EAI Endorsed Transactions
on Industrial Networks And Intelligent Systems
Research Article ICST.ORG

# Performance evaluation of composite Web services

Lynda Mokdad[1], Jalel Ben Othman [2] and Abdelkrim Abdelli [3,*]

[1]Lab. LACL, University of Paris-Est LACL (EA 4219), UPEC F-94010 Créteil, France
[2]Lab. L2TI, University of Paris 13- L2TI (EA 3043), UP13 F-93430 Villetaneuse, France
[3]Lab. LSI, USTHB university of Algiers, Algeria

## Abstract

Composite Web service architectures are demanding much guarantee on the Quality of Service (QoS) in order to meet user requirements. Performance evaluation of these architectures has become therefore a very challenging issue, as the task is very complex due to synchronization inside the orchestration of services. We propose in this paper to use stochastic automata networks which a is powerful formalism as it provides semantics to specify synchronization within a very smart formalism. Contrary to previous approaches, the modeling and the performance evaluation of a variable number of remote service invocation become possible. The reported simulation results advocate the use of our approach in the performance evaluation of composite Web service architectures.

**Keywords:** Stochastic Automata Networks (SAN), Markov Chain, response times, Composite Web services, performance evaluation.

Received on 25 September 2014; accepted on 10 February 2015; published on 04 June 2015

doi:10.4108/inis.2.4.e4

## 1. Introduction

The use of web services is continuously growing and the technological and economic potential is not yet tapped. The web services have become in recent times by far the technology application integration par excellence as it is now feasible to host basic web services on a smart phone without requiring additional technologies .

Web Services are software components that can be accessed over the Internet using well established web mechanisms. For instance, in the IT domain the impact of XML Web Services has increased during recent years, since the Extensible Markup Language (XML) has been enforced as a meta language for structured information and its representation [1].

Web services are self-descriptive loosely coupled and interact with each other. They are defined and described regardless of their platforms, implementation details. The biggest advantage of Web Services lies in their simplicity in expression, communication and servicing. The componentized architecture of Web Services also makes them reusable, thereby reducing the development time and costs [2].

Unlike its predecessors, such as the Common Request Broker Architecture (CORBA), Remote Method Invocation (RMI) and Distributed Component Object Model (DCOM), web services have responded satisfactorily to interoperability in the context of distributed systems, as well as to the scale of the Internet. Indeed, Web services can be seen as the standardized way to distribute services on the Internet. It uses Internet protocols to communicate and uses a standard language to describe its interface. The success of Web services is in fact due to the use of Internet technology as a communication infrastructure and the availability of a working framework based on a set of standards which are [3]:

- *SOAP* (Simple Object Access Protocol) a communication protocol for structuring the messages exchanged between software components [4]
- *WSDL* (Web Services Description Language) a specification for describing Web services interfaces [6]; and finally
- *UDDI* (Universal Description Discovery and Integration) a specification for publishing and localization of Web services [5].

There are, two key players, in *Web services architecture*: The service provider (which publishes the service), and the service requester. A third actor, the service registry, may be associated with this pair, but its presence is not essential as the service requester needs only to be aware of the address of the service provider. These three participants must be able to interact with each other.

*Corresponding author. E-mail: akabdelli@gmail.com

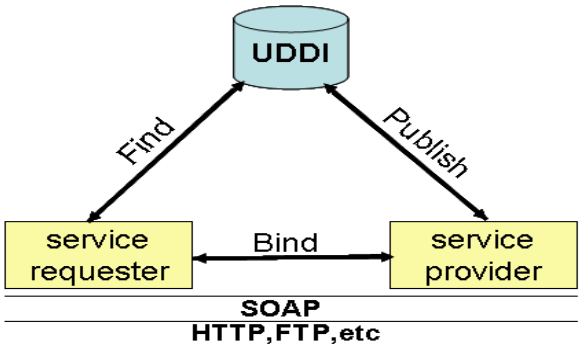



**Figure 1. Deployment, research and invocation of web service**

Hence, three types of interconnection have defined, as illustrated by the the figure 1.

Living in a competitive world, businesses are naturally interested in information technology supporting them for competitive advantage. As cooperation becomes increasingly important for companies, new challenges arise for the support of business to business scenarios by information technology. The emergence of Web services marks the beginning of a new evolution in this context. This development, a priori technical and architectural introduced a true revolution in how to design and build information systems. Therefore, it became possible to develop an application of high-level by federating some Web services already provided on the Internet by various organizations; hence the concept of *composite Web service*.

In actual fact, available services on the Internet have limited functionalities which do not always meet the user requirements. Therefore, services must often be composed to build more complex services to achieve specific user requirements. In their turn, these new created services are potential candidates for another composition [7]. A composite Web service may be distributed over a network, running on different platforms, implemented in different programming languages and offered by different vendors.

A Web service is said to be composite when its execution involves interactions with other Web services in order to call for their functionalities. The composition of Web services specifies which services need to be invoked, in what order, and how to manage the exceptions. Clearly, it describes a business process, involving different web services.

There are several languages to describe a composition which are classified into two major groups: non-semantic approaches and semantic approaches [8]. Within this context, BPEL4WS (Business Process Execution Language for Web Services), has emerged among non semantic approaches, whereas OWL-S (Web Ontology Language for Services) becomes the representative standard of semantic approaches. However, existing models have been also used to model and compose web services, such as: workflows graphs, Petri nets and currently programming languages as Java and C.

The composition languages show two levels of abstraction:

The *abstract level* (the abstract process), where the description of a process does not indicate the internal behavior of the parts involved in the process; and the *executable level* (the executable process), where the description is complete and specifies the order of execution of activities, the list of partners, the messages exchanged and treatment of exceptions. These languages provide two methods of execution: orchestration and choreography.

*Orchestration*: It describes the interaction of services at message level, including the business logics and the order of execution of interactions. The services run independently of the context of business processes. Only the coordinator of the orchestration needs to be aware about the orchestration.

*Choreography*: Unlike the first way to compose Web services, the choreography has no central coordinator. Each Web services involved in the composition knows the conditions of the execution of its operations and with which other services the interaction have to take place.

