SOASPE: a Framework for the Performance Analysis of Service Oriented Software

Henrique Jorge A. Holanda[1], Giovvani Cordeiro Barroso[2], A. B. Serra [3]

[1]Departamento de Informática UERN, Mossoró, Brazil / Dpto. de Inf. e Ing. de Sistemas, UNIZAR, Zaragoza, Spain

[2]Universidade Federal do Ceará
Campus do Pici, Fortaleza,- CE 60455 760 – Brazil

[3]Centro Federal de Educação Tecnológica do Ceará
Av. 13 de Maio, 2081, Fátima, Fortaleza – CE 60040-531 – Brazil

henriqueholanda@uern.br, gcb@ufc.br, prof.serra@gmail.com

Abstract. *Business Process Execution Language for Web Services (BPEL4WS) is a promising language describing the Service Oriented Software (SOS) orchestrations in form of Business Processes, but it lacks of a sound formal semantic, which hinders the formal analysis and verification of business processes specified in it. Formal methods, like Petri Nets (PN), may provide a means to analyse BPEL4WS processes, evaluating its performance, detecting weaknesses and errors in the process model already at design-time. This paper addresses quality of SOS orchestrations created using the BPEL4WS and a framework for transformation of BPEL4WS into* Generalized Stochastic Petri Nets *(GSPN) is proposed to analise the performance and throughput of SOS, based on the execution of orchestrated processes.*

1. Introduction

A Web Service (WS) is a software application identified by a Uniform Resource Identifier (URI), whose interfaces and bindings are capable of being defined, described and discovered by Extensible Markup Language (XML) artefacts´ and supports direct interactions with other software applications using XML based messages via Internet-based protocols [W3C 2002]. However, individual elementary services can only represent limited business functions, it is necessary and feasible to compose functions offered by different individual services, likely from different Service Providers (SP), into a composite service which is represented as a Business Process (BP) and can provide a more powerful and complex service.

Accordingly, a growing interest is to express a composite service using a Business Process Modeling Language tailored for Web Services. A landscape of these languages such as Business Process Modeling Language (BPML), Business Process Execution Language for Web Services (BPEL4WS, or BPEL), Web Service Choreography Interface (WSCI) and Web Services Choreography Description Language (WSCDL) has emerged and is continuously being enriched [OASIS 2007]. Nevertheless, all these proposals still remain at the descriptive level, without providing any kind of mechanisms or tools support for verifying an evaluation of performance specified in the proposed notations. Modeling and analyzing these proposals with a formal tool becomes critical. Formal analysis and verification techniques can enable

designers to detect performance problems and repair design errors even before actual running of a service process, or verify whether a service process does have certain desired properties, such as reachability, liveness, throughness, and so on.

In this paper, we are motivated by issues related to the definition of a framework for the transformation of WS orchestrating with BPEL4WS into GSPN and this way to evaluate its performance.

A BPEL4WS process implements a Web Service by specifying the interactions with other Web Services, which might be BPEL4WS processes as well, and their causality [OASIS 2007]. For evaluating the performance of Web Services, we assume that we know the BPEL codes and the Probability Distribution Function (PDF) of the response time of individual Service Providers (SP) where the services are executed.

We also assume that the SP´s provide the PDF of their Quality of Service (QoS) metrics. These can be either obtained by the SP´s themselves by analyzing historical data or by external agents that monitor the SP´s at regular intervals and fit the data for a distribution.

This paper is organized as follows. Section 2 gives a review of some related work. The Framework "SOASPE" is defined in Section 3 and its analysis is discussed in Section 4. Finally Section 5 concludes the paper and gives suggestions of future work.

2. Related Work

Software Performance Engineering (SPE) and QoS in the context of Web Service is the subject of many studies.

In [Menascé and Almeida 2001] the authors developed a methodology issue of performance evaluation of Web Services. While this methodology is focussed on capacity planning using Queuing networks (QN), we aim at evaluating performance of WS using GSPN.

