Planning Databases Service Level Agreements through Stochastic Petri Nets

Marcelo Teixeira[1], Pablo Sabadin Chaves[2]

[1] Universidade Federal de Santa Catarina
mt@das.ufsc.br
[2] Dueto Tecnologia Ltda
pablo@duetotecnologia.com.br

Abstract. The Service Oriented Architecture (SOA) has become a pattern for managing business transactions through distributed services, usually supported by third-party providers. In the SOA world, Service Level Agreements (SLA) are used to establish the requirements between customers and providers. Among the clauses agreed in SLA are those related with databases execution time, which have direct relationship with the overall web services performance. However, the high variability, typical of the SOA environments, makes difficult the negotiation of an appropriate SLA that could be guaranteed in practice. Thus, efforts to predict the quality of SOA-based transactions are justified by anticipate possible problems that tend to arise at run-time, disturbing the SLA clauses. In this article we propose a simulation modeling approach, based on stochastic Petri nets, for performance evaluation of databases requests in data-intensive business applications. Through our proposal it is possible to predict the resources consumption and performance degradation of databases, upon the variation of the workload levels, both at design-time and run-time. A case study was conducted in order to illustrate our contributions.

Categories and Subject Descriptors: D.4.8 [**Software**]: Performance—*Modeling and prediction*

Keywords: Databases, GSPN, Performance Evaluation, SLA compliance

1. INTRODUCTION

The new patterns for the informations systems are increasingly prioritizing the integration of organizational processes and the third partnerships enhancement, which results in distributed business models. In this context, the service-oriented architecture (SOA) has supported these new paradigms. Concisely, SOA is an architecture focused on heterogeneous environments, in which software components could be executed in platforms with distinct characteristics. Among the advantages achieved by adopting SOA paradigm are component reuse, interoperability, flexibility, integration, etc.

Within the SOA world, the quality of services (QoS) directly effects the quality of business transactions and the relationships between customers and service providers as well [Casati et al. 2003]. Usually, Service Level Agreements (SLA) are used to establish the legal commitments between them, whose breach may entail legal penalties. Among the clauses agreed in a SOA-aware SLA, are those related with databases (DB) spent time, which affect the web services overall performance.

However, the high variability typical of the SOA environments, makes difficult the negotiation of a SLA level that could, in fact, be guaranteed in practice. The ratio of the load variation for Internet applications can achieve the order of 300% [Chase et al. 2001], which makes critical the challenge of managing performance requirements of these systems. Moreover, in SOA it is common that the applications are constantly changing, with new services being introduced, updated and/or removed [Baresi and Guinea 2008] and it is essential to predict the impact of these modifications over the agreed contracts. In practice, the designers develop the application and stress it to measure its quality. This approach can be expensive and time consuming, since the analyst needs an executable version for that. Thus, estimating SOA metrics prior to develop or change the physical system, is essential to avoid undesirable behavior.

Recently proposed alternatives suggest adopting models for performance prediction in SOA-based systems [Rud et al. 2007; Teixeira et al. 2010], which certainly helps to establish some SLA clauses [Teixeira et al. 2011]. However, these approaches have basically focused on evaluating SOA network and web service transactions, generalizing that the time spent by DB queries is implicit into the service performance. From a SLA point of view it seems a gap, since some clauses can be specifically related with DB spent time. Moreover, this important drawback can affect a reliable SOA capacity planning, whereas one can not verify, for example, the DB performance degradation insofar as the workload increases nor the spent time by messages waiting in input and output database buffers, which is common in data-intensive applications.

Therefore, we propose in this work a GSPN (Generalized Stochastic Petri Nets [Marsan et al. 1995]) based simulation modeling for estimating the performance of databases operations in SOA-aware scenarios. Our model analyses the resource consumption and performance levels degradation in databases with highly variable workloads. Then, based on these information it is possible to elaborate, at modeling time, accurate agreements to be established between services customers and providers.

The main advantage of our proposal, with relation to another similar alternatives, is that it does not require real time measurements nor the complete system implementation to provide useful estimates. These information are not always available at design time, when a SOA capacity planning is useful. Instead, our model is supported by higher level parameters, collected from the Data Base Management System (DBMS) configuration and from a set of samples containing DB queries executions statistics. For this reason, the adopted technology, structure and/or particular type of operation, are irrelevant.

In order to illustrate our approach and analyze its accuracy, we develop a case study where were compare the estimated results against those measured from an evaluated DB system, performed in a real SOA environment. The remainder of this article is organized as follows: Section 2 describes some related works; Section 3 introduces the basic concepts of SOA, SLA and GSPN; Section 4 presents the proposed performance model. Finally, Section 5 presents the developed case study and Section 6 the final comments.

2. RELATED WORK

Performance Evaluation of DB systems has been explored since the initial proposals of DB technologies [Elhardt and Bayer 1984; Adams 1985]. However, with the web advent, DB systems have embodied new features, necessary to supply emergent requirements, as parallel and distributed extensions [Dewitt and Gray 1992], object [Kim et al. 2002] and service orientation [Tok and Bressan 2006], etc. Although these new concepts have played an essential technological role, evaluating their performance is difficult due to the variable and data-intensive environments where they are immersed.

In [Ranganathan et al. 1998] the authors discuss the impact of radically different workload levels on the performance of DB applications and how it becomes an important concern when it is necessary to provide service guarantees. Still, [Krompass et al. 2008] focuses on to separate the requests belonging to different levels of workloads, which allow to adopt particular policies when performing them.

Also, in [Lumb et al. 2003] is developed an approach for guaranteeing DB performance levels in highly variable scenarios. Through a developed tool, the authors suggest to retain and divert transactions that could saturate the system and cause performance loss. Thus, they avoid violating the agreed service clauses.

In [Schroeder et al. 2006], the authors develop the framework EQMS (External Queue Management System), that acts externally to the DB system, filtering the arrival of DB requests and so scheduling where and when each transaction is dispatched to the DBMS. Among the benefits provided by the EQMS, is the possibility of labeling requests before their execution, composing classes of similar transactions to be performed according to QoS rules and/or priorities polices.

In fact, the existent proposals seem effective for dealing with DB transactions in their real environment, estimating and improving their qualitative metrics. However, specially when interacting with service partnerships, the business managers need to be aware on the capacity planning and SLA negotiation issues. Usually, these information are required at design-time, in order to compose SLA clauses. Then, most of the existent alternatives for DB performance evaluation, although efficient, can be useless for this purpose.

