Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Performance Evaluation of Service-Oriented Architecture through Stochastic Petri Nets

Marcelo Teixeira, Ricardo Lima, Cesar Oliveira, Paulo Maciel
Center of Informatics
Federal University of Pernambuco
Recife, Brazil
{mt3, rmfl, calo, prmm}@cin.ufpe.br

***Abstract*—The Service-Oriented Architecture (SOA) has become an unifying technical architecture that can be embodied with Web Service technologies, in which the Web Service is thought as a fundamental building block. This paper proposes a simulation modeling approach based on stochastic Petri nets to estimate the performance of SOA applications. Using the proposed model it is possible to predict resource consumption and service levels degradation in scenarios with different compositions and workloads. A case study was conducted to validate the approach and to compare the results against an existing analytical modeling approach.**

***Index Terms*—SOA, BPEL, Capacity Planning.**

I. Introduction

The Service-Oriented Architecture (SOA) emerges as a new paradigm for processes modeling, allowing an efficient development of enterprise applications. SOA has become an unifying technical architecture focused on heterogeneous environments, in which software components execute in platforms with distinct characteristics. Web Service can be thought as the basic building block of SOA.

Despite the benefits of SOA, such as flexibility and interoperability, it is difficult to predict the performance and availability of SOA applications. It has been shown that the performance of SOA-applications can be adversely affected as the workload increases [3][4][2]. Moreover, the number of accesses to a service may be so high at a certain moment that it will become unavailable. The capability for estimating the performance and availability of SOA in scenarios with different workloads is important to predict undesirable behavior [5].

Usually, designers develop the application and subsequently stress it to measure performance metrics. This approach can be expensive and time consuming, since the performance analyst needs an executable version of the system and consumes limited computational resource of the company to analyze the system. Thus, the adoption of simulation or analytical modeling approaches could substantially reduce these costs, being useful to rapidly compare the performance of several design alternatives before implementing them.

In this context, Rud et al. [16][17] presented a model to predict the performance of BPEL processes. The work applies techniques of Operations Research, proposing a monitoring infra-structure that supports its utilization. However, the model relies on deterministic assumptions, which can affect its accuracy for modeling nondeterministic systems.

This paper proposes a simulation modeling approach for estimating the performance of SOA-applications based on Generalized Stochastic Petri Nets (GSPN) [8]. The models were developed to analyze resources consumption and service levels degradation in scenarios with different workloads. With the aim of validating the model and evaluating the accuracy of results, we compared the results obtained using our approach against the values measured from a real application. We also compared our results with that obtained through the model proposed by Rud et al.

The remainder of this paper is organized as follows: Section II describes the related works on capacity planning for SOA; Section III introduces the concepts of SOA and technologies involved, such as the BPEL language; Section IV describes the model proposed in this paper and the model proposed by Rud et al.; Section V presents a case study to compare the results obtained by each model with those obtained from the real application. Finally, the conclusions are discussed in Section VI, along with prospects of future works.

II. Related Works

Several works on SOA capacity planning have been proposed. O'Brien et al. [9] developed a list of Quality of Service (QoS) items which deserve further attention. Each item is characterized and some strategic challenges are proposed. O'Brien et al. [3] also addressed difficulties on establishing and measuring performance issues in SOA. According to them, the specific features of XML, the distributed characteristics of services, and unpredictability of load parameters make performance a critical factor of this architecture.

Another relevant study on performance analysis in SOA environments is described by D'Ambrogio and Bocciarelli [4], who proposed an extension to Web Service Definition Language (WSDL), the "P-WSDL", which provides the description of performance aspects in the language itself. Thus, performance properties of Web Services can easily be obtained, facilitating the monitoring and control of QoS levels.

Moreover, several tools have been proposed for evaluation of QoS in SOA environments. Mozart is a tool which provides an environment for business modeling, with support for syntax and semantic validations [1]. IBM offers a framework for strategic services management [18]. OpTier offers a tool called CoreFirst, which provides control of users and trans-

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

action numbers, as well as the performance and availability of services being offered [13]. The Apache Jakarta Project proposes a tool called JMeter, which provides an environment for workload simulation on various protocols such as HTTP, FTP, SOAP, and others. The central focus of this tool is the ability to conduct experiments with an unlimited number of simultaneous access to services [14].