The *BPEL4WS* has become a standard composition of business processes. It allows the manipulation of services as activities and processes. A process *BPEL4WS*, is a container where we find a list of external partners, declarations of data exchanged with these partners, managing exceptions and more importantly, the list of activities of the process.

The composition can be made at the time of design of composite Web services, or at runtime. At time of design, the composition is manual, and all the web services that take part, are known as well as their execution order. At runtime, the composition is dynamic and automatically performed and only a specification of the required abstract services is given. Hence, services must first be discovered, then selected and integrated. Microsoft BizTalk Server and BEA WebLogic are examples of platforms of static composition and eFlow and StarWSCoP are platforms for dynamic composition [9].

The composition of web services raises complex problem, being given the multitude of web services on the Internet located at different providers with identical functionalities but however with different qualities. Indeed, for both run time and design time compositions, the choice of concrete services for a particular abstract service may be based on non–functional parameters. Examples of such parameters are availability, throughput, response time, security and cost. In actual fact, web services operate in an

environment which is very dynamic (the Web) and thus, their *QoS* values are frequently changing due to updating in elementary services (disappearance of a service for example) or a change in the environment of their execution (a heavy load on the system). This shows the importance of dynamic service composition since services are selected and adapted dynamically at runtime, with services previously discovered to meet user requirements. Consequently, it becomes necessary to integrate, in the web services, such non-functional properties (QoS) in order to distinguish them at runtime [10], [11].

As the market capture of Web services is increasing significantly, in the past years, the applications are quite welcoming the ability to provide secure and reliable communication in the vulnerable and volatile mobile networks. Performance evaluation of these architectures is essential but complex due to synchronization inside the orchestration of services. Consequently, the increasing complexity of such architectures requires the development of methods and tools in order to monitor and evaluate their QoS. In fact, the QoS degradation can lead to serious consequences including a significant economic impact.

We propose in this paper to address this issue. We mainly focus on the composite Web service (CWS) response time computation, where the requests are decomposed into sub-queries to different elementary Web services and then merged into a final result. This includes the following models:

- parallel invocation of a constant number of elementary Web services merged by a federating component;
- parallel invocation of a variable number of elementary Web services merged by a federating component.

To this aim, we consider Stochastic Automata Network (SAN), to model such architectures. Contrarily to other probabilistic models, as Markov's chains for instance, SANs are a very powerful tool as they can specify systems with complex synchronization requirements in a very compact and elegant way. Hence, it becomes possible to specify the composition of a variable number of web services, when it is hard to achieve with other models. To advocate the use of our approach, we provide some empirical studies by simulating the obtained models to compute response times.

This remainder of the paper is organized as follows. In Section 2, we give the related work. In Section 3, we give a brief introduction to SAN model. In section 4, we describe our different models to specify web services composition, where in Section 5, we give some numerical results. Finally, we conclude and give future research perspectives in Section 6.

## 2. Related works

In the literature, performance evaluation of Web services have been conducted either by using tests or formal methods.

As concerns testing Web services, XML specification and SOAP protocol have been studied in [13–15] by testing and measuring their response times. In [13] a comparative study of existing protocols, like RMI, RMI/IIOP or CORBA/IIOP, is presented. A critical study of XML-based protocols for Web services is presented and binary encoded protocol has been proposed instead of text XML-based ones in [14]. In [16], information about past workflow executions is collected in a log. Starting from this log a continuous Markov chain is derived, in order to compute the execution response time and the cost of this workflow.

In [11], the composite Web service response time is considered as a response time of fork and join model. This model states that a single Internet application can invoke in parallel a set of elementary Web services and gather their responses from all these launched services in order to return the results to a client. In this considered study, authors analyze the effects of exponential response times based on their earlier work in [17].

An exact analysis of fork and join system is possible when the system is significantly simplified. This is the case for example when the job arrival process in the system follows a Poisson distribution with execution task having exponential distribution and the number of queues is equal to two. The exact computation response time of a such system can be found in [18], [19] and [20]. An approximation technique has been proposed in the case where the number of servers is greater than two and the servers are homogeneous [20]. This last study is extended in [21]. General arrival process and services times are considered in [22]. The most general case is considered in [23]. In this work, upper and lower bounds are proposed by assuming that the response times in each queue are mutually independent. Two approximation techniques are presented: one is based on a decomposition approach and the other is based on an iterative solution method.

In order to overcome the limitations of these studies and particularly the one presented in [11], we have proposed a general model taking into account the fact that elementary Web services are heterogenous and the number of invoked services can be variable (this is the case when we use for example the BPEL multi-choice constructor) [24]. More recently, the problem of computing the distribution of the throughput time in workflow nets has been studied in [25]. In this paper,

authors consider workflow with transition execution time having exponential distributions, and formulas have been proposed for each refinement rule (sequence, parallel, synchronization and loop execution pattern).

Response time of a Web service middleware is considered in [26], which follows a fork and join model of execution. The author proposes that while performing a join operation, servers with slow response times can be eliminated to maximize the performances. The work is more oriented towards studying fork and join model in order to understand how to optimally merge the results from various servers. In [27] we have proposed a generic transformation of the studied Markov chain which guarantees that the response time of the new Markov chain is an upper bound of the initial Markov chain response time. We instantiate this transformation in three ways, where each obtained new Markov chain is parameterized by a "quantitative" parameter. By an appropriate choice of the parameter, the recurrence equation systems can be resolved with an algorithm with $O(n)$ and $O(n\sqrt{n})$ respectively space complexity and time complexity, where $n$ is the number of invoked elementary Web services. However, some synchronization in the invocation of a variable number of web services can not be handled by the proposed models.

## 3. Stochastic Automata Networks

Stochastic Automata Networks have been introduced as an efficient method to represent complex systems with interacting components such as parallel systems or distributed systems [12]. This method automatically provides an analytic derivation of Markov chain generator matrix using tensor algebra. The SAN seems to be more efficient than Queueing Networks or Stochastic Petri Nets to model systems with a large number of states and complex synchronization.

Queueing Networks give a very compact representation of systems with resource contention among independent customers. Analytical methods and well known algorithms may be used to obtain either analytical or numerical results. However, queueing networks are inefficient whenever complex synchronization constraints are to be taken into account. On the other side, stochastic Petri Nets have been defined to represent synchronization constraints of parallel systems or protocols. However, they do not generally yield compact models as they build the transition matrix without any knowledge about its properties.