An option to cover these gaps is by adopting analytic models. In [Tomov et al. 2004], for example, the authors suggest adopting a queueing networks approach for DB response time estimation. The DB execution time is estimated by mapping DB queries according to patterns of resource consumption. Meanwhile, queue times are predicted through using heuristic rules [Zhou et al. 1997]. Similarly, [Osman et al. 2010] develop an approach to evaluate a particular DB design before its implementation. In fact, these proposals seem closed with ours, since are focused on predictive organizational support.

However, analytic models are shown to be predominantly deterministic, which does not often match the characteristics of the real web environments [Teixeira et al. 2010]. Moreover, they can be inflexible when adopting different probability distribution, in order to variate the modeled system behavior. Still, their accuracy can be degraded when representing queues times.

The problem is that, in practice, there is no predictable execution patterns for distributed web transactions and, certainly, it makes critical any type of performance estimation [Nicola and Jarke 2000]. Therefore, one can imagine how difficult is provisioning storage resources to ensure that database queries will execute enough quickly that will not delay the process more than the expected [Reiss and Kanungo 2005]. In this sense, we suggest that a stochastic simulation approach can absorb most of these drawbacks and, as we shall describe, it can be powerful from a SLA planning point of view.

3. PRELIMINARIES

In this section we present the general concepts associated with the SOA paradigm, implementation and relationship between SOA users, highlighting the legal clauses usually agreed among them, particularly those related with DB performance guarantees.

In order to provide a way for improve the SOA contracts negotiation, we also introduce the technical foundation used for that. In particular, we briefly describe GSPN and probability distributions concepts, essentials to develop the proposed model for DB performance evaluation.

3.1 SOA and Related Concepts

SOA emerges as a new paradigm for information planning and business processes integration. SOA is not a tool, but principles or concepts. This architecture is defined on the basis of three fundamental technical concepts [Josuttis 2008], as follows:

Functionality as services: service is a SOA element that operates independently from the other components of the system. Usually, a services receives one or more requests, processes them, and returns its contributions, through an interface;

Enterprise Service Bus (ESB): is the infrastructure that provides interoperability between different distributed systems and services. From a more practical perspective, the ESB can be understood as a mean by which a client invokes one or more services provided by suppliers;

Loose coupling: by focusing on large distributed systems, SOA supports the reduction of dependence between services, avoiding that a failure or maintenance affects significantly other services.

The most widely used language for SOA systems orchestrations is BPEL [Oasis 2011]. Through simple primitives and a distributed nature, BPEL provides facilities for orchestrating modern business logic flow, where complex operations are supported by nesting basic activities. The basic structures of BPEL are *Invoke, Receive, Assign, Wait* and *Reply.* The composite structures involve *Sequence, If, Pick, RepeatUntil/While/ForEach* and *Flow.*

In the SOA world, the service commitments between customers and providers are expressed by contracts, known as Service Level Agreements (SLA) [Sturm et al. 2000]. A SLA expresses responsibilities and rights with respect to QoS levels, such as but not limited to response time, availability, cost, etc. [Pinheiro 2009]. It also foresees the penalties for the cases where the quality falls below the promised standard [Raibulet and Massarelli 2008].

One can imagine how difficult is to fulfill this type of commitment, when dealing with highly variable environments, as is the case of the SOA scenarios. Successful examples have been achieved by adopting predictive models to support the SLA planning for network and web service transactions. Here, we suggest an approach that helps to define SLA clauses for SOA databases performance, which is disregarded by the existent alternatives. Before that, however, it is essential to clarify the main concepts related with our proposal, like those presented in the following.

3.2 Generalized Stochastic Petri Nets - GSPN

Among the several extensions of the Petri Nets, the timed ones [Merlin and Farber 1976; Murata 1989] are shown to be powerful for the modeling of time-dependent process, as communication protocols, systems performance, hardware design and so on.

For the cases where the explored timed processes demand a non-deterministic representation, an efficient alternative is to adopt GSPN (Generalized Stochastic Petri Nets [Marsan et al. 1995]) extensions, which associate timed and non-timed (immediate) transitions, in a way that the time is represented through random variables [Marsan et al. 1984].

The GSPN are extensively used, specially for systems performance evaluations since, in these cases, the analyzed behaviors are naturally stochastic. The model proposed in this work, was built and simulated through GSPN structures. Alternatively, extensions like CPN - Coloured Petri Nets [Jensen 1997] could be considered. However, CPN express the time through integer values, handled directly in the model, by the designer. Meanwhile, in GSPN it is implicit into continuous variables, which is much more efficient and useful.

In a GSPN structure, these variables are represented by timed transitions, with exponential distribution. Nevertheless, there may be modeling situations that require representing non exponential behaviors. Thus, it is essential to discuss the main distributions that a random variable can assume. In the GSPN model, it is represented by combining arrangements of exponential transitions, as discussed in the following.

3.3 Probability Mass Functions - PMF

Assuming a discrete scope, a Probability Mass Function (PMF) is a stochastic function that associates each random variable with each one of its possibly assumed values [Cassandras and Lafortune 2008; Jain 1991]. Among the most common distributions are the Binomial, Geometric, Poisson, Erlang, Hyper-exponential, Hypo-exponential and Exponential, where the last four mentioned ones, are depicted in Fig. 1 [Desrochers and Al'Jaar 1995].

In the following, we briefly discuss each of them, using average (μ) and standard deviation (σ)

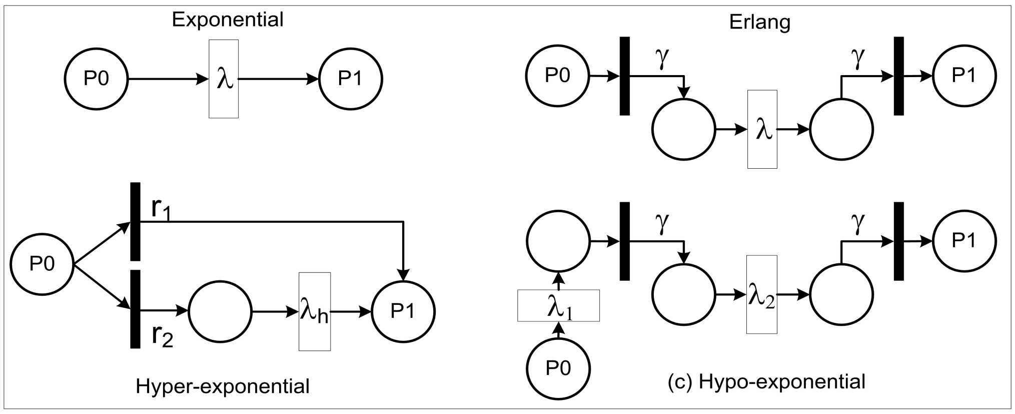

Fig. 1. Example of Probability Distributions

parameters to characterize them.