In fact there are many tools and techniques, both in industry and academy, focused on development and monitoring of QoS items. However, it seems that most initiatives focus on the evaluation of services that already operate in their real environment, or at least have some implementation (code). This occurs due to the difficulty in obtaining data for the modeling of unimplemented systems, based only on high-level specifications. Therefore, it is of great value the development of techniques to check previously the QoS properties, in order to avoid many problems, before the services are put into production, thus affecting people, schedules, resources, and others.

Rud et al. [16], address these issues. They develop an analytical model for predictive analysis of resources consumption in Web Services whose behavior is described through the Business Process Execution Language (BPEL). This model is applied to the optimization of service levels agreement (SLA) process between the involved parties. An updated version of the model addresses the lack of calculation on time spent by messages in the queue of service, also considering their size [17]. However, this model is still mostly deterministic, which does not often match the characteristics of the real system.

Oliveira [12] proposes a model for performance analysis and improvement of workflows, which uses Generalized Stochastic Petri Nets [8]. This model adopts an stochastic approach to represent the fundamental aspects of workflows. However, the work focuses on human-centered workflows and does not addresses Web Service orchestrations created using BPEL. In this paper, we extend the Oliveira's model to support the analysis of resources consumption and service levels degradation in SOA-applications with different workloads.

III. SOA Related Concepts

Our interest in this paper is restricted to computer applications based on service oriented architecture (SOA). This section briefly describes some important concepts connected with SOA. We will concentrate on aspects of interest for the simulation modeling approach proposed in this work.

A. Service-Oriented Architecture (SOA)

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 [7], as follows.

Functionality as services: service is an element of the business, which operates independently from the other components of the system and do not depend on the state of other processes or functions. Normally, the services function is to receive one or more requests, process them, and return their contributions, through an interface. By focusing on the business values, the services are able to establish a strong link between information technology and business. For this reason, the technologies involved in the services definition is not relevant;

Enterprise Service Bus (ESB): the infrastructure capable of providing interoperability between different distributed systems. 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. Thus, the characteristics of ESBs can make implementation complex, because it might require interconnection of different platforms of both hardware and software, different communication protocols, issues of security, reliability, etc.;

Loose coupling: SOA is focused on large distributed systems [7]. This fact makes the scalability and fault-tolerance to be critical factors for system maintainability. Thus, this architecture supports the reduction of dependence between services, in order to avoid that a failure or maintenance affect significantly other services. With this, it is possible to reach a more efficient management of the impact of changes.

Therefore, the use of SOA improves the capability of processes reengineering, allowing business models to be in frequent evolution and providing a reduction in development time. The expectation is that, with the support of SOA, the web will become a distributed business network, reducing the distance between information technology and business [6].

B. Business Process Execution Language (BPEL)

Currently, one of the most popular languages for service orchestration is BPEL [19]. This language enables the construction of service-oriented systems, and includes several facilities such as: the composition of Web Services; setting which operations need be invoked; ordering operations to be executed; exceptions handling; publishing which services are available; and many other essential aspects of the system execution. Some features of BPEL are [15] [11]:

Scope: the primitives of the language have adhesion to the characterization of business processes;

Orchestration: way to shape and reshape the business processes, from a perspective internal to the organization, preparing future services to converge with other services and external customers [10] [7];

Choreography: model business processes that communicate externally, interacting with processes of business partners. It deals with message exchange issues, protocols, and so on;

Structures: offers basic and simple primitives, whose combination support the modeling of complex operations. These structures are described below.

