*International Journal of Innovative Computing, Information and Control*
Volume **8**, Number **7(B)**, July **2012**

ICIC International ©2012 ISSN 1349-4198
pp. 5295–5316

# PERFORMANCE EVALUATION OF RESOURCE-AWARE BUSINESS PROCESSES USING STOCHASTIC AUTOMATA NETWORKS

Kelly Rosa Braghetto[1], João Eduardo Ferreira[1]
and Jean-Marc Vincent[2]

[1]Department of Computer Science
University of São Paulo
Rua do Matão, 1010, Cidade Universitária, 05508-090, São Paulo, Brasil
{ kellyrb; jef }@ime.usp.br

[2]LIG Laboratory – INRIA MESCAL Project
Joseph Fourier University
51, avenue Jean Kuntzmann, F-38330, Montbonnot, France
Jean-Marc.Vincent@imag.fr

Received March 2011; revised October 2011

Abstract. *In this work, we study the performance evaluation of resource-aware business process models. We define a new framework that allows the generation of analytical models for performance evaluation from business process models annotated with resource management information. This framework is composed of a new notation that allows the specification of resource management constraints and a method to convert a business process specification and its resource constraints into Stochastic Automata Networks (SANs). We show that the analysis of the generated SAN model provides several performance indices, such as average throughput of the system, average waiting time, average queues size, and utilization rate of resources. Using the* `BP2SAN` *tool – our implementation of the proposed framework – and a SAN solver (such as the PEPS tool) we show through a simple use-case how a business specialist with no skills in stochastic modeling can easily obtain performance indices that, in turn, can help to identify bottlenecks on the model, to perform workload characterization, to define the provisioning of resources, and to study other performance related aspects of the business process.*

**Keywords:** Business processes, Performance evaluation, Stochastic modeling, Stochastic automata networks

## 1. Introduction.

In order to meet the quality of service demanded by its users, a business process needs to be appropriately provisioned. The measure of the quality of service may be associated with factors that vary depending on the system type and on the user requirements, and may regard both qualitative and quantitative aspects of the system [12]. One important quantitative aspect that greatly impacts the system's quality of service is the *performance*. The performance analysis enables us to deal with problems frequently found in the computational systems such as comparison of systems, bottleneck identification, workload characterization, system tuning and performance forecasting [14].

Business process activities usually depend on different resources to be executed. The expected performance of a business process depends on how these resources are provisioned and used. The *resource management* defines (i) what are the resources required by the system, (ii) how many they are, and (iii) how they are accessed. In this work, we define a new proposal to enrich business process models with information concerning resources requirements in order to enable performance evaluation via analytical modeling.

The analytical modeling for performance evaluation is generally made over stochastic models that have as underlying formalism a Markov process. As happens with other concurrent systems, business processes are hard to be modeled using the traditional Markovian methods due to the complexity of their requirements, and even harder to be numerically solved due to their potentially large state spaces.

A feasible stochastic technique used to model systems with large state spaces in a structured way is the *Stochastic Automata Networks* (SANs) [18, 19]. Unlike other Markovian analysis techniques that require the generation of a state transition matrix, the internal representation of a SAN model (that is based on *Tensor Algebra*) remains compact even when the number of states of its underline Markov chain begins to explode.

The main contribution of this work is a framework for the stochastic modeling of resource management in business processes using SAN. Our framework is composed of (i) a novel notation to enrich BPMN process diagrams with annotations that define the resources requirements of the processes being modeled, and (ii) a methodology to generate SAN models contemplating resource management from the annotated business process models. We implemented this framework as part of a software tool called `BP2SAN`.

With `BP2SAN`, the resource management is first modeled in a high abstraction level. The stochastic model that expresses how the execution of the activities is impacted by the resources they depend on is automatically inferred by our method from the annotated business process models. Using a numerical solver for SAN, such as the open-source software tool PEPS [6], varied performance indices can be extracted from the business process SAN models generated through our modeling framework.

This text is organized as follows. Section 2 discusses other research works concerning the stochastic modeling of business processes. A brief presentation of SAN is given in Section 3. In Section 4, we define a notation to consider resource management in business process modeling. Section 5 deals with the problem of resource allocation under the perspective of stochastic modeling. With the support of a simple example, we define a methodology to model the resource management of business processes in SAN. Section 6 explains and illustrates how performance indices can be extracted from the SAN models generated through our modeling framework. Concluding remarks are made in Section 7.

## 2. Related Works.

The techniques generally used to model business processes, such as the *Business Process Model and Notation* (BPMN) [17], the *Unified Modeling Language* (UML), and the *Event-driven Process Chain* (EPC), do not directly support formal analysis. In addition, they are focused in the control-flow perspective or provide a view of resource management that is insufficient to support the performance analysis via analytical modeling. Several works such as [9, 10, 22] already discussed the conversion of business process models to formalisms (e.g., Petri nets and process algebras) aiming verification and validation. Other approaches such as [3, 7, 20] are devoted to the conversion of business process models to stochastic formalisms aiming quantitative analysis.

The work of Canevet et al. [7] proposed an automated mapping from UML state diagrams enhanced with performance information to *Performance Evaluation Process Algebra* (PEPA) [13]. This performance information they refer is probabilities attached to states and rates attached to transitions of the UML state model. In the proposed approach, the performance results obtained from the solution of the PEPA model can be reflected back to the UML level. However, the approach does not support functional rates, preventing some important aspects related to performance from being represented in the modeling.

Prandi et al. [20] proposed a mapping from BPMN to *Calculus for Orchestration of Web Services* (COWS), a process calculus inspired by the *Business Process Execution Language* (BPEL). The authors made a brief discussion about the use of a stochastic extension of COWS to support quantitative analysis of business processes. Despite being a compositional formalism, Stochastic COWS does not exploit the advantage of compositionality in the analysis method, and thus it suffers from the same state-space explosion problem that limits the use of other Markovian formalisms in the analysis of large-scale systems.

Braghetto et al. [3] discussed the viability of applying three different stochastic formalisms – the *Generalized Stochastic Petri Nets* (GSPN) [2], PEPA and SAN – in the modeling for numerical analysis of performance of business processes initially modeled in BPMN. They verified that the three formalisms are able to express with equivalent facilities basic business process scenarios. However, more advanced scenarios evidenced their pros et cons. Since SAN and PEPA are intrinsically compositional formalisms, they enable structured analyses, in addition to the facility to extend a model without impacting the previous modeled behavior. SAN and GSPN have the explicit notion of state and the concept of functional rates, which helps to model functional dependencies between the components of a process. The authors created an algorithm to automatically convert BPMN models to SAN [4]. However, the proposed conversion generates stochastic models with no information about execution rates and resource management.

Sauer and Chandy [21] stated that the *contention for resources* is a very significant factor in performance and the most difficult one to quantify. Some approaches [3, 7, 20] force the modeler to explicitly specify all the resources requirements in the business process model in order to have a stochastic model able to provide an accurate performance analysis.

In this work, we extend the mapping defined in [4], by simplifying the modeling of resource management for business process designers at the same time as the accuracy of the SAN models automatically generated from BPMN models is improved.

## 3. Stochastic Modeling.

The stochastic formalisms most used for performance analysis are the Markovian ones. A Markov process is a stochastic process that respects the *memoryless property*, which states that the conditional probability of a future state of the process depends only on the present state, and it is independent of the past states. As examples of well-known Markovian modeling techniques (i.e., techniques that generate models with an underlying Markov chain), we have the queueing networks [16], the stochastic Petri nets [11], and the stochastic process algebras [8].

We can make two kinds of numerical analysis of a stochastic model: *steady state analysis* and *transient analysis*. The steady state analysis gives us the *stationary probability distribution* of the process, i.e., the long-run average time the process spends in each one of its states. From this stationary distribution we can extract performance indices, such as average throughput of the system, average waiting time, average queues size, and utilization rate of resources. The transient analysis investigates the transient behavior of the process, e.g., the state of the process at the end of a time interval, the time until an event occurs, and the number of occurrences of an event in a time interval.

### 3.1. Stochastic automata networks – a short presentation.

The *Stochastic Automata Network* (SAN) is a technique used to model systems with large state spaces, introduced by Plateau in 1985 [18, 19]. SAN has been successfully applied to model parallel systems that can be viewed as collections of components that operate more or less independently, requiring only infrequent interactions such as synchronizing their actions, or operating at different rates depending on the state of parts of the overall system.

A system is described in SAN as a set of $N$ subsystems modeled as a set of stochastic automata $A^{(i)}$, $1 \leq i \leq N$, each one containing $n_i$ local states and transitions among them.

The global state of a SAN is the combination of the internal state of each automaton. A change in the state of a SAN is caused by the occurrence of an *event*. *Local events* cause a state transition in only one automaton (*local transition*), while *synchronizing events* cause simultaneous state transitions in more than one automaton (*synchronizing transitions*). A transition is labeled with the list of events that may trigger it.

Transitions are associated with rates that indicate the average frequency (i.e., the inverse of the average execution time) in which the transitions occur. The rate of an event may be constant (a nonnegative real number) or may depend upon the state in which it takes place. In this last case, the rate is a function from the global state space to the nonnegative real numbers and it is called *functional transition rate*.

The expression of the *infinitesimal generator* (transition rate matrix) of the Markov chain associated with a well defined SAN is given by the generators on these smaller spaces and by operators from the *Generalized Tensor Algebra* (GTA) [5], an extension of the *Classical Tensor Algebra* (CTA). The tensor formula that gives the infinitesimal generator of a SAN model is called *Markovian Descriptor*.

Each automaton $A^{(i)}$ of a SAN model is described by $n_i \times n_i$ square matrices. In the case of SAN models with synchronizing events, the descriptor is expressed in two parts: a *local* part (to group the local events) and a *synchronizing* part (to group the synchronizing events). The local part is defined by the tensor sum of $Q_l^{(i)}$, the infinitesimal generator matrices of the local transitions of each $A^{(i)}$. In the synchronizing part, each event corresponds to two tensor products: one for the occurrence matrices $Q_{s^+}^{(i)}$ (expressing the positive rates) and the other for the adjusting matrices $Q_{s^-}^{(i)}$ (expressing the negative rates). The descriptor is the sum of the local and the synchronizing parts, expressed as:

$$Q = \bigoplus_{i=1}^{N}{}_{g}\, Q_l^{(i)} + \sum_{s \in \varepsilon} \left( \bigotimes_{i=1}^{N}{}_{g}\, Q_{s^+}^{(i)} + \bigotimes_{i=1}^{N}{}_{g}\, Q_{s^-}^{(i)} \right)$$

where $\begin{cases} N & \text{is the number of automata of the SAN model} \\ \varepsilon & \text{is the set of identifiers of synchronizing events} \end{cases}$.