3.3.1 *Exponential.* This distribution is indicated for situations where:

$$\frac{\mu}{\sigma} = 1.$$

In GSPN, representing an exponential distribution requires assigning μ as the temporal parameter (λ) for a timed transition.

3.3.2 *Hyper-Exponential.* It is commonly used in a behavioral situation where:

$$\frac{\mu}{\sigma} < 1.$$

Thus, a GSPN Hyper-Exponential structure receives the following parameters:

$$\lambda_h = \frac{2\mu}{(\mu^2 + \sigma^2)}; \qquad r_1 = \frac{2\mu^2}{(\mu^2 + \sigma^2)}; \qquad r_2 = 1 - r_1.$$

3.3.3 *Erlang.* Erlang consists in a special case of an exponential distribution, that is triggered several times. Usually an Erlang is adopted for the cases whose behavior is characterized by:

$$\frac{\mu}{\sigma} \in \mathbb{Z} \,\wedge\, \frac{\mu}{\sigma} \neq 1.$$

The parameters of this distributions are as follows:

$$\gamma = (\frac{\mu}{\sigma})^2; \qquad \lambda = (\frac{\gamma}{\mu}).$$

3.3.4 *Hypo-exponential.* This distribution is particularly important for this work, since it is used for conduct the case study. A Hypo-exponential distribution is normally adopted to represent behaviors where:

$$\frac{\mu}{\sigma} > 1 \,\wedge\, \frac{\mu}{\sigma} \neq \mathbb{Z}.$$

Its parameters are obtained according to the following equations:

$$\left(\tfrac{\mu}{\sigma}\right)^2 - 1 \leq \gamma < \left(\tfrac{\mu}{\sigma}\right)^2; \qquad \lambda_1 = \frac{\gamma + 1}{\mu \mp \sqrt{\gamma(\gamma+1)\sigma^2 - \gamma\mu^2}}; \qquad \lambda_2 = \frac{\gamma + 1}{\gamma\mu \pm \sqrt{\gamma(\gamma+1)\sigma^2 - \gamma\mu^2}}.$$

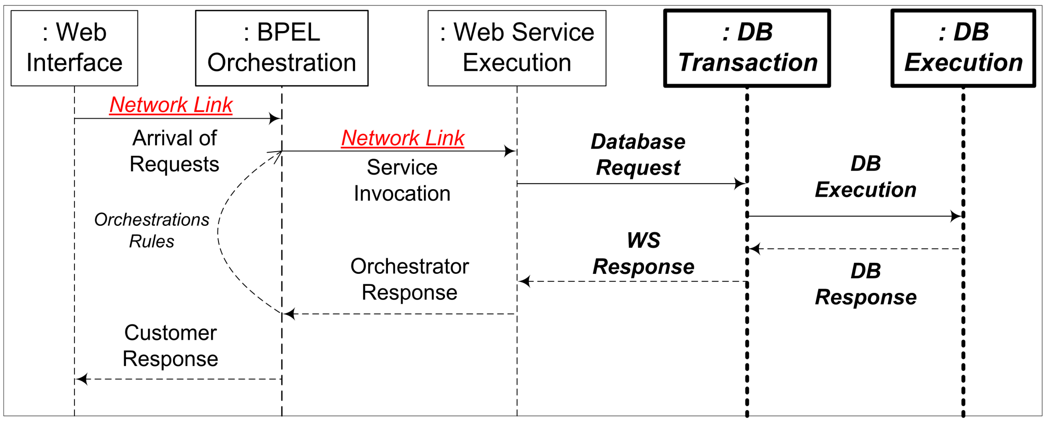

Fig. 2. Typical SOA Process Interaction

As can be seen, a Hypo-exponential function matches Exponential and Erlang distributions. Observe that λ_1 represents an exponential approximation, while λ_2 maps an Erlang distribution, since the γ parameter is used as a sequencer of Exponential triggers.

4. DATABASES PERFORMANCE MODEL

This section describes our proposal for modeling and performance evaluation of DB performing in Service-oriented environments. We are considering the flow of requests since their departure from the web service to the DB server, until a response is received back, by the web service. Alternatively, additional analysis of the SOA process could be associated to the present proposal, covering network links, orchestrations engine and web service evaluations [Teixeira et al. 2009; 2010].

The sequence diagram presented in Fig. 2 identifies our workspace within a SOA-aware scenario.

Usually, a web interface is used to compose packages sent from remote users to a BPEL server. After received, the messages are orchestrated by the BPEL's engine, that invokes several distributed WS iteratively, until a response be sent to requestors. Usually, web services operations require databases transactions (in bold), whose performance evaluation is our goal.

Based on the scenario from Fig. 2, we identify which ranks of activities and devices are responsible for time consumption in a DB transaction, such that the following delays are considered:

—Buffering: it is responsible for storing messages, before and after the DB execution;

—Parsing: corresponds to the stage of validation, syntactical and semantical, of the queries received for processing into DB server;

—Execution: consists in to perform the validated code. This stage involves accessing the data relations, building a package that answers each request.

From a stochastic point of view, *Parsing* and *Execution* stages could be evaluate together, since the largest portion of time is consumed by the "execution" rank. Hence, in Fig. 3 we present the proposed general structure to map the temporal behavior of DB transactions. The models building embodies the GSPN (Generalized Stochastic Petri Nets) formalism as modeling technique and composition rules among them. Thus, it can be simulated through specific tools (e.g. TimeNet [Zimmermann 2011]) and performance metrics can be obtained.

Table I presents the main notation for the model. Timely, we shall discuss how to assign the correspondent parameters (see in Subsection 4.1).

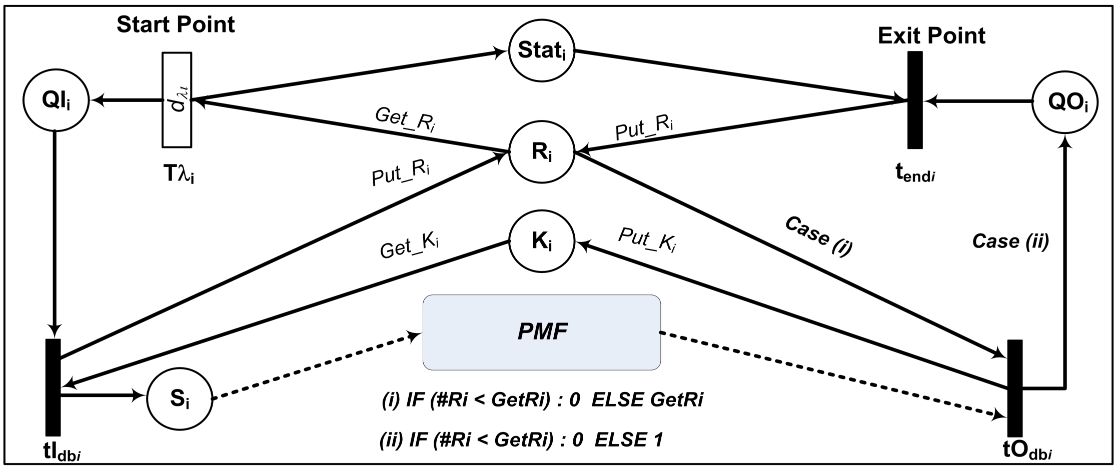

Fig. 3. Proposed GSPN structure for Databases Performance Evaluation.