- *Basic, primary or atomics structures*
  - *Invoke* - makes a synchronous or asynchronous request for the execution of an operation by a Web Service;
  - *Receive* - waits for a message to arrive;
  - *Wait* - delays the process by a period of time;
  - *Assign* - assigns a value to a variable;

  - *Reply* - sends a message.
- *Composite structures*
  - *Sequence* - executes a set of operations sequentially;
  - *If* - executes an operation if a boolean condition is satisfied;
  - *Pick* - performs an operation on the occurrence of a given event (blocks until the occurrence of the event);
  - *RepeatUntil/While* - repeats an operation while a condition is hold;
  - *Flow* - performs operations in parallel and synchronize them in the end. A *Link* command can be used inside the *Flow* structure in order to force synchronization between activities at arbitrary points.

In BPEL, services are composed independently of each other through the specification of their interfaces. This increases reuse and maintainability. Moreover, the BPEL execution engine does not need to know any detail about service implementation or execution platform. Therefore, it is portable and modifiable according to organizational needs. By supporting these business trends, BPEL has become a consolidated standard for organizational process modeling and SOA implementation.

IV. Models Descriptions

This section describes our proposal for modeling BPEL processes and the model proposed by Rud et al., which will be used for comparison. The stochastic simulation model proposed in this paper is an extension of a model from previous researches [12], which uses the formalism of Stochastic Petri Nets and consists of a set of basic models and composition rules that allow for the creation of process models to be evaluated. The model is then analyzed through specific tools and the performance metrics are obtained. The analytical model proposed by Rud et al. [16][17] is based on techniques of operations research and consists in a set of closed-form formulas by which the metrics of interest are obtained.

A. Rud's Model

This section describes the basics of the model from Rud et al. [17]

1) Notation: Table I presents the notation used in the Rud's model.

2) Performance Metrics: The performance calculation is defined on the basis of each BPEL structure.

- **Invoke**: $\tau[c] = \tau_{intr} + \tau_{comm}$, where τ_{comm} is the communication time, given by:
  For synchronous activities:
  $$\tau_{comm} = d[n] + \frac{hi[m] + ho[m]}{j[n]} + b[m] \ ;$$
  For assynchronous activity:
  $$\tau_{comm} = d[n] + \frac{hi[m]}{j[n]} \ ;$$
- **Receive and Wait:** $\tau[c] = \tau_{intr} + \tau_{wait}$, where τ_{wait} is the estimated waiting time;

TABLE I
Notation of the Rud's model

N	set of service providers (network nodes)
$S[n]$	set of web services available in $n \in N$
$M[s]$	set of operations provided by web service $s \in S[n]$
$M[n]$	set of operations provided by all services in node $n \in N$: $M[n] = \bigcup_{s \in S[n]} M[s]$
$M[q]$	set of operations that participate in the execution of process q
$hi[m]$	average size of input messages of an operation m
$ho[m]$	average size of output messages of an operation m
$b[m]$	compute time of operation m
$d[n]$	network latency (ping time)
$j[n]$	maximum throughput of the network link between customer and node $n \in N$
τ_{wait}	average time waiting for a message
$prob[i]$	probability of occurrence of event i
τ_{intr}	intrinsic time, which is the time spent by the BPEL engine for processing the referred command (XML parsing, object creation, validation, etc.

- **Reply:**
  $$\tau[c] = \tau_{intr} + d[n] + \frac{ho[m]}{j[n]} \ ;$$
- **Assign, Empty, Throw and Exit:** $\tau[c] = \tau_{intr}$;
- **Sequence:** considering $c_1, \ldots, c_k$ nested activities,
  $$\tau[c] = \tau_{intr} + \sum_{i=1}^{k} \tau[c_i] \ ;$$
- **Switch:** considering $c_1, \ldots, c_k$ branch activities, with choice probabilities $p[c_1], \ldots, p[c_k]$,
  $$\tau[c] = \tau_{intr} + \sum_{i=1}^{k} \tau[c_i] \cdot p[c_i] \ ;$$
- **RepeatUntil e While:**
  $$\tau[c] = \tau_{intr} + \upsilon \cdot \tau_{body} \ ,$$
  where υ is the expected number of iterations executed and τ_{body} is the estimated time for executing the body of the structure;
- **Flow:** considering $c_1, \ldots, c_k$ parallel activities,
  $$\tau[c] = \tau_{intr} + \max(\tau[c_1], \ldots, \tau[c_k]) \ .$$