The state space explosion problem associated with Markov chain models is attenuated by the fact that the state transition matrix is stored in a compact form, since it is represented by smaller matrices. All relevant information can be recovered from these matrices without explicitly building the global matrix.

A SAN model can be numerically solved using the PEPS tool [6][1]. PEPS includes several numerical iterative methods to solve SAN models and implements strategies to improve the time/space trade-off in the computation of solutions.

## 4. Considering Resource Management in Business Process Modeling.

If we want performance analyses that really approximate the results expected for real-world business processes, we need build SAN models that express the resource management policy associated with these processes, i.e., (i) which are the activities' resources requirements, and (ii) how the resources are shared between activities executed in parallel.

In BPMN, we can associate *Resource Roles* with activities. The resource is who will perform or be responsible for the activity, and it can be specified in the form of a specific individual, a group, an organization role or position, or an organization. This definition is considerable more restrictive than the general concept of resource in computer science.

In the context of this work, we define *resource* as something required to accomplish an activity of the business process model. It can be either a physical entity (processors,

[1]PEPS and details about it are available in http://www-id.imag.fr/Logiciels/peps/index.html.

memory, printers, human beings, organizations, etc.) or a virtual entity (software libraries, web services, databases, etc.). An activity may require the access to one or more resources to be performed. A resource may be required by one or more activities. As activities (of either a same type or different types) can be executed parallelly in business process management systems, resources can be concurrently accessed.

The execution time of an activity is related to its resources requirements and may vary with the workload of the system. For example, consider a web service that depends on a web server to be executed. If the server deals with only one request at a time, the execution time of the web service will not be impacted by the workload. If a new request arrives while the web server is busy, the new request will be enqueued to be processed posteriorly. However, in a more realistic assumption, the server will treat all its requests parallelly. In this case, the processing capacity of the server will be divided between the active requests, and the execution time of the web service will increase with the workload.

In the next sections, we define a new approach to specify resource management in business process models and to take it into account in the creation of a performance evaluation model using SAN. Our proposal for resource management description in business process models can be divided in two phases: (i) the description of the available resources, and (ii) the description of the resources requirements of each activity.

## 4.1. Phase 1 – description of the available resources.

In the following, we formally define the notion of resource used in this work.

**Definition 4.1.** *A* resource *is a quadruple*

$$R = ([\textit{resource id}], [\textit{quantity}], [\textit{work capacity}], [\textit{access discipline}]), \ \textit{where}$$

- [*resource id*] *is an unique identifier that specifies the type of the resource;*
- [*quantity*] *is a positive integer number expressing how many units of the resource are available to be accessed;*
- [*work capacity*] *is a positive decimal number defining the average quantity of work that one unit of the resource can process per unit of time;*
- [*access discipline*] *is a strategy that defines how activities are assigned to the resource.*

There are several access disciplines, such as the ones commonly found in the terminology of queueing networks: *first-in-first-out* (FIFO), *last-in-first-out* (LIFO) and *priority systems.* In this work, we consider two access disciplines: *random choice* and *time sharing.* Under the random choice, an activity will be randomly selected between the others waiting to access the resource. Under time sharing, the using time of the resource will be equally divided between all activities requesting access to it.

**Definition 4.2.** *The Resource Set (RS) of a business process* $\boldsymbol{p}$ *is the set* $RS(\boldsymbol{p}) = \{R_i \mid R_i \textit{ is a resource required by an activity of the business process } \boldsymbol{p}\}$.

**Example 4.1.** *The resource set*

$$RS(\boldsymbol{p}) = \{(\textit{“server1”}, 2, 5.0, \textit{“Time Sharing”}), (\textit{“server2”}, 1, 10.0, \textit{“Time Sharing”}), \\ (\textit{“printer”}, 3, 1.5, \textit{“Random Choice”})\}$$

*corresponds to a business process model* $\boldsymbol{p}$ *with three different types of resources, where*

- *there are 2 resources of type “server1”, each one processes 5 units of work per unit of time and has the time sharing as access discipline;*
- *there is only 1 resource of type “server2”, that processes 10 units of work per unit of time and has the time sharing as access discipline;*
- *there are 3 resources of type “printer”, each one processes 1.5 units of work per unit of time and has the random choice as access discipline.*

4.2. **Phase 2 – description of the resources requirements of activities.** In order to be able to consider resource management in the performance evaluation models automatically generated from business process models, we make two assumptions about the resource usage description of the business process models considered in this work:

- the resources requirements are time-homogeneous and state-independent, i.e., the quantity of work and the number of resources required by each activity do not vary with the time or with the state changes of the process;
- an activity only will be executed when *all* its required resources are available. In an analogous way, the used resources will be released all together, at the end of the execution of the activity. Otherwise, if the resources could be allocated or released in an independent way, one would note that the referred activity has not an indivisible behavior and could be remodeled as a set of parallel atomic activities.

Definitions 4.3 and 4.4 formalize our concept of activity's resources requirements.

**Definition 4.3.** *A Single Resource Requirement (SRR) of an activity can be expressed by a pair ([resource id], [quantity of work]), where*

- *[resource id] is the identifier of the resource type required by the activity;*
- *[quantity of work] is a positive decimal number that determines how many units of work the activity will require of the resource.*

**Definition 4.4.** *The Resources Requirements (RR) of an activity* $\boldsymbol{a}$*, denoted by* $RR(\boldsymbol{a})$*, can be described by an expression created using SRRs and two logical operators for composition:* $\wedge$ *(AND) or* $\vee$ *(OR).*

*In an RR expression, the AND operator has a higher precedence than OR; parentheses may be used to force the order of operations.*

**Example 4.2.** *Consider an activity* $\boldsymbol{a}$*, with*

$$RR(\boldsymbol{a}) = (\textit{``printer''}, 4.0) \wedge ((\textit{``server1''}, 5.5) \vee (\textit{``server2''}, 5.5)).$$

*The resources requirements of activity* $\boldsymbol{a}$ *express that the activity requires 1 resource of type "printer" to process 4.0 units of work and 1 resource of type "server1" or "server2" to process 5.5 units of work.*

If an activity requires more than one unity of a resource, then we can express this in its RR expression using different SRRs with the same resource type, as shown in Example 4.3.

**Example 4.3.** *Consider an activity* $\boldsymbol{b}$*, with*

$$RR(\boldsymbol{b}) = (\textit{``printer''}, 4.0) \wedge (\textit{``printer''}, 2.0).$$

*The resources requirements of* $\boldsymbol{b}$ *express that the activity requires two resources of type "printer" – one to process 4.0 units of work, and the other to process 2.0 units of work.*

We opted to represent a requirement for $x$ units of a resource $r$ as $x$ SRRs for $r$ (as we did in Example 4.3) because this enables the modeler to explicitly define how the total work required of $r$ will be divided between the $x$ units of the resource.

The set of the resources requirements of all activities of a business process model gives us the resource management policy of that business process. In the following, we make a complementary definition that will help us in the discussion made in Section 5.

**Definition 4.5.** *The Disjunctive Resources Requirements (DRR) of an activity* $\boldsymbol{a}$*, denoted by* $DRR(\boldsymbol{a})$*, is the set of all possible combinations of resources that enable the execution of activity a.*

*All set of resources* $\mathcal{S} \in DRR(\boldsymbol{a})$ *must respect the following property:* $\mathcal{S}$ *is a minimal set of SRRs of activity* $\boldsymbol{a}$ *that, when satisfied, can guarantee the execution of* $\boldsymbol{a}$*.*

We can obtain the DRR through the RR of an activity by the following simple steps:

- if the RR expression is in a *Conjunctive Normal Form* (CNF), then convert it to an equivalent expression in a *Disjunctive Normal Form* (DNF). A RR expression is in a CNF if it is a conjunction of clauses, where a clause is a disjunction of SRRs. Analogously, an RR expression is in a DNF if it is a disjunction of clauses, where a clause is a conjunction of SRRs. A conjunction is the operation associated with the operator AND, while a disjunction is the operation associated with the operator OR;
- each conjunctive clause of an RR expression in a DNF will originate a new set of the DRR. The elements of this set are the SRRs that appear in the conjunctive clause.

To illustrate the concept of DRR, consider Example 4.2. The RR expression of the example is in a CNF. The equivalent expression in a DNF is

$$RR(\boldsymbol{a}) = ((\text{“printer”}, 4.0) \wedge (\text{“server1”}, 5.5)) \vee ((\text{“printer”}, 4.0) \wedge (\text{“server2”}, 5.5)).$$

The disjunctive resources requirements associated to activity $\boldsymbol{a}$ are

$$DRR(\boldsymbol{a}) = \{\{(\text{“printer”}, 4.0), (\text{“server1”}, 5.5)\}, \{(\text{“printer”}, 4.0), (\text{“server2”}, 5.5)\}\}.$$

5. **Considering Resource Management in Performance Evaluation Models.** We will introduce our modeling approach using as support an example of a *production process* – the production of metal workpieces of a small machine shop – described in the sequence. The production processes are a subset of the business processes. Although a production process does not contain sophisticated control-flow structures, it contains complex and varied resource requirements that better illustrate our method.

**Example 5.1.** *The production of metal workpieces in a small machine shop.*

A machine shop is a place where workpieces are manufactured by machine tools. The machine shop of our example works with *Computer Numerical Controlled* (CNC) machines, i.e., the machining is controlled by computers, without human intervention. The CNC machines manufacture workpieces using 3D-models specified by designers. The machine shop has two identical CNC machines that can manufacture only small pieces and one CNC machine that can manufacture both small and big pieces. We will call the first type of machine `CNC1` and the second type `CNC2`, for short. Each CNC machine can only produce an item at a time. The production process starts with the arrival of a new workpiece order. Then, two designers are required to make the 3D-model and to program the CNC machine that will manufacture the workpiece. A designer can work in the model of only one workpiece at a time, and the machine shop has 4 designers. If the workpiece is a big one, the production will require the use of `CNC2`. In the other case, the production will require `CNC2` or one of the units of `CNC1`. After exiting the CNC machine, the workpiece is painted. The machine shop has only one painting machine. Since each workpiece must receive several layers of paint, the painting machine must wait for a layer to dry before painting a new layer in the same workpiece. Thus, in a given moment of the time, we can have several workpieces being painted. After exiting the painting machine, the workpiece will be packed by a human packer, and the production process will be finished. In order to be profitable, the machine shop can only accept orders requiring a quantity of workpieces that is in a pre-defined acceptable range.