Table I. Notation for the Proposed Model.

Notation		Description
Places	$Stat_i$	Used only for estimates purposes. It holds the tokens from the start to the exit point;
	R_i	Contains the available resources for input queue (size in bytes);
	K_i	Contains the available execution resources (parallel processing);
	S_i	Contains the requests waiting for processing;
	QI_i	Contains the requests that are being processed;
	QO_i	Contains the requests after processing;
Transitions	$T\lambda_i$	Simulates the arrival of requests in the system;
	tI_{dbi}	Represents the DB processing Input;
	tO_{dbi}	Represents the DB processing output;
	t_{endi}	Simulates the process exit point.

Let i be the indexer of the evaluated DB[1]. Basically, the model starts by firing the timed transition T_{λ_i}, according to rate $1/d_{\lambda_i}$, resulting from the delay d_{λ_i}. The fired requests are inserted into an input queue QI_i, bounded by the number of resources in the place R_i. According to the availability of resources in the place K_i, the transition tI_{dbi} fires, inserting the request into the place S_i, which contains the requests that are being executed in the DB. Hence, the tokens remain in S_i according to the adopted *Probability Mass Function - PMF* (gray block), whose choice is supported by measured DB statistics (see Section 3). Timely we turn to discuss the PMF selection (See Section 5).

After performing, the requests are inserted into the output queue QO_i, which represents also the model exit point. The supported number of requests in QO_i is coordinated by the availability of resources in R_i place. The weights of input and output arcs, from/to places R_i and K_i, express the impact caused by inserting and removing resources in each repository. This impact is conservative, that is, the number of removed and returned tokens is the same, for each request.

Notice that by sharing R_i with QI_i and QO_i places, it is possible a model deadlock. Supposing, for example, a situation where QI_i consumes all the resources from R_i. After perform, the requests can not be sent to QO_i, because there is no more resources for that. For the same reason, $T\lambda_i$ can not trigger requests toward QI_i. Therefore, the system is blocked.

In order to avoid that, we assign conditions for the arcs *Case (i)* and *Case (ii)*. Through *Case (i)* we avoid the deadlock by firing of tO_{dbi}, even if there is no enough resources in R_i. When it happens,

[1]Several DBs could be concurrently evaluated in a Web Service Composition

Case (ii) assigns 0 for the arc that leads to QO_i, discarding definitively the request.

4.1 Model Parameters

In order to simulate the proposed model, it is necessary to feed its input structures. In the following, we presented a practical manner to do that.

4.1.1 *Database Queue Parameters.* Let R_i be the available resources for the input and output DB queues occupation. The marking[2] of R_i is defined according to the buffer size, measured in the real DB system. In practice, each DBMS defines their own parameters for the amount of memory available for DB operations. These parameters are flexible and can be changed according to the system management rules. The most important parameters to be collected from DBMS are:

(1) Number of Memory Pages (NMP) (integer): refers to the available number of memory blocks destined to serve the DB operations[3];
(2) Size of the Memory Pages (SMP) (bytes): represents the amount of bytes assigned to each NMP.

From $M = NMP * SMP$, we have the amount of memory (M) available for storing input and output messages from/to DB system. Therefore, the marking of R_i is such that $\#R_i = M$. In addition, one need to establish the impact caused by the arcs from/to R_i. For that, we assign weights to the arcs (a) Get_R_i and (b) Put_R_i, according to the mean size of exchanged messages (bytes), such that:

$$(a)\ \ Get\_R_i = Msg_{arr}; \qquad\qquad (b)\ \ Put\_R_i = Msg_{dep},$$

where Msg_{arr} and Msg_{dep} are the measured mean size of messages arriving/departing, respectively, in/from DB system. Finally, the assigned parameters allow us to estimate the DB Queue Response Time (QRT), through the following equation:

$$QRT = \frac{E(QI_i) + E(QO_i)}{\lambda_i},$$

where, for $j = QI_i, QO_i$, $E(j)$ corresponds to the expectation of tokens in the place j and λ_i is the arrival rate of requests in the DB i. Observe that λ_i results from $1/d_{\lambda_i}$ and d_{λ_i} is the parameter of the transition $T\lambda_i$. The QRT represents the overall time spent by messages waiting before and after their processing.

4.1.2 *Database Execution Parameters.* Modeling DB processing operations encompasses requests since their arrival, in the S_i place, until that transition tO_{dbi} fires. Therefore, the adopted PMF makes part of the DB execution modeling so as K_i, that contains the available resources for DB processing. In the following we show how to calculate these parameters.

4.1.2.1 *Resources for Processing.* In order to establish the marking of K_i, it is necessary to measure the DB system (or its prototype when it is not available for). Specifically, one must collect the maximum number of parallel operations supported by the DB, without causing queues on the system.

Through gradually increasing the workload level, we observe the point where the queue appears on DB system. It is detected by an increase in the response time when the workload overcomes the resources available for processing. Thus, $\#K_i$ receives the value of the workload applied before observing the first signals of queue. The weight of the arcs Get_K_i and Put_K_i is 1, since their source place (K_i) contains requests and, therefore, each performed request has impact 1 on its resources repository.

[2]"#" is used to reference the number of tokens in the place p ($\#p \in \mathbb{N}$).
[3]Large database pages benefits database performance, usually decreasing I/O time.

