HAL
open science

Comparison of Modeling Approaches to Business Process Performance Evaluation

Kelly Rosa Braghetto, João Eduardo Ferreira, Jean-Marc Vincent

To cite this version:

Kelly Rosa Braghetto, João Eduardo Ferreira, Jean-Marc Vincent. Comparison of Modeling Approaches to Business Process Performance Evaluation. [Research Report] RR-7065, INRIA. 2009. inria-00424452

HAL Id: inria-00424452

https://inria.hal.science/inria-00424452v1

Submitted on 15 Oct 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Comparison of Modeling Approaches to Business Process Performance Evaluation

Kelly Rosa Braghetto — João Eduardo Ferreira — Jean-Marc Vincent

N° 7065

October 2009

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--7065--FR+ENG

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

INRIA

centre de recherche GRENOBLE - RHÔNE-ALPES

Comparison of Modeling Approaches to Business Process Performance Evaluation

Kelly Rosa Braghetto * † , João Eduardo Ferreira * ,
Jean-Marc Vincent ‡

Thème : Calcule distribué et applications à très haute performance
Équipe-Projet MESCAL

Rapport de recherche n° 7065 — October 2009 — 23 pages

Abstract: In order to improve efficiency in organizations, it is important to understand how organizational processes work and how they could be optimized. Our goal is to contribute with the Business Process Management (BPM) research area providing a summary of the advantages and disadvantages of applying three well-known high-level formalisms in the modeling and performance evaluation of business processes – the *Generalized Stochastic Petri Nets*, the *Performance Evaluation Process Algebra* and the *Stochastic Automata Networks*. To evaluate the feasibility of the approaches, we consider criteria regarding the modeling perspective, such as the expressive power, the facility of modeling, the readability of the models, and the efficacy of their supporting software tools. Different scenarios are used to illustrate the modeling characteristics commonly found in business processes and evidence the advantages and disadvantages of each approach.

Key-words: generalized stochastic Petri nets, stochastic process algebra, stochastic automata networks, performance evaluation, business process modeling

* Department of Computer Science, University of São Paulo
† Kelly Rosa Braghetto is supported by the brazilian government (CAPES and FAPESP)
‡ Joseph Fourier University (Grenoble) and Laboratory of Informatics of Grenoble

Centre de recherche INRIA Grenoble – Rhône-Alpes
655, avenue de l'Europe, 38334 Montbonnot Saint Ismier
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

Comparaison de méthodes de modélisation de "Business Processes" afin d'évaluer leurs performances

Résumé : Afin d'améliorer l'efficacité des processus de traitement d'information, il est fondamental de comprendre les mécanismes sous-jacents de ces processus puis de développer les méthodes pour les optimiser. Dans ce domaine de recherche ("Business Process Management"), l'objectif de ce rapport est de présenter une comparaison de trois approches de modélisation de haut-niveau permettant d'abord de modéliser puis d'évaluer les performances de processus de traitement: les réseaux de Petri stochastiques, les algèbres de processus et les réseaux d'automates stochastiques. L'évaluation de ces formalismes est faite, pour la partie modélisation, selon des critères d'expressivité, de facilité de modélisation et lecture de modèle et pour la partie analyse selon l'efficacité des environnement logiciels d'évaluation. Différents scénarios, classiques dans le domaine du "Business Process", sont étudiés et mettent en avant les avantages/désavantages des trois approches.

Mots-clés : réseaux de Petri stochastiques, algèbre de processus stochastique, réseaux d'automates stochastiques, évaluation de performance, modélisation des processus de traitement

1 Introduction

The *Business Process Management* (BPM) researches have been an important issue in improving efficiency in organizations. BPM can be defined as "supporting business processes using methods, techniques, and software to design, enact, control, and analyze operational processes involving humans, organizations, applications, documents and other sources of information" [Aalst et al., 2003]. BPM helps us to understand how organizational processes work and how they could be optimized.

Despite the fact of business processes are supported by a wide range of varied software technologies, most part of these softwares have as operational basis a *process model*. This model does more than allow the automated configuration and enactment of the business process; it boosts the *analysis capability*. Business processes can be analyzed under two perspectives: *qualitative* and *quantitative*. As examples of qualitative analysis we can cite the *validation* and the *verification*; the former tests whether the process behaves as expected (semantic correctness) while the later tests whether the process respects some structural criteria (syntactic correctness).

The most representative example of quantitative analysis is the *performance analysis*. The performance analysis of business processes evaluates the ability to meet requirements with respect to throughput times, waiting times, service levels, and resource utilization [Aalst, 1998]. The throughput time of a business process instance is the time spent in its complete execution, comprising the execution time of each activity that compounds the process and the waiting times (due, for example, to required synchronizations and queues to access resources).

Techniques for performance evaluation of computational systems can be based in three distinct approaches: *measurement*, *simulation* and *analytical modeling*. Most of the BPM tools offer support to business process performance evaluation by means of measurements made during the execution of the system. But, in many cases, it is desirable to know the expected dynamic behavior before the system becomes operational, since adjusting an "in production" system that does not achieve the required performance is always a costly task.

This work focus in performance evaluation techniques that take place via *analytical modeling*. Analytical models are usually based on stochastic models, that, in many cases, are assumed to be Markov processes. The main limitation of Markovian models is the well-known state space explosion problem associated with processes with complex structure or large number of components (as in business processes). Large state space models imply in computational difficulties – high memory requirements and high computation time – in the calculation of transient and stationary probability distributions. This problem hinders the application of Markovian models in the analysis of many real-world systems.

Our objective is to evaluate the feasibility of applying different performance analysis models in BPM domain. We present some scenarios of process models containing characteristics commonly found in business processes and their mapping to three high-level formalisms used to performance modeling: *Generalized Stochastic Petri Nets*, *Performance Evaluation Process Algebra* and *Stochastic Automata Networks*. These formalisms are vastly used to capture and analyze the dynamic behavior of parallel and distributed systems.

We can evaluate these formalisms under two perspectives – *modeling* and *solution*. Concerning the first perspective, possible evaluation criteria are: ex-

pressive power, modeling facility, model's readability (abstraction power, compositionality, etc.), and efficacy of their supporting software tools. In the second perspective, the computational resources required to perform the analysis are the points to be evaluated. In this work, we focus on the modeling perspective. Moreover, it is not our aim to compare the formalisms in a general way. Instead, we want to provide a summary of the pros and cons of their usage in the modeling and evaluation of business processes.

The remainder of this report is organized as follows. Section 2 discusses some works that apply formal techniques in business process performance evaluation modeling and briefly introduces the background concepts of the techniques used in this work. The scenarios chosen to illustrate the application of these techniques in business process domain are discussed in Section 3. Section 4 summarizes the results observed from the modeling of these scenarios and presents the concluding remarks.