Figure 1 shows the BPMN model of this production process. The thin-lined circle represents a *start event*, while the thick-lined circle represents an *end event*. Boxes represent atomic activities, while diamonds represent *gateways* that can be *divergent* (as the leftmost diamond in Figure 1) or *convergent* (as the rightmost diamond in Figure 1). A gateway labeled with “×” is an *exclusive gateway* (also known as *decision gateway* or, in

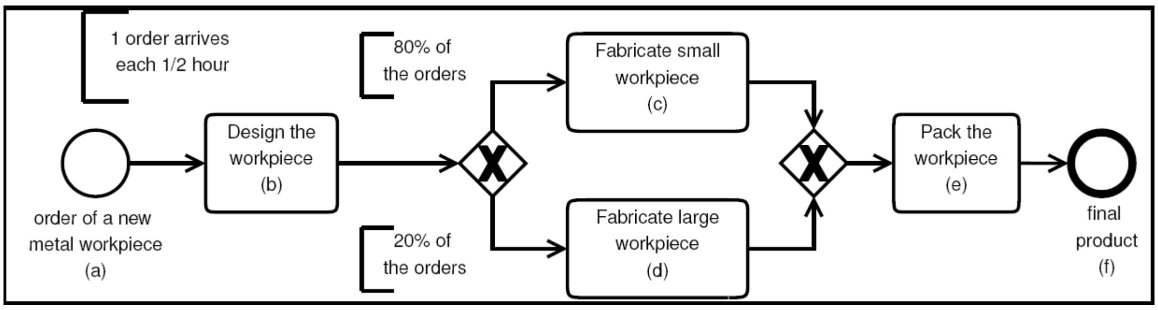

1 order arrives each 1/2 hour
80% of the orders
20% of the orders
order of a new metal workpiece (a)
Design the workpiece (b)
Fabricate small workpiece (c)
Fabricate large workpiece (d)
Pack the workpiece (e)
final product (f)

FIGURE 1. BPMN model of the production process of a small machine shop

the workflow terminology, OR-split/join). The directed arrows indicate the sequence flow of the process. The textual notes with a thin black border on the left are *annotations* to provide additional information for readers of the BPMN diagram[2].

To define the resource set and the resources requirements of the business process of Example 5.1, we are considering the following information:

- a designer takes in average 8 hours to make the 3D-model of a workpiece alone (work capacity = 0.125 3D-models per hour). But two designers together take in average 4 hours to model a workpiece;
- the average size of an order is 20 for a small workpiece, and 10 for a big workpiece;
- a `CNC1` wastes in average 10 minutes to manufacture 1 small metal volume (work capacity = 6 small metal volumes per hour);
- the `CNC2` wastes in average 30 minutes to manufacture 1 big metal volume (work capacity = 2 big metal volumes per hour);
- 1 big metal volume = 2 small metal volumes;
- 1 small workpiece in average requires 3 small metal volumes (or 1.5 big metal volumes) to be manufactured. Thus, an entire order of a small workpiece requires in average $3 \times 20 = 60$ small metal volumes to be manufactured;
- 1 big workpiece in average requires 2 big metal volumes to be manufactured. Thus, an entire order of a big workpiece requires in average $2 \times 10 = 20$ big metal volumes;
- the painting machine takes 6 minutes to paint 1 small volume (work capacity = 10 small metal volumes per hour);
- the packer wastes in average 1 hour to pack the products of an order (work capacity = 1 order per hour).

According to Definitions 4.2 and 4.4, the declaration of the resources set and the requirements set of the production process of metal workpieces would be

$$\begin{aligned} RS = \{ & (\text{“Designer”}, 4, 0.125, \text{“Random Choice”}), \\ & (\text{“CNC1”}, 2, 6.0, \text{“Random Choice”}), (\text{“CNC2”}, 1, 2.0, \text{“Random Choice”}) \\ & (\text{“PaintingM”}, 1, 10.0, \text{“Time Sharing”}), (\text{“Packer”}, 1, 1.0, \text{“Random Choice”})\} \end{aligned}$$

$$\begin{aligned} RR(b) &= (\text{“Designer”}, 0.5) \wedge (\text{“Designer”}, 0.5) \\ RR(c) &= ((\text{“CNC1”}, 60.0) \vee (\text{“CNC2”}, 30.0)) \wedge (\text{“PaintingM”}, 60.0) \\ RR(d) &= (\text{“CNC2”}, 20.0) \wedge (\text{“PaintingM”}, 40.0) \\ RR(e) &= (\text{“Packer”}, 1.0) \end{aligned}$$

[2]Details about the BPMN objects and their semantics can be found in the specification document [17].

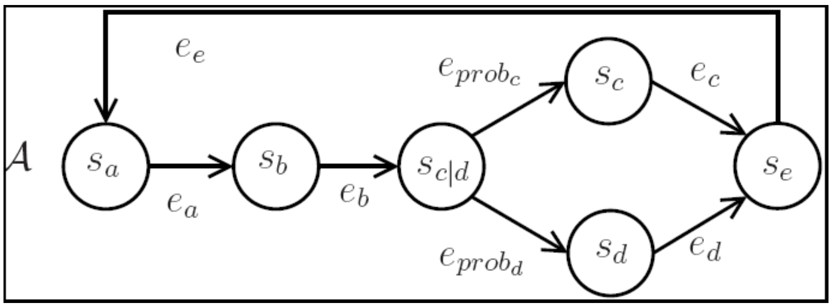


FIGURE 2. SAN model of the control-flow structure modeled in Figure 1

5.1. **The control-flow structure modeled in SAN.** A first SAN model of the process modeled in Figure 1 is illustrated by Figure 2. This SAN model only expresses the control-flow structure of the process. In this model, we have one SAN state and one SAN event associated with each activity and start event of the BPMN model. This SAN state indicates that the BPMN activity (event) is enabled to be executed. The execution of the activity (event) is expressed by a transition of state in the automaton caused by an occurrence of the SAN event that models the BPMN activity (event). This change of state will enable the subsequent activity in the sequence flow.

The divergent exclusive gateway of the BPMN model in Figure 2 was mapped to the SAN state $s_{c|d}$ and its output states $s_c$ and $s_d$. These two last states are connected to state $s_{c|d}$ by transitions associated with the events $e_{prob_c}$ and $e_{prob_d}$, respectively[3].

In the next sections, we explain how to consider the declaration of resources and requirements to improve this first SAN model. The approach can be summarized as follows:

1. enriching the control-flow automata with additional states, events, and transitions to express the resources requirements of the activities;
2. representing parallel instances of the entire process by means of replicated automata;
3. adding additional automata and synchronizing events to represent resources and their interactions with the "control-flow" automata;
4. defining functional rates to express: (i) the enabling of the activities when the resources they depend on are available; (ii) the variation of the activities' execution rates in function of the system workload.

5.2. **Associating resources requirements with activities in the SAN model.** A process model should consider that some activities depend on resources to be executed. For this reason, before enabling the execution of an activity, first we need to allocate the resources it needs. We express this by introducing in the SAN model additional states and events to express: (i) the necessity of waiting for the availability of resources and (ii) the disjunctive sets of resources requirements, that will force us to associate more than one SAN event with the same BPMN activity. For each activity of the BPMN model with resource requirements, we need as many additional states on the SAN model as the number of sets in the disjunctive resources requirements (DRR) of the activity.

As discussed in Section 4.2, the DRR is derived from the DNF of the RR expression of the activity. In the declaration of requirements of the production process of metal workpieces, only $RR(c)$ is in a CNF. The equivalent expression in a DNF is

$$RR(c) = ((\text{“CNC1”}, 60) \land (\text{“PaintingM”}, 60)) \lor ((\text{“CNC2”}, 30) \land (\text{“PaintingM”}, 60))$$

[3]For more details about how BPMN structures can be mapped into SAN, please refer to [3, 4].

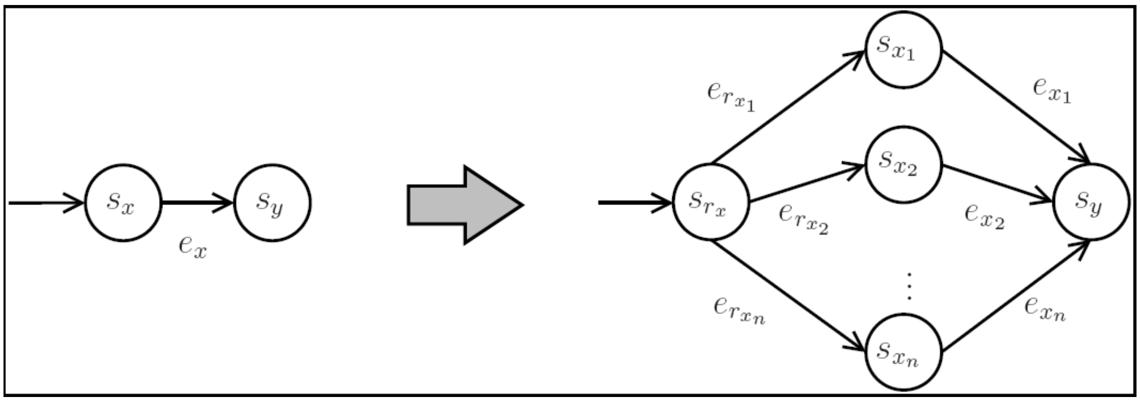


FIGURE 3. Modification that must be made in the SAN modeling of an activity, in order to express its resources requirements

Thus, the DRRs of Example 5.1 are

$$\begin{aligned} DRR(b) &= \{\{(\text{“Designer”}, 0.5), (\text{“Designer”}, 0.5)\}\} \\ DRR(c) &= \{\{(\text{“CNC1”}, 60.0), (\text{“PaintingM”}, 60.0)\}, \\ &= \{(\text{“CNC2”}, 30.0), (\text{“PaintingM”}, 60.0)\}\} \\ DRR(d) &= \{\{(\text{“CNC2”}, 20.0), (\text{“PaintingM”}, 40.0)\}\} \\ DRR(e) &= \{\{(\text{“Packer”}, 1.0)\}\} \end{aligned}$$

Figure 3 shows how the modeling of an activity $x$ must change in order to express its resources requirements. Consider that $x$ has $n$ disjunctive resources requirements (i.e., $DRR(x) = \{\mathcal{S}_{x_1}, \mathcal{S}_{x_2}, \dots, \mathcal{S}_{x_n}\}$ where $\mathcal{S}_{x_i}$, with $1 \leq i \leq n$, is a set of SRRs of $x$). In the leftmost partial SAN model in Figure 3, the activity $x$ was modeled at the same way as the activities were modeled in Figure 2. In the rightmost partial SAN model, we have a state $s_{r_x}$ expressing the process state where $x$ is waiting for the availability of the resources of at least one of its disjunctive requirements sets $\mathcal{S}_{x_i}$ before being enabled to execution. This process will move from $s_{r_x}$ to a state $s_{x_i}$ with the occurrence of an event $e_{r_{x_i}}$ (with $1 \leq i \leq n$). The occurrence of an event $e_{r_{x_i}}$ indicates that the SRRs in $\mathcal{S}_{x_i}$ were all satisfied and then the activity $x$ can be executed.

