Performance Analysis of Service Oriented Software

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

[1]Departamento de Informática UERN, Mossoró - RN, Brazil

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

[3]Instituto 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. *Web Services Business Process Execution Language (WS-BPEL) is a promising language describing the Service Oriented Software (SOS) orchestrations in form of Business Processes* (BP)*, 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, may provide a means to analyse WS-BPEL processes, evaluating its performance, detecting weaknesses and errors in the process model already at design-time. This article addresses quality of SOS orchestrations created using the WS-BPEL and a framework* ”SOASPE” *for transformation of WS-BPEL into* Generalized Stochastic Petri Nets *is proposed to analise the performance and throughput of SOS, based on the execution of orchestrated processes.*

1. Introduction

As the use of Web Services grows, organizations are increasingly choosing Business Process Modeling Notation (BPMN) for business process modelling while WS-BPEL for services orchestration. In addition to orchestrating organizations’ Web Services, *WS-BPEL’*s strengths include asynchronous message handling, reliability, and recovery [OASIS 2007].

WS-BPEL provides a powerful technology to aggregate encapsulated functionalities and define high-value Web Services. Even though *WS-BPEL* is a promising language describing the Web Services orchestrations in form of Business Processes, it lacks of a sound formal semantic, which hinders the formal analysis and verification of business processes specified in it [Dun, Xu and Wang 2008].

Formal methods, like Petri Nets (PN), may provide a means to analyse *WS-BPEL* processes, evaluating its performance, detecting weaknesses and errors in the process model already at design-time [Dun, Xu and Wang 2008].

This article presents a framework to transformation of Web Services (WS) orchestrating with *WS-BPEL* into Generalized Stochastic Petri Nets (GSPN) in a constructive way. Therefore we can translate orchestrating specified in *WS-BPEL* into GSPNs, which can be analyzed and verified by many specialized tools. So, we defined a framework for verifying WS performance.

The transformation of *WS-BPEL* into GSPN is proposed to estimate and forecast the influence of the execution of orchestrated processes on the utilization and throughput of individual involved Service Providers (SP) and of the whole WS. So, this translation is applied to the verification of Service Levels Agreement (SLA) document between the involved parties. We also present an implementation of our analysis techniques. This implementation takes as input a *WS-BPEL* orchestrating of a Web Service, and automatically generates a GSPN and performs the desired analysis. Such a tool has broad applicability both as a backend tool to existing manual Web service composition, and as a stand-alone tool for Web Service developers.

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), Web Services Business Process Execution Language (WS-BPEL, 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, and so on.

In this article, we are motivated by issues related to the definition of a framework for the transformation of WS´s orchestrating with WS-BPEL into GSPN and this way evaluate the performance of these WS.

A WS-BPEL process implements a Web Service by specifying the interactions with other Web Services, which might be WS-BPEL 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 where the services are executed.

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

This article is organized as follows. Section 2 gives a review of some related work. A backgroud from the technology used in the construction of the Framework ”SOASPE” is presented in Section 3. The Framework “SOASPE” is defined in Section

4 and its analysis is discussed in Section 5. Section 6 concludes the paper and gives suggestions of future work. Finally in Section 7 the acknowledgement are presented.

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. Their 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” [Rud et al. 2006], 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 difference 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 consistent with the topologies possibilities of services composition (sequential, parallel and free choice) of the WS. 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 define a framework that can be used to performance analysis of WS based on the execution of orchestrated processes that are executed by an engine.

Others related works of interest are cited next.

In [Chandrasekaran et al. 2003], Chandrasekaran proposes a simulation technique for analyzing performance of composite web services in order to obtain efficient web processes. Their paper describes the Service Composition and Execution Tool (SCET) and various methodologies that could be adopted for evaluating the performance of a Web process. SCET allows for composing services statically using its designer and storing them as Web Service Flow Language (WSFL) based specifications. Executing a process enables one to realize its functionality and also analyze its performance.

In [Narayanan and McIlraith 2002], Narayanan and his group take the DAML-S ontology for describing the capabilities of Web Services and define the semantics for a relevant subset of DAML-S in terms of a first-order logical language. With the semantics in hand, they encode service descriptions in Petri Net formalism and provide decision procedures for Web Service simulation, verification and composition.

Elena Gómez-Martíınez Elena and José Merseguer recall, from the literature, a performance study of a Web Service. This study, based on the Layered Queuing Network (LQN) paradigm, is now addressed following the PUMA approach to obtain a new performance model, in this case in terms of Petri Nets, for the target Web Service. Such Petri Net model is used to extend the previous LQN results with respect to some key Web Service performance aspects: the SOAP toolkit and the XML parsers [Martíınez and Merseguer 2006].

Menascé studies QoS issues of composite Web Services. In his work titled "A Heuristic Approach to Optimal Service Selection in Service Oriented Architectures" [Menascé et al. 2008], he believes that Service Oriented Architectures (SOA) enable a multitude of Service Providers to provide loosely coupled and interoperable services at different Quality of Service (QoS) and cost levels. His paper considers Business Processes composed of activities that are supported by Service Providers. The structure of a business process may be expressed by languages such as BPEL and allows for constructs such as sequence, switch, while, flow, and pick. So his paper considers the problem of finding the set of service providers that minimizes the total execution time of the business process subject to cost and execution time constraints. The problem is clearly NP-hard. However, the paper presents an optimized algorithm that finds the optimal solution without having to explore the entire solution space. This algorithm can be used to find the optimal solution in problems of moderate size. A heuristic solution is also presented and experimental studies that compare the optimal and heuristic solution show that the average execution time obtained with a heuristic allocation of providers to activities does not exceed 6% of that of the optimal solution.

