IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 4, No 6, July 2010
ISSN (Online): 1694-0784
ISSN (Print): 1694-0814

QoS Assurance for Service-Based Applications Using Discrete-Event Simulation

Yassine Jamoussi[1], Maha Driss[1,2], Jean-Marc Jézéquel[2], and Henda Hajjami Ben Ghézala[1]

[1]ENSI, RIADI-GDL Laboratory, University of Manouba
La Manouba, 2010, Tunisia

[2]IRISA/INRIA, University of Rennes I
Rennes, 35042, France

Abstract

The new paradigm for distributed computing over the Internet is that of Web services. The goal of Web services is to achieve universal interoperability between applications by using standardized protocols and languages. One of the key ideas of the Web service paradigm is the ability of building complex and value-added service-based applications by composing pre-existing services. For a service-based application, in addition to its functional requirements, Quality of service (QoS) requirements are important and deserve a special attention. In this paper, we introduce a discrete-event modeling approach for service-based application. This approach is oriented towards QoS assurance through discrete-event simulation.

Keywords: *Web Services, Service-based applications, QoS assurance, Discrete-event simulation.*

1. Introduction

In the last ten years, the Service-Oriented Architecture (SOA) emerged as a powerful solution to enable interoperability between distributed software components known as Web Services (WSs) [1, 2]. WSs are universally accessible software components that are advertised, discovered, and invoked through the Web. The key aspect of the SOA is the use of standard technologies such as: WSDL, UDDI, and SOAP. These technologies define standard ways of WSs definition, discovery, and invocation.

SOA is the best solution for composite application integration. Indeed, WSs may be easily composed/aggregated together into a new application, regardless specific implementation platforms and technologies [3]. The obtained Service-Based Application (SBA) may be further published as a new service creating a collaboration network between different organizations. For example, telecommunication companies can be considered as an example of service aggregators [4]. Multiple and different services such as calling services (e.g., call forwarding and call barring), messaging services (e.g., text messaging and video messaging), and internet services (e.g., chat and e-mail) are brought together and offered via telephone.

For an SBA, in addition to its functional requirements, Quality of Service (QoS) requirements are important and deserve a special attention. QoS requirements for SBAs include response time, throughput, availability and security [4, 5]. Being able to characterize SBAs based on QoS has three distinct advantages [6]:

- It allows for the design of SBAs according to QoS requirements. Indeed, it is important for service providers to know the QoS of a SBA at prior before offering it to their clients.
- It allows for the selection and the execution of SBAs based on their QoS. Since many services provide overlapping or identical functionality, different SBAs can be composed, satisfying the same functional requirement. A choice needs to be made to determine which SBA is to be used to provide with the more beneficial QoS.
- It allows for the evaluation of alternative adaptation strategies. The dynamic and unpredictable nature of the execution environment (e.g., network resources and devices characteristics) has an important impact on QoS of SBAs. Thus, in order to better fulfill QoS requirements, it is necessary to adapt SBAs in response to an unexpected evolution of the execution environment.

To assure the desired QoS requirements for an SBA, different analytical quality assurance techniques can be used. The goal of these techniques is to evaluate QoS and uncover quality defects in the SBAs after they have been created. An example for analytical quality assurance techniques is simulation. The goal of the simulation technique is to emulate the conversational behavior of the atomic WSs of an SBA. In this paper, we adopt a special case of simulation that is Discrete-Event Simulation (DES) to assure QoS of SBAs. DES has proved its effectiveness for diagnosing the QoS of software applications [7, 8]. To perform DES, we propose a discrete-event modeling approach for SBAs. This approach enables analytical description of SBAs and allows QoS evaluation in different status and conditions of the execution environment. To assure QoS evaluation, we define a lightweight quality model for WSs focusing on essential properties of QoS that play a critical role for the effective management of WSs. These properties are measured by DES technique. We propose also a context model that supports an explicit description of the execution environment. This model is depicted into the simulation model in order to provide a context-based approach for evaluating QoS of SBAs. Our approach is supported by a simulation framework named *SBAS.*

The remainder of the paper is structured as follows: Section 2 introduces an overview of the Web services architecture. Section 3 reviews QoS assurance techniques for SBAs. In section 4, discrete-event simulation issues are addressed. We describe, in Section 5, the proposed context model. In Section 6, we introduce our quality model and explain metrics used to measure considered QoS properties. Section 7 introduces our simulation framework *SBAS.* Case study experimental results are documented in Section 8. This paper ends with concluding remarks and future work.

2. Web Services Architecture Overview

SOA is an architecture that functions are defined as WSs. According to [1, 2], WSs are self-contained, modular applications that can be described, published, located, and invoked over a network, generally, the World Wide Web. The SOA is described through three different roles: service provider, service requester and service registry. SOA requires three fundamental operations: publishing, finding, and binding. The key idea of the SOA is the following: A service provider publishes services in a service registry. The service requester searches for a service in the registry. He finds one or more by browsing or querying the registry. The service requester uses the service description to bind a service. These ideas are shown in the following figure 1. The above operations are supported by standard technologies that are: UDDI, WSDL, and SOAP [2].

- Universal Description, Discovery, and Integration (UDDI) [9]: provides a registry where service providers can register and publish their services.
- Web Services Description Language (WSDL) [10]: is an XML based language for describing WSs. It specifies the location of the WS and the operations exposed by the WS.
- Simple Object Access Protocol (SOAP) [11]: is an XML based protocol for exchanging information between WSs or between a client and a WS in a decentralized and distributed environment.