In the SAN approach, the dynamic behavior of each system component is modeled by an automaton and the interactions between the different components by labels on the directed edges which may represent synchronization events and transition rates [12]. A Stochastic Automata Network is a set of automata. Each automaton $A_i$ is defined by the tuple $(S_i, L, Q_i)$ where $S_i$ is the set of states of the automaton. $Q_i$ is the transition function of the automaton $A_i$ which associates a label from $L$ to every arc of $A_i$. Labels describe the rate and the type of the transition. The rate may be dependent of the others automata states. There are 2 types of transitions: *local* and *synchronized*. A local transition occurs only within the automaton, whereas the synchronized transition occurs in several automata at the same time. An automaton itself is not Markovian. The Markovian assumption holds only for the global SAN behavior if we assume exponential distribution and independence for the firing of the transitions.

The transition rate matrix $Q$ is automatically translated from the SAN description. This translation is based on tensor algebra of matrices (see [12] for more details and proofs).

First let us define tensor operators:

**Definition 1.** Let $A$ be a matrix of order $n \times n$, and $B$ a matrix of order $p \times p$. The **tensor product** of $A$ and $B$ is a matrix $C$ of order $np \times np$ such that $C$ may be decomposed into $n^2$ blocks of size $p$.

$$C = A \bigotimes B = \begin{bmatrix} a_{11}B & \dots & \dots & a_{1n}B \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ a_{n1}B & \dots & \dots & a_{nn}B \end{bmatrix}$$

$A$ and $B$ are matrices of real values but we generalize the definition of tensor product on matrices of functional values (i.e. the elements of $A$ and $B$ are functions using states as arguments). Some properties of the classical tensor product still hold.

**Definition 2.** Let $A$ be a matrix of order $n \times n$, and $B$ a matrix of order $p \times p$. The **tensor sum** of $A$ and $B$ is defined by :

$$E = A \bigoplus B = A \bigotimes I_B + I_A \bigotimes B$$

where $I_D$ represents the identity matrix with the same size as matrix $D$.

It has been proved in [12] that, if the states are in a lexicographic order, then the generator matrix $Q$ of the Markov chain associated to a continuous-time SAN is given by:

$$Q = \bigoplus_{i=1}^{n} F_i + \sum_{j=1}^{c} \bigotimes_{i=1}^{n} S_{i,j} + \sum_{j=1}^{c} \bigotimes_{i=1}^{n} R_{i,j}$$

Where:

- $n$ is the total number of automata in the network and $c$ is the number of synchronization.
- $F_i$ is the transition matrix of automaton $i$ without synchronization.

- $S_{i,j}$ is the transition matrix of automaton $i$ due to synchronization $j$.
- $R_{i,j}$ is a matrix representing the normalization associated to the synchronization $j$ on automaton $i$.
- $\bigoplus$ and $\bigotimes$ denote tensor sum and product, respectively.

The transition matrix $P$ of a Markov chain associated to a discrete-time Stochastic Automata Network can be obtained by a slightly different formula, given by:

$$P = \bigotimes_{i=1}^{n} F_i + \sum_{j=1}^{c} (\bigotimes_{i=1}^{n} S_{i,j} - \bigotimes_{i=1}^{n} R_{i,j}$$

The main advantage of this methodology is its ability to represent the Markov chain associated to the SAN model by a compact formula. This point is particularly important since it allows us to deal with systems which may have very large state spaces. In the following section, we show how we model our system using the SAN methodology.

## 4. Considered composite Web service model

We focus on the composite Web service (CWS) response time computation, where the requests are decomposed into sub-queries to different elementary Web services and then merged into a final result. The control patterns considered here are not directly supported by BPEL [28]:

- parallel invocation of a constant number of elementary Web services merged by a federation component. This model is described in section 4.1.
- parallel invocation of a variable number of elementary Web services merged by a federation component. This model is described in section 4.2.

### 4.1. Case $n$ is constant

We consider a composite Web service where the data is stored in databases and can be accessed using XML-based protocols noted $s_i$ for $1 \leq i \leq n$. We assume that when a composite Web service is invoked, $n$ elementary Web services are invoked in parallel and the partial responses are then integrated into the global response to provide to the client. We assume that the arrival of the composite Web services follow a Poisson process with rate $\lambda$. The response times of the servers $s_i$ for $1 \leq i \leq n$ are also assumed to be of exponential distributions with rate $\mu_i$ for $1 \leq i \leq n$. We assume that the merging time of the $n$ elementary Web services is an exponential distribution with rate $\mu$ (batch service).

As we have defined the behavior of the system as Markov process, the considered model can be described by a continuous time Markov chain denoted by $X(t)$. To describe this chain, we define the state $x$ by $(x_1, s_1, s_2, \ldots, s_n, a_1, a_2, \ldots, a_n)$, where:

- $x_1$ is the number of the composite Web services requests.
- $s_1, s_2, \ldots, s_n$ are the elementary Web services in each queue.
- $a_1, a_2, \ldots, a_n$ are the elementary Web service responses in the related queue waiting to be merged in the global responses to give back to the clients.

**Associate queueing model** Hereafter, we define the queueing model when $n$ is constant. Our model can be specified by queuing network. We have one finite buffer for requests with size $b$. $n$ queues with finite buffers with size $b_i$ for $1 \leq i \leq n$. $n$ others queues with finite buffers with size $b_i$ for $1 \leq i \leq n$ where the elementary Web services waiting for the batch service.