In [Dun, Xu and Wang 2008], for the authors, Web Service composition involves the combination of a number of existing Web Services to create a value-added service in way that may not be foreseen at the time when a Web Service is written. BPEL is a promising language which describes Web Service composition in form of Business Processes. However, BPEL is an XML-based language and may suffer from ambiguities or some erroneous properties. It is necessary to analyze business processes specified in BPEL with a formal tool. In their paper, the authors put forward an approach to model and verify BPEL based on ServiceNet, a special class of Petri nets. They present some transformation rules of BPEL business processes into ServiceNet. Then the throughness of a BPEL business process can be verified by reducing the corresponding ServiceNet based on some reduction rules.

3. Background: SOA, BPEL, SPE, PETRI NETS (PN)

In this section will be made a retrospectively of technologies used in the construction of the Framework ”SOASPE”.

3.1 SOA

Service-Oriented Architecture (SOA) is an architecture for building distributed software systems on the basis of autonomous, interoperable, discoverable and reusable software entities called services [W3C 2002]. Services incapsulate business functionalities of involved parties and in that way make the Business Process Management (BPM) to a natural application of the Service-Oriented Architecture.

The SOA Architecture expresses a concept where applications or routines are available as services in a computer network (Internet or Intranet) and they communicate through open standards [Menascé 2004]. Most implementations of SOA are used for Web Services. A SOA can be used in any standard based on web technology.

The basis of the standard Web Services relevant to the SOA include:

- **XML (*eXtensible Markup Language*)** - is a markup language to describe data in loads of messages in a document format;
- **SOAP (*Simple Object Access Protocol*)** - is an XML-based protocol used for exchanging information in a distributed environment;
- **WSDL (*Web Services Description Language*)** - is an XML document that describes a set of SOAP messages and how these messages are exchanged. As the WSDL is XML, it is readable and editable, but in most cases, it is generated and consumed by software; and
- **UDDI (*Universal Description, Discovery, and Integration)*** - is a way to locate a Web Service in a register, like a catalog of yellow pages, so that a program in search of a particular service can easily find and understand what the service does.

According to [Arsanjani 2004] the SOA can be well represented from the following model involving three main parts: the Service Providers, the Service Consumer and the Registry. This model is also called "find-bind-execute paradigm" as best shown in Figure 1.

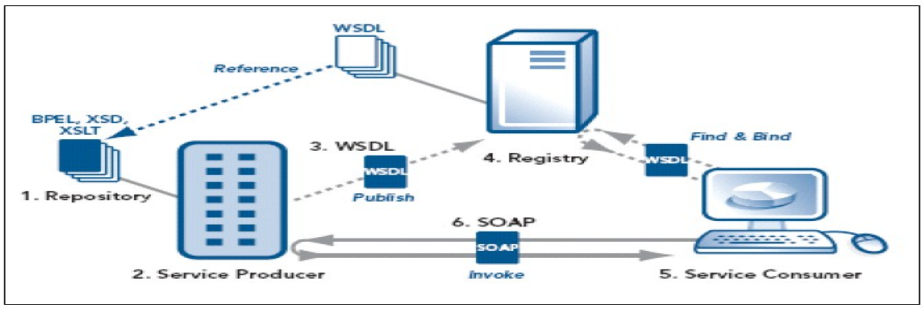


Figura 1. The SOA Architecture [Arsanjani 2004]

In this work we utilise the "find-bind-execute paradigm" for SOA implementation, i.e., SOA is implemented as Web Services.

3.2 BPEL

The Business Process Execution Language (BPEL) is used as a language to build Web Services compositions. The business activities, which compose the BP, are implemented by services and executed by Service Providers, and invoked by service consumers using the WS-BPEL and Web Services communication paradigm. WS-BPEL has emerged as a standard-based service orchestration technology that provides an XML-based grammar for describing the control logic required to orchestrate Web Services participating in a process flow. Business Processes defined in BPEL are portable and can be executed in any BPEL-compliant process engine [OASIS 2007].

In our work, WS-BPEL is used on Web Service orchestration. We will provide the rules of translation from *WS-BPEL* to GSPN for evaluation of performance of the WS. This is presented in section 4.3.1.

3.3 SPE

Software Performance Engineering (SPE) has been defined as a method for constructing software systems to meet performance objectives. Smith’s book, “Performance Solutions”, is an excellent bibliography to understand the methodologies of SPE. It states: “The process begins early in the software life cycle and uses quantitative methods to identify satisfactory designs and to eliminate those that are likely to have unacceptable performance, before developers invest significant time in implementation. SPE continues through the detailed design, coding and testing stages to predict and manage performance of the evolving software and to monitor and report actual performance against specifications and predictions. SPE methods cover performance data collection, quantitative analysis techniques, prediction strategies, management of uncertainties, data presentation and tracking, model verification and validation, critical success factors, and performance design principles” [Smith and Williams 2001].

By the prospect of being able to evaluate the performance of *WS-BPEL processes*, before it is implemented, we use the principles of SPE in our work.

3.4 PETRI NETS (PN)

A Petri Net is a mathematical formalism represented graphically by a bipartite graph containing *places* (drawn as circles) and *transitions* (drawn as rectangles). Places hold *tokens* and represent predicates about the world state. Transitions are the active component. When all of the places pointing into a transition contain an adequate number of tokens, the transition is *enabled* and may *fire*, removing tokens from its input places and depositing a new set of tokens in its output places. The most relevant features of Petri Nets for our purposes is their ability to model events and states in a distributed system and capture sequentiality, concurrency and event-based asynchronous control [Natkin 1980].