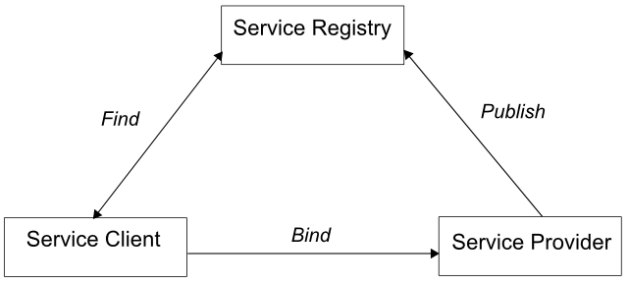

Fig. 1 Service-Oriented Architecture

What makes the SOA attractive is the ability of creating SBAs by composing existing WSs. Such a composition is based on the common standards of WS interfaces regardless of the languages that are used to implement the WSs and the platforms where the WSs are executed. In general, the WSs have the following features that make them better in composition inside the heterogeneous environments [3]:

- Loosely coupled: WSs are autonomous and can operate independently one from another. The loosely coupled feature enables WSs to locate and communicate with each other dynamically at runtime.
- Universal accessibility: WSs can be defined, described and discovered through the Web that enables an easy accessibility.
- Standard languages: WSs are described by standard XML languages that have been considered as parts of the Web technology.

3. Quality of Service Assurance techniques for Service-Based Applications

By QoS, we refer to non-functional requirements of SBAs such as response time, throughput, availability, and security [4, 5]. Thanks to the dynamic and unpredictable nature of the execution environment, the management and assurance of the QoS aspects of SBAs become of utmost importance.
To achieve the desired QoS of an SBA, two complementary kinds of techniques can be employed: constructive and analytical quality assurance techniques. Figure 2 provides an overview of these techniques.

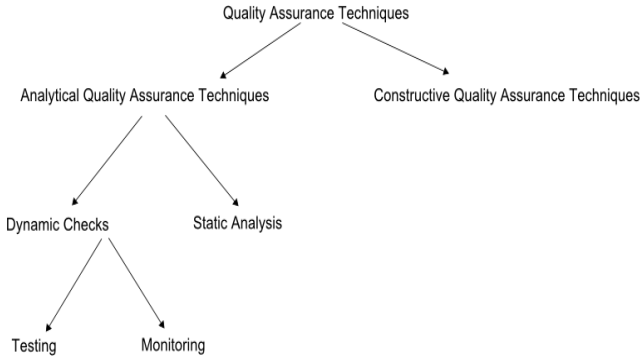

Fig. 2 Overview of Quality Assurance Techniques for Service-Based Applications

The goal of constructive quality assurance techniques is to ensure QoS and prevent the introduction of quality defects while the SBA is created. Examples of such techniques include code generation, software development guidelines, as well as templates. The goal of analytical quality assurance techniques is to evaluate QoS and uncover quality defects in an SBA after it has been created. We sub-divide the analytical quality assurance techniques into three major classes: static analysis, monitoring, and testing. These classes have been proposed in the software quality assurance literature [12, 13] and have been used in a recent overview of quality assurance approaches for SBAs [14].
This section will cover the state of the art in analytical quality assurance techniques for SBAs as our work deals with proposing DES as a new analytical testing technique to assure QoS for SBAs.

3.1 Static Analysis

Numerous efforts have been made by leading research groups to use static analysis to evaluate QoS and to uncover quality defects in SBAs.

The aim of static analysis is to systematically examine an SBA in order to ascertain whether some predefined QoS properties are met. Examples of static analysis techniques include formal ones, like data flow analysis, model checking, symbolic execution, type checking and correctness proofs. We present, in the following paragraphs, some relevant approaches applying static analysis to SBAs.
In [15], Nakajima uses the model-checker SPIN to verify a set of QoS properties related to SBAs. SPIN provides a specification language that describes the SBA to be a collection of automata. The properties to be checked are reachability, deadlock, and freedom. These QoS properties are expressed as formulas of linear temporal logic.

In [16], *Salaün et al.* propose an approach that uses process algebra as an abstract representation means to describe, compose, and reason on SBAs. The techniques used to check whether an SBA described in process-algebraic notations respects temporal QoS properties (e.g. safety and liveness) are referred to as model checking methods.
In [17], Foster *et al.* propose the tool LTSA-WS to verify SBAs. This tool supports verification of QoS properties (e.g. absence of deadlock and liveness) created from design specifications and implementation models of SBAs to confirm expected QoS results from the viewpoints of both the designer and implementer. Scenarios are modeled in UML, in the form of message sequence charts, and then compiled into the finite state process algebra to concisely model the required choreography behavior and to verify the required QoS properties.
In [18, 19], Kazhamiakin *et al.* and Osterweil address the problem of the verification and the analysis of SBAs defined as a set of behavioral models against various QoS requirements. The works focus on modeling and analyzing specific QoS behavioral properties of SBAs, namely asynchronous communications, data and time-related properties. In [18], Kazhamiakin *et al.* present a framework which relies on a formal model where temporal logics are exploited for the specification and the verification of the above QoS behavioral properties.

3.1 Monitoring

Monitoring has been widely used in many disciplines and, in particular, in service-oriented engineering.
Monitoring is defined as a process of observing, collecting, and reporting information about the execution and the evolution of SBAs. The relevant references of monitoring are summarized in the following paragraphs.
In [20], Keller and Ludwig propose the WSLA framework for the specification and the monitoring of service-level agreements. The WSLA framework defines a language for

the specification of contract information that allows for describing the parties involved in the agreement, the relevant QoS properties, as well as the ways to observe and measure them and the obligations and constraints imposed on these properties.

In [21], Ludwig *et al.* propose an architecture and implementation for the creation, the management, and the monitoring of service-level agreements represented as WS-Agreement documents. WS-Agreement specification provides a standardized way of defining contractual information between service provider and customer. The proposed architecture is called *CREMONA*. The monitoring module of *CREMONA* is not only used to observe and detect contract violations, but also to predict future violations and to engage appropriate adaptation strategies in advance.

In [22], Curbera *et al.* propose the *COLOMBO* platform for developing, deploying, and executing SBAs. The *COLOMBO* platform incorporates the tools and facilities for checking, monitoring, and enforcing service requirements expressed in WS-Policy notations. WS-Policy notations define the QoS assertions that can be attached to a particular WS, operation, or a message type.