To illustrate the behavior of the system, we give the behavior equations of the considered model in the following:

$x \rightarrow$ CWS arrival
$\rightarrow (x_1 + 1, s_1, s_2, \ldots, s_n, a_1, a_2, \ldots, a_n)$
$with\ rate\ \lambda \times \mathbb{1}_{x_1 < b}$
$\rightarrow$ Decomposition of CWC into n elementary WS
$\rightarrow (x_1 - 1, s_1 + 1, s_2 + 1, \ldots, s_n + 1, a_1, a_2, \ldots, a_n)$
$with\ rate\ \mu_\lambda \times \mathbb{1}_{x_1 > 0} \times \mathbb{1}_{s_i < b_i}$
$\rightarrow$ End of service of WS1
$\rightarrow (x_1, s_1 - 1, s_2, \ldots, s_n, a_1 + 1, a_2, \ldots, a_n)$
$with\ rate\ \mu_1 \times \mathbb{1}_{s_1 > 0} \times \mathbb{1}_{a_1 < b_1}$
$\rightarrow$ End of service of WS2
$\rightarrow (x_1, s_1, s_2 - 1, \ldots, s_n, a_1, a_2 + 1, \ldots, a_n)$
$with\ rate\ \mu_2 \times \mathbb{1}_{s_2 > 0} \times \mathbb{1}_{a_2 < b_2}$
$\rightarrow \ldots$
$\rightarrow$ End of service of WSn
$\rightarrow (x_1, s_1, s_2, \ldots, s_n - 1, a_1, a_2, \ldots, a_n + 1)$
$with\ rate\ \mu_n \times \mathbb{1}_{s_n > 0} \times \mathbb{1}_{a_n < b_n}$
$\rightarrow$ Synchronization of n WSs, so response of CWS
$\rightarrow (x_1, s_1, s_2, \ldots, s_n, a_1 - 1, a_2 - 1, \ldots, a_n - 1)$
$with\ rate\ \mu \times \mathbb{1}_{a_i > 0}$

### 4.2. Case $n$ is variable

We assume for this model that once a composite Web service (CWS) is requested, it can be decomposed into $k$ Elementary Web Services (EWS) where $1 \leq k \leq n$.

The case is a generalization of the precedent case ($n$ constant).

Thus we define the following probabilities :

- $p_1$ is the probability that CWB, is decomposed into one elementary Web service
- $p_2$ is the probability that CWB, is decomposed into two elementary Web services
- $p_k$ is the probability that CWB, is decomposed into $k$ elementary Web services
- $p_n$ is the probability that CWB, is decomposed into $n$ elementary Web services, which is the case of the first model.

**Associate queueing model** The considered model can also be described by a continuous time Markov chain denoted by $X(t)$. To describe this chain, we define the state $x$ by $(x_1, s_1, s_2, \ldots, s_n, a_1, a_2, \ldots, a_n)$, where:

- $x_1$ is the number of the composite Web services requests
- $s_1, s_2, \ldots, s_n$ are the elementary Web services in each queue.
- $a_1, a_2, \ldots, a_n)$ are the number elementary Web services responses in each queue waiting for merging and giving the responses to the clients.

As the system is complex, we give the behavior equations only for the case that a CWS can be decomposed into 1, 2 or 3 EWS ($n = 3$). We denote by $s_1$, $s_2$ and $s_3$ the elementary Web service that can compose the CWS denoted by $Sw$. Thus we have the following combinations:

- $Sw$ is composed only by $s_1$ with probability $p_1$
- $Sw$ is composed only by $s_2$ with probability $p_2$
- $Sw$ is composed only by $s_3$ with probability $p_3$
- $Sw$ is composed by $s_1$ and $s_2$ with probability $p_{12}$
- $Sw$ is composed by $s_1$ and $s_3$ with probability $p_{13}$
- $Sw$ is composed by $s_2$ and $s_3$ with probability $p_{23}$
- $Sw$ is composed by $s_1$, $s_2$ and $s_3$ with probability $p_{123}$

Thus, the behavior equations are given as follows: $x$

$\rightarrow$ CWS arrival

$\rightarrow$ $(x_1 + 1, s_1, s_2, s_3, a_1, a_2, a_3)$
$with\ rate\ \lambda \times \mathbb{1}_{x_1<b}$

$\rightarrow$ Decomposition of CWS into EWS 1

$\rightarrow$ $(x_1 - 1, s_1 + 1, s_2, s_3, a_1, a_2, a_3)$
$with\ rate\ \mu_\lambda \times p_1 \times \mathbb{1}_{x_1>0} \times \mathbb{1}_{s_1<b_1}$

$\rightarrow$ Decomposition of CWS into EWS 2

$\rightarrow$ $(x_1 - 1, s_1, s_2 + 1, s_3, a_1, a_2, a_3)$
$with\ rate\ \mu_\lambda \times p_2 \times \mathbb{1}_{x_1>0} \times \mathbb{1}_{s_2<b_2}$

$\rightarrow$ Decomposition of CWS into EWS 3

$\rightarrow$ $(x_1 - 1, s_1, s_2, s_3 + 1, a_1, a_2, a_3)$
$with\ rate\ \mu_\lambda \times p_3 \times \mathbb{1}_{x_1>0} \times \mathbb{1}_{s_3<b_3}$

$\rightarrow$ Decomposition of CWS into EWS 1 and EWS 2

$\rightarrow$ $(x_1 - 1, s_1 + 1, s_2 + 1, s_3, a_1, a_2, a_3)$
with rate $\mu_\lambda \times p_{12} \times \mathbb{1}_{x_1>0} \times$
$\mathbb{1}_{s_1<b_1} \times \mathbb{1}_{s_2<b_2}$

$\rightarrow$ Decomposition of CWS into EWS 1 and EWS 3

$\rightarrow$ $(x_1 - 1, s_1 + 1, s_2, s_3 + 1, a_1, a_2, a_3)$
with rate $\mu_\lambda \times p_{13} \times$
$\mathbb{1}_{x_1>0} \times \mathbb{1}_{s_1<b_1} \times \mathbb{1}_{s_3<b_3}$

$\rightarrow$ Decomposition of CWS into EWS 2 and EWS 3

$\rightarrow$ $(x_1 - 1, s_1, s_2 + 1, s_3 + 1, a_1, a_2, a_3)$
with rate $\mu_\lambda \times p_{23} \times$
$\mathbb{1}_{x_1>0} \times \mathbb{1}_{s_2<b_2} \times \mathbb{1}_{s_3<b_3}$

$\rightarrow$ Decomposition of CWS into EWS 1, EWS 2 and EWS 3

$\rightarrow$ $(x_1 - 1, s_1 + 1, s_2 + 1, s_3 + 1, a_1, a_2, a_3)$
$with\ rate\ \mu_\lambda \times p_{123} \times$
$\mathbb{1}_{x_1>0} \times \mathbb{1}_{s_1<b_1} \times \mathbb{1}_{s_2<b_2} \times \mathbb{1}_{s_3<b_3}$