Extensions to the basic Petri Net formalism that include typed arcs, hierarchical control, timed transitions, parameterization, typed (individual) tokens and stochasticity are called SPN (Stochastic Petri nets).

Stochastic Petri nets are a mathematical formalism that allows graphical representation, and has powerful methods of formal analysis and performance (using its extensions timer) of a system. The “places” and “transitions” of Petri Nets are used to shape vision points of logic systems. The idea of involving random variables of delay exponentially distributed to transitions has been explored for the first time by Natkin [Natkin 1980] and Molloy [Molloy 1981], allowing the emergence of Stochastic Petri Nets (SPN´s) and its extensions. SPN´s have the property of being similar to Markov Chains of Continuous Time, allowing the same analysis stationary and transient known that can be applied to those nets. The extension Generalized Stochastic Petri Nets (GSPN), originally proposed by Marsan [Marsan *et al*. 1995], is one of the most popular extensions.

A GSPN is defined by an 8-upla GSPN = (P, T, I, O, H, Π, W, M0) [Marsan *et al*. 1995]. The set of places P is the availability of resources (marks on the net) system, states and local variables of the system. The set of transitions T is the set of actions that can cause change of state. This collection is divided in the subset of timed transitions and the subset of transitions that describe immediate instant actions, with higher priority, the immediate transitions. The function “W” combines a non-negative real number that indicates the rate of exponential distribution to each timed transition. The function “Π” defines the level of priority of each timed immediate transition (the priority of a immediate transition is zero). We have “I” and “O” functions such as mapping of places to transitions and transitions to places, respectively, while the function “H” represents the inhibitors arcs. The marking M0 is the initial state of the system. The GSPN´s have been widely used for modeling various types of systems, from manufacturing systems to wireless networks.

For this article, the crucial fact about the utilization of GSPN in representation is that it is active with a well defined durative transition semantics for service descriptions.

In the next sections we utilise this technologies to define the Framework “SOASPE”, that is the objective of this work.

4. Framework: SOASPE

The related works showed that there has been a lot of studies and researches 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, concerning the performance evaluation of the WS based *on the execution of orchestrated processes and this WS modeled with GSPN.*

We propose a framework for performance analysis of Web Services orchestrated with WS-BPEL. The structure of 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**.

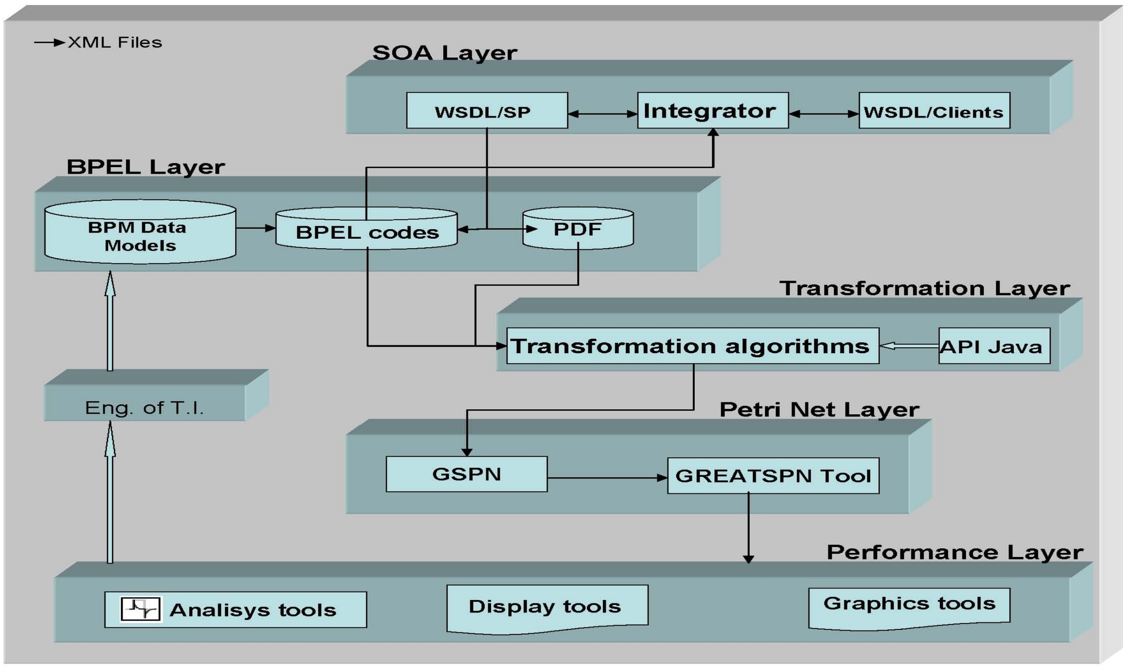


Figure 2. Architecture of Framework “SOASPE”

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 inside it. The execution of this BP, that invoke these services, which execute in the SP is coordinated by a **Business Process Integrator (BPI).**

The services can be of two types: **Basic** and **Orchestrated** [Rud et al. 2006].

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 other Business Processes that compose the Business Processes Integrator.

The composition of Business Processes is shown in Figure 3.