The Web Service Trust Center (WSTC) is a platform for development and evaluation of measurement tools and techniques in the field of Service Oriented Architectures (SOA) and web services. One of their publishers titled "Performance Modeling of WS-BPEL-Based Web Service Compositions**"**, addresses quality of service aspects of web service orchestrations created using WS-BPEL from the standpoint of a web service integrator. A mathematical model based on operations research techniques and formal semantics of WS-BPEL is proposed to estimate and forecast the influence of the execution of orchestrated processes on utilization and throughput of individual involved nodes and of the whole system. This model is applied to the optimization of Service Levels Agreement process between the involved parties [Rud et al. 2006].

Our work is different from the work presented in [Rud et al. 2006] , in fact we use GSPN to evaluate the performance of WS orchestrating with BPEL and not a pure mathematical model as the authors of that proposal. The advantage of using GSPN is that they are also mathematical models with the advantage of providing a good view of the system model.

For Silva and Lins, Web Services have played an important role in the development of Distributed Systems. In particular, the possibility of composing already implemented Web Services in order to provide a new functionality is an interesting approach for building Distributed Systems. However, choosing the better composition is still a challenger as different qualities may be observed in the composition, such as security, performance, fault tolerance, and so on. In this context, the paper [Silva and Lins 2006] proposes a methodology based on Stochastic Petri Nets to model, evaluate and help to choose Web Service compositions considering performance aspects.

Regarding the work [Silva and Lins 2006] which proposes a methodology based on analytical models of Generalized Stochastic Petri Nets in an effort to assess possibilities for composition of Web Services, with the main focus in the performance. Our work differs from it, because we are interested in evaluating the performance of WS based on the execution of orchestrated processes and not only in its composition. Our biggest interest is determined performance analysis of WS based on the execution of orchestrated processes that are executed by an engine in WS of greater complexity.

3. Framework: SOASPE

The related work showed that there has been a lot of studied and researched in the performance of Web Services. However most of these studies and research promote the evaluation of the performance of Web Services focussing on optimizing their composition.

In our work we want to address the issue, wich has not been explored yet, concerning the performance evaluation of the WS based *on the execution of orchestrated processes and this WS modeled with GSPN.*

Therefore, it is our intention to develop a framework for performance analysis of Web Services orchestrated with BPEL4WS. This framework, see Figure 2, is composed of five layers: **SOA Layer, BPEL Layer, Transformation Layer, Petri Net Layer** and finally the **Performance Evaluation Layer** .

The architecture of the Framework "SOASPE" is based on the principles of SOA, where a Business Process (BP) is composed of one or more services, which in turn may be composed of several subservices ans it, and their execution is coordinated by a **Business Process Integrator (BPI).**

The services can be of two types: **Basic** and **Orchestrated**.

The **Basic Services** are services that are processed by computer systems belonging to a Service Provider that return an Extensible Markup Language (XML) message as a result of processing.

The **Orchestrated Services** are services that are BPEL codes that serve to orchestrate new Business Processes that compose the Business Processes Integrator. The composition of Business Processes is shown in Figure 1.

Services can and almost always stay available on the Internet Service Providers. These services are mostly orchestrated with BPEL composing Web Services. The evaliation of the performance of the WS based on the execution of orchestrated processes is importante. In this regard, we define the FrameWork "SOASPE."