$\rightarrow$ End of service EWS1

$\rightarrow$ $(x_1, s_1 - 1, s_2, s_3, a_1 + 1, a_2, a_3)$
with rate $\mu_1 \times \mathbb{1}_{s_1>0} \times \mathbb{1}_{a_1<B_1}$

$\rightarrow$ End of service EWS2

$\rightarrow$ $(x_1, s_1, s_2 - 3, s_3, a_1, a_2 + 1, a_3)$
$with\ rate\ \mu_2 \times \mathbb{1}_{s_2>0} \times \mathbb{1}_{a_2<B_2}$

$\rightarrow$ End of service EWS3

$\rightarrow$ $(x_1, s_1, s_2, s_3 - 1, a_1, a_2, a_3 + 1)$
with rate $\mu_3 \times \mathbb{1}_{s_3>0} \times \mathbb{1}_{a_3<b_3}$

$\rightarrow$ Response to CWS composed only by EWS1

$\rightarrow$ $(x_1, s_1, s_2, s_3, a_1 - 1, a_2, a_3)$
with rate $\mu \times p_1 \times \mathbb{1}_{a_1>0}$

$\rightarrow$ Response to CWS composed only by EWS2

$\rightarrow$ $(x_1, s_1, s_2, s_3, a_1, a_2 - 1, a_3)$
with rate $\mu \times p_2 \times \mathbb{1}_{a_2>0}$

$\rightarrow$ Response to CWS composed only by EWS3

$\rightarrow$ $(x_1, s_1, s_2, s_3, a_1, a_2, a_3 - 1)$
with rate $\mu \times p_3 \times \mathbb{1}_{a_3>0}$
$\rightarrow$ Response to CWS composed only by EWS1and EWS2
$\rightarrow$ $(x_1, s_1, s_2, s_3, a_1 - 1, a_2 - 1, a_3)$
*with rate* $\mu \times p_{12} \times \mathbb{1}_{a_1>0} \times \mathbb{1}_{a_2>0}$
$\rightarrow$ Response to CWS composed only by EWS1and EWS3
$\rightarrow$ $(x_1, s_1, s_2, s_3, a_1 - 1, a_2, a_3 - 1)$
with rate $\mu \times p_{13} \times \mathbb{1}_{a_1>0} \times \mathbb{1}_{a_3>0}$
$\rightarrow$ Response to CWS composed only by EWS2 and EWS3
$\rightarrow$ $(x_1, s_1, s_2, s_3, a_1, a_2 - 1, a_3 - 1)$
*with rate* $\mu \times p_{23} \times \mathbb{1}_{a_2>0} \times \mathbb{1}_{a_3>0}$
$\rightarrow$ Response to CWS composed only by EWS1, EWS2 and EWS3
$\rightarrow$ $(x_1, s_1, s_2, s_3, a_1 - 1, a_2 - 1, a_3 - 1)$
with rate $\mu \times p_{123} \times \mathbb{1}_{a_i>0}$

## 4.3. Associated SAN

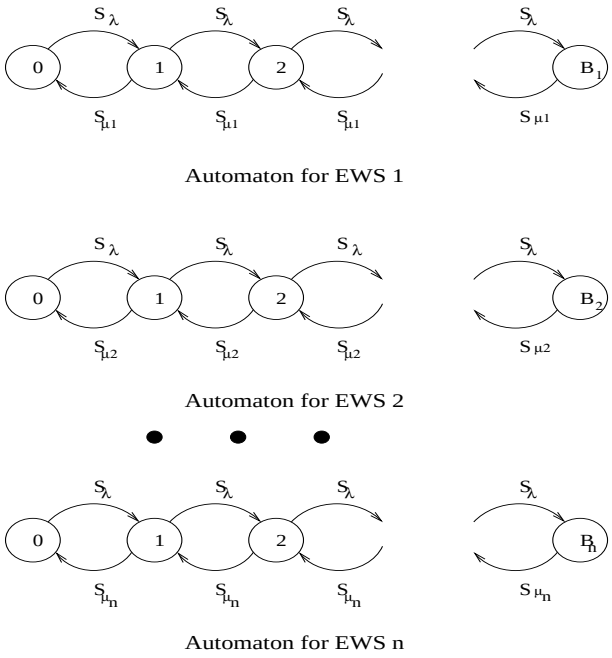



**Figure 2. stochastic Automata for elementary service execution**

We present in this section how we specify our model using SAN for the both considered models.

## 4.4. Case $n$ is constant

Before giving the automata, we give the description of the different synchronization in the following.

- $S_\lambda$ is a synchronization which corresponds to the fork which means that we decompose the request (CWS) into $n$ elementary Web services. Thus, its rate is equal to $\lambda$.

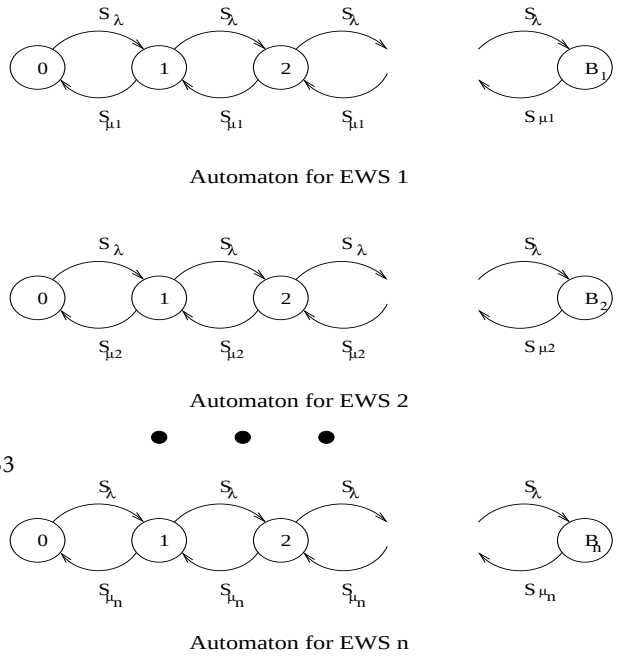



**Figure 3. stochastic Automata for elementary service execution**

- $S_{\mu_1}$ is a synchronization which corresponds to the service of elementary Web service from server 1. Its rate is equal to $\mu_1$.
- Thus, we can note that $S_{\mu_i}$ is the synchronization which corresponds to the service of elementary Web service from server $i$ for $1 \leq i \leq n$. Hence, its rate is equal to $\mu_i$.
- $S_\nu$ is a synchronization which corresponds to the batch service of elementary Web service from all servers. Thus, its rate is equal to $\nu$.