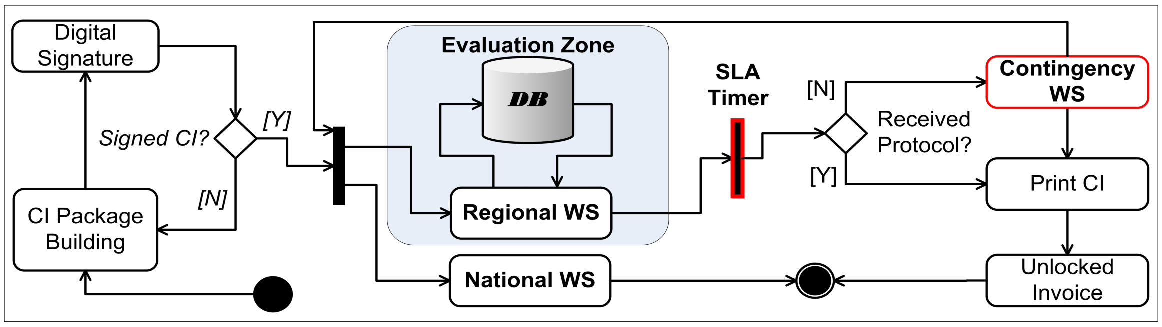

Fig. 4. Activity Diagram of Evaluated Process

4.1.2.2 *Processing Response Time.* These assigned parameters enable us to estimate the DB Processing Response Time (PRT), through the following equation:

$$PRT = \frac{E(S_i) + E(PMF)}{\lambda_i},$$

where, $E(S_i)$ is the expectation of tokens in the place S_i and $E(PMF)$ is the expectation of tokens in all the places contained in the PMF structure. Finally, λ_i is the arrival rate of requests in DB system i. The value of PRT represents the amount of time spent when processing DB operations.

4.1.3 *Overall Database Response Time.* Finally, one can estimate the overall DB Response Time (DBRT), including queues and processing time, according to one of the following equations:

$$DBRT = \frac{E(Stat_i)}{\lambda_i}, \quad \textit{or even} \quad DBRT = QRT + PRT,$$

where $E(Stat_i)$ contains the expectation of tokens in the place $Stat_i$, which does not exert a functional role, but it is used for storing the overall process expectation. Hence, the value of $DBRT$ means the amount of time spent by a DB transaction, considering all its execution stages.

Summarizing, the proposed DB performance evaluation starts by receiving a set of parameters, without including any evidence of queue. As a result, we inform the variable behavior assumed by the DB system, insofar as the workload level increases. For that, of course, the GSPN must be simulated, as presented in the following.

5. CASE STUDY

We developed a case study in order to validate our approach. For that, we explore a DB system, that makes part of a SOA application, implemented into the scope of this work. This application represents a real process, currently used in practice by Brazilian government for the issuance and the management of electronic invoices. It was necessary to propose our own implementation due to the impossibility of accessing the real DB, from the federal agency, since the stored data are confidential.

Even so, our application covers the usual SOA features, since its workflow is orchestrated by using BPEL language [Oasis 2011], its services are deployed on remote servers and interfaced with external users, through the World Wide Web. These users order different types of distributed operations, generating a random system workload. The developed SOA process is depicted in Fig. 4.

Basically, for each finalized sell, a commercial invoice is transmitted in parallel toward a regional and national web services. For now, our scope covers the regional operations, although it could involve additional web services (reason for the notation i, in the proposed model). The *Evaluation zone* contains the DB system that receives the regional transactions. After recording each remittance and replying a response message, the merchandises are released to transport or transfer.

Otherwise, if the request does not respond timely, according to the SLA definitions, a contingency operation is invoked. In this case, the overall system response time tends to be increased, disagreements with service commitment are susceptible to occur and the data security is affected, since the remittance must be retransmitted posteriorly. Besides this, any problem delaying the process directly affects customers and/or inspection agencies, that usually are waiting for the invoices validation.

For these reasons, it is essential that a contractual structure is preventing eventual service interruptions or even delays longer than the expected to answer, defining the penalties for possible interventions of the contingency solution. By providing juridical validity for web electronic transactions, it is reasonable to supply technological support for the lawyer's minds.

Our goal, in this work, is focused on investigating how to establish and accomplish legal rules for SOA DB transactions. For the described process, we estimate the mean spent time for receiving a response from the invoices transmitters and its degradation insofar as the workload increases. It helps to acquire and provide electronic DB operations with QoS guarantees. Moreover, it avoids to develop a SOA system and, only then, to observe unexpected low performances, which certainly has already consumed money, time, employees, etc.

5.1 Database Structure and Primary Measurements

We are interested on to estimate the performance of DB requests, through using the proposed GSPN model. But this model receives a set of input parameters, obtained from the DB system. So, for the particular case, the DB structure was built over a *Java DB* technology[4], such that the experiments posteriorly performed, require only partially its relations, involving the following tables:

—*Products*: containing more than 700 records;
—*Clients*: with more than 500 records;
—*Invoices*: currently, it includes more than 10.000 received records;
—*MovInvoices*: it also contains more than 10.000 records.

In the sequence, is presented an example of a problem, whose solution is achieved through bringing information from the DB recorded data.

PROBLEM 1. *Given the previously described DB system, implement a query that returns all the clients and their respective negotiated invoices, admitting the following requirements:*
(i) the merchandises were already shipped;
(ii) the deadline for the payment is in, at most, one month.
Besides this, sort the results by the invoice deadline.

Let us assume the following SQL query, in order to resolve problem 1:

```
select *
from CLIENTS, INVOICES, MOV_INVOICES, PRODUCTS
where INVOICES.Clients_IdClient           = CLIENTS.idClients
and INVOICES.IdInvoices                   = MOV_INVOICES.Invoices_IdInvoices
and MOV_INVOICES.Products_IdProducts      = PRODUCTS.IdProduct
and INVOICES.Shipment_Date               <= 'Informed current date'
and INVOICES.Deadline                    <= 'Informed Limit date for payment'
```

[4]DB Java: http://www.oracle.com/technetwork/java/javadb

Table II. Model Input Parameters.

Type of collected Parameter							
DBMS Configuration		**Buffering**			**Processing**		
NMP	SMP	$\#R_i$ (bytes)	Get_R_i	Put_R_i	$\#K_i$ (req)	Get_K_i	Put_K_i
1000	$4(KB) = 4096(B)$	4096000	1022	1022	2	1	1

order by INVOICES.Deadline

By the sake of convenience, we are evaluating a single query, although it does not make difference if considering more than one. Actually, by receiving a set of initial measurements and DBMS parameters, issues involving DB technology, type of performed operation, access policies, etc., are irrelevant for the estimates, since they are absorbed by each particular set of collected parameters. It enables us to generically apply our proposal over any size and type of DB and different operation as well, without compromising its accuracy.

Then, using the proposed code, we obtain the parameters to feed the GSPN model, by collecting the DB statistics when answering the mentioned query. Through JMeter tool[5], an Apache software designed mainly for workload generation and performance evaluation, we build a test plan that performs it repeatedly.