In [23], Baresi and Guinea propose the run-time monitoring framework *DYNAMO*. *DYNAMO* uses an expressive monitoring language namely WSCoL for specifying monitoring rules. *DYNAMO* oversees the execution of SBAs by checking monitoring rules and by reacting as soon as they are violated by means of the associated adaptation strategies. In [24], Baresi *et al.* extend this work for what concerns the kind of properties the approach can monitor. The extended specification language, namely Timed WSCoL, allows for specifying temporal QoS properties over the events that occur during the SBA execution.

3.1 Testing

Testing is a frequently used technique for the analysis and the prediction of QoS of SBAs.

The goal of testing is to systematically execute SBAs in order to uncover QoS defects. During testing, the SBA which is tested is fed with concrete inputs and the produced outputs are observed. The observed outputs can deviate from the expected outputs with respect to functionality as well as QoS. When the observed outputs deviate from the expected outputs, a defect is uncovered.

A special case of testing is simulation. Simulation allows us to predict software applications performance in different status and load conditions of the execution environment. The predicted results are used to provide feedback on the efficiency of the application. Simulating SBAs for QoS evaluation is a research area with little previous work. Works in simulation that are the closest to ours are described by [25], [26], and [27].

In [25], Narayanan and McIlraith propose a model-theoretic semantics as well as distributed operational semantics that can be used for the simulation, the validation, the verification, the automated composition and the enactment of DAMLS-described SBAs. To provide a full service description, Narayanan and McIlraith use the machinery of situation calculus and its execution behaviour described with Petri Nets. They use the simulation and modeling environment *KarmaSIM* to translate DAML-S markups to situation calculus and Petri Nets. In this work, three QoS properties are analyzed: reachability, liveness and the existence of deadlocks.

In [26], Chandrasekaran *et al.* focus on problems related to SBA specification, evaluation, and execution using Service Composition and Execution Tool (*SCET*).

SCET allows to compose statically a WS process with WSFL and to generate a simulation model that can be processed by the *JSIM* simulation environment. In this work, Chandrasekaran *et al.* have enhanced WSFL to include QoS measures obtained by performing simulation tests.

In [27], Mancini *et al.* present a framework which is aimed at supporting the development of self-optimizing, predictive and autonomic systems for WS architectures. It adopts a simulation-based methodology which allows predicting QoS properties in different status and load conditions. In contrast to [25] and [26], this work considers execution environment information in the simulation models. This work focuses on simulating only atomic WSs. It proposes also only one possible QoS optimization that is response time minimization. Enhancements are needed to simulate SBAs and to add more optimization rules for QoS properties.

4. Discrete-Event Simulation Modeling of Service-Based Applications

Our work deals with using simulation as an analytical testing technique to assure QoS of SBAs.

There are two main reasons for adopting simulation techniques: first, simulation is a dynamic analytical technique that allows QoS predictions for software applications in different status and conditions of the execution environment. Second, simulation allows to tune and to evaluate software applications without experiencing the cost of enacting them. The originality of our work is the adoption of a special case of simulation that is the Discrete-Event Simulation (DES) to evaluate and assure QoS of SBAs.

In this paper, we propose a SBA modeling approach that is oriented towards QoS evaluation through DES. DES is a

kind of qualitative description of a dynamic system the behavior of which is event-driven. This technique is frequently used to analyze and predict the QoS of software applications. Giving the evolution of the operation of an application, we can analyze its behavior and evaluate appropriate quality measures. [7] and [8] are fundamental works about discrete-event systems diagnosis. DES is suitable to model the behavior of a SBA since it is composed of WSs which are decentralized and dynamic. The interactions between WSs can be modeled by a synchronized composition of several local models.

To elaborate our simulation model for SBAs, we are based on the work presented in [28] that focuses on modeling distributed applications. We model an SBA as a combination of two types of entities: distributed application and network infrastructure entities [29]. Our simulation model is shown in figure 3.

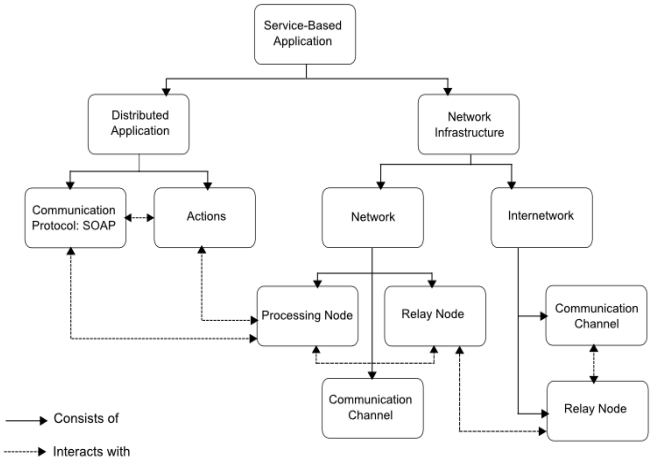

Fig. 3 Discrete-Event Simulation Model for Service-Based Applications

4.1 Distributed Application Modeling

The operation of the distributed applications is based on the client-server model. In this model, the client sends a set of requests to the server and the server sends a response back to the client for each request. The operation scenario is supported through specifying groups of actions:

- Processing: indicating data processing;
- Request: indicating invocation of a server process;
- Write: indicating data storage;
- Read: indicating data retrieval;
- Transfer: indicating data transfer between client and server processes;
- Synchronize: indicating replica synchronization.

Each WS is executed on a processing node. Processing action indicates invocation of the processing unit of the corresponding node and is characterized by the amount of data to be processed.

Request action indicates invocation of a server process and is characterized by the name of the server, the name of the WS, its invoked interface and the required inputs. Request action implies activation of the network, since the request and the reply must be transferred from the invoking to the invoked process, and vice versa.

There are two available actions for data storing, reading and writing, which are respectively characterized by the amount of the stored and retrieved data and the invoked server. The observations and performance analysis of SBAs have proven that SOAP messages are small and simple [30].