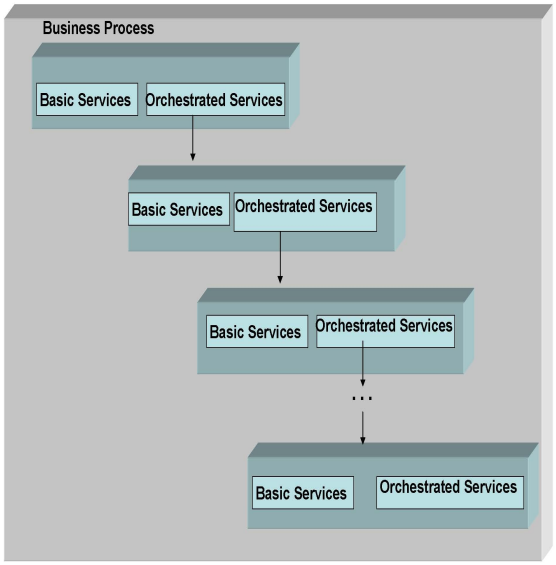

Figure 1. Business Processes Composition

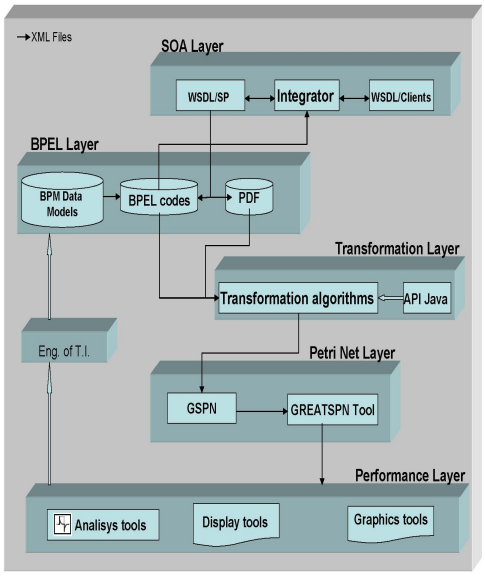

Figure 2. Architecture of Framework "SOASPE"

3.1. The SOA Layer

One of the possible scenarios of SOA implementation is a system consisted of a **Set of Service Providers**, a **Integrator** an operator of an orchestration engine and a **Set of Clients** of the latter, i.e. Business Process Consumers. This scenario is used as the basis for SOA Layer of the Framework "SOASPE".

The mission of the Integrator is execute the BPI Code of WS, to orchestrate a composite service from it by filling out a Business Process description template with all information necessary to start the process - i.e. with partner links, addresses, etc., and finally to provide the latter to the customers. This BPI Code will be written with BPEL4WS and for having evaluated the performance of the WS orchestrated by it, we will transform it into GSPN.

The relationship between the Integrator and the Service Providers as well as between the Integrator and the Clients is based on Service Level Agreements (SLA) which, in particular, determines: pricing, conditions and Quality of Service (QoS) warranties.

3.2. The BPEL Layer

In the **BPEL Layer** are finding the BPEL codes that make the orchestration of Web Services, the values of PDF of the response time of each of Service Providers (SP) of the WS and the Business Process Management (BPM) data models.

Since BPEL is not very friendly to developers, most of them prefer to model their applications using BPM tools. For this reason, BPM data models constitute the other component of this layer.

The leading standards for Business Process Modeling in SOA are the Business Process Modeling Notation for graphical modelling of Business Processes and the XML-based BPEL4WS for their execution. The corresponding centralised approach is referred to as Web Service orchestration. The main component of an orchestration infrastructure is a BPEL4WS engine that drives the execution of Business Processes by carrying out given algorithmic constructs and communicating with involved Web Services and clients.

From the BPM data models, the BPM tools generate all the BPEL code required for orchestration of the Web Service.

The BPEL codes are used also to be transformed into GSPN, which enabled make performance analysis of WS based on *the execution of orchestrated processes*.

3.3. The Transformation Layer

In **Transformation Layer** are presented the transformation algorithms of the BPEL codes into GSPN. In this transformation, the BPEL code that is present in BPI is transformed into the main GSPN, while Orchestrated Services are transformed in subnets of the main GSPN and the Basic Services that are processed in the Service Providers are modelled on GSPN as transitions and the PDF of the response time of each Service Provider will be assigned to the Delay Time of this transitions in GSPN. The firing of these transitions models the executions of the services in a Service Providers.

The representation of the transformation of each Basic Service into GSPN is modeled by a transition “t”, by two places “p1” and “p2”, and two arches linking it places to a transition, as shown in Figure 3. A token in place “p1” represents that the Basic Service modeled by the transition “t” is able to execute. The place “p2” will contain tokens after the firing of transition “t”, and this represent that Basic Service was executed.

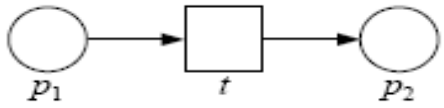

Figure 3. Representation of Basic Services and Basic Activities

The other component of this layer is the API´s Java that is added to the transformation algorithms to generate the executable codes.

The functionality of the Transformation Layer is illustrated in Figure 4.

The rules of transformation of BPEL code into GSPN are specified in the next section.

3.3.1. Transformation of BPEL into PETRI NETS (GSPN)

The purpose of this section is to provide a translation of BPEL into GSPN. We present the representation of the Basic Activities and Structured Activities of BPEL into

GSPN. We stress that layer of management is relevant to the implementation of WS, but for our purposes it will be omitted here.