According to Rud et al. [17], it is possible to estimate an additional waiting time for a process q caused by the time spent in the service's incoming queues. This can be estimated as follows:

$$\tau_{queue}[q, n] = \sum_{s \in S[n]} (a'[s, q] \cdot b[s, q] \cdot mi[s, q]) \ .$$

Where $a'[s, q]$ is the *measured* average number of requests in the incoming queue of service s during the execution of the process q, $b[s, q]$ is the average processing time of operations of the service s and $mi[s, q]$ is the number of incoming messages in the service's buffer which relate to process q. Some more specific details about the calculations can be found in [16] [17].

B. Proposed Stochastic Simulation Model

This section describes the model we propose for performance evaluation of BPEL processes. We employ GSPN (Generalized Stochastic Petri Nets) as the modeling technique.

1) Model Description: The model consists in a set of basic GSPN blocks and composition rules that are used to model a large number of different complex processes. Fig. 1 shows the most basic type of process.

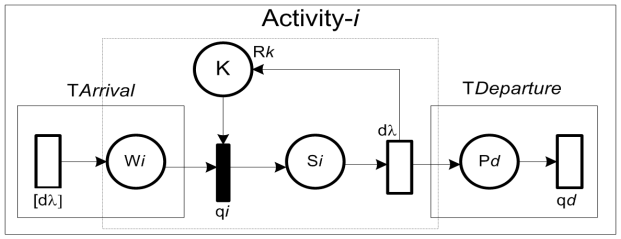

Fig. 1. Basic Model Process

In the *Activity* block, place W represents the queue of customers waiting for service; place S represent requests being executed; place R_k contains the resources that perform the activity. The activity block is composed with the *Case Arrival* block on the left and the *Case Departure* block on the right to model the most basic *Workflow System* (a running workflow containing only a single activity to be executed).

In order to represent particular characteristics of SOA, the model is extended with two new elements:
1) the delay d_r (d_s) required to receive (or send) a message of size m_r (m_s) in a network with bandwidth b ($d_r = m_r/b$; $d_s = m_s/b$);
2) the delay d_{bpel} required by the BPEL engine to delivery the message to the service invoker after process completion. Fig. 2 depict the model with the SOA characteristics.

Delays d_r and d_s are represented as basic activities, which are sequentially composed with the corresponding Web Service. Similarly, the d_{bpel} is represented as a basic activity sequentially composed with the process exit point.

Fig. 3 shows the basic structures of composition: a) Sequence; b) Alternate path (XOR); c) Parallel (AND); d) Iteration. Each one can be applied to compose entire sub-processes (denoted by U). A sub-process can be formed by a single activity or a composition of various activities.

The following notation is used:

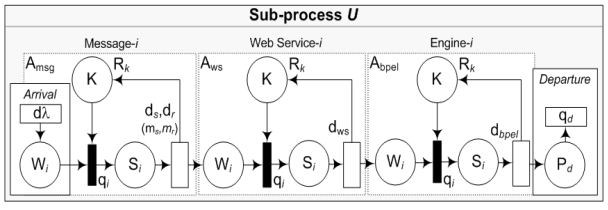

Fig. 2. Extended Basic Model Process

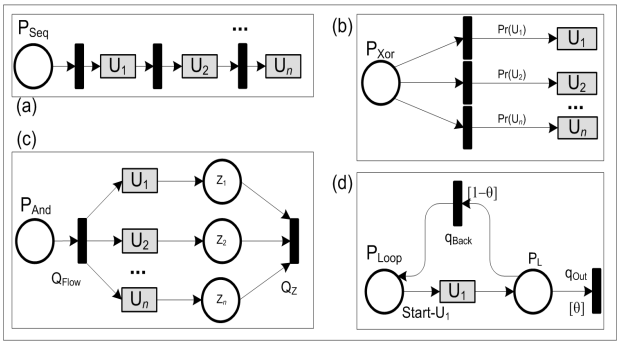

Fig. 3. Process Composition Rules

- K: number of resources in an organizational role;
- d_i: mean time for performing the activity A_i;
- λ_i: arrival rate of process instances to activity A_i;
- $E(n_i)$: average number of instances of process in activity A_i;
- $E(Z_i)$: average number of synchronizing instances in a parallel structure;
- $E(\tau_i)$: expected duration of a sub-process U_i.