For the considered model, we need several automata as described in the following:

- We need one automaton for request arrival which decompose the CWS into $n$ EWS. The action is done by the synchronization $S_\lambda$. The considered automaton is given in figure 4.
- We describe one automaton for each servers where are executed the elementary Web services. As we consider that a CWS can be decomposed in $n$ EWS, we need $n$ automata. These automata are described in figure 3.
- We also need one automaton for each EWS waiting for the batch service. The batch service is provided by synchronization $S_\nu$ as shown in figure 5.

In the following, we give the associated stochastic automata for the case where $n$ is variable.

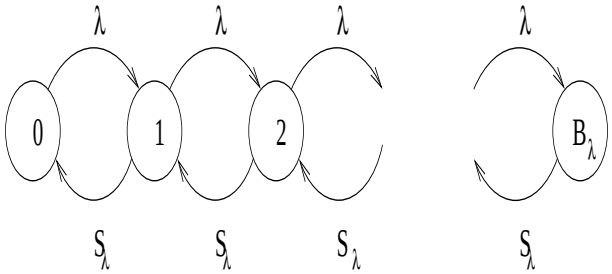



Figure 4. Automaton for request arrival

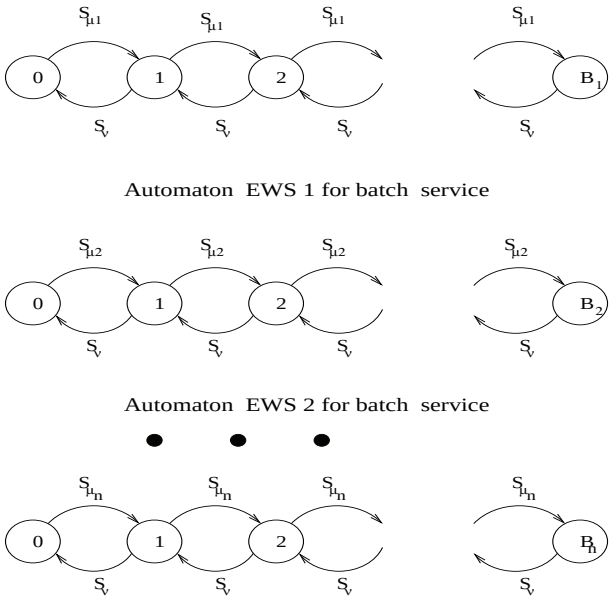



Figure 5. Automata for batch service

## 4.5. Case $n$ is variable

For a sake of readability, we represent only the case where a CWS can be decomposed into 1, 2 or 3 EWS ($n = 3$) and where several automata are needed. Before we describe this modeling, we give the used synchronization:

- The synchronization $S_1, S_2, \ldots, S_{123}$ are used for the decompositions of CWS into 1, 2 or 3 EWS ($n = 3$).
- The synchronization $S_{\mu 1}, S_{\mu 2}, S_{\mu 3}$ are used for the service of EWS
- The synchronization $S_{\nu 1}, S_{\nu 2}, \ldots, S_{\nu 123}$ are used for the merge of 1, 2 or 3 EWS.

We present, now, the associated stochastic automata network.

- We need one automaton for request arrival, which decomposes the CWS into $n$ EWS. The action is done by the synchronization $S_\lambda$. The considered automaton is given in figure 6.

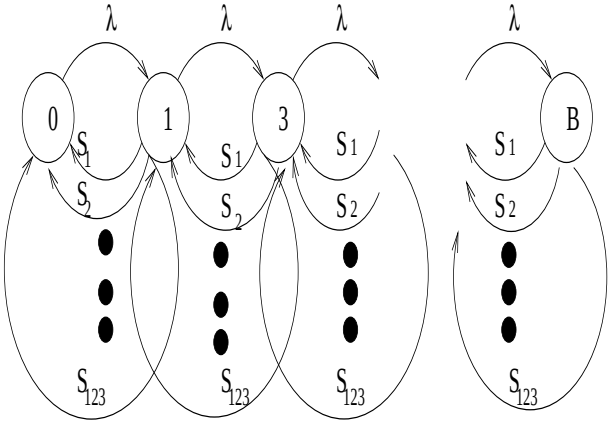



Figure 6. Automaton for request arrival

- We need one automaton for each server where are executed the elementary Web services. As we consider that a CWS can be decomposed in $n$ EWS, where $n$ is variable, we use the corresponding synchronization. These automata are described in figure 7.
- We also need one Automaton for each EWS waiting for the batch service. The batch service is provided by synchronization $S_{\nu 1}, S_{\nu 2}, \ldots, S_{\nu 123}$ as shown in figures 8, 9. and 10.

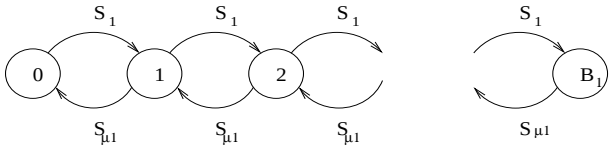
Once the automata are built, we develop, first, for each automata the local and synchronization matrices. Then we apply the compact formula.

$$Q = \bigoplus_{i=1}^{n} F_i + \sum_{j=1}^{c} \bigotimes_{i=1}^{n} S_{i,j} + \sum_{j=1}^{c} \bigotimes_{i=1}^{n} R_{i,j}$$

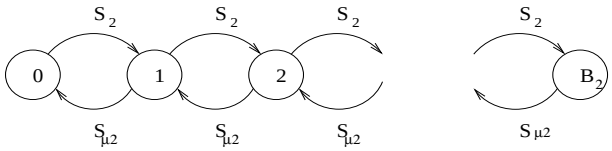
The latter makes it possible to compute the rewards as it is discussed in the next section.