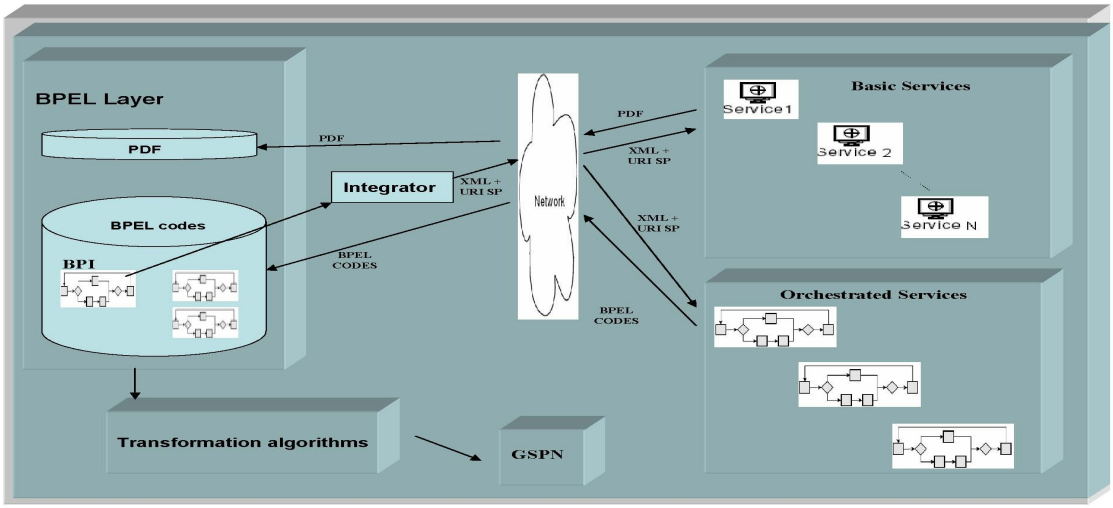

Figure 4. The Functionality of Transformation Layer

3.3.1.1. Transformation of Basic Activities

The Basic Activities are those that describe the steps of an elementary activity. BPEL defines the following Basic Activities: <Process>, <Invoke>, <Receive>, <Reply>, <Wait>, <Empty> and so on. The representation of the Basic Activities is the same of Basic Service and is shown in Figure 3.

3.3.1.2. Transformation of Structure Ativities

The Structured Activities prescribe the order in which a set of Basic Activities is executed. To enable the representation of complex structures, BPEL defines the following Structured Activities: <Sequence>, <Switch>, <While>, <Pick>, <Flow> and <Control Link>. Here we present their transformation into GSPN.

- **Sequence Structure**: this structure contains one or more activities that are carried out consecutively. Its representation is shown in Figure 5.
- ***Switch Structure:*** this structure supports conditional choices. Where only one of the transitions (t1 to tn) is fired when the arrival of a token on p1. Its representation is shown in Figure 5.
- ***While Structure:*** this structure allows one or a series of activities executives: none, one or more times. Figure 5 shows the representation of this structure. The transitions “t2” to “tn” can fire in a repetitive way, until the transition “t1” fires and shuts down the cycle of repetitions.
- ***Pick Structure*****:** the *pick* construct awaits the occurrence of one of a set of events and then it performs the activity associated with the event that occurred. The representation of the *pick* structure is the same as the representation of the *switch* structure shown in Figure 5.

- ***Flow Structure*:** the BPEL flow lets specify one or more activities to be carried out simultaneously. This fact leads to the definition of Flow Structure which is shown in Figure 5. In this representation the weight of the arc output of the transition "t_0"is "n", then the transitions "t1" to "tn" can fire simultaneously.
- ***Control Link Translation:*** more generally, the Flow activities allow the dependence of synchronization between the activities that directly or indirectly are nested within it. The Control Link structure is used to express these dependencies of synchronization. The sequence of representation of this structure is shown in Figure 5. This representation shows that there is a synchronism between the transitions t2 and tn. The transition tn will fire after t2 finishes its processing to be put a token in the place P5 and therefore make the transition tn enabled.