A transfer action is used to indicate SOAP messages exchanged between processes.

A synchronize action is needed since the replication of data is a common practice in such distributed applications. Synchronize action parameters include the process replicas that must be synchronized and the amount of transferred data.

To describe the operation of a SBA, we proceed by transforming the process behavior written in BPEL [31] into discrete-event actions. BPEL is a standard proposed by IBM and Microsoft along with several other companies to model composed Web services. BPEL defines a grammar for describing the behavior of a SBA. It is composed of fifteen activity types, some of them are basic activities and the others are structured activities. Among the basic activities, the most important ones are the following:

- The <receive> activity: is for accepting the triggering message from another WS;
- The <reply> activity: is for returning the response to its requestor;
- The <invoke> activity: is for invoking another WS.

The structured activities define the execution orders of the activities inside their scopes. For example:

- The <sequence> activity: defines the sequential order of the activities inside its scope;
- The <flow> activity: defines the concurrent relations of the activities inside its scope.

Each activity can be translated into the discrete-event formalism as one or several actions.

Basic activities involve processing, request, and data storing actions, while structured ones involve transferered and synchronized actions.

4.1 Network Infrastructure Modeling

In the proposed modeling scheme, the network infrastructure is considered as a collection of individual networks and internetworks, exchanging messages through relay nodes (active communication devices e.g. routers and switches). Communication channels represent protocol suites (i.e. routing protocols (OSI layers 2 and 3) and peer-to-peer protocols (OSI layers 4-7)).
According to the SOA, communication between WSs is performed through exchanging SOAP messages.
figure 4 illustrates one way of making a remote call using SOAP in OSI network reference model [32].

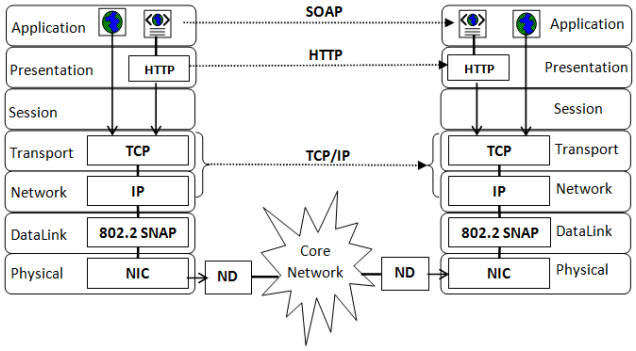

Fig. 4 Sending a SOAP Request under OSI

First at application level, a native data object needs to be serialized into XML as SOAP request. Then, the SOAP message is passed to HTTP level. The HTTP layer, on the client-side, needs to "handshake" with service-side by sending a "POST" request. This request initiated a TCP connection. Once receiving "HTTP: ACK", the client-side HTTP begins to send the whole SOAP message via TCP/IP. The SOAP message may be partitioned into a set of small segments at TCP layer. Appropriate headers and footers are attached to each segment as the segments are passed through Transport, Network, Data Link layers, until reaching the Network Interface Card (NIC) at the physical layer. The NIC is responsible for putting the packages onto the wire at a specific speed (network bandwidth) to next network device (such as a router or a switch), till server NIC [32]. The path from bottom (physical layer) to the top (application layer) on the service-side is opposite to the process on the client-side: the received packages are unpacked at each layer and forwarded to next layer for further retrieving.

4.1 Context Model

By the term "context", we mean "information utilized by the web service to adjust execution and output to provide the client with a customized and personalized behavior"[33].
Since SBAs are operating in dynamic environments, variations of execution context lead to variations in QoS expectations. In this work, we propose a context-based approach for evaluating SBAs performances. We consider a context model which consists of a set of elements grouped in 2 axes.

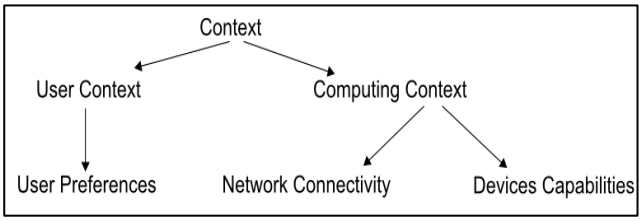

Fig. 5 Context Model

- User context: describes user preferences. To each QoS property, user attributes a weight. He chooses the value of this weight according to the level of the QoS property he needs. (e.g., execution time←0.8).
- Computing context: describes network connectivity (e.g., Internet connectivity, locality, and bandwidth) and devices capabilities (e.g., memory capacity and CPU speed).

Context information are described in the simulation model. Context changes are modeled as discrete events.

4.2 QoS Model

In [34], we define a light-weight quality model focusing on essential properties of QoS that play critical role for the effective management of WSs and that can be measured by DES technique. The QoS properties detailed above are defined in the context of atomic WSs. They are also used to evaluate the QoS of composite WSs. To provide aggregation functions for computing the QoS of composite WSs, we use the QoS computation models described by [6] and [35]. In these works, authors propose aggregation formulae for each pair QoS property/control statement (e.g., Sequence, Switch, Flow, and Loop). QoS aggregation functions are summarized in table 1.

- Response Time: it corresponds to the total time needed by a WS to transform a set of inputs into

outputs. Response Time (*RT*) for a service *s* can be computed as follows:

$$RT(s) = ST(s) + DT(s) \quad (1)$$

- Service Time (*ST*) is the time that the WS takes to perform its task.
- Delay Time (*DT*) is the time taken to send/receive SOAP messages.

- Reliability: it corresponds to the likelihood that the service will perform for its users on demand. Reliability (*R*) of a service *s* is function of the Failure Rate (*FR*):

$$R(s) = (1-FR(s))*100 \quad (2)$$

  - *FR*= successful executions/scheduled executions

- Availability: it refers to the rate of Service Activity (*SA*). Availability (*A*), during a time interval *I*, for a service *s* corresponds to:

$$A(s) = SA(s)/I \quad (3)$$

- Scalability: it computes the capacity of the service to manage loads. To test the scalability of a WS, we conducted the simulation while changing the number of concurrent clients.