2 Related Work

Reijers [2003] discussed analytical methods to the performance analysis of *Stochastic Workflows Nets* (SWN) models. The SWNs are sets of loosely coupled business processes modeled using *Workflow Nets* (WN) – a sub class of Petri nets specially designed to workflow modeling – and some interactions between then. Li et al. [2008] introduced a method to evaluate performance measures of inter-organizational workflows based on SWN models. They propose a simplification of the net into a simple system, in which the performance analysis is applied. This gives an approximation of the performance measures of the original net. The technique was inspired by the compositional characteristic of the Stochastic Process Algebras and their basic operational elements. The authors claim that, with this simplification framework, it is possible to analyze large-scale and complicated systems more effectively than with other commonplace methods.

Yaikhom et al. [2007] proposed an approach to performance modeling of workflow systems based on PEPA and the notion of algorithmic skeletons. The complexity of the models can be contained by restricting the mechanisms through which parallelism can be introduced to some basic skeletons. The authors presented an algorithm to the automatic generation of PEPA performance models from workflow system descriptions based on these skeletons. But the skeletons proposed are only feasible to specify pipeline-based workflow models; they are not appropriate to specify the behavior of workflows with complex routing structure of components.

2.1 Applied Formalisms

The first stochastic models applied to performance analysis were the *queueing networks*. Although being an useful analysis tool, the queueing networks are inefficient to express the complex synchronization constraints required in the modeling of modern systems. Accounting these limitations, more high level modeling techniques started to be introduced during the 1980s. Most part of these techniques are based on models from which a continuous time Markov chain (CTMC) can be derived as underlying stochastic model [Hillston and Kloul, 2007].

A Markov process allow the modeling of the uncertainty in real-word systems that evolve dynamically in time. A Markov process is composed of *states* and *state transitions*. The states are *discrete* and *countable*. The state transitions are modeled by a *discrete-time* or *continuous-time* stochastic process, defined by a *geometric* or *exponential* distribution respectively.

In a Markov process, the future probabilistic behavior of the process depends only on the present state of the process and it is not influenced by its past history (*Markovian property*). Two kinds of analysis can be made over Markovian models: *steady state* and *transient state* analysis. In the steady state analysis, we aim to find the *stationary probability distribution* (also known as *equilibrium probabilities*) of the chain, i.e., the long-run fraction of time the process will be in each state with probability 1. In many practical situations, one is interested in the transient behavior of a system, rather than in its log-run behavior. By the transient analysis, it is possible to answer questions such as: (i) the state of a model at the end of a time interval, (ii) the time until an event occurs, (iii) the residence time in a set of states during a given interval, and (iv) the number of given events in an interval.

In this work, we utilize three high level Markovian modeling techniques for performance evaluation based on three different formal methods: *Generalized Stochastic Petri Nets* (GSPN), *Performance Evaluation Process Algebra* (PEPA), and *Stochastic Automata Networks* (SAN). The following subsections present a short introduction of these techniques.

2.1.1 Generalized Stochastic Petri Nets

Stochastic Petri Nets (SPN) are timed transitions Petri nets with atomic firing and in which transition firing delays are exponentially distributed random variables: each transition t_i is associated with a random firing delay whose probability density function is a negative exponential with rate w_i. The original SPN proposal assumed a race execution policy (i.e., when multiple transitions are simultaneously enabled, the transition with the statistically minimum delay to fire is selected).

The Generalized Stochastic Petri Net (GSPN) extend the modeling power of SPN introducing two types of transitions: *immediate transitions* (fire in zero time) and *timed transitions* (fire after a random, exponentially distributed, enabling time) [Balbo, 2007]. Immediate transitions are fired with priority over timed transitions. If only one immediate transition is enabled, it fires and the following marking is produced. When several immediate transitions are enabled, it is necessary use a metric to establish the transition that will fire. If the enabled transitions are concurrent, they can be fired in any order. But when they are in conflict, the selection of the transition to be fired becomes relevant. For this reason, GSPN associate weights (probabilities) with immediate transitions belonging to the same conflict set. A GSPN model is formally defined by an 8-tuple $GSPN = \{P, T, \Pi(.), I(.), O(.), W(.), m_0\}$, where

- P is a set of places and T is a set of transitions;
- $\Pi(.)$ is the priority function that maps transitions into nonnegative natural numbers representing their priority level. Timed transitions are associated with priority zero, whereas all other priority levels are reserved for immediate transitions;

- $I(.)$ is the input function that maps "bags" of places into transitions, represented by directed arcs;
- $O(.)$ is the output function that maps transitions into "bags" of places, represented by directed arcs;
- m_0 is the initial marking;
- $W(.)$ maps transitions into real positive functions of the SPN marking. The quantity $W(t_k, m)$ (or w_k) is called the *rate* of transition t_k in marking m if t_k is timed, and the *weight* of transition t_k in marking m if t_k is immediate.

A k-bounded GSPN with m_0 as initial marking has a finite state space, irreducible, homogeneous and continuous-time semi-Markov process. A semi-Markov process can be analyzed identifying an embedded Markov chain (EMC) that describes the transitions from state to state of the process, disregarding the concept of time and focusing on the set of states of the semi-Markov process.

2.1.2 Performance Evaluation Process Algebra

The first works formalizing the ideas of extending process algebras to delay actions by means of exponential distributions appeared in the early nineties, with the *TImed Processes and Performance* (TIPP) [Hermanns et al., 1998] and the *Performance Evaluation Process Algebra* (PEPA) [Hillston, 1996] – both based on *Communicating Sequential Processes* (CSP). After these approaches, other were created having in common the underlying semantical model closely related to continuous-time Markov chains.

PEPA has two kinds of basic elements: *components* and *activities*. Each activity is represented by a pair (α, r), where α is its *action type* and r is its *activity rate* – the parameter of the negative exponential distribution determining its duration. The set of all possible action types $\mathcal{A}$ includes a distinguished type τ, denoting internal (or "unknown") activities.