The next section gives the rules to calculate the Delay Time of the GSPN transitions. Note that the fault handlers will not be considered for purposes of the calculus of performance because exceptions are not part of the normal behavior of the execution of Web Services. Compensation handlers and activities <Compensate> will also be ignored, because they can only be activated from failures or other compensation handlers.

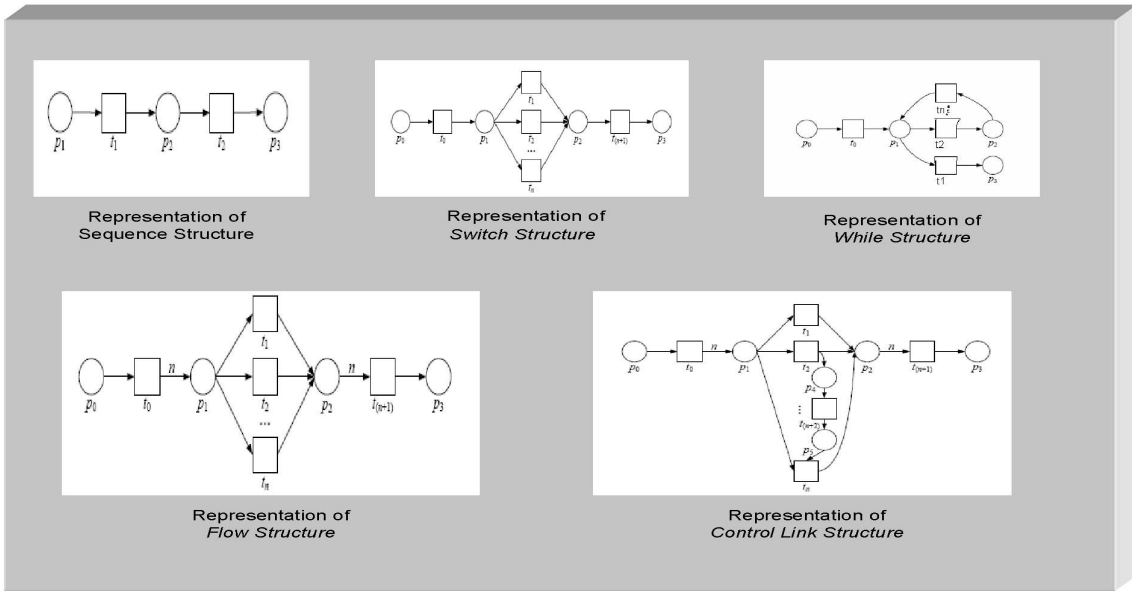

Figure 5. Logic of Representation of Structure Activities

3.3.1.3. Attribution Time to GSPN

In the transformation of BPEL codes into GSPN, the firing of transitions is immediate, except in the transitions that represent the Basic Activity <Invoke> that recive as Delay Time, the values of PDF of the response time of each of these Service Providers (SP), where services is executed.

To model the estochastic behavior of response time of Service Providers (SP´s), we will make use of PDF.

As entries of the PDF, it will be used the Average and Standard Deviation of the response time of Service Providers, while the output is expected the value of Delay Time of the transition (λ).

These response times of SP´s provide a sample with unknown distribution with Average (μ) and Standard Deviation (σ).

The Average (μ) is calculated as the arithmetic average of the response time of Service Providers and the Standard Deviation (σ) is calculated as shown in Figure 6.

Depending on the value of the Coefficient of Variation (CV), wich is calculated as shown in Figure 6, these response times are approximate to one of the distributions: Erlang, Hiperexponential or Hipoexponential. This makes it possible to represent the probable issue involved in the approximation of these response times of Service Providers for a Delay Time (λ) of the transition that it model.

If the Coefficient of Variation is greater than 1 (CV> 1) and the same is an integer value, the sample must be empirical approximate with Erlang Distribution. In this case the Delay Time (λ) of the transition that shapes this Service Providers will be calculated as shown in Figure 6.

If CV> 1 (CV is not a integer number), the distribution should be approximated with Hiperexponecial Distribution and the Delay Time (λ) of the transition that shapes this Service Providers will be calculated as shown in Figure 6.