Table 1: QoS Aggregation Functions per control statement

Aggregation function	Response Time (RT)	Reliability (R)	Availability (A)	Scalability (S)
Sequence	$\sum_{i=1}^{n} RT(s_i)$	$\prod_{i=1}^{n} R(s_i)$	$\prod_{i=1}^{n} A(s_i)$	$\prod_{i=1}^{n} S(s_i)$
Switch	$\sum_{i=1}^{n} p_i * RT(s_i)$	$\sum_{i=1}^{n} p_i * R(s_i)$	$\sum_{i=1}^{n} p_i * A(s_i)$	$\sum_{i=1}^{n} p_i * S(s_i)$
Flow	$Max\{RT(s_i)_{i\in[1..n]}\}$	$\prod_{i=1}^{n} R(s_i)$	$\prod_{i=1}^{n} A(s_i)$	$\prod_{i=1}^{n} S(s_i)$
Loop	$k * RT(s)$	$R(s)^k$	$A(s)^k$	$S(s)^k$

4. Our Simulation Framework SBAS

We have conducted simulation experiments using NS-2 simulator [36]. NS-2 is a discrete-events simulator; its code is written in C++ with an OTcl interpreter as a front end. NS-2 is targeted at networking research. It provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless networks. The main advantages of such an object-oriented simulator are reusability and easy maintenance. To support SBA simulation, we have extended the C++ class hierarchy of NS-2 in order to implement HTTP, SMTP, and SOAP protocols.

Our simulation framework (*SBAS*) is modular and includes: a graphical user interface, a BPEL generator, a simulation model generator, a models library and NS-2 simulator. The architecture of SBAS is presented in figure 6. User specifies the SBA under study. SBAS constructs corresponding BPEL model. Simulation model is implemented as actions organized in the object hierarchy of the NS-2 simulator. When simulation has been completed, results are collected and subjected to output QoS analysis.

5. Experimentations

In this section, we describe three SBAs which satisfy the same functional requirement. We use our discrete-event simulation approach to evaluate QoS of each of these SBAs.

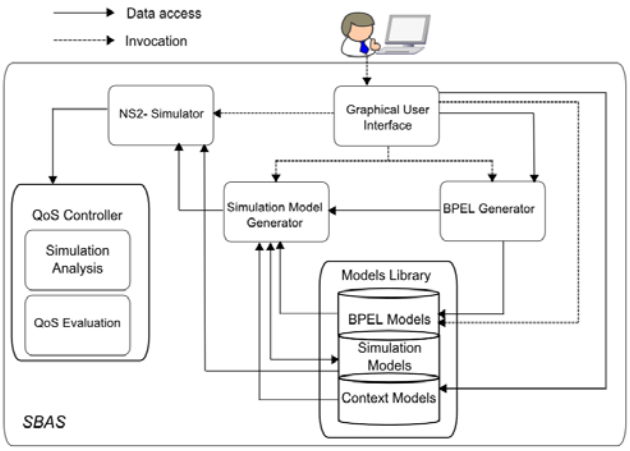

Fig. 6 Our Framework for Simulating Service-Based Applications: SBAS

To express variability modeling of SBAs, we adopt the MAP formalism [37]. A map is a labeled directed graph with intentions as nodes and strategies as edges between intentions. An intention is a goal that can be achieved by the performance of the process. Each map has two distinct intentions Start and Stop to respectively begin and end the navigation into the map. A strategy is an approach, a manner to achieve an intention. The MAP permits to capture variability by focusing on the strategy to achieve an intention and the potential alternatives to accomplish the same intention.

We consider an abstract intention *Buy books online*. Maps (described in Figure 7, Figure 8, and Figure 9) present possible refinements of this abstract intention. They model different SBAs (SBA1, SBA2, and SBA3) that ensure the same functional requirement *Buy books online*.

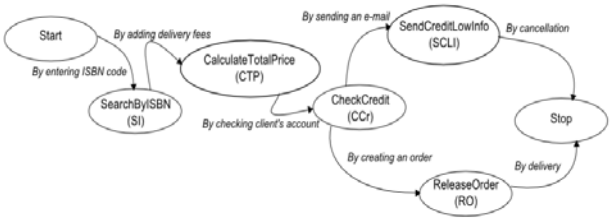

Fig. 7 Map of SBA1

SBA1 begins by invoking *SearchByISBN* (SI) service. This service allows the customer to search a book by entering its ISBN code. The total price to pay is calculated using the *CalculateTotalPrice* (CTP) service. The client's account is then checked for sufficient funds using the *CheckCredit* (CCr) service. If the client has sufficient credit, the *ReleaseOrder* (RO) service is invoked in order to send the book. Otherwise, the *SendCreditLowInfo* (SCLI) service is invoked.

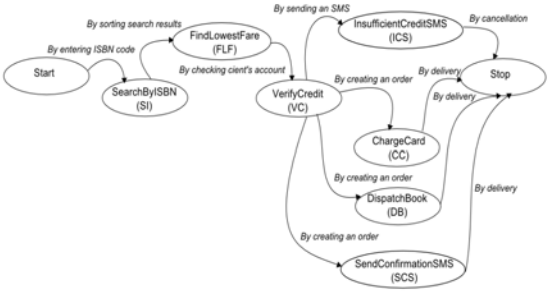

Fig. 8 Map of SBA2

In SBA2, the client begins also by searching the book that he wants to buy using the *SearchByISBN* (SI) service. The *FindLowestFare* (FLF) service allows him to find the cheapest bookstore. The client's credit is then checked by *VerifyCredit* (VC) service. If the client has sufficient credit, *ChargeCard* (CC), *DispatchBook* (DB), and *SendConfirmationSMS* (SCS) are invoked. Otherwise, *InsufficientCreditSMS* (ICS) is executed in order to inform the client of his insufficient credit.