PEPA has a small set of combinators that enable the construction of components formed of activities and interactions between them. The syntax for its terms is formally defined as:

$$\begin{array}{l} S ::= (\alpha, r).S | S + S | C_s \\ P ::= P \underset{L}{\bowtie} P | P/L | C \end{array}$$

where S denotes a *sequential component* and P denotes a *model component* which executes in parallel; C stands for a constant which denotes either a sequential or a model component, while C_S stands for constants which denote sequential components. An informal description and interpretation of these combinators is provided in the following:

- *prefix* – $(\alpha, r).S$: this component carries out activity (α, r) and subsequently behaves as S;
- *choice* – $S_1 + S_2$: this component represents a system which may behave either as S_1 or as S_2;

- *cooperation* – $P_1 \underset{L}{\bowtie} P_2$: this component represents a system where P_1 and P_2 must cooperate to achieve any activity whose type is in the cooperation set L (i.e., the activities whose type is in L are only enabled in $P_1 \underset{L}{\bowtie} P_2$ when they are enabled in both P_1 and P_2). When two components cooperate to carry out α, their total capacity to complete α type activities is limited to the capacity of the slower component. In cases where an activity is known to be carried out in cooperation with another component, a component may be *passive* with respect to that activity, denoted by $(\alpha; \top)$. In this case, the rate of the activity is determined by the rate of the activity in the active component;
- *hiding* – P/L : this component behaves as P except that any activities of types $\in L$ are *hidden* (i.e., they appear as the internal behavior τ);
- *constant* – C : is a component whose meaning is given by a definition equation. For example, $C \stackrel{def}{=} P$ assigns the name C to the behavior P.

The semantics of each term in PEPA is given via a labeled multi-transition system. A state corresponds to a *derivative* (i.e., a syntactic term of the language) and an arc represents the activity which causes one derivative to evolve into another. The *derivative set* is the complete set of reachable states. By applying the semantic rules exhaustively over these states we obtain the *derivative graph* (DG) of the model.

The timing aspects of components' behavior are represented in the arcs of the DG as the parameter of the negative exponential distribution governing the duration of the corresponding activity. When an activity $a = (\alpha, r)$ is enabled, it will delay for a period sampled from the negative exponential distribution with parameter r. If more than one activity is enabled concurrently, we assume that a *race condition* exists between them. In this case, the activity whose delay before completion is the least will be the one to succeed.

The DG is the basis of the underlying CTMC of a PEPA model. In order for the CTMC to be ergodic, its DG must be strongly connected. The grammar of PEPA imposes the necessary syntactic conditions for ergodicity.

2.1.3 Stochastic Automata Networks

The *Stochastic Automata Networks* (SAN) is a technique used to model systems with large state spaces, introduced by Plateau [1985]. SAN is specially appropriated to model parallel and distributed systems that can be viewed as collections of components that operate more or less independently, requiring only infrequent interaction 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 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 defined by the combinations 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 *synchronization 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.

The rate at which event transitions occur may be constant (nonnegative real numbers) or may depend upon the state in which they take place. Transitions in which the rate is a function from the global state space to the nonnegative real numbers are called *functional transitions*.

The expression of the infinitesimal generator of the Markov chain associated with a well defined SAN is given using only the generators on these smaller spaces and operators from the *Generalized Tensor Algebra* (GTA) [Brenner et al., 2005], an extension of the *Classical Tensor Algebra* (CTA), also known as *Kronecker Algebra*. 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 a set of $n_i \times n_i$ 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:

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

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

Given that all tensor sum is equivalent to a sum of particular tensor products, the descriptor can be expressed by:

$$Q = \sum_{j=1}^{(N+2E)} \bigotimes_{i=1}^{N}{}_g \, Q_j^{(i)} \tag{2}$$

$$\text{where } Q_j^{(i)} = \begin{cases} I_{n_i} & \text{for } j \leq N \text{ and } j \neq i; \\ Q_l^{(i)} & \text{for } j \leq N \text{ and } j = i; \\ Q_{(j-N)^+}^{(i)} & \text{for } N < j \leq N+E; \\ Q_{(j-(N+E))^-}^{(i)} & \text{for } j > N+E. \end{cases}$$

The state space explosion problem associated with Markov chain models is attenuated by the fact that the state transition matrix is not stored, since it is represented by smaller matrices. All relevant information can be recovered from these matrices without explicitly form the global matrix.

3 Business Process Scenarios

In order to evidence the similarities and differences between the formalisms examined in this work, we will consider different scenarios which illustrate important characteristics of business process models.

Symbol	Meaning
[Activity Name]	Activity (that can be atomic or non-atomic)
	Start Event, End Event, Error End Event
	Exclusive Decision, Inclusive Decision
	Parallel Gateway (AND-Split/AND-Join)
	Simple Gateway, Complex Gateway
text	Annotation

Table 1: Basic flow and connecting objects of BPMN.

The first scenario (Section 3.1) comprises four control-flow structures considered by the *Workflow Management Coalition* (WfMC) (`http://www.wfmc.org`) as the basic ones in business process modeling: sequences, OR splits and joins (choices and joins), AND splits and joins (parallelizations and synchronizations), and iterations. In the second scenario (Section 3.2) the modeling of more complex branching and merging structures is discussed, while the third scenario (Section 3.3) explores features of performance modeling formalisms that are not commonly found in business process modeling languages.

Each scenario is illustrated by a simple real-world business process example, described using the *Business Process Modeling Notation* (BPMN). The BPMN is a standard developed by the *Business Process Management Initiative* (`http://www.bpmn.org`) for graphical representation of business processes. Table 1 shows the meaning of the BPMN flow and connecting objects employed in this text.

The examples presented here were implemented and analyzed[1] using the following tools: PEPS[2] – a software which allows to solve numerically very large Markov chains using an input interface based on SAN; PEPA *Plug-in Project*[3] – a software tool that supports the stochastic process algebra PEPA; SMART[4] – a software package to study complex discrete-state systems, in particular Markov chains and SPN / GSPN. Even though Hillston and Kloul [2007] has already discussed the introduction of functional dependencies (extending the activity rates to include functional rates) in PEPA, the PEPA *Plug-in Project* (that is the most indicated supporting tool for PEPA) does not cover yet this extended version of the formalism. For this reason, we did not use functional dependencies to model our examples in PEPA.

3.1 Scenario 1: Basic Structures

This scenario presents an example of on-line order-through-to-delivery process – a typical order processing, commonly found in e-Commerce applications –,

[1] The results of this implementation are available in `http://www.ime.usp.br/~kellyrb/bp_performance_evaluation`.

[2] `http://www-id.imag.fr/Logiciels/peps`

[3] `http://www.dcs.ed.ac.uk/pepa/tools/plugin/index.html`

[4] `http://www.cs.ucr.edu/~ciardo/SMART`

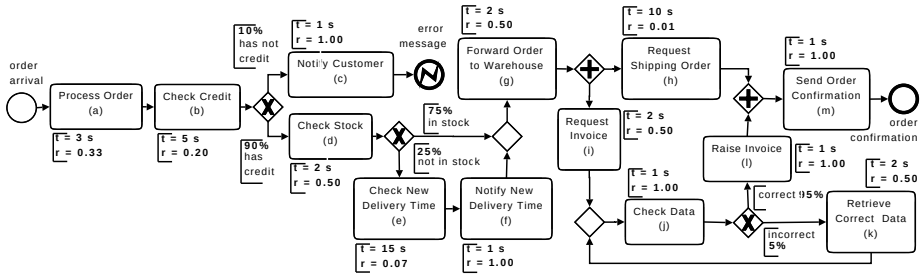


Figure 1: A typical "order processing" (modeled using BPMN).

