petri Nets II: Applications*, 1998.

[17] John Jeston and Johan Nelis. *Business Process Management : Practical Guidelines to Successful Implementations*. Elsevier/Butterworth- Heinemann, Amsterdam, 2006.

[18] D. Kreische. Performance and dependability in business process modeling. In *Proceedings of 5th Int. Workshop on Performability Modeling of Computer and Communication Systems PMCCS 5*, Erlangen-N¨urnberg, 2001.

[19] JianQiang Li, Yushun Fan, and MengChu Zhou. Performance modeling and analysis of workflow. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 34(2):229–242, 2004.

[20] Shuxia Li and Haiping Zhu. Generalized stochastic workflow netbased quantitative analysis of business process performance. In *ICIA '08: Proceedings of the IEEE International Conference on Information and Automation*, pages 1040–1044, Washington, DC, USA, 2008. IEEE Computer Society.

[21] Dongsheng Liu, Jianmin Wang, Stephen C. F. Chan, Jiaguang Sun, and Li Zhang. Modeling workflow processes with colored Petri nets. *Computers in Industry*, 49(3):267–281, 2002.

[22] Menish Malhotra and Andrew Reibman. Selecting and implementing phase approximations for semi-Markov models. *Stochastic Models*, 9(4):473–506, 1993.

[23] M. Ajmone Marsan, G. Balbo, and G. Conte et al. *Modelling with Generalized Stochastic Petri Nets*. Wiley series in parallel computing. Wiley, New York, 1995.

[24] Michael K. Molloy. Performance analysis using stochastic Petri nets. *IEEE Trans. Computers*, 31(9):913–917, 1982.

[25] M. Netjes, W.M.P. van der Aalst, and H. A. Reijers. Analysis of resource-constrained processes with colored Petri nets. In K. Jensen, editor, *Proceedings of the Sixth Workshop on the Practical Use of Coloured Petri Nets and CPN Tools (CPN 2005)*, volume 576 of *DAIMI*, pages 251–266, Aarhus, Denmark, October 2005. University of Aarhus.

[26] Cesar Oliveira and Ricardo Lima. Performance analysis of resource constrained business processes: A formal approach based on stochastic Petri nets. Technical report, Federal University of Pernambuco, 2009.

[27] OASIS Open. Web service business process execution language (wsbpel) version 2.0. Technical report, OASIS Open, 2007.

[28] Oracle. Oracle business process management (BPM). web site, 2009. http://www.oracle.com/technologies/bpm/bpm.html.

[29] C. A. Petri. *Kommunikation mut Automaten*. PhD thesis, Schriften des IIM Nr. 2, Bonn, 1962.

[30] Michael E. Porter. *Competitive Advantage: Creating and Sustaining Superior Performance*. The Free Press, New York, 1985.

[31] H.A. Reijers. *Design and Control of Workflow Processes: Business Process Management for the Service Industry*. PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 2002.

[32] H.A. Reijers, M. Song, and B. Jeong. Analysis of a collaborative workflow process with distributed actors. *Information Systems Frontiers*, 11(3):307–322, 2009.

[33] Robert Harper. *Programming in Standard ML*. Carnegie Mellon University, 2005.

[34] A. Rozinat, RS Mans, M. Song, and WMP van der Aalst. Discovering simulation models. *Information Systems*, 34(3):305–327, 2009.

[35] Dmytro Rud, Andreas Schmietendorf, and Reiner Dumke. Performance modeling of ws-bpel-based web service compositions. In *SCW '06: Proceedings of the IEEE Services Computing Workshops*, pages 140– 147, Washington, DC, USA, 2006. IEEE Computer Society.

[36] A. K. Schomig and H. Rau. A Petri net approach for the performance analysis of business processes. Technical report, University ofW¨urzburg, 1995.

[37] TIBCO. TIBCO staffware process monitor (SPM). web site, 2005. http://www.tibco.com.

[38] CPN Tools. Cpn tools help pages, aug 2008.

[39] WFMC. Workflow management coalition workflow standard: Workflow process definition interface – XML process definition language (XPDL) (WFMC-TC-1025). Technical report, Workflow Management Coalition, Lighthouse Point, Florida, USA, 2002.

[40] Stephen A. White. Introduction to bpmn. Technical report, IBM Software Group, 2006.