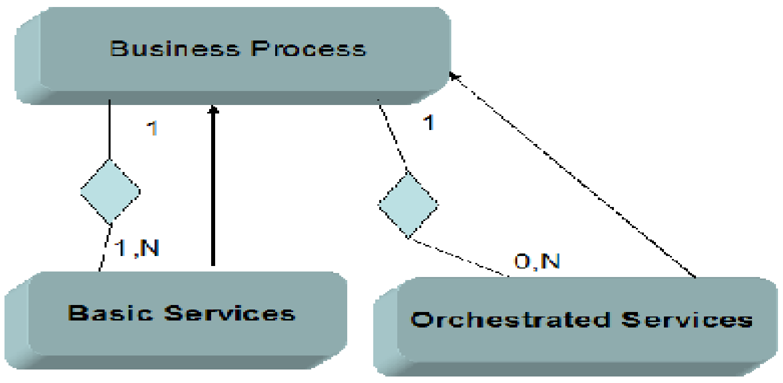


Figure 3. Business Processes Composition [Rud et al. 2006]

Services are available by Service Providers. These services are mostly orchestrated with BPEL composing Web Services. The evaluation of the performance of the WS based on the execution of orchestrated processes is important [Rud et al. 2006]. In this regard, we define the FrameWork "SOASPE".

4.1. The SOA Layer

One of the possible scenarios of SOA implementation is a system consisted of a **Set of Service Providers**, an **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 BPEL code of the BPI, 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 consumers. This BPI Code will be written with *WS-BPEL* 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 and Quality of Service (QoS) guarantees as performance.

4.2. 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 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 *WS-BPEL* for their execution. The corresponding centralised approach is referred to as Web Service orchestration. The main component of an orchestration infrastructure is a WS-BPEL engine that drives the execution of Business Processes by carrying out given algorithmic constructs and communicating with involved Web Services and clients.

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

4.3. 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. The Basic Services, that are processed in the Service Providers, are modeled into GSPN as in Figure 4 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 these 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 arcs linking each places to a transition, as shown in Figure 4. 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.

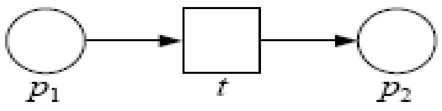


Figure 4. Representation of Basic Services and Basic Activities

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 5.

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

4.3.1. Transformation of BPEL into Generalized 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.

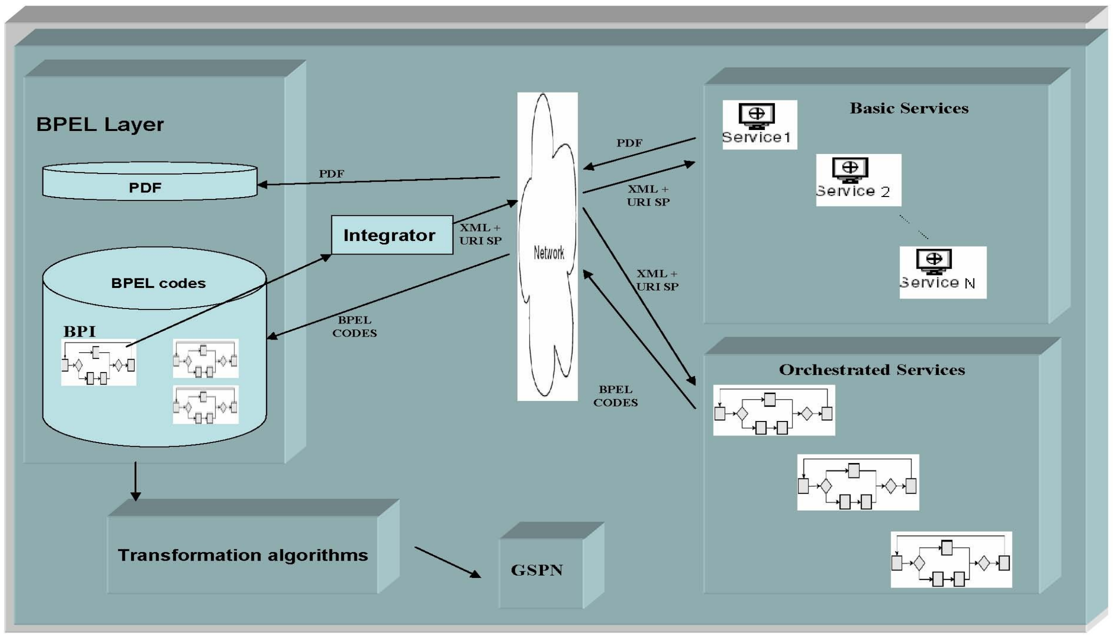


Figure 5. The Functionality of Transformation Layer

4.3.1.1. 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>. The representation into GSPN of the Basic Activities is the same of Basic Service and is shown in Figure 4.

4.3.1.2. Transformation of Structured Activities

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 6.
- ***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 6.
- ***While Structure:*** this structure allows one or a series of activities executives: none, one or more times. Figure 6 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* structure 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 6.
- ***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 6. In this representation the weight of the output arc 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 6. 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.