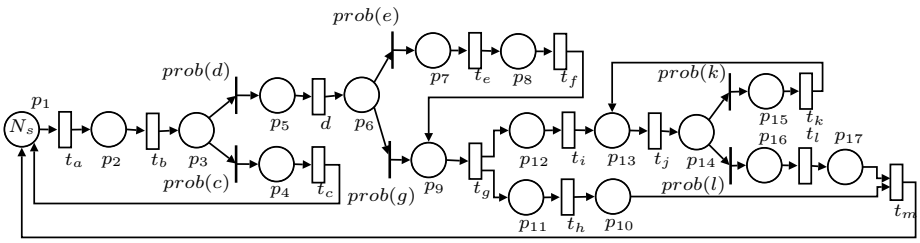
described using BPMN in Figure 1[5]. The process starts with the reception and pre-processing of a new client's order. After, the application checks the client's credit; if some problem is found, the client receives an error notification and the order processing is interrupted. In the other case, the application verifies if the order items are available in the stock warehousing. In the case of unavailability of items, the delivery time is recalculated and then notified. After forwarding the order to the warehouse, the shipping is requested at the same time that the invoice is generated. The generation of the invoice is composed of 4 steps: (i) the invoice is requested, (ii) its data is verified and (iii) updated until the data is correct, and then (iv) the invoice is raised. After the conclusion of the shipping request and the invoice generation, the order processing finishes with the sending of a confirmation to the client.

Figure 1 has, for each activity *act* belonging to the model, an annotation indicating its *average execution time* $t(act)$ and its execution rate $r(act)$ (equivalent to $\frac{1}{t(act)}$). We will assume that the execution time of an activity *act* is an exponentially distributed variable with rate $r(act)$. In Figure 1, we have also annotations indicating the probabilities associated to the branches of the decision gateways. We will also assume that the activities do not share CPU, i.e., the execution rates presented in Figure 1 are constant, and they do not vary in function of the number of activities in execution in a given time.

The activity `Forward Order to Warehouse` marks the start of a parallel branching in the "order processing" model. As we can see in Figure 2, we can denote the start of a parallel branching in a GSPN model by a transition with as many output places as the number of branches in the parallel gateway (in our example, the transition g with its two output places p_{10} and p_{12}). By the same way, a synchronization point (as the one that precedes the activity `Send Order Confirmation`) can be modeled as a transition with as many input places as the number of branches to be synchronized (transition m and its input places p_{11} and p_{17}).

The representation of a parallel gateway in a PEPA model requires more than one component, as we can see in Figure 3 with `POrder` and `PInvoice`. These components are composed by means of the cooperation combinator, using g and m as synchronizing activities – the former marks the start of the parallel branching, while the later delimits the end of the parallelism. After the execu-

[5]The example of Figure 1 is based on a business process model diagram available in `http://www.businessballs.com/business-process-modelling.htm#BPM-example`.



Timed	t_a	t_b	t_c	t_d	t_e	t_f	t_g	t_h	t_i	t_j	t_k	t_l, t_m
Rate	0.33	0.2	1	0.5	0.07	1	0.5	0.01	0.5	1	0.5	1

Imm.	$prob(c)$	$prob(d)$	$prob(e)$	$prob(g)$	$prob(k)$	$prob(l)$
Prob.	0.1	0.9	0.25	0.75	0.05	0.95

Figure 2: GSPN model of the "order processing" example.

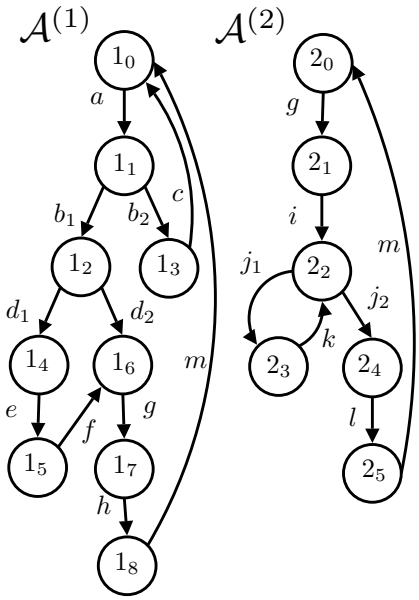
tion of g and before the execution of m, the two components are constrained to act together[6].

```
// Execution rates associated to each activity
r_a = 0.33;   r_b = 0.20;   r_c = 1.00;   r_d = 0.50;
r_e = 0.07;   r_f = 1.00;   r_g = 0.50;   r_h = 0.01;
r_i = 0.50;   r_j = 1.00;   r_k = 0.50;   r_l = 1.00;
r_m = 1.00;

// Routing probabilities associated to the choices
prob_c = 0.10;   prob_d = 1 - prob_c;
prob_e = 0.25;   prob_g = 1 - prob_e;
prob_k = 0.95;   prob_l = 1 - prob_k;

num_servers = 1; // Number of servers

// Order processing
POrder = (a,r_a).((b,prob_c * r_b).(c,r_c).POrder +
                  (b,prob_d * r_b).PStock);
PStock = (d,prob_g * r_d).PFinalize +
         (d,prob_e * r_d).(e,r_e).(f,r_f).PFinalize;
PFinalize = (g,r_g).(h, r_h).(m,r_m).POrder;
PInvoice  = (g,⊤).(i,r_i).PCheck;
PCheck    = (j,prob_l * r_j).(l,r_l).(m,⊤).PInvoice +
            (j,prob_k * r_j).(k,r_k).PCheck;

POrder[num_servers] <g,m> PInvoice[num_servers]
```

Figure 3: PEPA model of the "order processing" example.

[6]The cooperation operator $\underset{L}{\bowtie}$ of PEPA is represented in the PEPA *Plug-in Project* compiler as `<L>`.

In SAN models, parallel behaviors are expressed by different automata. As can be observed in Figure 4, two automata are required (one for each branch of the parallelism) to represent the parallel gateway of the "order processing" example. The automata synchronization is made by the events g and m.



Event	**Rate**
a	0.33
b_1	$prob(d) * 0.20$
b_2	$prob(c) * 0.20$
c	1.00
d_1	$prob(e) * 0.50$
d_2	$prob(g) * 0.50$
e	0.07
f	1.00
$\boldsymbol{g}$	0.50
h	0.01
i	0.50
j_1	$prob(k) * 1.00$
j_2	$prob(l) * 1.00$
k	0.50
l	1.00
$\boldsymbol{m}$	1.00

Figure 4: SAN model of the "order processing" example.