In JMeter, we gradually increase the workload of requests. However, notice that the model input parameters, should not include queue time. This is part of the dynamic system behavior and should be estimated through subsequent simulations. Instead, when measuring, we impose the highest possible workload level, before observing evidences of queue appearing in the system. It is detected by a perceptible increases of the response time, when the workload overcome the systems resources. Hence, we collected the parameters presented in Table II, complemented by those information from the DBMS.

First two columns bring the DBMS configuration parameters, where NMP represents the number of memory pages available for database operations, whose size[6] is defined by SMP.

Columns labeled *Buffering*, show the input and output queue resources, responsible for storing the database requests before and after their execution, which is supplied by the parameters presented in *Processing* columns. For clarifying the notations meaning, see the model description in Section 4.

The presented parameters are not the only ones necessary for allowing the GSPN simulations. A probability function must be also defined, as discussed in the following.

5.2 Establishing and Feeding the PMF

Assuming the proposed GSPN topology for performance evaluation of DB systems (see Fig. 3), notice that input and output memory spaces (places QI_i and QO_i) mediate the access to the DB core S_i. These buffers express the DB extra-execution spent time.

Modeling the DB processing time, however, requires to chose a PMF that better fits with the real system behavior. This decision is made during the initial measurements, when the model input parameters are being collected. Once chosen, the same PMF can be used along all the simulation process, since it should stochastically follow the changes of the DB dynamic behavior.

The calculation of an appropriate PMF, requires the average - μ and standard deviation - σ of the DB requests response time. Usually, these metrics are only defined after collecting a set of DB transactions, containing an enough number of samples that allows observing a tendency to a stationary behavior.

[5]JMeter: `http://jakarta.apache.org/site/news/news-2011-q3.html`
[6]The used DB automatically tunes the database page size, although it can be changed.

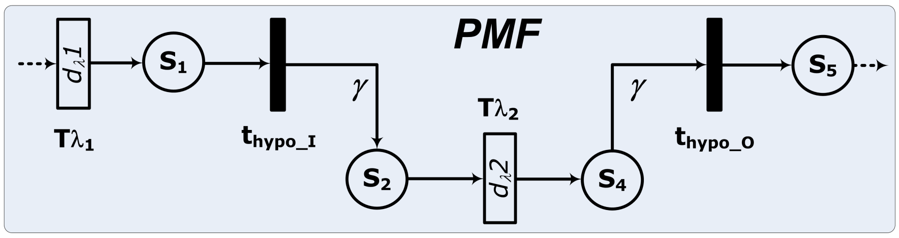

Fig. 5. Adopted Hypo-exponential distribution

For the particular case, for example, when JMeter was requesting around the hundredth sample, it was already possible to observe a stationary condition. Then, the values assigned for μ and σ are those from (a) and (b) as below. Equation (c) defines how establishing the coefficient of variation - Π, with result in (d):

$$(a)\ \ \mu = 36; \qquad (b)\ \ \sigma = 17; \qquad (c)\ \ \Pi = \frac{\sigma}{\mu}; \qquad (d)\ \ \Pi = 0,47.$$

Depending on the value assumed by Π, one can identify the appropriated PMF that would better represent each type of stochastic process. For the particular example, $\Pi = 0,47$. Then, as discussed in Section 3.3, whenever $\Pi < 1$ it is suggested adopting a Hypo-exponential distribution, whose structure is depicted in Fig. 5. Therefore, this GSPN block takes the place of the gray box in the initial model, presented in Section 4, Fig. 3.

A Hypo-exponential probability function matches Exponential and Erlang distributions. In fact, $T\lambda_1$ represents an exponential approximation, while $T\lambda_2$ maps an Erlang distribution, since the γ parameter is used as a sequencer of Exponential triggers, both with delay $d_\lambda 2$.

Supported by the formulation discussed in Section 3.3 and variables μ and σ previously presented, we establish the Hypo-exponential parameters, as follows:

$$d_\lambda 1 = 12; \qquad d_\lambda 2 = 23; \qquad \gamma = 3.$$

At this moment we have the necessary informations to proceed with the GSPN simulations, providing organizational informations and planning SLA for DB transactions, as discussed in the following.

5.3 Defining SLA clauses for DB systems

Let us start supposing usual situations, faced when defining SLA clauses for SOA-based systems. Assuming, for example, that it is necessary to answer the following question.

Question 1*: For the particular DB system discussed in this section, let W be a predefined workload level of requests arriving at DB server (requests per second). Which SLA, for the DB mean response time, could be guaranteed in practice*?

In this case, a typical range of workload levels (variating W) is known, but the response times for the DB requests under these workloads, are quite variables and difficult to predict.

In a similar way, let us suppose that a performance engineer is challenged to answer the following question, to the company's legal department, in order to elaborate contractual partnership clauses.

Question 2*: For the same discussed DB system, let RT be an established SLA for the response time (in milliseconds) of a particular DB operation. Which SLA, for the higher supported workload, could be guaranteed in practice, such that the mentioned RT is not exceeded*?

Table III. Comparison between Simulated and Measured Response Times

Applied Workload Level (req/sec)														
1	2	3	4	5	6	7	8	9	10	20	30	40	50	100
Estimated Response Time (s)														
0,06	0,12	0,18	0,29	0,34	0,41	0,51	0,65	0,78	0,94	1,18	1,59	2,17	3,84	8,16
Measured Response Time (s)														
0,04	0,08	0,20	0,25	0,38	0,46	0,56	0,68	0,84	0,92	1,62	2,22	3,51	4,50	9,08
Closeness Percentage between Estimations and Measurements														
62%	67%	89%	86%	90%	89%	90%	97%	93%	98%	73%	72%	62%	85%	90%

In this case, the service supplier is committed to deliver each request with response time no less than the clause RT. But, certainly, the RT degrades when the workload increases and, in SOA, it is unexpected. So, it is essential to know which number of requests per second could be received by the application, such that RT is kept on track in the SLA.

The capability for answering questions 1 and 2 is one of the keys for elaborating realistic and safer service contracts. In practice, it is difficult to find ways to do so. Usually, the performance engineer waits for the system implementation and, then, for historical execution traces in order to supply those information and, finally, the lawyers could plan appropriated SLA. It really could take years. In the following we present an alternative to quickly and efficiently answer these questions.

5.4 Model Simulations

We use the statistical data from Table II and the PMF parameters previously discussed, to feed the input structures of the GSPN depicted in Fig. 3. As output, we intend to estimate the DB variable behavior, including queue occupation, processing performance degradation and so on, when increasing the workload level of requests. For that, we conduct GSPN simulations using *TimeNet* tool[7] [Zimmermann 2011], considering a confidence level in 95% and relative error of 10%.