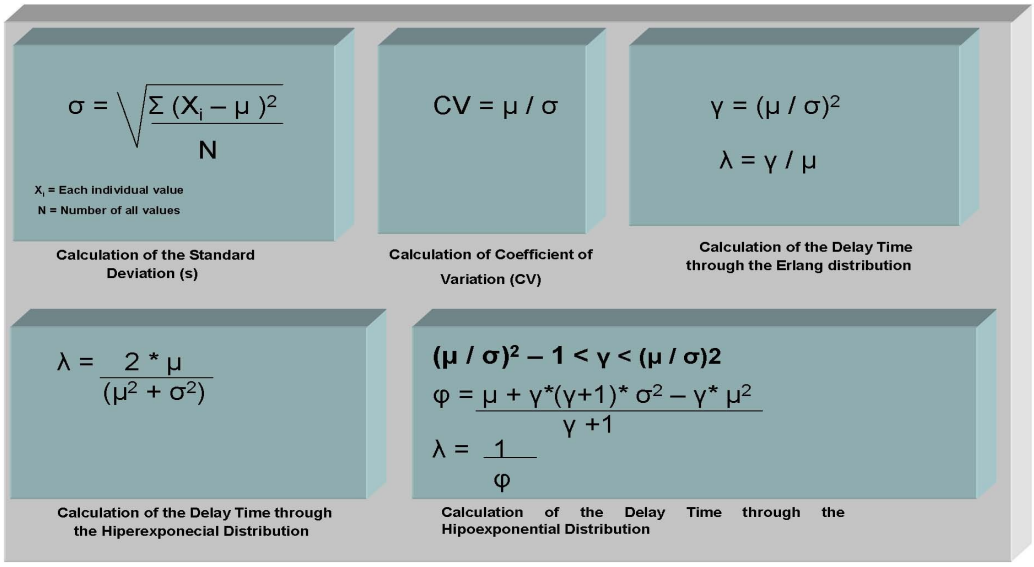

Figure 6. Calculation of the σ, **CV** and λ

And if CV< 1, the distribution should be approximated with Hipoexponential Distribution and the Delay Time (λ) of the transition that shapes this Service Providers will be calculated as shown in Figure 6.

To illustrate the attribution time to GSPN, suppose that you have the following BPEL code, as shown in Figure 7.

```
<Process>
<Receive createInstance="yes" />
<Switch>
    <Case name="Usa">
        <Invoke name="install_firmware" /> </Case>
    <Case name="France">
        <Invoke name="install_firmware" /> </Case>
 </Switch>
<Reply variabele="status" />
 </Process>
```

Figure 7. BPEL Code

In the code of Figure 7 the basic activities: <Process>, <Receive>, <Switch>, <Reply> will be shaped by an immediate transitions (represented in the model with a thin line), as shown in Figure 8. Already the transitions that shape the activities <Invoke> (in the model represented by a rectangle) will receive as Delay Times the values of PDF of the response time of each of Service Providers (SP), where invoked services will be executed, as shown in Figure 8.