Using this new mapping for activities in SAN, we have $n$ events ($e_{x_i}$) to represent the execution of the activity, each one with its respective rate. This reflects the fact that the execution rate of the activity is given by the resources it depends on. Since an activity has alternative sets of resources requirements, it will have alternative rates expressed in function of these alternative sets. Modifying the modeling of the activities in Figure 2 according to the example presented in Figure 3, we have the SAN model of Figure 4.

5.3. **Expressing parallel instances with replicated automata.** The model of Figure 4 expresses the behavior of only one instance of the production process. However, new orders for metal workpieces may arrive while another order is being processed and this may cause resource contention. To analyze how the performance is impacted by the workload, we need to consider the system behavior when several instances are being executed in parallel. We can do this by replicating the automata of the SAN model; each replica represents one instance. The number of replicas defines the workload of the system.

Figure 5(a) shows a simple SAN model constituted by $n$ replicas of the same automaton. We will use this model to illustrate the simplified notation that we will adopt from now on to represent replicas of automata and their events. A set of $n$ replicated events as $\{e_x[1], e_x[2], \dots, e_x[n]\}$ will be denoted by $e_x[1..n]$. In an analogous way, a set of $n$

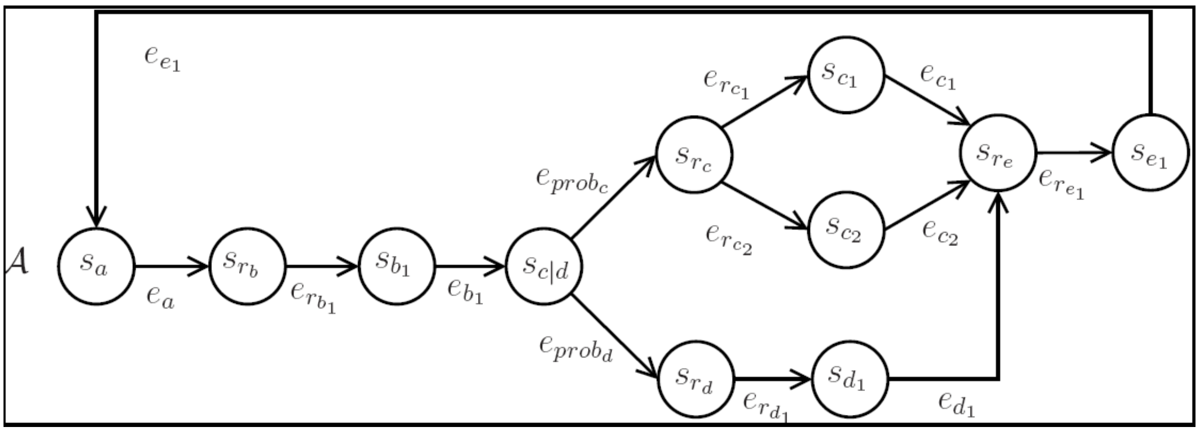
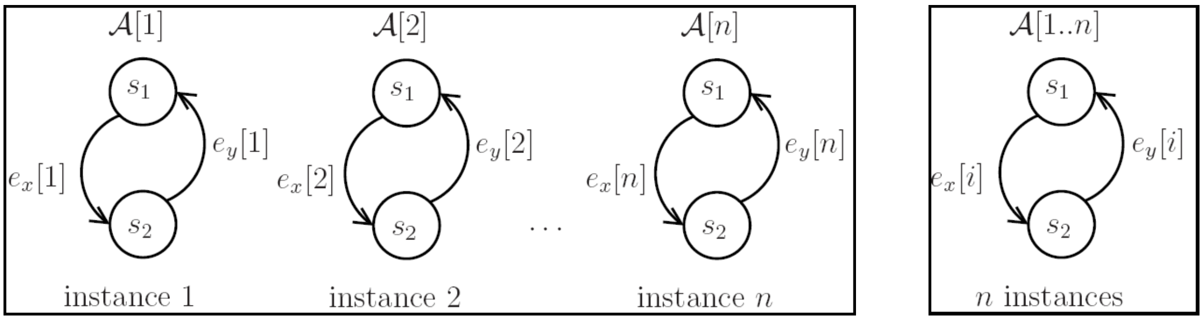


FIGURE 4. SAN model of Figure 2 enriched with additional states and events to express resources requirements



(a) A model in SAN with replicated automata (b) Simplified notation

FIGURE 5. Simplified notation to denote replicated automata in SAN

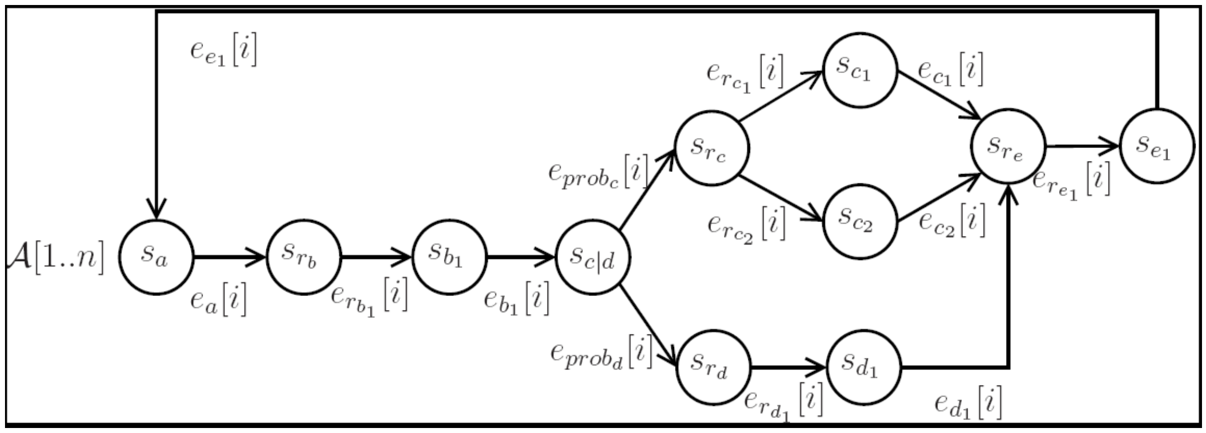


FIGURE 6. SAN model of Figure 4 with $n$ replications

identical automata $\{\mathcal{A}[1], \mathcal{A}[2], \ldots, \mathcal{A}[n]\}$ will be denoted by $\mathcal{A}[1..n]$. Figure 5(b) shows this simplified notation applied to the model of Figure 5(a). In this figure, we have the events $e_x[i]$ and $e_y[i]$, where $i$ indicates the index of the current automaton. Therefore, for the automaton $\mathcal{A}[1]$ we have $e_x[1]$ and $e_y[1]$, for $\mathcal{A}[2]$ we have $e_x[2]$ and $e_y[2]$, and so on.

Figure 6 shows the replicated model of the fabrication process of the machine shop, using the simplified notation to denote the $n$ replicas.