## 5. Numerical results

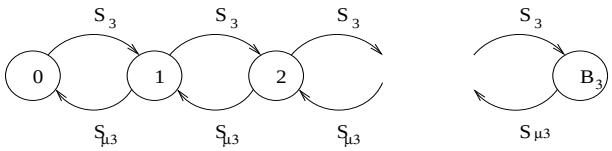
In this paper, we are interested in assessing the response times as it is most important Qos parameter in the performance evaluation of Web services architecture. We have solved the previous obtained models using the numerical method (Gauss-Seidel) to obtain the steady-state distribution which provides the performance measures. The latter give the response times and the mean requests ratio of the system. In figure 11, we have plotted the response times results according to the system load by considering that a composite Web service can be decompose into $n$ elementary Web services where $n$ is a constant. As we expected, the response times increase when the system load increases. In figure 12, we have considered the basic model ($n$




Automaton for EWS 1




Automaton for EWS 2




Automaton for EWS 3

**Figure 7. Automata for elementary Web services**

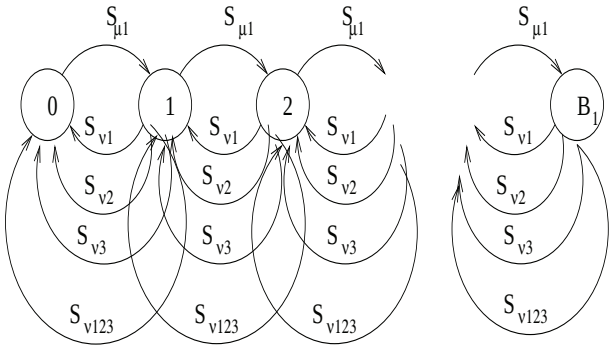



**Figure 8. Automate A1**

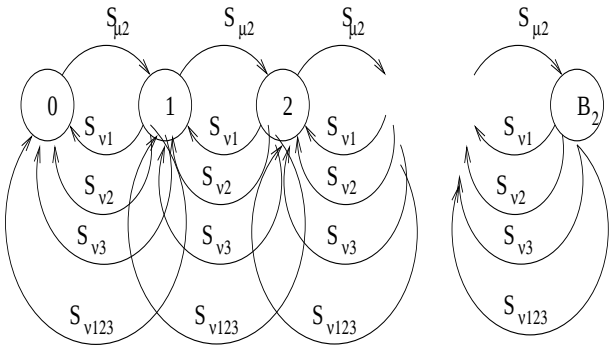



**Figure 9. Automate A2**

constant) and the general model ( *n* variable); response

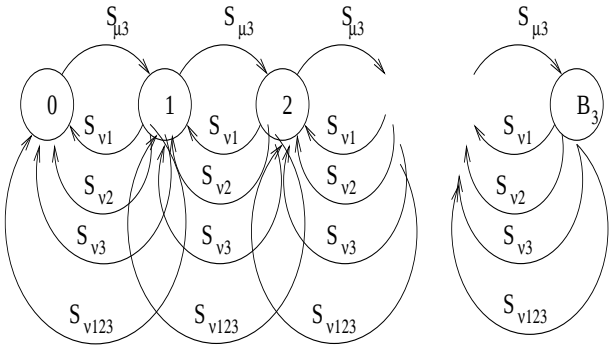



**Figure 10. Automate A3**

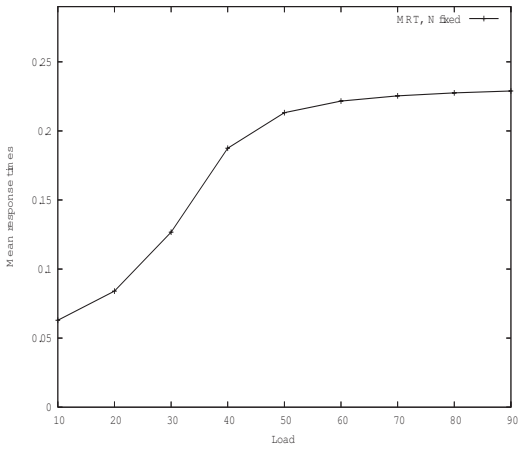



**Figure 11. Mean Response times for the case n is constant.**

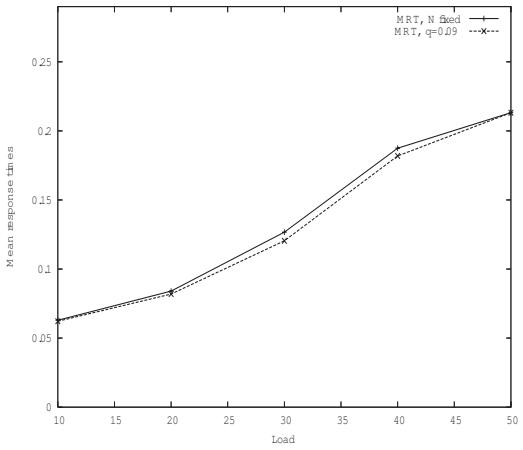



**Figure 12. Mean Response times for both cases.**

times according to the system load are reported, thus comparing both models. In figure 13, we compare the performances of both models by computing the mean request ratio of both models. As we can see the variability of the number of composed elementary web services has no effect on the behavior of the system comparing where *n* is fixed.

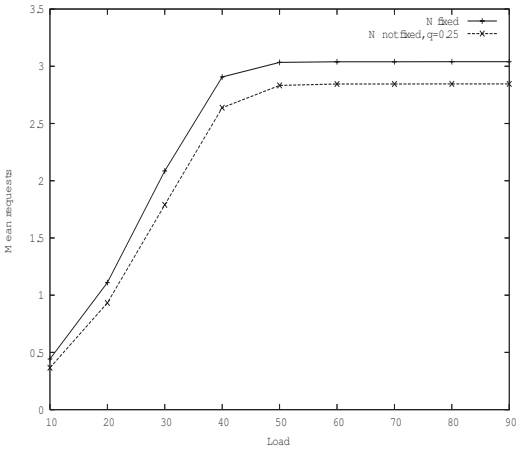



**Figure 13. Mean Requests ratio of both cases.**

## 6. Conclusion

The objective of this paper was to present the SAN tool and to show how it can be used to evaluate Web service architectures. Indeed, the proliferation of Web services on the Internet and their interoperable nature have shown very quickly the importance of having the QoS in their operating model. This enables the distinction between them and provides the users with tools to select those which are the most suitable for their needs. The performance evaluation of such systems is therefore crucial to guarantee their reliability and their QoS requirements. Because of the complexity in the modeling of such requirements, new tools and methods are therefore needed to handle complex synchronization as well as variable number of elementary services.