2) Performance Metrics: The model provides analytic formulae for obtaining a number of metrics. The simulation of the GSPN is used for obtaining complementary metrics. The metrics for a basic activity are calculated as follows:

- Expected number of activity A_i instances:

$$E(s_i) = d_i \cdot \lambda_i \ ,$$

  where d_i represents the delay of the activity and λ_i is the mean rate to cases arriving in the block (this is independent of the arrival distribution).
- Expected number of customers executing activity A_i:

$$E(n_i) = E(w_i) + E(s_i) \ ,$$

  where $E(w_i)$ is the queue size, obtained from GSPN simulation.
- Mean response time:

$$E(\tau_i) = E(n_i)/\lambda_i \ .$$

The composition of activities causes the metrics to be composed as a consequence. Below, these metric compositions are defined:

Sequence: considering $a_1, \ldots, a_k$ activities in sequence:
Arrival rate: $\lambda_i = \lambda$;
Expected number of instances:

$$E(n_{Seq}) = \sum_{i=1}^{k} E(n_i) \ ;$$

Mean response time: $E(\tau_{Seq}) = E(n_{Seq})/\lambda$.

Xor (BPEL's If): considering $a_1, \ldots, a_k$ branching activities, with probabilities of occurrence given by $Prob(a_i)$,
Arrival rate: $\lambda_i = \lambda \cdot Prob(a_i)$;

Expected number of instances:

$$E(n_{Xor}) = \sum_{i=1}^{k} E(n_i) \ ;$$

Mean response time: $E(\tau_{Xor}) = E(n_{Xor})/\lambda$.

And (BPEL's Flow): considering $a_1, \ldots, a_k$ parallel activities,

Arrival rate: $\lambda_i = \lambda$;

Expected number of instances:

$$E(n_{And}) = \frac{1}{k} \sum_{i=1}^{k} [E(n_i) + E(Z_i)] \ ,$$

where $E(n_i)$ represents the expected number of instances performing in the flow and $E(Z_i)$ the expected number of instances waiting for synchronization;

Mean time for synchronization:

$$E(\tau_\xi) = \frac{1}{k \cdot \lambda} \sum_{i=1}^{k} E(Z_i) \ ;$$

Mean response time including synchronization:

$$E(\tau_{And}) = E(\tau_\xi) + E(n_{And})/\lambda \ .$$

Iteration (BPEL's While and RepeatUntil):

Arrival rate: $\lambda_i = \frac{\lambda}{\theta}$, where θ is the probability of the output condition to be true;

Expected number of instances: $E(n_{Loop}) = E(n_1)$;

Mean response time:

$$E(\tau_{Loop}) = \frac{E(n_{Loop})}{\lambda} \ .$$

V. Case Study

In order to validate the proposed model, we developed a real loan processing application which consists of two Web Services sequentially composed to form a process described in BPEL. The first Web Service, called Assessor, is responsible for reviewing the client documentation and then, call the second Web Service, named Approver. The Approver can approve or reject the loan request, providing also the necessary justifications for the decision. Each of these Web Services where also used to create two basic BPEL processes with a single activity, in order to give a sight on how each one behaves when running alone.

To simulate different scenarios, we implemented a random number generator for variables such as amount required, monthly salary, and age. The applications are both simple to facilitate the understanding of the performance analysis and general enough to represent characteristics found in most SOA applications.

We used the JMeter tool [14] for generating the workload and measuring the execution time of each Web Service (see Table II). These measures neither include the influence of the queue nor the time consumed with the BPEL engine. The latter is predominantly the time required to send and receive messages. Thus, it can be easily calculated from the message size and the network bandwidth. Table II also presents the messages sizes handled by each Web Service. We used the statistical data in Table II as parameter to the models. We adopted exponential distributions to represent the response times in the model, because it is the natural distribution for modeling time in GSPN. If the results were not satisfactory, we could refinate the model in order to match the real distributions measured. As will be shown along this section, the results with exponential distributions were satisfactory.