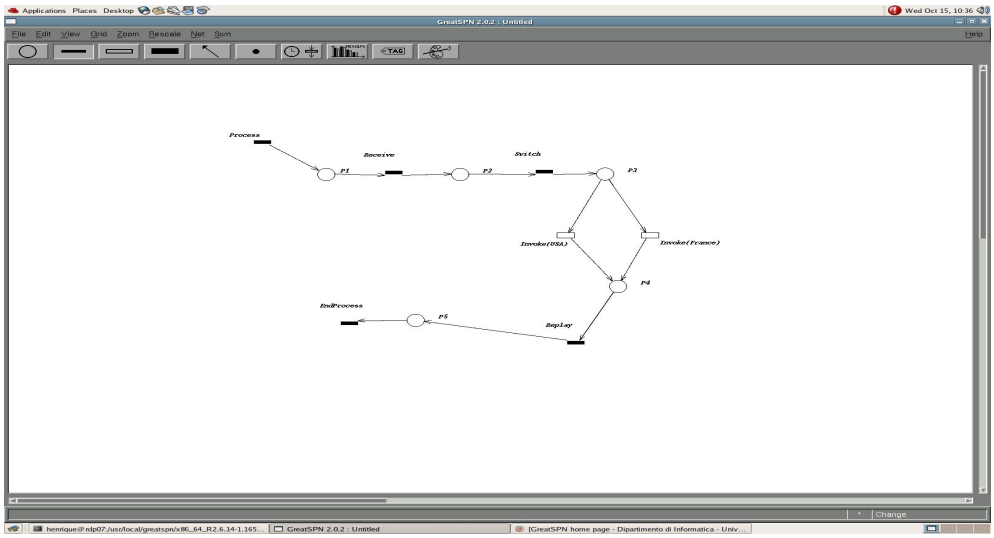

Figura 8. GREATSPN Tool

As a result of the Transformation Layer we have the necessary files to load the GSPN Nets in a GSPN tools.

3.4. The Petri Net Layer

The **Petri Net Layer** is composed of the GSPN obtained by transformation algorithms of the previous layer. This GSPN should be loaded into a GSPN tool, the other component of this Layer. At first time the tool user is the GEATSPN.

The Figure 8. shows the GSPN of the BPEL code presented in Figure 7, modeling inside the GREATSPN Tool [PE group (2006)].

3.5. The Performance Evaluation Layer

The **Performance Evaluation Layer** is defined as the layer responsible for the viewing (Graphics and Display) of the GSPN Nets and for the performance Analysis that the model will be submitted to investigate the evaluation of performance of WS based on the execution of orchestrated processes

4. Analysis of Framework "SOASPE"

With the Framework "SOASPE" defined, this section presents a case study – "WS SodaSys" with the objective to verify the usability and validity of it. The Bpel code of "WS SodaSys" is showed in Figure 7.

The analysis of the case study was performed in a machine with Intel Core Duo 1.86 GHz processor, motherboard with on-board and with 2 GB RAM. The installed operating system is Windows XP Professional.

In the implementation of Web Services, various artefacts of software were used. The Web Services (written in Java) were available in a Tomcat server, version 5.0.28. Additionally, we used the Ant (Apache AntUnit - version 1.6), the module SOAP for Apache (Java Web Services - version 2.3.1) and Apache Axis (Version 1.4). All these softwares are needed for the implementation of Web Services. Additional information about the need of them and how to use them can be found in "Java Web Services" [Hendricks *et al.* 2002].

To enable the orchestration of the "WS SodaSys", an engine of BPEL was used. For the reason of being Open-Source, the ActiveBPEL engine (version 1.2) is adopted.

4.1. Performance of "WS SodaSys"

The "WS SodaSys" is orchestrated by a BPI, which invokes the services of two Service Providers: **WS (USA) and WS (France).**

We measured the response times for Business Process Integrator (BPI) and for the Service Providers of an individual way. The response time of the BPI extends from the time in which an <Invoke> is made by the issuance of its response, including the execution time of its own Service Providers.

The measure of the response times of the Service Providers: WS (USA) was 5,191 ms and WS (France) was 4,919 ms.

Figure 10. shows the graph of measures of the response times for the Business Process Integrator of "WS SodaSys", when it meets with a number of requests ranging from 130 to 290 requests.

4.2. Performance of The Model Generated by Framework "SOASPE"

This section `deals` the use of the Framework "SOASPE" to simulate the evaluation of performance of "WS SodaSys".

As it has been said, the Transformation Layer receives as input: the BPEL code of the Business Process Integrator (BPI), the BPEL's codes of the Orchestrated Services and the Probability Distribution Function of the Service Providers (WS) that execute the Basic Services.

In the case study on the issue, the Transformation Layer receives the BPEL code of BPI of the “WS SodaSys” and the response time of Service Providers (Basic services) of WS (USA and France). With these data as entry the Transformation algorithms generate the GSPN that model the “WS SodaSys” in accordance with the guidelines set out in section 3.3.1.

The calculation of the Delay Time of the transitions that compose the Basic Activities <Invoke> of the “WS SodaSys” is shown next.

Each invoked Service Provider sends a SOAP message containing a list of response times of this service. With the list of response time of each Service Provider, the value of the Coefficient of Variation (CV) is calculated. With the CV it is possible to make an approximation of these response times by a Probability Distribution Function and thus have one close value to Delay Time of the transition that shape this Service Provide.

Figure 9 shows the Average, Standard Deviator, Coefficient of Variation, Approximation and Delay Time of Service Providers of the “WS SodaSys”.