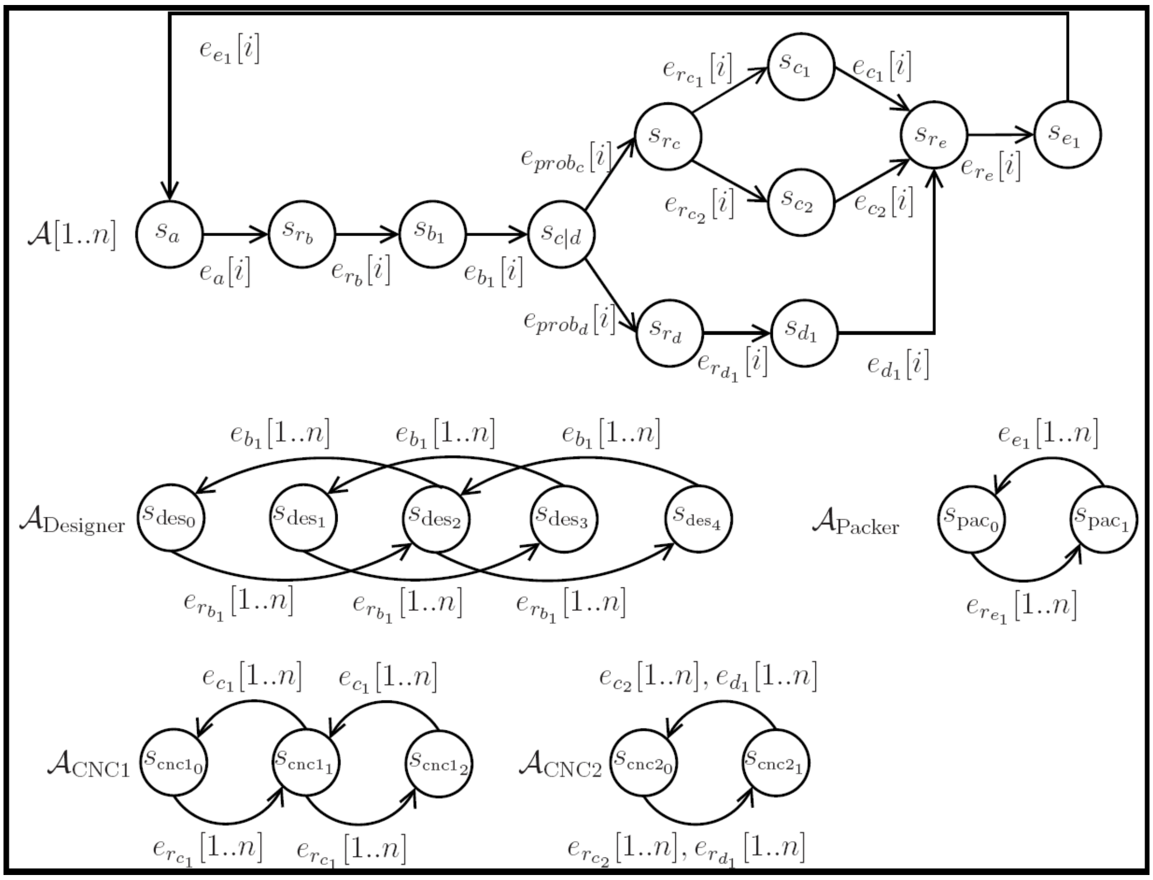

$e_{e_1}[i]$
$\mathcal{A}[1..n]$
$s_a$
$e_a[i]$
$s_{r_b}$
$e_{r_b}[i]$
$s_{b_1}$
$e_{b_1}[i]$
$s_{c|d}$
$e_{prob_c}[i]$
$e_{prob_d}[i]$
$s_{r_c}$
$e_{r_{c_1}}[i]$
$s_{c_1}$
$e_{c_1}[i]$
$e_{r_{c_2}}[i]$
$s_{c_2}$
$e_{c_2}[i]$
$s_{r_e}$
$e_{r_e}[i]$
$s_{e_1}$
$s_{r_d}$
$e_{r_{d_1}}[i]$
$s_{d_1}$
$e_{d_1}[i]$
$\mathcal{A}_{\text{Designer}}$
$s_{\text{des}_0}$ $s_{\text{des}_1}$ $s_{\text{des}_2}$ $s_{\text{des}_3}$ $s_{\text{des}_4}$
$e_{b_1}[1..n]$
$e_{r_{b_1}}[1..n]$
$\mathcal{A}_{\text{Packer}}$
$s_{\text{pac}_0}$ $s_{\text{pac}_1}$
$e_{e_1}[1..n]$
$e_{r_{e_1}}[1..n]$
$\mathcal{A}_{\text{CNC1}}$
$s_{\text{cnc1}_0}$ $s_{\text{cnc1}_1}$ $s_{\text{cnc1}_2}$
$e_{c_1}[1..n]$
$e_{r_{c_1}}[1..n]$
$\mathcal{A}_{\text{CNC2}}$
$s_{\text{cnc2}_0}$ $s_{\text{cnc2}_1}$
$e_{c_2}[1..n], e_{d_1}[1..n]$
$e_{r_{c_2}}[1..n], e_{r_{d_1}}[1..n]$

FIGURE 7. Complete SAN model of the production process

The number of instances in a model defines the analyzed system workload. As this parameter can be easily configured, our suggestion is to solve the SAN model for varied numbers of instances, to evaluate how the performance is impacted by different workloads.

5.4. **Including additional automa to represent resources.** In this work, we consider two types of access disciplines: *random choice* and *time sharing*. Under the random choice access discipline, only one activity can access the resource at a time. When a resource in use is released, one of the activities waiting for the resource will be randomly selected to access it. In practice, generally there is a "non-random" strategy (e.g., FIFO and LIFO) to select an activity among the others that are also waiting for the availability of the resource. However, the number of activities waiting for the resource in a given time will not change in function of the selection strategy used to control the access to it. Under the performance evaluation point of view, the random choice generalizes the access disciplines of resources that must be accessed in a mutually exclusive manner.

A random choice resource can be expressed in the SAN model by a dedicated automaton. Figure 7 shows the model of Figure 6 enriched with the automata $\mathcal{A}_{\text{Designer}}$, $\mathcal{A}_{\text{CNC1}}$, $\mathcal{A}_{\text{CNC2}}$ and $\mathcal{A}_{\text{Packer}}$, correspondent to the random choice resources of the machine shop process.

The automaton of a resource *res* with $m$ units has $m+1$ states ($s_{\text{res}_i}, 0 \leq i \leq m$), each one expressing one of the possible quantities in use of the resource in a given time (i.e., $s_{\text{res}_0}$ corresponds to 0 units of the resource in use, $s_{\text{res}_1}$ corresponds to 1 unit of the resource in use, and so forth).

According to its resources requirements, activity $b$ of the fabrication process of the machine shop (the design of the workpiece's 3D-model) requires two units of the resource "Designer" to be executed. The automaton $\mathcal{A}_{\text{Designer}}$ change from a state $s_{\text{des}_i}$ to the state

$s_{\text{des}_{i+2}}$ (where $0 \leq i \leq 2$) with the occurrence of one event in $e_{r_{b_1}}[1..n]$. As these events also appear in the automata $\mathcal{A}[1\dots n]$, they are *synchronizing events.* So, an automaton $\mathcal{A}[j]$ (where $1 \leq j \leq n$) will only pass from state $s_{r_b}$ to state $s_{b_1}$ when $\mathcal{A}_{\text{Designer}}$ passes from a state $s_{\text{des}_i}$ to state $s_{\text{des}_{i+2}}$ (where $0 \leq i \leq 2$) – indicating that two available resources of type "Designer" were allocated by the activity $b$ of instance $j$ and are now in use.

When the state of $\mathcal{A}_{\text{Designer}}$ is $s_{\text{des}_4}$, all the resources are already in use, and no transitions from state $s_{r_b}$ to state $s_{b_1}$ will be possible while two designers are not released. Two resources are released with the occurrence of one event of the set $e_{b_1}[1..n]$, i.e., when $\mathcal{A}_{\text{Designer}}$ passes from a state $s_{\text{des}_i}$ to state $s_{\text{des}_{i-2}}$ (where $2 \leq i \leq 4$) and an automaton $\mathcal{A}[j]$ (where $1 \leq j \leq n$) passes from state $s_{b_1}$ to state $s_{c|d}$ – indicating that instance $j$ has finished the execution of activity $b$ and has released two designers.

The same synchronized behavior described for $\mathcal{A}_{\text{Designer}}$ also occurs between the instance automata ($\mathcal{A}[1..n]$) and the other resource automata $\mathcal{A}_{\text{cnc1}}$, $\mathcal{A}_{\text{cnc2}}$ and $\mathcal{A}_{\text{Packer}}$. However, in these other resources, only one unit of each is allocated (or released) at a time.

It is important to notice that in $\mathcal{A}_{\text{cnc2}}$, the transition from state $s_{\text{cnc2}_0}$ to state $s_{\text{cnc2}_1}$ occurs with the execution of an event that can be in $e_{r_{c_2}}[1..n]$ or in $e_{r_{d_1}}[1..n]$, to express the fact the `CNC1` resource appears in the requirements of both activities $c$ and $d$. Analogously, the resource can be released by an event in $e_{c_2}[1..n]$ or in $e_{d_1}[1..n]$.

Under the time sharing access discipline, the using time of the resource will be equally divided between all the activities that requested the access to it. Differently from the random choice resources, the time sharing resources cannot be expressed by additional automata in the model. They need to be directly expressed in the rates of the events that represent the activities' execution in the model. Even the random choice resources can be modeled using only functional rates, with no need of additional automata. The next section will show how to use functional rates to model resources.

5.5. **Defining the rates of the SAN model.** The last step involved in the inclusion of the resources/requirements in the SAN model of Example 5.1 is the definition of the rates associated with the events. We will do this by means of a textual declaration of constants and functions, using a syntax similar to the one used by the PEPS tool.

5.5.1. *Declaration of the constants.* We will define some constants that work as the configuration parameters of our SAN model and that support the definition of the event rates. They are extracted from the business process model and from its declaration of resources and requirements. Their values can be conveniently changed, in order to provide varied analyses without the necessity of changes in the structure of the model.

For each resource of the business process, we define a constant indicating its number of units and other indicating its work capacity (in units of work per unit of time):

```
qty_Designer   =  4;    workCapacity_Designer   =  0.125;
qty_CNC1       =  2;    workCapacity_CNC1       =  6.0;
qty_CNC2       =  1;    workCapacity_CNC2       =  2.0;
qty_PaintingM  =  1;    workCapacity_PaintingM  =  10.0;
qty_Packer     =  1;    workCapacity_Packer     =  1.0;
```

For each activity in the model, we define a constant indicating the quantity of work that the activity requires of each one of the resources in its requirements set:

```
requiredWork_B1_srr1_Designer  =  0.5;   requiredWork_B1_srr2_Designer   =  0.5;
requiredWork_C1_srr1_CNC1      =  60.0;  requiredWork_C1_srr2_PaintingM  =  60.0;
requiredWork_C2_srr1_CNC2      =  30.0;  requiredWork_C2_srr2_PaintingM  =  60.0;
requiredWork_D1_srr1_CNC2      =  20.0;  requiredWork_D1_srr2_PaintingM  =  40.0;
requiredWork_E1_srr1_Packer    =  1.0;
```

For each divergent decision gateway in the model, we define the probabilities associated with its branches: $\boxed{\texttt{prob\_C} = 0.8; \quad \texttt{prob\_D} = 0.2;}$.

Finally, we need to define a constant rate – the `ins_rate` – to express the (small) execution time associated with the events artificially inserted in the model to express decision routings, such as the events $e_{prob_c}[1..n]$ and $e_{prob_d}[1..n]$, or allocation of resources, such as $e_{r_{b_1}}[1..n], e_{r_{c_1}}[1..n]$. Since we cannot have instantaneous transitions in SAN, the value for this rate should be any arbitrary value higher than the other rates in the model, because it should not greatly impact the performance indices extracted from the SAN model. In this example, we arbitrary set the value to 50.0: $\boxed{\texttt{ins\_rate} = 50.0;}$.

It is possible to divide the events we created in our SAN model in four groups:

1. events to indicate the occurrence of one start event of the BPMN model;
2. events to express a probabilistic routing in the BPMN model;
3. events to indicate that the resources requirements of an activity were satisfied;
4. events to express the execution of one activity of the BPMN model.

The rate associated with the events of the first group is a constant value and cannot be automatically inferred from the data provided in the description of the process (BPMN model + resources/requirements declaration). A specialist of the domain can define an appropriate value, or (as can occur with the number of process instances in the model) it should be varied in the analyses, in order to express different system workloads. In the SAN model of our example, the events of this type are the events in the set $e_a[1..n]$.

For the events of the second group, we need to define rates able to express the probability of execution associated with the routing decision that they correspond to. In the SAN model of our example, we have two sets of events of this type: $e_{prob_c}[1..n]$ and $e_{prob_d}[1..n]$.

In a SAN model, a *race condition* governs the dynamic behavior of the model when it will evolve from a state with more than one enabled output transition. In this case, the "faster" the output transition is (i.e., the greater the transition rate), more frequently it will win the race (i.e., it will occurs). For this reason, we can use pondered rates to express probabilistic routings. For each branch of a decision, we define a constant rate (to express the routing cost) multiplied by the probability of the branch:

$$\boxed{\texttt{rate\_prob\_C} = \texttt{ins\_rate} \times \texttt{prob\_C}; \qquad \texttt{rate\_prob\_D} = \texttt{ins\_rate} \times \texttt{prob\_D};}$$

As mentioned in Section 3.1, SAN has the concept of *functional rates* – event rates given in function of the current state of the system. We will use these powerful functions to define the rates of the events in the groups 3 and 4.