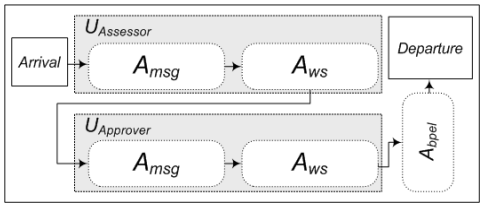

Fig. 4. Experiment structure

After the real data has been collected, the GSPN for each application was modeled, receiving as input the data depicted in Table II. Fig. 4 depicts how the GSPN blocks were composed to assembly the Loan BPEL model. The GSPNs were simulated with a level of confidence of 95%.

Similarly, the data in Table II was used to calculate the performance prediction by employing the analytical model from Rud et al.

When conducting the experiments, we gradually increased the workload until observing the point when a queue appear in the system. This can be detected by a significant increase in response time when the workload overcomes that mark. We collected the response time of the entire BPEL processes, which includes BPEL engine computation, in three different scenarios: 1) without causing queues (one customer at a time); 2) workload of 0.005 requests/ms; 3) workload of 0.01

TABLE II
Real Application Measures (ms)

Web Service	*WS*	*Msg Size (Bytes)*
Assessor	2215.00	714.00
Approver	1075.00	718.00

TABLE III
Response time of the BPEL process (ms)

(a) without queue			
Application	*Measured*	*Analytical (Rud et al)*	*Simulation (our)*
Assessor BPEL	3,286.00	2,931.00	2,953.00
Approver BPEL	2,152.00	1,795.00	1,812.00
Loan BPEL	**5,075.00**	**5,092.00**	**5,050.00**
(b) workload: 0.005 requests/ms			
Application	*Measured*	*Analytical (Rud et al)*	*Simulation (our)*
Assessor BPEL	9,511.00	14,957.00	9,189.00
Approver BPEL	3,397.00	13,437.00	4,860.00
Loan BPEL	**11,454.00**	**28,395.00**	**11,589.00**
(c) workload: 0.01 requests/ms			
Application	*Measured*	*Analytical (Rud et al)*	*Simulation (our)*
Assessor BPEL	12,721.00	85,290.00	11,992.00
Approver BPEL	5,697.00	53,750.00	6,393.00
Loan BPEL	**18,242.00**	**139,040.00**	**16,278.00**

requests/ms. These workloads are generated with deterministic distributions.

The "Measured" column of Table III (a), (b), and (c) shows the response time measured for these three scenarios for each application modeled: Assessor BPEL process, Approver BPEL process and Loan BPEL process.

Analyzing the results presented in Table III (a), both the analytical (Rud et al) and GSPN (our) approaches are adherent to values measured in the real application.

When we increase the workloads, the system becomes less deterministic due to the presence of queues. As we can see in Table III (b), and (c), the performance prediction of the analytical model was unable to reproduce the mean times measured in the real applications. On the other hand, the response times calculated through our approach (stochastic) were very close to the mean execution times measured from the three applications.

The analytical approach cannot capture the stochastic behavior when the queue is inserted in the system. It depends on real-time measuring of queue occupations in order to provide an useful estimate. This information is not available at design time. Therefore, it is impossible to feed the model with them before deploying the system. On the other hand, our approach does not require information about the queue to represent the dynamic behavior of the system, which is more realistic and natural.

VI. Conclusions

In this paper, a stochastic performance simulation model for capacity planning in business enviroments was proposed. This model uses the Petri Net formalism to represent the process and estimate its performance. Thus, it can be used to predict the expected durations of new processes, resulting from different service orchestrations. This allows for the optimization of service level agreements between Web Service suppliers and their clients.