	μ	σ	CV	Approximation	λ (in ms)
WS (Usa)	5,351	3,37	0,63	Hipoexponential	4,337812499
WS (France)	4,819	2,505	0,52	Hipoexponential	2,998373314

Figure 9. μ, σ, CV, approximation and λ of Service Providers.

Ended the activities carried out by the Transformation Layer is obtained the files needed to load the specification of GSPN for a GSPN tool. The files generated contained the format of the GSPN to be loaded on GREATSPN tool.

With the Petri Net loaded on GREATSPN tool it begins the activities of the Performance Evaluation Layer.

The performance analysis of the model is made from simulations with the same amount of requests made in the “WS SodaSys”.

Figure 10. presents the graph of response time of the GSPN model of the “WS SodaSys” when it meets with a amount of requests ranging from 130 to 290 requests

These results shows that the response times of the model generated by the Framework "SOASPE" and the response times of the “WS SodaSys” not differ by more than 5.3%, proving itself as the usability and validity of the Framework “SOASPE” in Performance Analysis of Web Services.

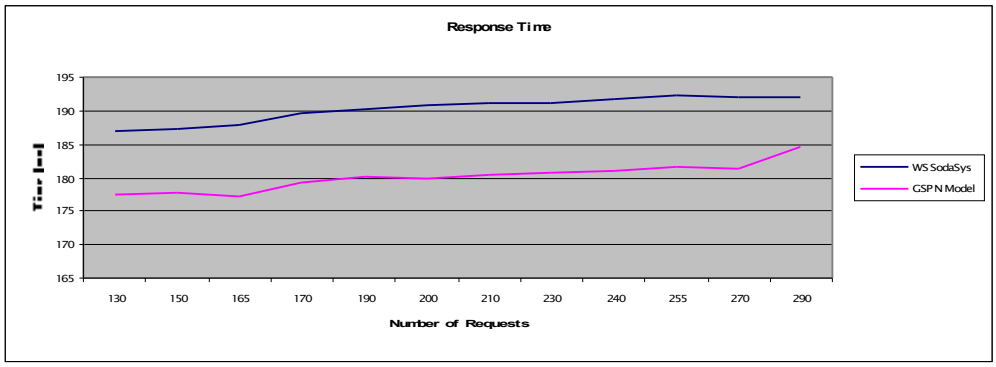

Figure 10. Response Time of "WS SodaSys" and Response Time of Model of the "WS SodaSys"

5. Concluding Remarks and Future Work

The SOA model brings several new benefits to software design and architecture by enabling re-use and sharing of components through dynamic discovery. Service orchestrations enable complex applications to be put together in a variety of ways. Each possible service selection of services brings different levels of QoS. Thus, there is a need to devise fast and efficient mechanisms that can be used for performance analysis of WS among a set of service providers. This paper presented such an efficient mechanism that, in all experiments reported, comes very close to the real response time of WS (less than 6% worse) after having compare with the time of the model generated by the Framework "SOASPE".

As future work, we want to continue the issues seen in this work, particularly a more deeper refinements in view of modeling other aspects that were not included in this article, such as the extension of this work to support Grid Services, recent technology that adds new features to the design of Web Services.

References

W3C. (2002). Web Services Description Requirements.

OASIS. (2007). Web Services Business Process Execution Language 2.0.

Menascé, D. A. and Almeida, V. F. (2001). Capacity Planning for Web Service: metrics, models, and métodos. Prentice Hall. 608p.

Rud D., Schmietendorf A. and Dumke R (2006). Performance modeling of WS-BPEL-based web service compositions. In Proceedings of the IEEE Service Computing Workshops (SCW 2006), pages 140-147, Los Alamitos, CA, USA, September 2006.

Silva, A. N. And Lins, F. A. (2006). Avaliação de Desempenho da Composição de *Web Services* Usando Redes de Petri. In: SBRC, 2006, Curitiba. 24o. Simpósio Brasileiro de Redes de Computadores.

PE group (2006) . GreatSPN User's Manual (version 2.0.2). University of Torino, Italy. http://www.di.whito.it/greatspn

Hendricks, M. *et al.* (2002). "*Java Web Services*", Alta Books.