In this paper we have investigated the use of stochastic automata networks and we have shown how we can use this formalism to model and evaluate these systems. The main objective was to compute composite Web services response times when the number of invoked elementary Web services can be variable. Future work will lead us to generalize the study by taking into account more complex patterns (e.g. hierarchical composite Web services). Secondly, we plan to consider time requirements in the modeling as well as in the performance evaluation of Web service architectures.

## References

[1] D. Tidwell, "Web services - The Web's next revolution", IBM developerWorks, 2000

[2] David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, and Chris Ferris David Orchard. Web Service Architecture. Published on the internet, February 2004. URLhttp://www.w3.org/TR/ws-arch/. W3C Recommendation.

[3] M. Hu, "Web Services Composition, Partition, and Quality of Service in Distributed System Integration and Re-engineering", In IEEE, Internet Computing, 2005.

[4] M. Gudgin, and M. Hadley, and N. Mendelsohn, and J. Moreau and H. Nielsen, "Simple Object Access Protocol (SOAP) 1.1", World Wide Web Consortium,

[5] T. Bellwood, and L. Clément, and C. von Riegen, "Universal Description, Discovery and Integration", OASIS UDDI Specification Technical Committee,

[6] E. Christensen, and F. Curbera, and G. Meredith and S. Weerawarana, "Web Services Description Language (WSDL) 1.1", World Wide Web Consortium,

[7] Francisco Curbera and Rania Khalaf and Nirmal Mukhi and Stefan Tai and Sanjiva Weerawarana, The next step in Web services, Commun. ACM journal, volume 46, number 10, pages 29-34, year 2003.

[8] Daniela Claro, Patrick Albers and Jin-Kao Hao, Web services composition, In J. Cardoso, Sheth, Amit (Eds.), Semantic Web Services, Processes and Applications » Chapter 8, 2006.

[9] S. Dustdar, W. Schreiner, A survey on web services composition, International Journal Web and Grid Services 1, pages 1-30, 2005

[10] D.A. Menascé. QoS Issues in Web Services", IEEE Internet Computing, vol. 6, no. 6, pp. 72-74, 2002.

[11] D.A. Menascé et al., Response Time Analysis of Composite Web Services," IEEE Internet Computing, vol. 8, no. 1, pp.90-92, January/February 2004.

[12] B. Plateau and J.M. Fourneau "A Methodology for Solving Complex Markov Models of Parallel Systems", Journal of Parallel and Distributed Computing, Vol. 12, pp. 370-387, 1991.

[13] D. Davis and M. P. Parashar, "Latency performance of soap implementations", In CCGRID '02: Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, page 407, Washington, USA, IEEE Computer Society, pp. 407-415, 2002.

[14] C. Kohlhoff and R. Steele, "Evaluating soap for high performance business applications: Real-time trading systems", In Proceedings of WWW, pp. 1-8, 2003.

[15] P. Sandoz, S. Pericas-Geertsen, K. Kawaguchi, M. Hadley, and E. Pelegri-Llopart, "Fast web services", 2009.

[16] J. Klingemann, J. Wäsch, and K. Aberer, "Deriving Service Models in Cross-Organizational Workflows", In proceedings of RIDE -Information Technology for Virtual Enterprises, Sydney, Australia, pp. 100-107, 1999.

[17] D.A. Menascé et al., "Static and Dynamic Processor Scheduling Disciplines in Heterogeneous Parallel Architectures," J. Parallel and Distributed Computing, vol. 28, no. 1, pp. 1-18, 1995.

[18] S. Hahn and L. Fatto, "Two parallel queues created by arrivals with two demands", Applied Mathematics, Vo. 44, pp. 1041-1053, October, 1984.

[19] L. Fatto, "Two parallel queues created by arrivals with two demands II", Applied Mathematics, Vo. 45, pp. 861-878, October, 1985.

[20] R. Nelson and A.N. Tantawi, "Approximate Analysis of Fork/Join Synchronization in Parallel Queues", IEEE Transaction Computer, vol. 37, No. 6, pp. 739-743, 1998.

[21] A. Makawski and S. Verma, "Interpolation approximations for symmetric Fork-Join Queues", Perform. Evaluation journal. 20(1-3), pp. 245-265, 1994.
[22] F. Bacelli and A.M. Makowski, "Simple computable bounds for the fork-join queue", Proceeding Information Science, pp. 436-441, March, 1985.
[23] P. Heidlberg and K.S. Trivedi, "Analytic queuing models for programs with internal concurrency", IEEE Trans. Computer, vol. C-32, pp. 73-82, Nov, 1993.
[24] S. Haddad, L. Mokdad and S. Youcef, Response time analysis of composite Web services", Communication Systems, Networks and Digital Signal Processing (CSNDSP), IEEE Computer Society, Graz University of Technology, pp. 42-49, July 2008.
[25] C.W. Piotr, F.Pawel and W. Grzegorz, "Time Distribution in Structural Workflow Nets", Fundam. Inf., vol. 85, n. 1-4, IOS Press, Amsterdam, Netherlands, pp. 67-87, 2008.
[26] M. Sharf, "On the response time of the large-scale composite Web services", Proceedings of the 19th International Teletraffic Congress (ITC 19), Beijing, pp. 1807-1816, 2005.
[27] S. Haddad, L. Mokdad, S. Youcef: Bounding models families for performance evaluation in composite Web services. J. Comput. Science 4(4): 232-241 (2013)
[28] S. Weerawarana and F. Curbera, "Business Process Execution Language for Web Services", IBM Corporation, 2002.