5.5.2. *Declaration of the auxiliary functions.* Definition 5.1 describes an important function that exists in the implementation of SAN made in the PEPS tool.

**Definition 5.1.** *Let nb([automata set], [state]) be a function that returns the number of automata in [automata set] whose current state is [state].*

Using the function *nb*, we define a function to give the number of units currently in use for each resource of the model:

$$\boxed{\begin{array}{lcl}
\texttt{f\_usedQty\_Designer} & = & 2 \times nb(\mathcal{A}[1..n], s_{b_1}); \\
\texttt{f\_usedQty\_CNC1} & = & nb(\mathcal{A}[1..n], s_{c_1}); \\
\texttt{f\_usedQty\_CNC2} & = & nb(\mathcal{A}[1..n], s_{c_2}) + nb(\mathcal{A}[1..n], s_{d_1}); \\
\texttt{f\_usedQty\_PaintingM} & = & nb(\mathcal{A}[1..n], s_{c_1}) + nb(\mathcal{A}[1..n], s_{c_2}) + nb(\mathcal{A}[1..n], s_{d_1}); \\
\texttt{f\_usedQty\_Packer} & = & nb(\mathcal{A}[1..n], s_{e_1});
\end{array}}$$

From the discussions made in Sections 5.3 and 5.4, we know that when an automaton in $\mathcal{A}[1..n]$ is in the state $s_{b_1}$, for example, the set of resources required by activity $b$ to execute was already allocated for its use (in this case, two designers). For this reason, the total number of designers currently in use (`f_usedQty_Designer`) is given by the number of instances in the state $s_{b_1}$ ($nb(\mathcal{A}[1..n], s_{b_1})$) multiplied by two. For the other resources, the reasoning is analogous. It is important to notice that the resources `CNC2` and Painting Machine are required for more than one activity. For this reason, their number of units currently in use is given by a sum of counts ($nb$) over different states.

Now, we will define boolean functions to return the availability of the resources. For each resource, we will create a function for each different quantity of the resource that an activity of the model may require. In the case of resources whose access discipline is random choice, these functions will return *true* (1) when the indicated quantity of the resource is available for use, and *false* (0) in the other case[4]. In the case of a time sharing resource, we are only interested in knowing if there exists the resource in the quantity required by the activities of the model (because it will never be "busy" to an activity).

```
* f_isAvailable_Designer_2 = ((qty_Designer - f_usedQty_Designer) ≥ 2);
* f_isAvailable_CNC1_1     = ((qty_CNC1 - f_usedQty_CNC1) ≥ 1);
* f_isAvailable_CNC2_1     = ((qty_CNC2 - f_usedQty_CNC2) ≥ 1);
* f_isAvailable_Packer_1   = ((qty_Packer - f_usedQty_Packer) ≥ 1);
  f_exists_PaintingM_1     = (qty_PaintingM ≥ 1);
```

Using the functions of availability, we define functions to indicate when an activity is enabled to be executed according to the availability of the resources it requires.

```
 * f_isEnabled_r_B1 = f_isAvailable_Designer_2;
 * f_isEnabled_r_C1 = (f_isAvailable_CNC1_1 AND f_exists_PaintingM_1);
** f_isEnabled_r_C1 = f_exists_PaintingM_1;
 * f_isEnabled_r_C2 = (f_isAvailable_CNC2_1 AND f_exists_PaintingM_1);
** f_isEnabled_r_C2 = f_exists_PaintingM_1;
 * f_isEnabled_r_D1 = (f_isAvailable_CNC2_1 AND f_exists_PaintingM_1);
** f_isEnabled_r_D1 = f_exists_PaintingM_1;
 * f_isEnabled_r_E1 = f_isAvailable_Packer_1;
```

The time sharing resources are expressed in the model by means of functional rates. However, the resources whose access discipline is random choice can be expressed in the SAN model in two manners: (i) using functional rates only, and (ii) using additional automata (as made in Figure 7) combined with functional rates. When we use additional automata to model random choice resources, the allocation and the liberation of a resource for an activity is controlled by synchronizing events.

For this reason, in the declarations above, the functions marked with * are only required for a SAN model that does not include additional automata to represent the random choice resources. Contrarily, the functions marked with ** are required for a SAN model that has additional automata to represent the random choice resources (since these functions express only the dependency for time sharing resources). In the rest of this section, we will continue using the mark * to denote a function that is required in a model without additional automata, and ** to denote a function that is required in a model with additional automata for resources.

For each time sharing resource, we define a function that returns the work capacity of the resource under the point of view of an activity that is using it, i.e., its nominal capacity divided by the number of activities currently sharing the resource:

```
f_sharedWorkCapacity_PaintingM = workCapacity_PaintingM / f_usedQty_PaintingM;
```

[4]A function in SAN will always return a numerical value. If the return value of the function is the evaluation of a boolean expression, the return value will be 1 to indicate *true*, and 0 to indicate *false*.

In the following, we define the processing rates of the resources for each activity single requirement in that they appear in the business process. This rate is the inverse of the time required of the resource to process the work demanded by the activity.

```
f_rate_B1_srr1_Designer    =  workCapacity_Designer / requiredWork_B1_srr1_Designer;
f_rate_B1_srr2_Designer    =  workCapacity_Designer / requiredWork_B1_srr2_Designer;
f_rate_C1_srr1_CNC1        =  workCapacity_CNC1 / requiredWork_C1_srr1_CNC1;
f_rate_C1_srr2_PaintingM   =  f_sharedWorkCapacity_PaintingM /
                                   requiredWork_C1_srr2_PaintingM;
f_rate_C2_srr1_CNC2        =  workCapacity_CNC2 / requiredWork_C2_srr1_CNC2;
f_rate_C2_srr2_PaintingM   =  f_sharedWorkCapacity_PaintingM /
                                   requiredWork_C2_srr2_PaintingM;
f_rate_D1_srr1_CNC2        =  workCapacity_CNC2 / requiredWork_D1_srr1_CNC2;
f_rate_D1_srr2_PaintingM   =  f_sharedWorkCapacity_PaintingM /
                                   requiredWork_D1_srr2_PaintingM;
f_rate_E1_srr1_Packer      =  workCapacity_Packer / requiredWork_E1_srr1_Packer;
```

5.5.3. *Declaration of the functional rates.* The events of group 3 (that indicate that the resources requirements of an activity were satisfied) are associated with the following functional rates:

```
*  f_rate_r_B1  =  f_isEnabled_r_B1 × ins_rate;     **  f_rate_r_B1  =  ins_rate;
*  f_rate_r_C1  =  f_isEnabled_r_C_1 × ins_rate;    **  f_rate_r_C1  =  ins_rate;
*  f_rate_r_C2  =  f_isEnabled_r_C_2 × ins_rate;    **  f_rate_r_C2  =  ins_rate;
*  f_rate_r_D1  =  f_isEnabled_r_D1 × ins_rate;     **  f_rate_r_D1  =  ins_rate;
*  f_rate_r_E1  =  f_isEnabled_r_E1 × ins_rate;     **  f_rate_r_E1  =  ins_rate;
```

When `f_isEnabled_r_B` returns 0 (indicating that the required resources for activity $b$ are not available), the return value of `f_rate_r_B1` will be also 0, indicating that the associated event $e_{r_{b_1}}$ (of allocation of the resources for activity $b$) cannot occur in the current state of the system. In the other case, the rate of $e_{r_{b_1}}$ will be greater than 0, indicating that the allocation of the resources for $b$ can occur at the current state of the system. This same interpretation is valid for the other functions in the block above.

The functional rates associated with the events of group 4, that represent the execution of the activities of the business process, are given in the sequence. In the case of the activities that depend on only one resource, the execution rate will be determined by the time demanded by this resource to perform the required work. For activities that depend on more than one resource, the execution rate will be determined by the slower resource (i.e., the minimum rate among the rates of the resources that the activity depends on).

```
f_rate_B1  =  min(f_rate_B1_srr1_Designer,f_rate_B1_srr2_Designer);
f_rate_C1  =  min(f_rate_C1_srr1_CNC1,f_rate_C1_srr2_PaintingM);
f_rate_C2  =  min(f_rate_C2_srr1_CNC2,f_rate_C2_srr2_PaintingM);
f_rate_D1  =  min(f_rate_D1_srr1_CNC2,f_rate_D1_srr2_PaintingM);
f_rate_E1  =  f_rate_E1_srr1_Packer;
```

Table 1 shows the (functional) rates associated with the events in the SAN model of Figure 7. Remember that we cannot automatically define a rate for the start events ($e_a[1..n]$), since it is not given by the resource management of the business process. The same will occur with an activity that has no resource requirements.

## 6. Implementation of the Method and Extraction of Performance Indices.

Our method to specify resource management in business process models and its mapping to SAN models was implemented as part of a software tool, the BP2SAN[5]. BP2SAN is able to convert a subclass of the BPMN process diagrams to SAN models that reflect the

[5]BP2SAN is publicly available at http://www.ime.usp.br/~kellyrb/bp2san.

Table 1. The events of the SAN model of Figure 7 and their respective rates

| **Event(s)** | **Rate** | **Function in the BP model** |
|---|---|---|
| $e_{r_{b_1}}[1..n]$ | `f_rate_r_B1` | allocation of resources |
| $e_{b_1}[1..n]$ | `f_rate_B1` | execution of activity |
| $e_{prob_c}[1..n]$ | `rate_prob_C` | probabilistic routing |
| $e_{prob_d}[1..n]$ | `rate_prob_D` | probabilistic routing |
| $e_{r_{c_1}}[1..n]$ | `f_rate_r_C1` | allocation of resources |
| $e_{c_1}[1..n]$ | `f_rate_C1` | execution of activity |
| $e_{r_{c_2}}[1..n]$ | `f_rate_r_C2` | allocation of resources |
| $e_{c_2}[1..n]$ | `f_rate_C2` | execution of activity |
| $e_{r_{d_1}}[1..n]$ | `f_rate_r_D1` | allocation of resources |
| $e_{d_1}[1..n]$ | `f_rate_D1` | execution of activity |
| $e_{r_{e_1}}[1..n]$ | `f_rate_r_E1` | allocation of resources |
| $e_{e_1}[1..n]$ | `f_rate_E1` | execution of activity |

modeled control-flow of the business processes. In addition, when annotations concerning the resources requirements of the BPMN model are provided to the tool, the automatically generated SAN models will also include the resource management policy of the process. The resulted SAN models are textually expressed using the syntax accepted by PEPS [6].