For the sake of convenience, we establish the workload levels (in requests per second - *req/sec*), to be used during the experiments. We start by applying 1 req/sec and gradually increase until achieving 10 req/sec. After that, we turn to increase it from 10 to 10, until 50. Finally, in order to verify the accuracy of the proposed approach under a more extreme variability condition, we simulate the model by applying 100 req/sec.

We also measure the implemented DB system, with the purpose of comparing the collected real samples, against the estimates provided by our model. This comparative analysis represents a way to validate our approach. The workloads levels used for simulations and measurements were the same and the obtained results are presented in Table III.

First line presents the workload level (number of requests per second) applied on the experiments, which is achieved by properly variating the delay $d\lambda_i$ of the timed transition $T\lambda_i$ in the GSPN model. *Estimated* and *Measured* response times, bring the results achieved by simulating the proposed model and by measuring the real DB system, respectively.

5.4.1 *Discussing the achieved results.*

As can be seen, according to the workload increases, the system becomes less deterministic due to presence of queues. Nevertheless, the response times calculated through the proposed approach, maintain their accuracy in relation to the measures taken from the real DB application, whose percentage may be checked in the last line (*Closeness Percentage*). Fig. 6 helps to graphically analyze the results from Table III.

[7] *TimeNet - Timed Net Evaluation Tool* is a software for the modeling and analysis of stochastic Petri nets with non-exponentially distributed firing times.

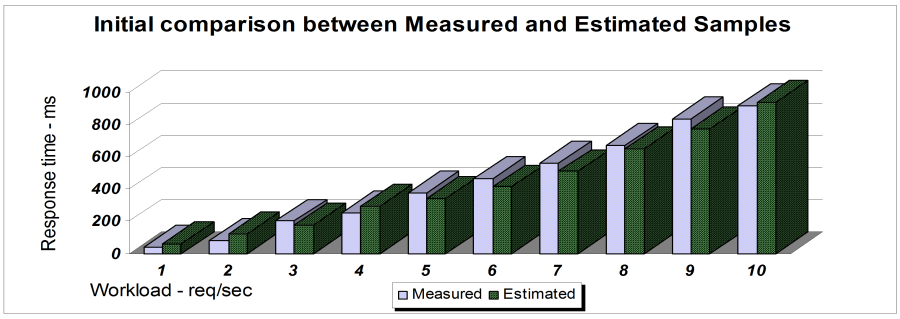

Fig. 6. Analysis of the Measured and Estimated Database Behaviors

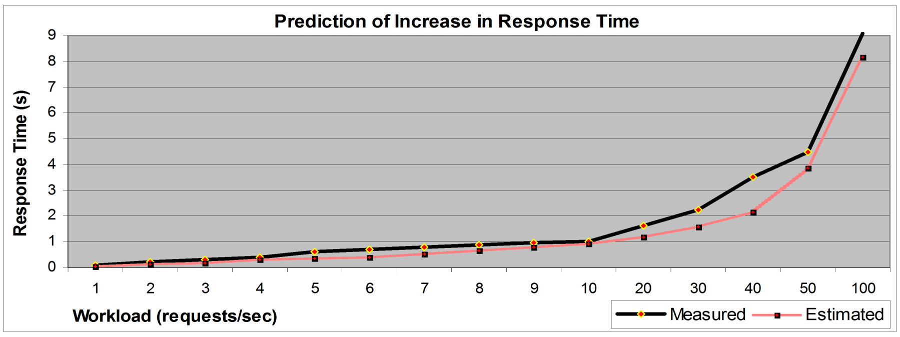

Fig. 7. General Closeness between Measured and Estimated Response Times

Firstly, let us discuss the statistics from the first ten levels of workload, which were increased individually. A comparison between estimated and measured results, as presented in Fig. 6, leads us to recognize that our approach remains close to the measured samples, whatever the existent variability. For these results, the accuracy is on the order of 86%. We also graphically analyze the Table III as a whole, as shows Fig. 7.

One can observe that, even increasing the workload, our estimates follow the behavior of the real measurements, which indicates that a more extreme variability does not affect our predictions. In a general case, the closeness observed between measured and estimated results is on the order of 83%, which certainly is reasonable from a stochastic point of view. It is also essential to notice that, for several times, the curves tend to intercept to each other. This fact indicates that nowhere our model lost the tracing of the real evaluated behavior.

5.5 Composing the SLA

Given the SLA requirements, defined in the subsection 5.3, it is opportune to analyze the range of information available for answering the questions 1 and 2, after obtained the results from Table III.

For question 1, let us suppose that we know the mean workload usually imposed to the system. Let us assume $W = 50$, for example. It is easy to note that for this workload level, the DB system would take 4436 ms to reply each request, while we have estimated a response time of 3842 ms. Although we have not provided an exact estimative, which was not our proposal, certainly it contributes for the SLA planning since, in practice, this difference should be absorbed by an usual adopted margin of error. Moreover, our estimative is on the order of 85% closed with the measured sample, which surely is stochastically acceptable.

In a inverse way, suppose that we know the minimum response time for each DB requests, which is defined in SLA, as introduced by question 2. So, let us assume $RT = 900\ ms$, for instance. It is expected to establish a SLA for the maximum supported workload, such that the RT is not exceeded. According to the Table III, it is easy to show that 9 req/sec is the higher supported workload that satisfies the assumed assumptions. The estimated mean response time, in this case, is 778 ms while the measured one is 835 ms, with accuracy between them on the order of 93%.

6. CONCLUSIONS

In this article, we proposed a stochastic approach for managing database service requirements, in SOA-based systems. By predicting the performance of DB transactions upon the variation of the workload, we provided support for planning suitable SLA for the response time, considering a range of possible arrival rates. In a inverse way, we also established SLA for the highest workload supported by a DB system, without overcome a previously agreed response time. These are only examples of possible information mined from our approach.

In order to illustrate our contributions, we compared the simulated results against those measured from a real DB implementation. The comparison shows that our estimates remain close with the measured samples, whatever the existent variability. Moreover, we established two examples of SLA requirements and, based on our estimates, we identified possible solutions for them. We remember that these analysis were conducted without requiring real-time measures. However, by focusing on representing the dynamic system behavior, we provide support at design-time and/or run-time, which contributes to avoid legal issues with SLA compliance.

Prospects of future works aim to extend the performance model, inserting a timeout mechanism. So, would be possible to register the DB failure rate, establishing availability metrics. Moreover, by crossing failure and performance metrics, we could discover possible bottlenecks delaying DB transactions, so planning structural upgrades, access policies, load balancing and so one.

REFERENCES

Adams, E. J. Workload models for dbms performance evaluation. In *Proceedings of the ACM Annual Conference on Computer Science*. New York, NY, USA, pp. 185–195, 1985.