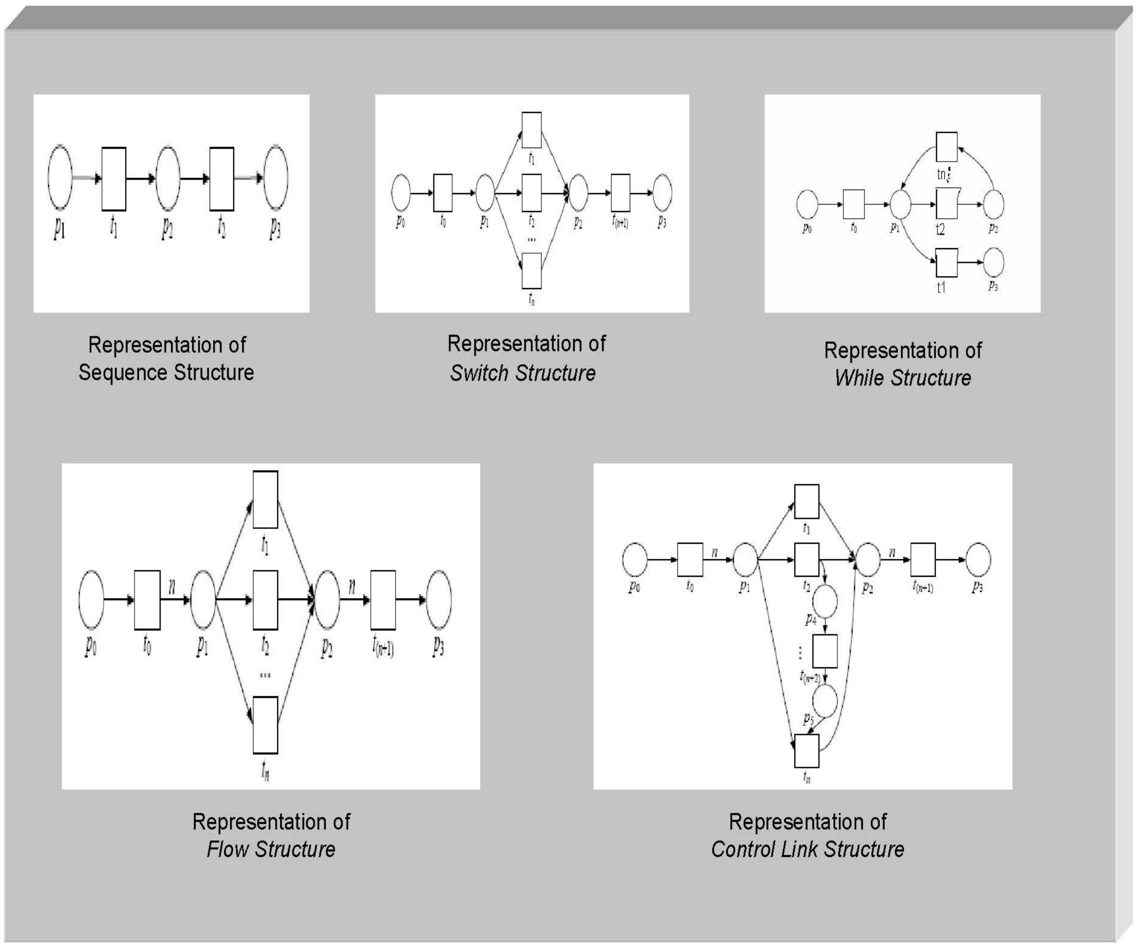

Representation of Sequence Structure
Representation of *Switch Structure*
Representation of *While Structure*
Representation of *Flow Structure*
Representation of *Control Link Structure*

Figure 6. Logic of Representation of Structure Activities

4.3.1.3. Attribution Time to GSPN

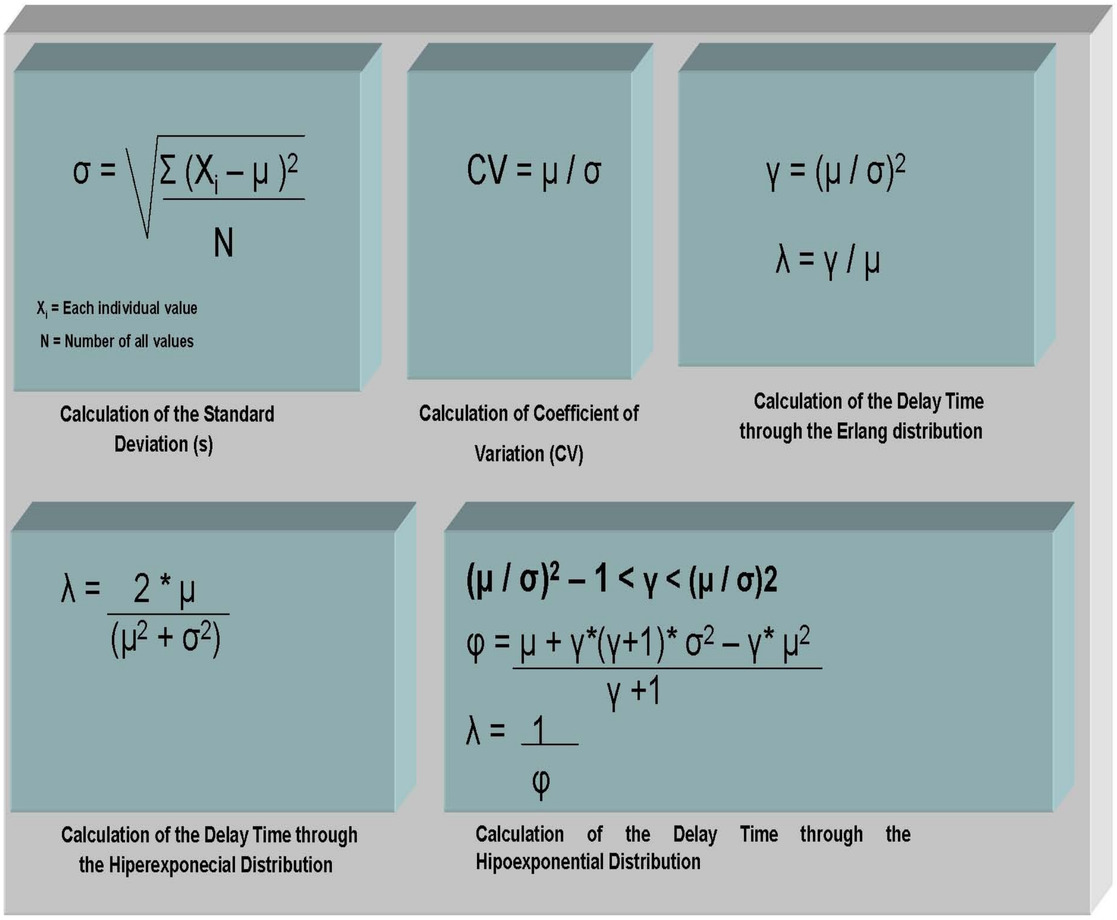
In the transformation of BPEL codes into GSPN, the firing of transitions is immediate, except in the transitions that represent the Basic Activity <Invoke> that receive as Delay Time, the values of Probability Distribution Function (PDF) of the response time of each of these Service Providers, where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