An exclusive decision gateway can be modeled in GSPN by transitions that share a same input place. If there is a probability associated to each branch of the decision, we can use immediate transitions to logically represent them. In the "order processing" example, the activities `Notify Customer` and `Check Stock` are in an exclusive decision gateway. In the GSPN model of Figure 2, this decision gateway is represented by the place p_3 and the two immediate transitions labeled with their probabilities – $prob(c)$ and $prob(d)$.

To represent an exclusive decision in PEPA, we use the choice combinator "+". The rate of the activity that precedes the decision gateway can be adjusted to capture the probability of each possible branch of the decision, as made with the activity b in the component `POrder` of the model in Figure 3.

In a SAN model, an exclusive decision is represented by a state with two or more output transitions (like the state 1_1 in automaton $\mathcal{A}^{(1)}$ of Figure 4). As occurs in PEPA models, the race condition that governs the dynamic behavior of a model when more than one activity is enabled allows us to specify probabilistic branchings. To express the probabilities associated to the branches, we can associate to each transition an event which rate is given by the rate of the event that precedes the decision point multiplied by the probability of the transition (as made with the transitions associated to the events b_1 and b_2 in automaton $\mathcal{A}^{(1)}$ of Figure 4).

The underlying Markov chains derived of the models in Figures 2, 3 and 4 are equals: they have the same number of reachable states (for only 1 server,

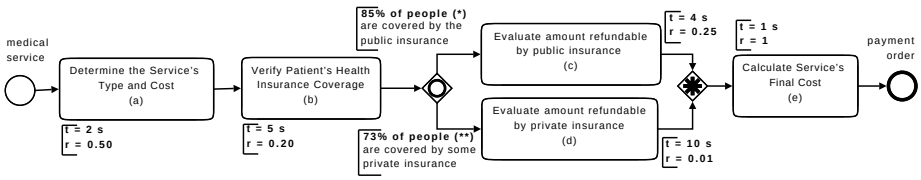
we have 17 states) and the same steady state probability distribution for these states.

In the GSPN models of this work we are assuming a *infinite-server* semantics for the firing of the transitions: every enabling set of tokens are processed as soon as it forms in the input places of the timed transition. Its corresponding firing delay is generated at time, and the timers associated with all these enabling sets run down to zero in parallel. Under this assumption, the process in Figure 2 models multiple servers by setting the number of tokens N_s in place p_1 to a number greater than 1. In a PEPA model, more servers can be added to the system by the cooperation combinator $\underset{L}{\bowtie}$. For example, `POrder` $\bowtie$ `POrder` signalizes two servers of the "order processing" in execution. The PEPA *Plug-in Project* represents the same expression as `POrder[2]`; in the model of Figure 3, the number of servers is parameterized by the variable `num_servers`. Following the same idea, SAN models can express multiple servers by means of replicated automata.

Varying the number of servers of the "order processing" models (as previously described) and applying aggregation techniques for PEPA and SAN models with replicas [Benoit et al., 2004; Ribaudo, 1995], we obtain the same reachable state-space in the three formalisms. The idea behind the aggregation techniques is to use some notion of equivalence of states to partition the underlying state space of a model into equivalence classes, reducing the complexity of the analysis.

3.2 Scenario 2: Advanced Branching/Merging

This scenario considers business process models with sophisticated structures of branching and merging. Well-known examples of structures with these characteristics are the second category of control-flow patterns described by Aalst et al. [2003]: *multi choice*, *synchronized merge*, *multi merge*, and *discriminator*.



(*) Source: *Portail de la Sécurité Sociale* (`http://www.securite-sociale.fr/chiffres/stat/statistiques.htm`).

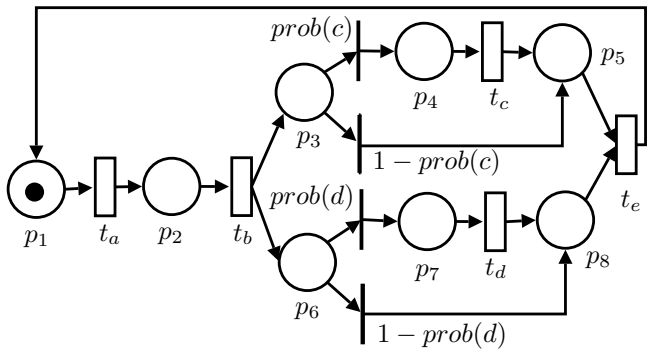
(**) This percentage may not reflect the current situation in France.

Figure 5: A simplified view of the french process to determine the cost of a medical service (modeled using BPMN).

To illustrate the patterns, we will use a process that occurs in french health-care system. The health-care system in France involves a mix of public and private financing. The public financing offers the coverage (sometimes, partially) of basic medical services. But the French may also buy supplemental insurance which reduces their out-of-pocket costs and possibly covers extra expenses (as private hospital rooms, eyeglasses, and dental care). So, to determine the final costs of a medical service to a patient in France, it is necessary: (i) to

verify if the patient is covered by public and/or private health insurance; (ii) for each applicable insurance, to evaluate the refundable amount according to the executed medical service; and, finally, (iii) to combine the covered values (if they exist!) in order to calculate the final costs. Figure 5 shows the BPMN model of the described process.

The activity `b` in Figure 5 marks the beginning of a *multi choice*: after its execution, none or several branches of a choice can be selected to be executed (in the example, there are only two choices to be considered – activities `c` and `d`). As they can be executed in a parallel way, a special structure to merge the selected branches of a *multi choice* is needed. The three kinds of merge usually found in business processes are: (i) the *synchronized merge*, that synchronizes the end of the execution of all selected branches before enabling the next activities (sub-process); (ii) the *multi-merge*, that enables the next sub-process every time the execution of a selected branch finishes; and (iii) the *discriminator*, that enables the next sub-process only once, when the execution of the fastest selected branch finishes. By the given description of the french health-care process, it is easy to identify that the merging pattern associated to the *multi choice* of Figure 5 is a *synchronized merge*. Figures 6, 7 and 8 show the mapping of this process in GSPN, PEPA and SAN, respectively.



Timed	t_a	t_b	t_c	t_d	t_e	**Immed.**	$prob(c)$	$prob(d)$
Rate	0.5	0.2	0.25	0.01	1	**Prob.**	0.85	0.73

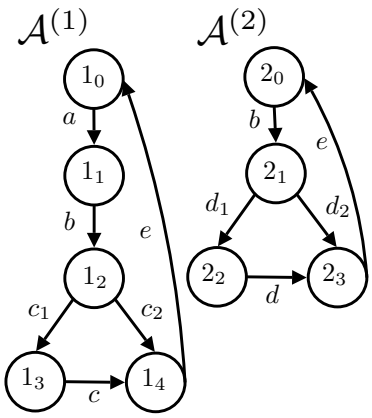
Figure 6: GSPN model of the "calculating cost" process.