Baresi, L. and Guinea, S. A dynamic and reactive approach to the supervision of bpel processes. In *Proceedings of the Annual India Software Engineering Conference*. Hyderabad, India, pp. 39–48, 2008.

Casati, F., Shan, E., Dayal, U., and Shan, M. Business-oriented management of web services. *Communications of the ACM* 46 (10): 55–60, 2003.

Cassandras, C. G. and Lafortune, S. *Introduction to Discrete Event Systems*. Springer Science, New York, 2008.

Chase, J. S., Anderson, D. C., Thakar, P. N., Vahdat, A. M., and Doyle, R. P. Managing energy and server resources in hosting centers. In *Proceedings of the ACM Symposium on Operating Systems Principles*. Alberta, Canada, pp. 103–116, 2001.

Desrochers, A. A. and Al'Jaar, R. *Applications of Petri Nets em Manufacturing Systems: Modeling, Control and Performance Analysis*. IEEE Press, 1995.

Dewitt, D. J. and Gray, J. Parallel database systems: the future of high performance database systems. *Communications of the ACM* 35 (6): 85–98, 1992.

Elhardt, K. and Bayer, R. A database cache for high performance and fast restart in database systems. *ACM Transactions on Database Systems* 9 (4): 503–525, 1984.

Jain, R. *Art of Computer Systems Performance Analysis Techniques For Experimental Design Measurements Simulation And Modeling*. John Wiley & Sons, Inc., New York, 1991.

Jensen, K. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use*. Vol. 1. Springer-Verlag, Berlin, 1997.

Josuttis, N. M. *SOA in practice*. O'reilly, 2008.

Kim, S., Son, S., and Stankovic, J. Performance evaluation on a real-time database. In *Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium*. San Jose, CA, pp. 253–265, 2002.

Krompass, S., Scholz, A., cezara Albutiu, M., Kuno, H. A., Wiener, J. L., Dayal, U., and Kemper, A. Quality of service-enabled management of database workloads. *IEEE Data(base) Engineering Bulletin* 31 (1): 20–27, 2008.

Lumb, C. R., Merchant, A., and Alvarez, G. A. Façade: Virtual storage devices with performance guarantees. In *Proceedings of the USENIX Conference on File and Storage Technologies*. Berkeley, CA, USA, pp. 131–144, 2003.

Marsan, M. A., Balbo, G., and Conte, G. A class of generalized stochastic Petri nets for the performance analysis of multiprocessor systems. *ACM Transactions on Computer Systems* 2 (2): 93–122, 1984.

Marsan, M. A., Balbo, G., Conte, G., Donatelli, S., and Franceschinis, G. *Modelling with Generalized Stochastic Petri Nets*. Wiley Series in Parallel Computing - John Wiley and Sons, New York, 1995.

Marsan, M. A., Balbo, G., and et al., G. C. *Modelling with Generalized Stochastic Petri Nets*. Wiley, 1995.

Merlin, P. M. and Farber, D. J. Recoverability of communication protocols: Implications of a theoretical study. *IEEE Transection in Communications* 24 (9): 1036–1043, 1976.

Murata, T. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE* 77 (4): 541–580, 1989.

Nicola, M. and Jarke, M. Performance modeling of distributed and replicated databases. *IEEE Transactions on Knowledge and Data Engineering* 12 (4): 645–672, 2000.

Oasis. *Web Services Business Process Execution Language*. http://www.oasis-open.org/committees/wsbpel, 2011.

Osman, R., Awan, I., and Woodward, M. Performance evaluation of database designs. In *Proceedings of the IEEE International Conference on Advanced Information Networking and Applications*. Perth, Australia, pp. 42–49, 2010.

Pinheiro, P. P. *Direito Digital*. Saraiva, São Paulo, 2009.

Raibulet, C. and Massarelli, M. Managing non-functional aspects in soa through sla. In *Proceedings of the International Conference on Database and Expert Systems Application*. Turin, Italy, pp. 701–705, 2008.

Ranganathan, P., Gharachorloo, K., Adve, S. V., and Barroso, L. A. Performance of database workloads on shared-memory systems with out-of-order processors. *Operating Systems Review* 32 (5): 307–318, 1998.

Reiss, F. R. and Kanungo, T. Satisfying database service level agreements while minimizing cost through storage qos. In *Proceedings of the IEEE International Conference on Services Computing*. Washington DC, USA, pp. 13–21, 2005.

Rud, D., Schmietendorf, A., and Dumke, R. Performance annotated business processes in serviceoriented architectures. *International Journal of Simulation: Systems, Science & Technology. Special Issue on Performance Modelling of Computer Networks, Systems and Services* 8 (3): 61–71, 2007.

Schroeder, B., Harchol-Balter, M., Iyengar, A., and Nahum, E. Achieving class-based qos for transactional workloads. In *Proceedings of the International Conference on Data Engineering*. Washington, DC, USA, pp. 153, 2006.

Sturm, R., Morris, W., and Jander, M. *Foundations of Service Level Management*. Sams Publishing, 2000.

Teixeira, M., Lima, R., Oliveira, C., and Maciel, P. Performance evaluation of service-oriented architecture through stochastic petri nets. In *Proceedings of the International Conference on Systems, Man, and Cybernetics*. Texas, USA, pp. 2831–2836, 2009.

Teixeira, M., Lima, R., Oliveira, C., and Maciel, P. A stochastic model for performance evaluation and bottleneck discovering on soa-based systems. In *Proceedings of the International Conference on Systems, Man, and Cybernetics*. Istanbul, Turkey, pp. 358–365, 2010.

Teixeira, M., Lima, R., Oliveira, C., and Maciel, P. Planning service agreements in soa-based systems through stochastic models. In *Proceedings of the ACM Symposium On Applied Computing*. TaiChung, Taiwan, pp. 1576–1581, 2011.

Tok, W. H. and Bressan, S. Dbnet: A service-oriented database architecture. In *Proceedings of the International Workshop on Database and Expert Systems Applications*. Los Alamitos, CA, USA, pp. 727–731, 2006.

Tomov, N., Dempster, E., Williams, M. H., Burger, A., Taylor, H., King, P. J. B., and Broughton, P. Analytical response time estimation in parallel relational database systems. *Parallel Computing* 30 (2): 249–283, 2004.

Zhou, S., Tomov, N., Williams, M. H., Burger, A., and Taylor, H. Cache modeling in a performance evaluator for parallel database systems. In *Proceedings of the International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*. Haifa, Israel, pp. 46–50, 1997.

Zimmermann, A. *TimeNET 4.0*. Technische Universität Ilmenau, http://www.tu-ilmenau.de/TimeNET, 2011.