To model the estochastic behavior of response time of Service Providers, 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 7.



$$\sigma = \sqrt{\frac{\Sigma\,(X_i - \mu)^2}{N}}$$

X_i = Each individual value

N = Number of all values

Calculation of the Standard Deviation (s)

$$CV = \mu / \sigma$$

Calculation of Coefficient of Variation (CV)

$$\gamma = (\mu / \sigma)^2$$

$$\lambda = \gamma / \mu$$

Calculation of the Delay Time through the Erlang distribution

$$\lambda = \frac{2 * \mu}{(\mu^2 + \sigma^2)}$$

Calculation of the Delay Time through the Hiperexponecial Distribution

$$(\mu / \sigma)^2 - 1 < \gamma < (\mu / \sigma)2$$

$$\varphi = \frac{\mu + \gamma*(\gamma+1)*\sigma^2 - \gamma*\mu^2}{\gamma+1}$$

$$\lambda = \frac{1}{\varphi}$$

Calculation of the Delay Time through the Hipoexponential Distribution

Figure 7. Calculation of the σ, CV and λ [Silva and Lins 2006]

Depending on the value of the Coefficient of Variation (CV), wich is calculated as shown in Figure 7, these response times are approximate to one of the distributions: Erlang, Hiperexponential or Hipoexponential [Silva and Lins 2006]. 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 7.

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 7.

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 7.

To illustrate the attribution time to GSPN, suppose that you have the following BPEL code, as shown in Figure 8.

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

Figure 8. BPEL Code

In the code of Figure 8 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 9. Already the transitions that shape the activities <Invoke> (in the model in Figure 9 represented by a rectangle) will be shaped by an timed transitions and receive as Delay Times the values of PDF of the response time of each of Service Providers, where invoked services will be executed.

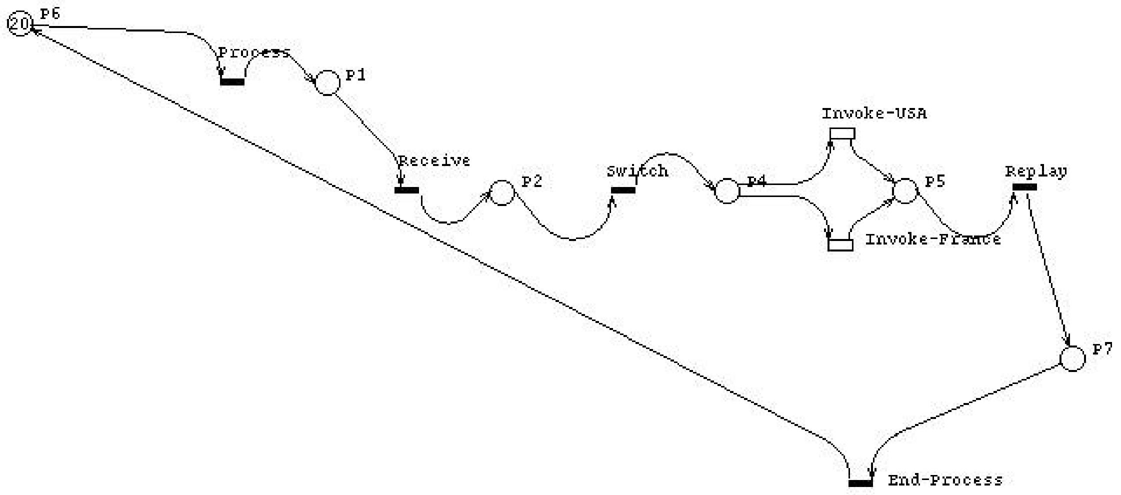


Figure 9. Rede de Petri (GSPN)

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

4.4. 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. Figure 10 shows the GSPN of the BPEL code presented in Figure 8, modeled inside the GREATSPN Tool [PE group (2006)]..

4.5. 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.