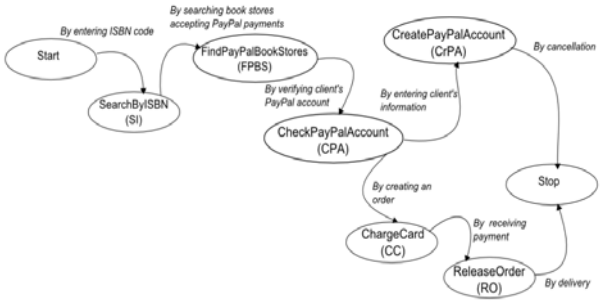

Fig. 9 Map of SBA3

SBA3 begins also by invoking the *SearchByISBN* (SI) service. The *FindPayPalBookStores* (FPBS) service allows the client to find book stores accepting PayPal payments. The client's PayPal account is checked by *CheckPayPalAccount* (CPA) service. If the client has a PayPal account, the *ChargeCard* (CC) and the *ReleaseOrder* (RO) services are invoked. Otherwise, the *CreatePayPalAccount* (CrPA) is executed to permit to the client to create a PayPal account.
Figure 10 shows the execution context of the user.

- Devices: PC, CPU: 1.86GHz, RAM: 2Go.
- Internet connectivity: NUMERIS, Modem speed: 512KB/s.
- Network: topology: user is connected to an Internet Service Provider (ISP), which is in its turn connected to Server through a Router (R).
  - Link User ISP: throughput: 512KB/s, delay: 50ms.
  - Link ISP R: throughput: 1MB/s, delay: 25ms.
  - Link R Server: throughput: 512KB/s, delay: 50ms.

Fig. 10 User's Execution Context

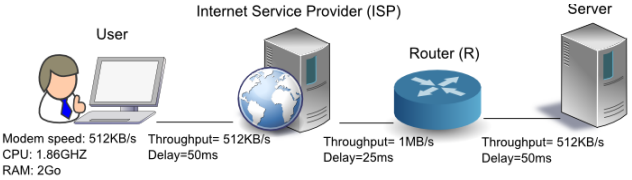

Table 2 illustrates the user's preferences. For each QoS property i, the user attribute a weight.

Table 2: User's QoS Preferences

QoS property	Weight
Response Time	0.35
Reliability	0.3
Availability	0.2
Scalability	0.15

The configuration of the simulation platform is Dual-Core based Windows XP system. For each QoS property, we performed a set of simulation experiments, and we have considered the average value. Simulation results (figure 11, figure12, figure 13, and figure 14) show that SBA1 is more reliable, available, and scalable than SBA2 and SBA3, but SBA2 is faster than SBA1 and SBA3.

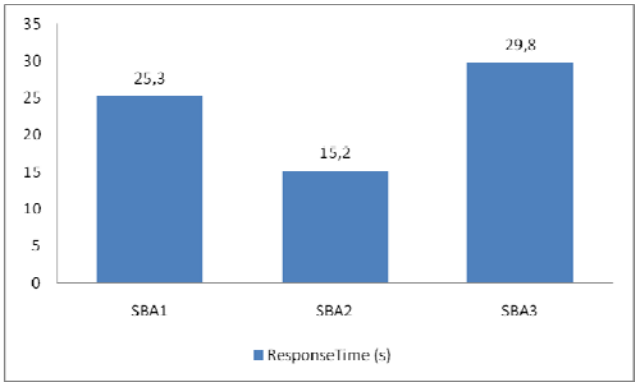

Fig. 8 Response Time Simulation Results

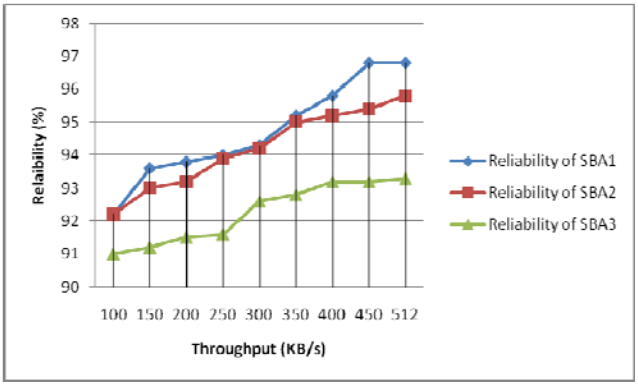

Fig. 9 Reliability Simulation Results

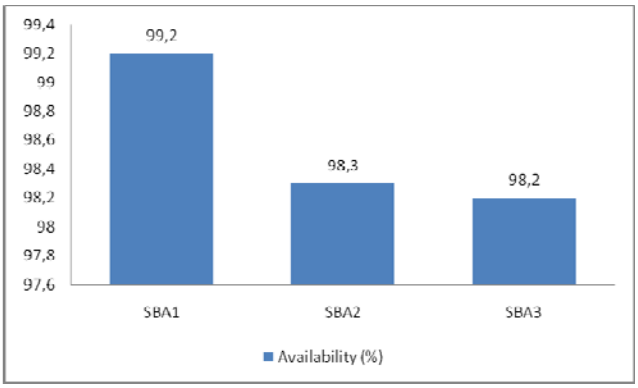

Fig. 10 Availability Simulation Results

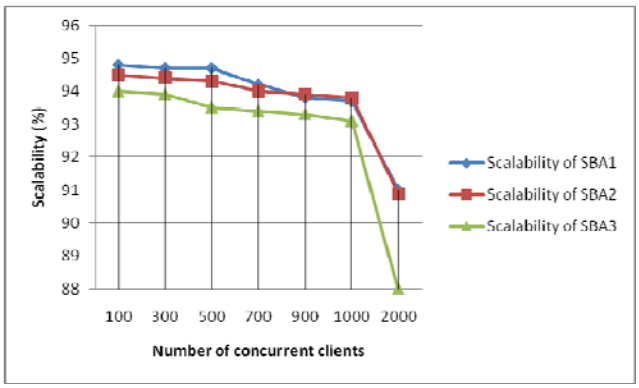

Fig. 11 Scalability Simulation Results

To validate these results in order to help the user to choose the appropriate SBA, we use a Benefit Function (BF). BF is computed as follows:

$$BF = \sum_{i=1}^{n} d_i^{'} * \omega_i \quad with \sum_{i=1}^{n} \omega_i = 1 \qquad (4)$$