In the GSPN model, to express the *multi choice* we used two immediate transitions to each branch of the choice – one models the probability of the branch be executed, and the other models the probability of it not be executed. Since the immediate transitions in Figure 6 will feed the places p_5 and p_8 even if the branches are not selected to be executed, the *synchronized merge* is directly modeled by transition t_e and its two input places p_5 and p_8 (as made in the simple synchronization example presented in the first scenario).

As they have no execution rates, the immediate transitions of the GSPN formalism help us to express more sophisticated branching and merging structures without impacting the performance analysis results. To automatically map the *multi choice* pattern in PEPA (SAN), we need to introduce "artificial" actions (events) – i.e., actions (events) which do not exist in the real process. This artifact helps the modelling task, but must be used consciously, since it may bring

```
// Execution rates associated to each activity
r_a = 0.50;   r_b = 0.20;   r_c = 0.25;   r_d = 0.01;
r_e = 1.00;   r_immediate = 50.00;

// Routing probabilities associated to the multi-choice
prob_c = 0.85;   prob_d = 0.73;

// Medical service cost calculation process
PCalc = (a,r_a).(b,r_b).(e,r_e).PCalc;
P1 = (b,⊤).((c1,prob_c * r_immediate).(c,r_c).(e, ⊤).P1 +
            (c2,(1-prob_c) * r_immediate).(e,⊤).P1);
P2 = (b,⊤).((d1,prob_d * r_immediate).(d,r_d).(e,⊤).P2 +
            (d2,(1-prob_d) * r_immediate).(e,⊤).P2);

PCalc <b,e> P1 <b,e> P2
```

Figure 7: PEPA model of the "calculating cost" process.



Event	Rate
a	0.50
$\boldsymbol{b}$	0.20
c_1	$prob(c) * 50$
c_2	$(1 - prob(c)) * 50$
c	0.25
d_1	$prob(d) * 50$
d_2	$(1 - prob(d)) * 50$
d	0.01
$\boldsymbol{e}$	1.00

Figure 8: SAN model of the "calculating cost" process.

undesirable results. The time spent in an artificial action or event influence the performance measures. We can minimize this impact in the analysis results by attributing high execution rates (relative to the other rates in the model) to these artificial actions or events. Furthermore, in most part of the real systems it is reasonable to assume that, attached to each routing decision or synchronization, we have a computational effort that should also be considered in the modeling.

The artificial action types created to model the example in PEPA was `c1`, `c2`, `d1` and `d2` of Figure 7. The PEPA processes `P1` and `P2` represents the branches of the *multi choice*; they cooperate with the main process `PCalc`, using `<b,e>` as synchronizing action set. In an equivalent way, the local events c_1, c_2, d_1 and d_2 were introduced in the SAN model of Figure 8 to express the execution probabilities of each branch of the *multi choice*; the synchronizing events b and

e are responsible for delimiting, respectively, the beginning of the choice and the synchronizing merge.

3.3 Scenario 3: Functional Dependencies

This scenario represents business processes with activities whose execution rates depend on the state of the system. As illustration, consider Figure 9, which defines a simple "producer / packer" process. The process is composed of three sub-processes: one representing a producer of items (that uninterruptedly operates while the stock is not full), a second one representing a packer (that needs to group items to create a new package and send it from the stock to the transportation service), and a last sub-process representing the limited size stock itself. In the example, we consider that a new package must always contain three items, and the stock can only keep nine items per time. As the stock is a sub-process with a passive characteristic, it was modeled in BPMN as a data object. The data objects do not have any direct effect on the sequence flow of the process, but they provide information about what activities require to be performed and/or what they produce.

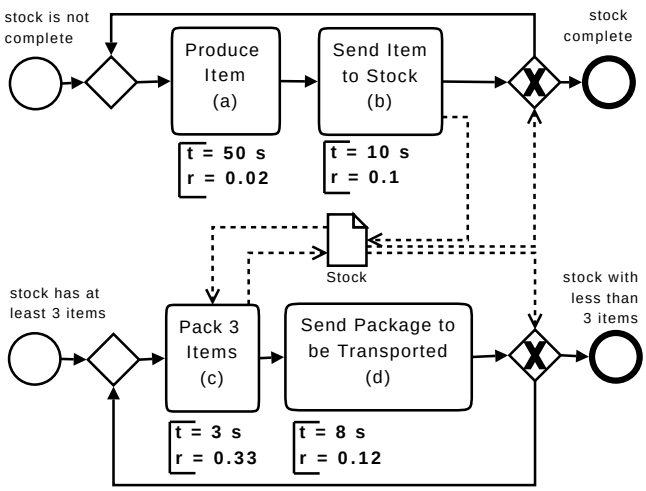
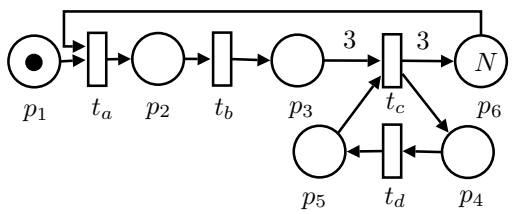


Figure 9: A simple "producer / packer" process (modeled using BPMN).

Figures 10, 11 and 12 show the modeling of the example in GSPN, PEPA, and SAN, respectively.



Timed	Rate
t_a	0.02
t_b	0.1
t_c	0.33
t_d	0.12

Figure 10: GSPN model of the "producer/packer" process.

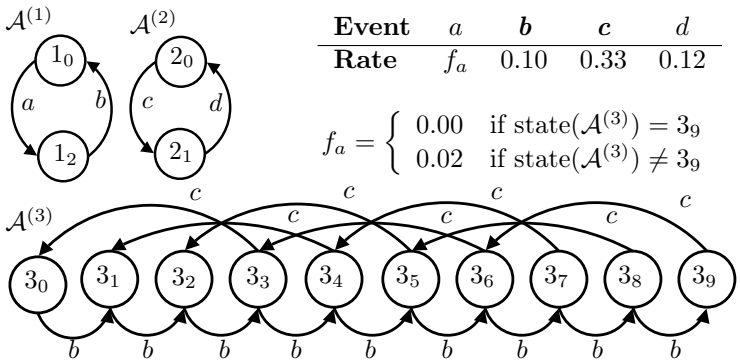
In the GSPN model of Figure 10, the stock is modeled by the net places p_3 and p_6; p_3 keeps the current number of produced items in stock, while p_6 indicates the number of available places in the stock. Transition t_a is enabled only when there is some token in p_6, i.e., when the stock is not complete. The initial number of tokens in place p_6 is given by N – a variable that represents the max number of items that the stock can keep. With this modeling approach, we can easily change the stock capacity without structurally changing the model. As the produced items are modeled as tokens, we can use arc weights to model the requirement of three items per package generated.