To verify the acuracy of the proposed model with other aproaches, with different characteristics, we compared our aproach with a model proposed by Rud et al. [16][17]. Their model is based on techniques of Operations Research and consist in a set of formulas that provide an estimate for the performance of BPEL processes and Web Service utilization. This approach considers the details inherent to distributed environments, such as network speed and capacity, message sizes, network latency, and so on. However, the model uses only averages as input paremeters for performing the estimates. It considers the behavior of the system as being deterministic, ignoring variability of time. Furthermore, the adoption of this model requires the establishment of an infra-structure for the collection of data such as number of incoming messages for a service, current occupation of server queues, queue increasing rate, etc.

The experiments performed to compare the results obtained from the models to that measured from the real application showed that the analytical model can predict the overall average behavior of the real system in optimistic scenarios. However, as the workload increases, this model loses its accuracy. On the other hand, the analysis of the stochastic simulation model proposed in this paper produced accurate results for both scenarios evaluated when compared with the real application.

We plan to conduct further experiments to explore different enviroments, simulating other common characteristics of distributed systems such as network bottleneck, availability of services, performance degradation, which are inquiring aspects of service level negotiation between supplier and its clients.

Acknowledgments

We would like to thank Cenpes, Petrobras, Finepe and CNPq for supporting this research.

References

[1] M. Abe and J. Jeng. Authoring tool for business performance monitoring and control. In *IEEE International Conference on Service-Oriented Computing and Applications*, Newport Beach, California, 2007.

[2] V. A. F. Almeida and D. A. Menasce. *Planejamento de capacidade para servios na web*. Campus, 2003.

[3] L. OBrien; P. Brebner and J. Gray. Business transformation to soa: Aspects of the migration and performance and qos issues. In *ACM International Workshop on Systems Development in SOA Environments*, pages 35–37, Leipzig, Germany, 2008.

[4] A. Dambrogio and P. Bocciarelli. A model-driven approach to describe and predict the performance of composite services. In *ACM Workshop on Software and Performance*, pages 78–80, Buenos Aires, Argentina, 2007.

[5] Y. Liu; I. Gorton and L. Zhu. Performance prediction of service-oriented applications based on an enterprise. In *IEEE Annual International Computer Software and Applications Conference*, Beijing, China, 2007.

[6] Y. Li; J. Shen; J. Shi; W. Shen; Y. Huang and Y. Xu. Multi-model driven collaborative development platform for service-oriented e-business systems. In *Elsevier Advanced Engineering Informatics*, pages 328–329, 2007.

[7] N. M. Josuttis. *SOA in practice*. Oreilly, 1 edition, 2008.

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

[9] L. OBrien; P. Merson and L. Bass. Quality attributes for service-oriented architectures. In *IEEE International Workshop on Systems Development in SOA Environments*, Minneapolis, Minnesota, USA, 2007.

[10] E. Newcomer and G. Lomow. *Understanding SOA with Web Services*. Addison-Wesley, 2004.

[11] Oasis. *Web Services Business Process Execution Language*, 2007. Oasis Standard.

[12] C. A. L. Oliveira. An approach for improvement of workflow based on generalized stochastic petri nets. Master's thesis, University of Pernambuco, 2008.

[13] Optier. *Monitor transactions across multiple tiers*. URL: 'http://www.optier.com/technology', October 2008.

[14] Apache Jakarta Project. *jMeter 2.3.2*. URL:'http://jakarta.apache.org', January 2009.

[15] R. J. Rabelo. *Integration of Systems Corporate*, 2008.

[16] D. Rud; A. Schmietendorf and R. Dumke. Performance modeling of ws-bpel-based web service compositions. In *IEEE Services Computing Workshops*, pages 140–147, Los Alamitos, California, USA, 2006.

[17] D. Rud; M. Kunz; A. Schmietendorf and R. Dumke. Performance analysis in ws-bpel-based infrastructures. In *UK Performance Engineering Workshop*, Edge Hill University, England, 2007.

[18] IBM Tivoli. *Gerenciamento e orquestração de negócios*. URL: 'http://www01.ibm.com/software/br/demos', September 2008.

[19] Y. Vasiliev. *SOA and WS-BPEL. Composing Service-Oriented Solutions with PHP and ActiveBPEL*. Packt Publishing, 2007.