Where d'_i is a normalized value of a QoS dimension (i.e., QoS property) d_i and w_i denotes the user's assigned relative importance to the dimension. As dimensions can be of different units (e.g., response time is in second and availability in percentage), in order to allow for a uniform measurement of WS QoS independent of units, data normalization is applied, which essentially transforms values of different units into comparable ones. By considering a 75% confidence interval, the dimensions that are stronger with larger values (e.g., reliability, availability and scalability) are normalized according to the following equation:

$$f(n) = \begin{cases} d_i^{'} = 1 & if\ d_i - m(d) > 2 * \delta(d) \\ d_i^{'} = 0 & if\ d_i - m(d) < 2 * \delta(d) \\ d_i^{'} = \dfrac{d_i - m(d)}{4 * \delta(d)} + 0.5 & otherwise \end{cases} \qquad (5)$$

While for QoS dimensions that are stronger with smaller values (e.g., response time), they are normalized according to the following equation so that smaller values contribute more to the user's benefit:

$$f(n) = \begin{cases} d_i^{'} = 0 & if\ d_i - m(d) > 2 * \delta(d) \\ d_i^{'} = 1 & if\ d_i - m(d) < 2 * \delta(d) \\ d_i^{'} = 0.5 - \dfrac{d_i - m(d)}{4 * \delta(d)} & otherwise \end{cases} \qquad (6)$$

Where d_i is the value of dimension d for the service instance *i*, and *m(d)* and *δ(d)* are the mean and standard deviation values for dimension *d* respectively.
The validation of SBA1, SBA2, and SBA3 has given the results shown in table 3. This validation proves that SBA1 is more appropriate for users' expectations than the SBA2 and SBA3 are.

Table 3: Validation Results

	SBA1	SBA2	SBA3
BF	0.31	0.265	0.12

5. Conclusion and Future Work

In this work, we adopted a discrete-events simulation approach to evaluate QoS of SBAs. We presented a simulation modeling approach. This approach enables an analytical description of SBAs and allows QoS predictions in the different status and conditions of the execution context.

We defined a light-weight quality model considering a set of QoS properties that can be measured by simulation techniques. We proposed also a context model that describes execution environment and the user's preferences. This model is depicted into the simulation model in order to provide a context-based approach for evaluating SBAs.

To show the effectiveness of our approach, we have conducted a set of simulation experiments in order to evaluate and to validate three SBAs that provide the same required functionality.

One possible extension of our work is the support of dynamic adaptations of SBAs. It requires extensive simulation experiments to define, validate and enhance the adaptation strategies.

Acknowledgments

This work has been supported by the European Communitys Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube). (http://www.s-cube-network.eu/).

References

[1] M. N. Huhns, and M. P. Singh, "Service-oriented Computing: Key Concepts and Principles", in IEEE Internet Computing, Vol. 9, No. 1, 2005, pp. 75–81.

[2] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana, "Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI", IEEE Internet Computing, Vol. 6, No. 2, 2002, pp. 86–93.

[3] M. P. Papazoglou, "Service-oriented Computing: Concepts, Characteristics and Directions", in Proceedings of WISE '03: International Conference on Web Information Systems Engineering, 2003, pp. 3–12.

[4] J. O'Sullivan, D. Edmond, and A. Hofstede, "What's in a Service? Towards Accurate Description of Non-functional Service properties", Distributed and Parallel Databases, Vol. 12, No. 2, 2002, pp. 117–133.

[5] D. A. Menascé, "QoS Issues in Web Services", IEEE Internet Computing, Vol. 6, No. 6, pp. 72–75.

[6] J. Cardoso, J. Miller, A. Sheth, and J. Arnold, "Modeling Quality of Service for Workflows and Web Service Processes", Journal of Web Semantics, Vol. 1, No. 3, 2004, pp. 281-308.

[7] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Diagnosability of Discrete-Event Systems", IEEE Transactions on Automatic Control, Vol. 40, No. 9, 1995, pp. 1555–1575.

[8] M. Cordier, and S. Thiébaux, "Event-based Diagnosis for Evolutive Systems", in Proceedings of DX '94: International workshop on Principles of diagnosis, 1994, pp. 64-69.

[9] Universal Description, Discovery and Integration specification 3.0.2, http://uddi.org/pubs/uddi-v3.0.2-20041019.htm

[10] Web Services Description Language 2.0, http://www.w3.org/TR/wsdl20/

[11] Simple Object Access Protocol 1.2, http://www.w3.org/TR/SOAP/

[12] L. J. Osterweil, "Strategic Directions in Software Quality", ACM Computing Surveys, Vol. 28, No. 4, 1996, pp. 738-750.

[13] G. J. Myers, "Art of Software Testing", John Wiley & Sons (Eds.), ISBN: 978-0-471-04328-7, 1979.

[14] L. Baresi, and E. DiNitto, "Test and Analysis of Web Services", Elisabetta (Eds.), ISBN: 978-3-540-72911-2, 2007.

[15] S. Nakajima, "Model Checking Verification for Reliable Web Service", in Proceedings of OOPSLA '02: Workshop on Object-Oriented Web Services, 2002, pp. 20.

[16] G. Salaün, L. Bordeaux, and M. Schaerf, " Describing and Reasoning on Web Services using Process Alge bra ", in Proceedings of ICWS '04: IEEE International Conference on Web Services, 2004, pp. 43.