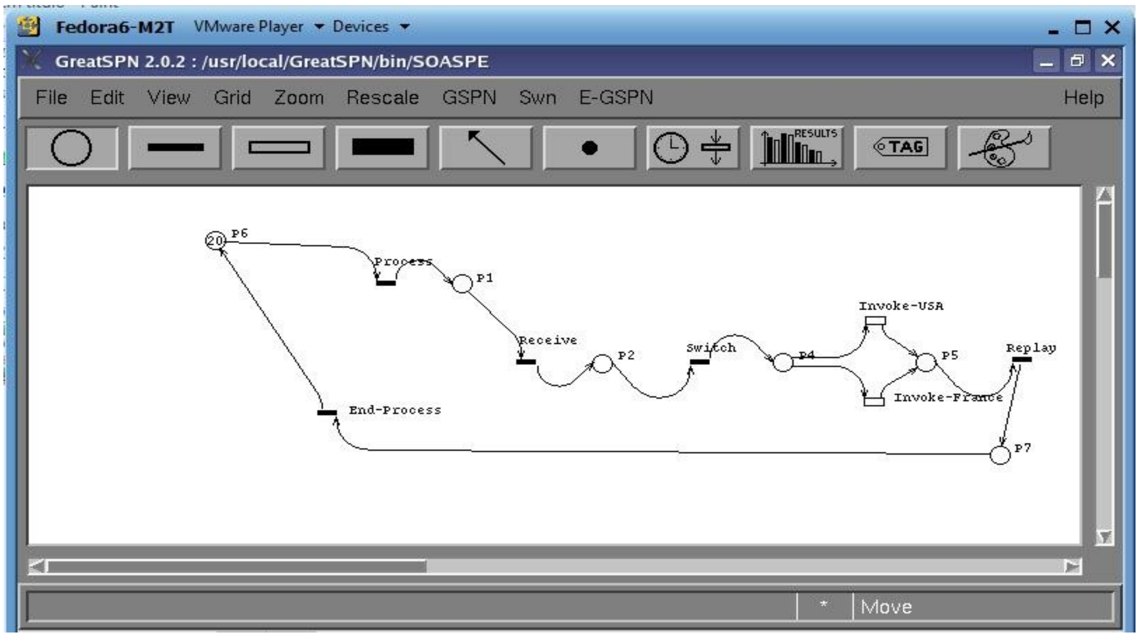


Figure 10. GREATSPN Tool

5. 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 8.

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.

5.1. 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 11 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.

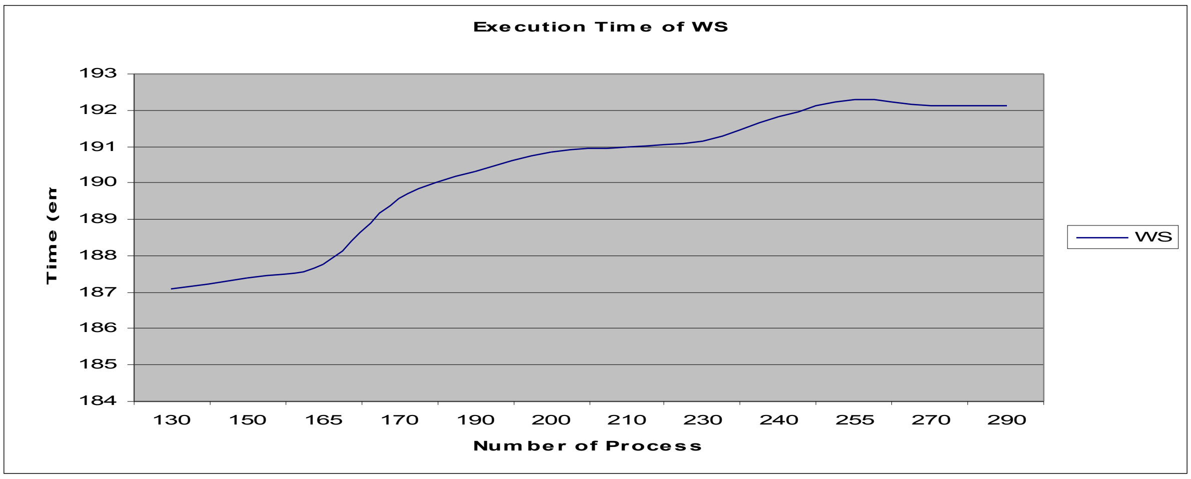


Figure 11. Response Time of “WS SodaSys”

5.2. 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 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 4.3.1.

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 Provider.

Figure 12 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 12. 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 13 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

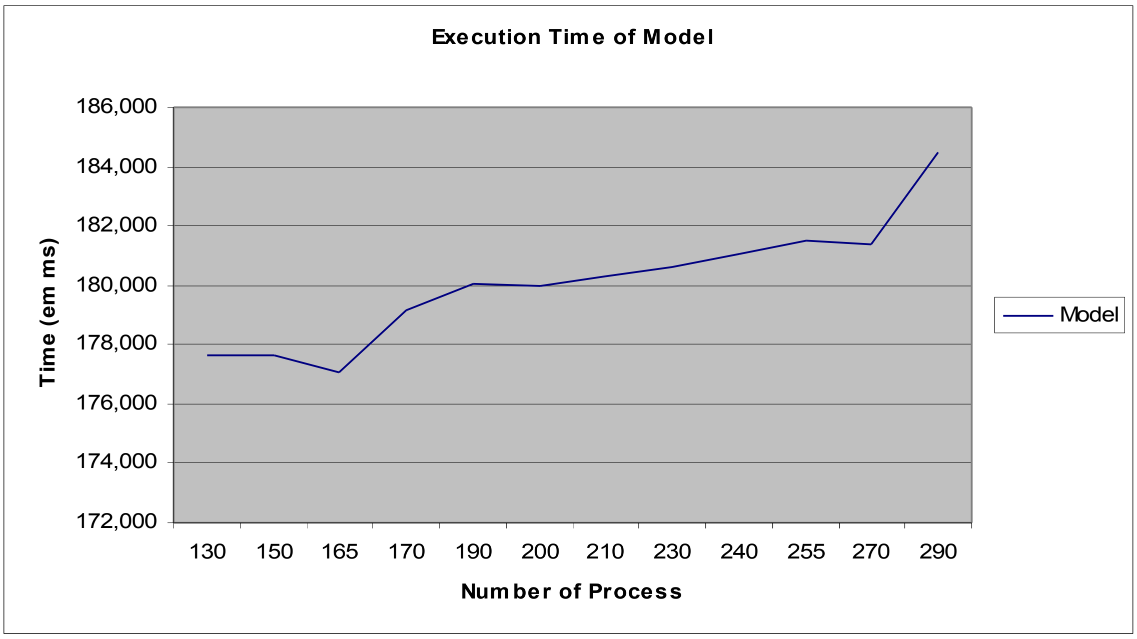


Figure 13. Execution Time of Model of the “WS SodaSys”

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 14 presents a comparison of response times of the model generated by the Framework "SOASPE" for "WS SodaSys" and the response times of the "WS SodaSys".