The next section provides examples of performance indices that can be obtained from a SAN model generated by BP2SAN with the support of the numerical solver PEPS.

6.1. **Extracting performance indices from the generated SAN model.** We can extract performance indices from a SAN model by defining numerical functions over the state space of the system and integrating them with the stationary probability distribution of the model. For example, the throughput of resource CNC2 of Example 5.1 is workCapacity_CNC2 $\times \sum_{i \in S} \pi_i$, where $S$ is the set of all global states of the SAN model of Figure 7 in which the resource CNC2 is in use (i.e., in which f_qtyInUse_CNC2 $> 0$), and $\pi_i$ is the probability of the state $i$ in the stationary distribution of the model.

In PEPS, we can specify the functions that define the performance indices together with the SAN model. After solving the model, the tool returns as result the integrated values of these functions. From now on, we will illustrate some functions to extract performance indices as they can be defined in the PEPS tool. Some of the functions that we will use were previously defined in Section 5.5.

6.1.1. *Average number of resource units in use.* Integrating f_usedQty_Designer over the steady state probabilities, for example, we have the *average number* of occupied designers in the machine shop in the steady state of the system.

6.1.2. *Resource utilization rate.* We will define a function that returns *true* (1) when the resource CNC2 is in use:

$$\texttt{f\_isInUse\_CNC2} = \texttt{f\_usedQty\_CNC2} == 1\ .$$

The integration of f_isInUse_CNC2 over the steady state probabilities gives us the *utilization rate* of CNC2 (i.e., the percentage of the time in which the resource is in use).

For resources with more than one unit, we can define an utilization rate for each number of units that can be simultaneously in use. For example, consider the following function:

$$\texttt{f\_isInUse\_Designer\_4} = \texttt{f\_usedQty\_Designer} == 4\ .$$

The integration of f_isInUse_Designer_4 over the steady state probabilities gives us the percentage of the time in which the four designers are simultaneously busy.

6.1.3. *Throughput of a resource.* Let define a function that returns the rate (work capacity) of the resource `CNC1` multiplied by its number of units currently in use:

$$\texttt{f\_usefulRate\_CNC1} = \texttt{f\_usedQty\_CNC1} \times \texttt{workCapacity\_CNC1} \ .$$

The integration of `f_usefulRate_CNC1` over the steady state probabilities gives us the *throughput* of `CNC1` (i.e., the effective number of small metal volumes processed by the `CNC1` machines together per hour).

6.1.4. *Throughput of an activity.* To compute the throughput of an activity, we can define an auxiliary function that returns the rate of $b$ multiplied by the number of instances of activity $b$ currently enabled to execution:

$$\texttt{f\_usefulRate\_B} = nb(\mathcal{A}[1..n], s_{b_1}) \times \ \texttt{f\_rate\_B1} \ .$$

Integrating `f_usefulRate_B` over the steady state probabilities, we obtain the *throughput* of $b$ (i.e., the number of workpieces designed in the machine shop per hour).

6.1.5. *Average "Queue" size.* To have the average number of workpiece orders waiting for the availability of designers, we can integrate the function $nb(A[1..n], s_{r_b})$ over the steady state probabilities. For each state of the model, this function gives the number of automata in state $s_{r_b}$. The state $s_{r_b}$ indicates that the process instance is waiting for the availability of the resources required to the execution of activity $b$.

6.1.6. *Service time.* The service time of the business process is the average time required for the complete execution of an instance. To calculate it, we need the probability of the initial state of an instance automaton, that is given by the following function:

$$\texttt{f\_probInitialState} = nb(\{\mathcal{A}[1]\}, s_a) > 0 \ .$$

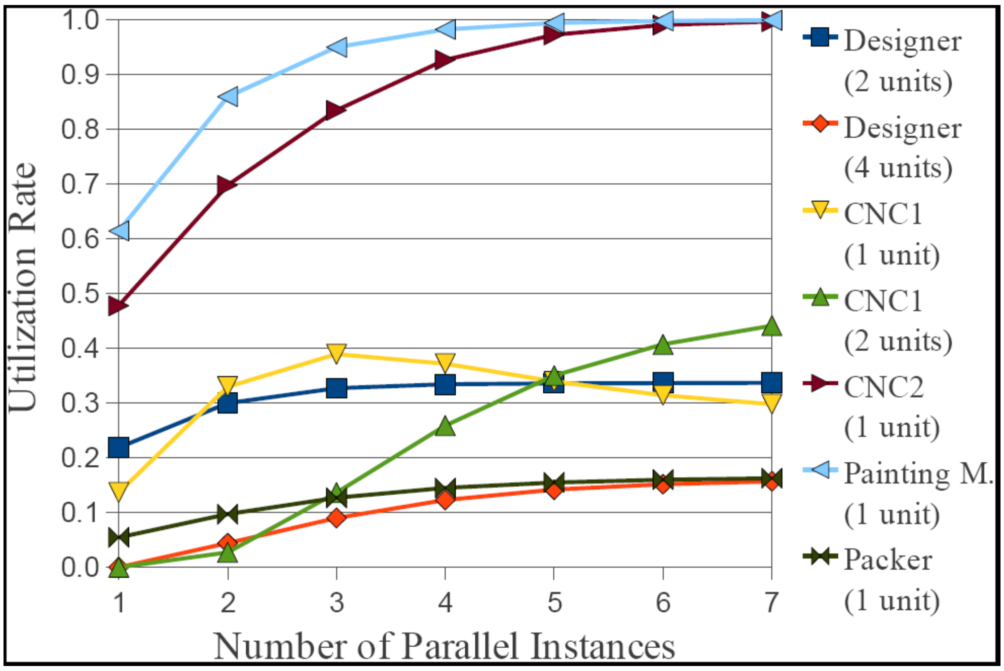
From the integration of `f_probInitialState` over the steady state probabilities we obtain the probability $\pi_{s_a}$ of an instance to be in the initial state $s_a$.

Knowing that $\pi_{s_a} = \frac{\text{time between order arrivals}}{\text{time between order arrivals + service time}}$ and that the time between order arrivals is $\frac{1}{\text{rate of } e_a}$, the service time is given by the formula: $\frac{1-\pi_{s_a}}{\text{rate of } e_a \times \pi_{s_a}}$.

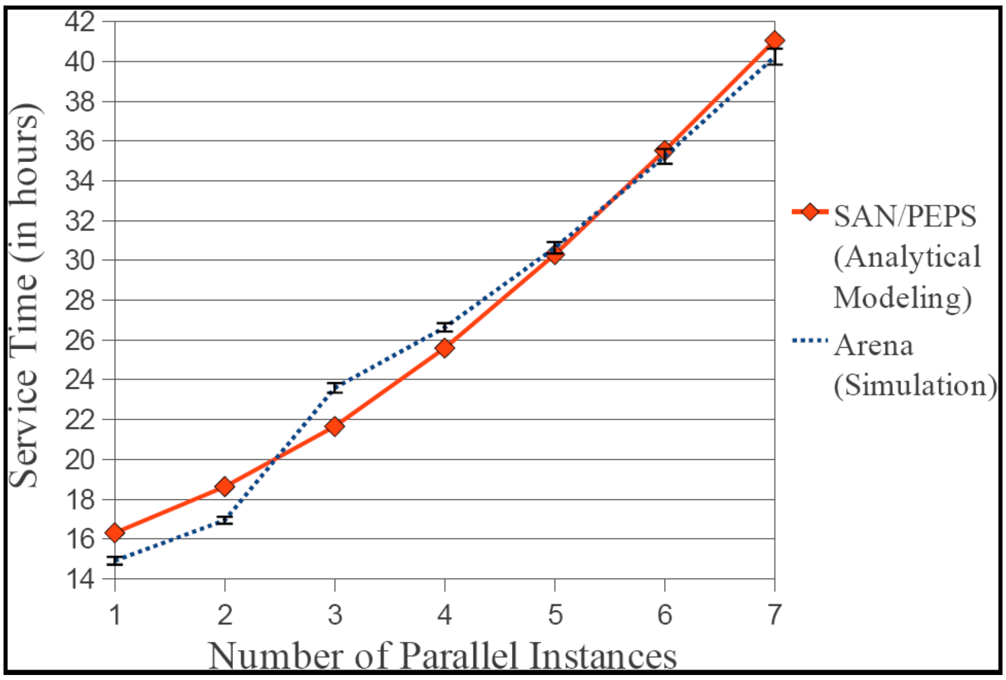
6.2. **Some performance indices for the production process of the machine shop.** Using `BP2SAN`, we generated a SAN model from the annotated BPMN model of Figure 1. Then, using PEPS, we extracted some performance indices for the business process. We obtained the utilization rate of the resources and the service time of the business process. We set the arrival rate to 0.5 orders by hour, i.e., each two hours (in average) a new order arrives for the machine shop. We analyzed the model for a number of parallel instances varying from one up to seven, in order to express different workloads.

The chart in Figure 8(a) shows the variation of the utilization rate of the resources with the number of parallel instances. For `CNC1` and Designer, we have a different utilization rate for each possible number of units of the resource that can be simultaneously used during the system execution. In this chart, we can observe that the CNC machines and the painting machine become the most overloaded resources as the workload increases. With six parallel instances, the resources `CNC2` and Painting Machine almost reach the saturation level, while the designers and the packer are idle more than half of the time. These results suggest that the machine shop should consider the acquisition of new machines to improve the utilization of its human resources, or reduce the number of employed designers.

The filled line in the chart of Figure 8(b) shows how the service time of the business process is impacted by the number of orders being parallelly treated in the machine shop (according to results obtained from the SAN model). From the chart, we can observe



(a) Resource utilization



(b) Service time

FIGURE 8. Variation of performance indices with workload in the fabrication process of the machine shop

that the time to process one order at a time is approximately 16 hours. This service time exceeds 40 hours when the number of orders being parallelly treated is seven.

Table 2 shows the state space size of the SAN model of Figure 7, for each number of parallel instances analyzed in our experiments. The table also shows the order of magnitude of the computation time required by the PEPS tool to numerically solve each model in an Intel® Xeon® machine with 2.6 GHz and 32 GB of RAM. It is important to notice that, with seven parallel instances, the model exceeds five million reachable

Table 2. Size of the state space of the SAN model and the order of magnitude of the computation time required to solve the model using PEPS

| Parallel Instances | Size of Product State Space | Size of Reachable State Space | Computation Time of the Model's Solution |
|---|---|---|---|
| 1 | 11 | 11 | $\approx 0$ seconds |
| 2 | 121 | 111 | $10^{-2}$ seconds |
| 3 | 1,331 | 1,056 | $10^{-1}$ seconds |
| 4 | 14,641 | 9,612 | $10^{1}$ seconds |
| 5 | 161,051 | 84,456 | $10^{2}$ seconds |
| 6 | 1,771,561 | 720,576 | $10^{3}$ seconds |
| 7 | 19,487,171 | 5,995,296 | $10^{4}$ seconds |

states. Such models with a huge state space are generally considered intractable for other analytical modeling techniques.