To model the example in PEPA without using functional rates, we have to use additional synchronizing actions, as showed in Figure 11. The stock is represented by the sub-processes `PStock0`, `PStock1` ..., `PStock9`. Generally, to represent set of resources in PEPA models we need at least as much sub-processes as the number of resources. The execution of action `(a, rate_a)` in `PProducer` will only occur in cooperation with one sub-process $\texttt{PStock}_i$, where $0 \leq i \leq 8$.

```
// Execution rates associated to each activity
rate_a = 0.02; rate_b = 0.1; rate_c = 0.33; rate_d = 0.12;

// Producer and Packer processes
PProducer = (a, rate_a).(b, rate_b).PProducer;
PPacker   = (c, rate_c).(d, rate_d).PPacker;

// Stock - with max number of items equals to 9
PStock0 = (a,⊤).(b,⊤).PStock1;
PStock1 = (a,⊤).(b,⊤).PStock2;
PStock2 = (a,⊤).(b,⊤).PStock3;
PStock3 = (a,⊤).(b,⊤).PStock4 + (c,⊤).PStock0;
PStock4 = (a,⊤).(b,⊤).PStock5 + (c,⊤).PStock1;
PStock5 = (a,⊤).(b,⊤).PStock6 + (c,⊤).PStock2;
PStock6 = (a,⊤).(b,⊤).PStock7 + (c,⊤).PStock3;
PStock7 = (a,⊤).(b,⊤).PStock8 + (c,⊤).PStock4;
PStock8 = (a,⊤).(b,⊤).PStock9 + (c,⊤).PStock5;
PStock9 = (c,⊤).PStock6;

PProducer <a,b> PStock0 <c> PPacker
```

Figure 11: PEPA model of the "producer/packer" process.

In SAN, we have an automaton to represent each entity of the system – $\mathcal{A}^{(1)}$ is the producer, $\mathcal{A}^{(2)}$ is the packer and $\mathcal{A}^{(3)}$ represents the stock. In $\mathcal{A}^{(3)}$ we have a state for each possible number of items in the stock in a given moment. To avoid the occurrence of the event a (the production of an item) when the stock is already full, we can made use of the powerful concept of functional transitions of SAN. The rate of a is defined in function of the current state of the automaton $\mathcal{A}^{(3)}$: if the state of $\mathcal{A}^{(3)}$ is 3_9 (i.e., the stock is full), then the rate of a will be 0 (i.e., the event will not occur); in the other case, the rate of a will be 0.02.



Event	a	$\boldsymbol{b}$	$\boldsymbol{c}$	d
Rate	f_a	0.10	0.33	0.12

$$f_a = \begin{cases} 0.00 & \text{if state}(\mathcal{A}^{(3)}) = 3_9 \\ 0.02 & \text{if state}(\mathcal{A}^{(3)}) \neq 3_9 \end{cases}$$

Figure 12: SAN model of the "producer/packer" process.

4 Synthesis

As we illustrated in Section 3.1, the basic control-flow structures used in business process modeling can be represented in the three formalisms with equivalent facilities. But more complex modeling requirements, as the one illustrated in Sections 3.2 and 3.3, evidence their pros and cons.

To model advanced branching and merging structures, the immediate transitions of the GSPN formalism proved to be a powerful tool. They facilitate the modeling task without impacting the readability of the model or the analysis results (as illustrated in Section 3.2).

Functional dependencies between the activities of a process can be directly expressed using functional rates (present both in GSPN and SAN). Section 3.3 discussed one special use case, but modeling using functional rates is also particularly interesting in cases where the rates of the activities vary according to the load of the system or the number of available physical resources.

The compositional characteristic of PEPA and SAN formalisms helps us to model the system in a modular way. These kinds of models provide a good insight in how the system should be implemented, at the same time as they improve the expansion capability of the system. In the biggest scenario of this work, presented in Section 3.1, the advantages of the compositional aspect of PEPA and SAN can be easily noticed.

Both PEPA and GSPN models suffer from state space explosion; in the SAN models, this problem is attenuated by the use of tensor algebra (incorporated into the formalism) for state space representation. It is important to notice that the use of a tensor representation of the underlying Markov process is not a state space reduction method, but, instead, it is an alternative approach to state space explosion which handles the model solution in a decomposed form [Hillston and Kloul, 2007]. The SAN formalism was the first to use tensor algebra to represent the models, but, as discussed by Donatelli [1994]; Hillston and Kloul [2001], it is also possible to apply the tensor representation in GSPN and PEPA models.

In general, performance measures are derived from the steady state solution of a Markov chain by associating a reward structure to its states. But it is also possible to define rewards at the level of the high level model (GSPN, PEPA and

SAN), rather than at the level of the its underlying Markov process. In PEPA models, it is possible to associate rewards with certain activities within the system. The reward associated with a component, and the corresponding state, is then the sum of the rewards attached to the activities it enables. This works well when the measure of interest may be phrased in terms of some identifiable aspect of the system behavior that is associated with activities; but in some cases, it may be important to consider a measure which is explicitly formulated over states. In these cases, GSPN and SAN models are more advantageous.

The following subsections recall the main characteristics of each considered stochastic formalism in order to compare them under the modeling perspective, using as evaluation criteria the expressive power, the facility of modeling and the readability of the models (abstraction power, compositionality, etc.), always regarding the business process requirements. Some of these criteria were already discussed in a more general context by Donatelli et al. [1995]; Hillston [1996].

4.1 GSPN

Positive aspects: (i) GSPN have a graphical notation that provides a clear image of the dynamic behavior of the model. (ii) In addition, the presence of immediate transitions and markings facilitate the modeling abstraction. The fire of a timed transition represents in a natural way the completion of a time consuming activity; in the other hand, an immediate transition can represent a routing decision or a merely synchronization. Places and tokens permit a more directly modeling of countable resources. (iv) GSPN models have a clear notion of states. As consequence, performance indices involving "state-based" information can be efficiently computed.
Negative aspects: (i) At the same time that GSPN's graphical notation facilitates the comprehension of the dynamic behavior, it provides little insight into the structure of the system. (ii) GSPN models are not intrinsically compositional; building a new large-scale business process model from the sketch or expanding an existing one using GSPN is not a trivial task.