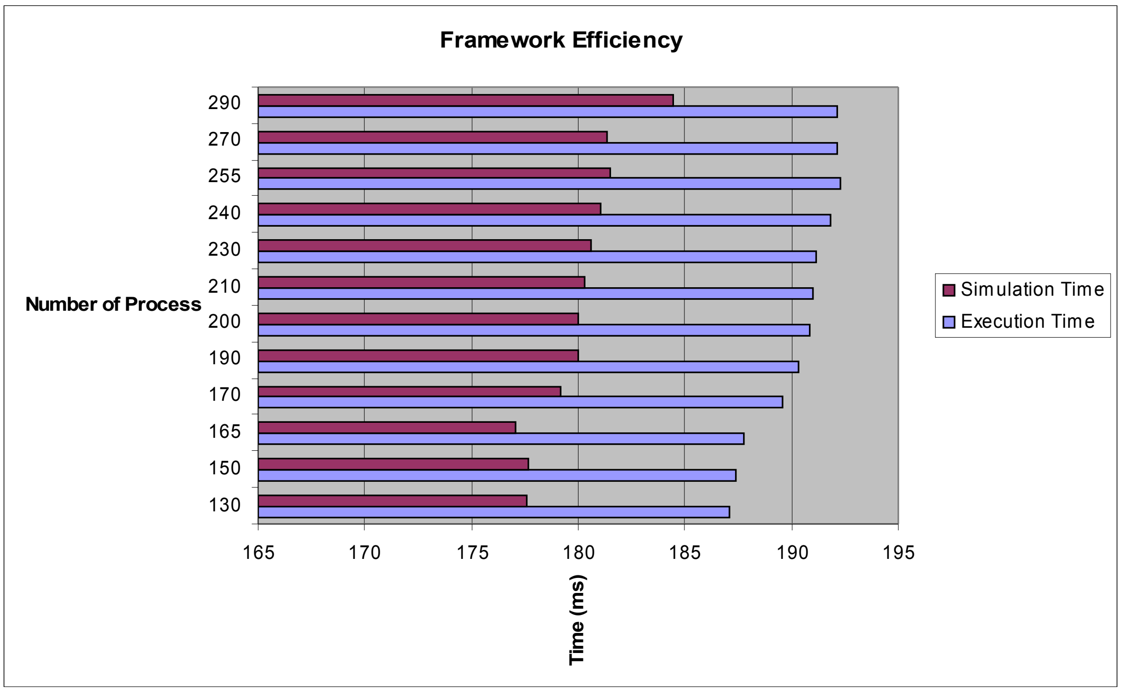


Figure 14. Framework "SOASPE" Efficiency

6. 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 article 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 compared 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 and considered fault handlers, compensation handlers and activities <Compensate> for purposes of the calculus of performance.

7. Acknowledgements

We would like to acknowledge our colleagues of the Centro Politécnico Superior (CPS) of University of Zaragoza (UNIZAR) in the help and collaboration in the development of the Framework “SOASPE". We would also like to thanks the Professor Giovanni Cordeiro Barroso for our interesting discussions on various aspects of this work. We particularly thank the Professor José Merseguer Hernáiz for believing in my abilities and always be present in the development of this work and for your review for an informative and thorough review of this article. Finally we gratefully acknowledge all those who directly and indirectly involved in execution of this work

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.

Menascé. (2004) “Composing Web Services: A QoS View“. IEEE Internet Computing, 8(6):88–90.

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.

S. Chandrasekaran, J. Miller, G. Silver, I. Arpinar, and A. Sheth. (2003) “Performance Analysis and Simulation of Composite Web Services”. *Electronic Markets*.

Srini Narayanan**,** Sheila McIlraith. (2002) “Analysis and Simulation of Web Services“. Eleventh International World Wide Web Conference (WWW11).

Gómez-Martíınez Elena, Merseguer José. (2006) “Impact of SOAP implementation in the performance of a Web-Server-based Application. LNCS 4331, pp. 884–896.

Daniel A. Menascé, Emiliano Casalicchio, Vinod Dubey. (2008) “A Heuristic Approach to Optimal Service Selection in Service Oriented Architectures“. *WOSP* ’08 Princeton, NJ, USA.

Haiqiang Dun, Haiying Xu, Lifu Wang. (2008) “Transformation of BPEL Processes to Petri Nets“. 2nd IFIP/IEEE International Symposium on Theoretical Aspects of Software Engineering.

Arsanjani, A. (2004) “Service-oriented modeling and architecture: How to identify, specify, and realize services for your SOA“. IBM developerWorks.

C. Smith and L. Williams. (2001) “Performance Solutions“. Addison-Wesley.

Natkin, S. (1980) “Les Réseaux de Petri Stochastiques et Leur Application a L’evaluation des Systemes Informatiques”, Thèse de Docteur Ingegneur, CNAM, Paris, France.

Molloy, M. (1981) “On the Integration of Delay and Throughput Measures In Distributed Processing Models”, Ph.D.Thesis, UCLA.

Marsan, M.A. *et al*. (1995) “*Modelling with generalized stochastic Petri Nets*”, Università degli studi di Torino, Dipartimento di informática.