6.2.1. *Validation of the obtained results.* In order to validate the results obtained through our analysis framework, we compared these results with the ones obtained from other well-established analysis method for business processes: the simulation.

There are several simulation software tools specifically created to the BPM domain (Jansen-vullers and Netjes [15] provided a survey of these tools). Most of these tools do not support the definition of accurate models for performance evaluation. They only enable us to associate average execution times with activities, and probabilities with branches of decisions. Nevertheless, this kind of model does not enable us to capture the performance degradation caused by resource contention.

There are several general-purpose, discrete-event simulation tools that have more expressive modeling languages than the ones found in BPM tools. But specifying complex business process models using these tools demands a lot of time and deep knowledge of the functioning of the simulator. The control-flow structures of the business process models must be denoted in terms of the commands available in the simulator. On the one hand, some tools provide facilities to model resources accessed in a mutually exclusive manner. On the other hand, these facilities do not account time sharing resources, which are hard to be implemented in simulations.

We designed and analyzed the production process of the machine shop using a popular general-purpose simulator called Arena (Version 13.9) [1]. For each number of parallel process instances between one and seven, we performed 30 simulations of the model (with the same parameter values used in the SAN model), each one simulating the work performed during a period of one year.

The dashed line in the chart of Figure 8(b) shows the average service time obtained from these experiments. Each measurement is presented with a 95% confidence interval. In the chart, we can observe that the results obtained through the two analysis methods were satisfactorily similar. The differences between the lines are due to the absence of support for the modeling of time sharing resources in Arena.

The accuracy of the obtained measures is an important issue in the comparison between simulation and analytical modeling, independently of the application domain. If we want to estimate indices given by mean values, a simulation may give satisfactory results. However, when we need to estimate the probabilities of occurrence of rare events (i.e., very small probabilities), simulation is not the best approach, since the number of experiments required to give estimated probabilities in an acceptable confidence interval may be prohibitive. In both cases, the analytical modeling gives us more accurate results.

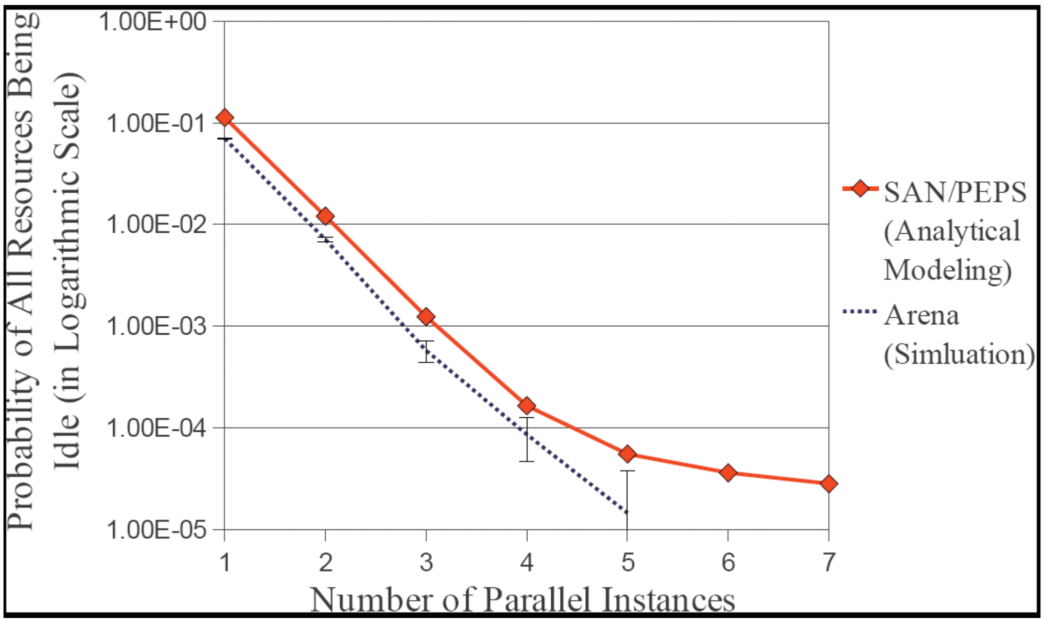


Figure 9. Variation of the probability of all resources being idle with workload in the fabrication process of the machine shop

To illustrate the accuracy problem above-mentioned, we have also computed the probability of all resources being simultaneously idle in the machine shop process. The chart in Figure 9 shows the probabilities obtained through analytical modeling with SAN/PEPS and through simulation with Arena. In the chart, we can observe that the accuracy of the probabilities obtained through simulation decreases with the increasing of the workload (for a fixed number of simulations). Furthermore, we could not compute with Arena the probability of the rare event for six and seven parallel instances. In these two cases, the simulator returned zero for the probability.

## 7. Conclusions.

In this work, we defined a new framework for the generation of resource-aware, performance evaluation models from business process models annotated with resource management information. This framework is composed of a new notation for the specification of the resources and how they are used by a business process and a method to generate SAN models from business processes modeled using BPMN and this notation. Supported by a use case, we introduced our method and illustrated its results.

The SAN features contributed to the simplicity of this approach. Using SAN, we were able to model resources and requirements in a straightforward way. Parallel instances of processes can be modeled as automata replicas. Functional rates model the relationships between the execution rates of the activities and their resources requirements. The SAN models generated through our method reflect how the process' performance is affected by resource contention as the workload of the system increases.

Business specialists do not require skills in stochastic modeling to specify the resource management of a business process using the proposed notation. The specification of the resource management is made in a high abstraction level. All the remaining stochastic framework required to model the randomness and the variability of the business processes and their resources requirements is automatically inferred by our method.

Our method was implemented as part of `BP2SAN`, a software tool that automatically converts annotated BPMN process diagrams into SAN models. Using a numerical solver for SAN, such as PEPS, the business specialists can predict varied performance indices directly from the generated SAN models. The parameters that describe the quantifiable behavior of the system can be easily adjusted in order to express different system workloads or resource capacities. Analyzing how the performance indices are impacted by varied parameter values, we can identify unsuspected dependencies that may exist between the activities and make better resource provisions for the business processes.

**Acknowledgment.** This work was supported by the *São Paulo Research Foundation* (FAPESP) and the *Brazilian Federal Agency for the Support and Evaluation of Graduate Education* (CAPES).

## REFERENCES

[1] T. Altiok and B. Melamed, *Simulation Modeling and Analysis with Arena*, Academic Press, 2007.
[2] G. Balbo, Introduction to generalized stochastic Petri nets, *Proc. of the 7th Int. Conf. on Formal Methods for Performance Evaluation*, Bertinoro, Italy, pp.83-131, 2007.
[3] K. R. Braghetto et al., Performance analysis modeling applied to business processes, *Proc. of 2010 Spring Simulation Multiconference*, Orlando, FL, USA, pp.122:1-122:8, 2010.
[4] K. R. Braghetto et al., Performance evaluation of business processes through a formal transformation to SAN, *Proc. of the 8th European Performance Engineering Wksp*, Borrowdale, UK, pp.42-56, 2011.
[5] L. Brenner et al., The need for and the advantages of generalized tensor algebra for Kronecker structured representations, *Int. J. Simul. Syst. Sci. Technol.*, vol.6, no.3-4, pp.52-60, 2005.
[6] L. Brenner et al., PEPS2007 – Stochastic automata networks software tool, *Proc. of the 4th Int. Conf. on the Quantitative Evaluation of Systems*, Edinburgh, UK, pp.163-164, 2007.
[7] C. Canevet et al., Performance modelling with the unified modelling language and stochastic process algebras, *IEE Proc. of Comput. Digit. Tech.*, vol.150, no.2, pp.107-120, 2003.
[8] A. Clark et al., Stochastic process algebras, *Proc. of the 7th Int. Conf. on Formal Methods for Performance Evaluation*, Bertinoro, Italy, pp.132-179, 2007.
[9] R. M. Dijkman et al., Semantics and analysis of business process models in BPMN, *Inf. Softw. Technol.*, vol.50, no.12, pp.1281-1294, 2008.
[10] H. Eshuis, *Semantics and Verification of UML Activity Diagrams for Workflow Modelling*, Ph.D. Thesis, University of Twente, 2002.
[11] G. Florin et al., Stochastic Petri nets: Properties, applications and tools, *Microelectron. Reliab.*, vol.31, no.4, pp.669-697, 1991.
[12] P. J. Fortier and H. E. Michel, *Computer Systems Performance Evaluation and Prediction*, Digital Press, Newton, MA, USA, 2003.
[13] J. Hillston, *A Compositional Approach to Performance Modelling*, Cambridge University Press, New York, NY, USA, 1996.
[14] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*, Wiley, New York, NY, USA, 1991.
[15] M. H. Jansen-Vullers and M. Netjes, Business process simulation – A tool survey, *Proc. of the 7th Wksp and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*, Aarhus, Denmark, pp.80-96, 2006.
[16] E. D. Lazowska et al., *Quantitative System Performance: Computer System Analysis Using Queueing Network Models*, Prentice-Hall, Upper Saddle River, NJ, USA, 1984.
[17] OMG, *Business Process Model and Notation (BPMN)*, Version 2.0, 2010.
[18] B. Plateau, On the stochastic structure of parallelism and synchronization models for distributed algorithms, *SIGMETRICS Perform. Eval. Rev.*, vol.13, no.2, pp.147-154, 1985.
[19] B. Plateau and K. Atif, Stochastic automata network for modeling parallel systems, *IEEE Trans. Software Eng.*, vol.17, no.10, pp.1093-1108, 1991.
[20] D. Prandi et al., Formal analysis of BPMN via a translation into COWS, *Proc. of the 10th Int. Conf. on Coordination Models and Languages*, Oslo, Norway, pp.249-263, 2008.
[21] C. H. Sauer and K. M. Chandy, *Computer Systems Performance Modeling*, Prentice Hall, Englewood Cliffs, NJ, USA, 1981.
[22] W. M. P. van der Aalst, Formalization and verification of event-driven process chains, *Inf. Softw. Technol.*, vol.41, no.10, pp.639-650, 1999.