4.2 PEPA

Positive aspects: (i) PEPA allows to model system's behavior as separated components, since it counts with compositional constructors and other abstraction mechanisms (as the τ actions). (ii) Additionally, being based on process algebra, PEPA is equipped with facilities for reasoning about models, since the notions of equivalence are defined in terms of the operational semantics.
Negative aspects: (i) PEPA is focused on actions and does not have (explicitly) the notion of state; a state is associated with each vertex of a PEPA model's DG. (ii) Moreover, it does not have the concept of "immediate" actions; this lack brings difficulties in the modeling of advanced branchings and merges without affecting performance measures. (iii) The most recommended supporting tool for PEPA (the PEPA *Plug-in Project*) does not implement yet the concept of functional rates.

	GSPN	PEPA	SAN
Modeling criteria			
Expressive power	+	-	+
Abstraction power	+	+	+
Facility to enlarge	-	+	+
Readability	-	+	+

Table 2: Comparison summary of the performance evaluation formalisms having as basis business process modeling.

4.3 SAN

Positive aspects: (i) As the GSPN, SAN models have a clear notion of states. (ii) The SAN formalism counts with a powerful abstraction mechanism – the *functional transitions* – that allows a system to be modeled using fewer automata and fewer synchronizing transitions (reducing, as consequence, the computational effort involved in the solution of the model). (iii) The global matrix of the underlying Markov chain of a SAN model is never explicitly generated. Individual component matrices and information concerning component interactions are combined into the SAN descriptor, that is written as a sum of tensor products. For that, a SAN model (generally) requires less memory during its solution than other kind of models; the representation of the model remains compact even when the underlying Markov chain of the model is very large. (iv) Finally, SAN models are suitable for structured analysis, since the state space of the system is represented as a product of smaller state spaces.
Negative aspects: (i) The functional transitions adds flexibility with respect to the model construction but do not allow to abstract time. (ii) The computation time of the solution of a SAN model may be excessively long due to the cost of computing the tensorial product-vector multiplications. (iii) SAN models are useful for automata which have some interaction (otherwise the stochastic behavior could be modeled with separated Markov chains). However, too much interaction among the automata can complicate the SAN model to the point that its use is questionable, since increasing the number of synchronizing events increases also the complexity of the model described by Equation 2.

Table 2 summarizes the advantages and disadvantages of the formalisms observed over the mapping of the scenarios presented in Section 3. The symbol "+" in the table indicates that the formalism satisfactorily meets the criterion, while "-" indicates that it does not sufficiently support the analyzed criterion. It is important to reinforce that this evaluation is made in the BPM context, since these criteria can seem excessively "subjective" if we consider them in a more general way.

References

Aalst, W. (1998). The application of petri nets to workflow management. *The Journal of Circuits, Systems and Computers 8*(1), 21–66.

Aalst, W., A. Hofstede, B. Kiepuszewski, and A. P. Barros (2003). Workflow patterns. *Distributed and Parallel Databases 14*(1), 5–51.

Aalst, W., A. H. M. ter Hofstede, and M. Weske (2003). Business process management: a survey. In *BPM 2003: Business Process Management International Conference*, Volume 2678 of *Lecture Notes in Computer Science*, pp. 1–12. Springer Berlin.

Balbo, G. (2007). Introduction to generalized stochastic Petri nets. In *SFM 2007: 7th International School on Formal Methods for the Design of Computer, Communication, and Software Systems*, Volume 4486 of *Lecture Notes in Computer Science*, pp. 83–131. Springer Berlin.

Benoit, A., L. Brenner, P. Fernandes, and B. Plateau (2004). Aggregation of stochastic automata networks with replicas. *Linear Algebra and its Applications 386*(1), 111–136.

Brenner, L., P. Fernandes, and A. Sales (2005). The need for and the advantages of generalized tensor algebra for Kronecker structured representations. *International Journal of Simulation: Systems, Science & Technology 6*(3-4), 52–60.

Donatelli, S. (1994). Superposed generalized stochastic Petri nets: definition and efficient solution. In *15th International Conference on Application and Theory of Petri Nets 1994*, Volume 815 of *Lecture Notes in Computer Science*, pp. 258–277. Springer Berlin.

Donatelli, S., M. Ribaudo, and J. Hillston (1995). A comparison of performance evaluation process algebra and generalized stochastic Petri nets. In *PNPM '95: 6th International Workshop on Petri Nets and Performance Models*, pp. 158–168. IEEE Computer Society.

Hermanns, H., U. Herzog, and V. Mertsiotakis (1998). Stochastic process algebras - between LOTOS and Markov chains. *Computer Networks 30*(9-10), 901–924.

Hillston, J. (1996). *A Compositional Approach to Performance Modelling*. New York, NY, USA: Cambridge University Press.

Hillston, J. and L. Kloul (2001). An efficient Kronecker representation for PEPA models. In *PAPM-PROBMIV '01: Joint International Workshop on Process Algebra and Probabilistic Methods, Performance Modeling and Verification*, Volume 2165 of *Lecture Notes in Computer Science*, pp. 120–135. Springer Berlin.

Hillston, J. and L. Kloul (2007). Formal techniques for performance analysis: Blending SAN and PEPA. *Formal Aspects of Computing 19*(1), 3–33.

Li, Y., C. Lin, and Q.-L. Li (2008). A simplified framework for stochastic workflow networks. *Computers & Mathematics with Applications 56*(10), 2700–2715.

Plateau, B. (1985). On the stochastic structure of parallelism and synchronization models for distributed algorithms. *SIGMETRICS Performance Evaluation Review 13*(2), 147–154.

Reijers, H. A. (2003). *Design and Control of Workflow Processes: Business Process Management for the Service Industry.* Secaucus, NJ, USA: Springer-Verlag New York.

Ribaudo, M. (1995). On the aggregation techniques in stochastic petri nets and stochastic process algebras. In *Proceedings of the Third International Workshop on Process Algebras and Performance Modelling*, pp. 600–611. British Computer Society.

Yaikhom, G., M. Cole, S. Gilmore, and J. Hillston (2007). A structural approach for modelling performance of systems using skeletons. *Electronic Notes in Theoretical Computer Science 190*(3), 167 – 183.

Contents

1 Introduction **3**

2 Related Work **4**
2.1 Applied Formalisms 4
2.1.1 Generalized Stochastic Petri Nets 5
2.1.2 Performance Evaluation Process Algebra 6
2.1.3 Stochastic Automata Networks 7

3 Business Process Scenarios **8**
3.1 Scenario 1: Basic Structures 9
3.2 Scenario 2: Advanced Branching/Merging 13
3.3 Scenario 3: Functional Dependencies 16

4 Synthesis **18**
4.1 GSPN 19
4.2 PEPA 19
4.3 SAN 20

Centre de recherche INRIA Grenoble – Rhône-Alpes
655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399