[17] H. Foster, S. Uchitel, J. Magee, and J. Kramer, "LTSA-WS: A Tool for Model-based Verification of Web Service Compositions and Choreography", in Proceedings of ICSE '06: International Conference on Software Engineering, 2006, pp. 771– 774.

[18] L. J. Osterweil, "Formal Analysis of Web Service Compositions", Ph.D. Dissertation, 2007.

[19] R. Kazhamiakin, M. Pistore, and L. Santuari, "Analysis of Communication Models in Web Service Compositions", in Proceedings of WWW'06: International Conference on World Wide Web, 2006, pp. 267-276.

[20] A. Keller, and H. Ludwig, "The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services", Journal of Network and Systems Management, Vol. 11, No. 1, 2003, pp. 57–81.

[21] H. Ludwig, A. Dan, and R. Kearney, "CREMONA: An Architecture and Library for Creation and Monitoring of WS-Agreements", in Proceedings of ICSOC '04: IEEE International Conference on Service Oriented Computing, 2004, pp. 65–74.

[22] F. Curbera, M. J. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana, "Colombo: Lightweight Middleware for Service-Oriented Computing", IBM Systems Journal, Vol. 44, No. 4, 2005, pp. 799-820.

[23] L. Baresi, and S. Guinea, "Towards Dynamic Monitoring of WS-BPEL Processes", in Proceedings of ICSOC '05: International Conference of Service-Oriented Computing, 2005, pp. 269–282.

[24] L. Baresi, D. Bianculli, C. Ghezzi, S. Guinea, and P. Spoletini, "A Timed Extension of WSCoL", in Proceedings of ICWS '07: IEEE International Conference on Web Services, 2007, pp. 663–670.

[25] S. Narayanan, and S. A. McIlraith, "Simulation, Verification, and Automated Composition of Web Services", in

Proceedings of WWW '02: International Conference on World Wide Web, 2002, pp. 77–88.
[26] S. Chandrasekaran, J. A. Miller, G. A. Silver, I. B. Arpinar, and A. P. Sheth, "Performance Analysis and Simulation of Composite Web Services", Electronic Markets, Vol. 13, No. 2, 2003.
[27] E. Mancini, U. Villano, M. Rak, and R. Torella, "A Simulation-based Framework for Autonomic Web Services", in Proceedings of ICPADS '05: IEEE International Conference on Parallel and Distributed Systems, 2005, pp. 433–437.
[28] M. Nikolaidou, and D. Angnostopoulos, "An Application-oriented Approach for Distributed System Modeling and Simulation", in Proceedings of ICDCS '01: International Conference on Distributed Computing Systems, 2001, pp. 165.
[29] M. Driss, Y. Jamoussi, and H. Hajjami Ben Ghézala, "QoS Testing of Service-Based Applications", in Proceedings of IDT '08: IEEE International Design and Test Workshop, 2008, pp. 45-50.
[30] S. Chen, B. Yan, J. Zic, R. Liu, and A. Ng, "Evaluation and Modeling of Web Services Performances", in Proceedings of ICWS '06: IEEE International Conference on Web Services, 2006, pp. 437-444.
[31] Business Process Execution Language for Web Services 2.0, http://www.oasisopen.org/committees/tc_home.php?wg_abbrev=wsbpel
[32] D. Bertsekas, and R. Gallager, "Data Networks", Prentice Hall (Eds.), ISBN: 978-0-132-00916-4, 1992.
[33] M. Keidl, and A. Kemper, "Towards Context-Aware Adaptable Web Services", in Proceedings of WWW '04: International Conference on World Wide Web, 2004, pp. 55–65.
[34] M. Driss, Y. Jamoussi, J. M. Jézéquel, H. Hajjami Ben Ghézala, "A Discrete-Events Simulation Approach for Evaluation of Service-Based Applications", in Proceedings of ECOWS '08 : IEEE European Conference on Web Services, 2008, pp.73-78.
[35] L. Zeng, B. Benatallah, A. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-Aware Middleware for Web Services Composition", IEEE Transactions on Software Engineering, Vol. 30, No. 5, pp. 311–327.
[36] The ns manual, http://www.isi.edu/nsnam/ns/doc/index.html
[37] C. Rolland, and N. Prakash, "Bridging the Gap between Organizational Needs and ERP Functionality", Requirements Engineering, Vol. 5, No. 3, 2000, pp. 180-193.

Y. Jamoussi received the Engineering degree from the Computer Engineering Faculty of the University of Tunisia, Tunis, in1989, and the Ph.D. degree from the same Faculty, in 1998. He is currently an Assistant Professor at the National School of Information Sciences, University of Manouba, Manouba, Tunisia. His current research interests focus on the enactment, guidance and the monitoring of strategic process. He is a co-author of a book on Method Engineering. He is an MVP on Biztalk. Recently, his research interests include web services composition.

M. Driss received an engineer degree in computer science in 2006 with distinction (Major of promotion) and master degree in software engineering in 2007 from the National School of Computer Science (ENSI), University of Manouba, Tunisia. She is currently a permanent researcher in the laboratory RIADI-GDL, ENSI, University of Manouba, Tunisia, and in the INRIA team-project Triskell, University of Rennes I, France. Her research interests include Web services composition, QoS of Web services, and QoS assurance techniques.

J-M.Jézéquel received an engineering degree in Telecommunications from the ENSTB in 1986, and a Ph.D. degree in Computer Science from the University of Rennes, France, in 1989. He first worked in Telecom industry (at Transpac) before joining the CNRS (Centre National de la Recherche Scientifique) in 1991. Since October 2000, he is a Professor at the University of Rennes, leading the INRIA research team Triskell. His interests include model driven software engineering based on object oriented technologies for telecommunications and distributed systems.

H. Hajjami Ben Ghézala received Ph.D. degree from the Computer Engineering Faculty of the University of Tunisia, Tunis, in 1987. She is a Professor of Software Engineering at the National School of Information Sciences, University of Manouba, Manouba, Tunisia, leading the RIADI laboratory. Since the beginning of 2009, she is the rector of the University of Manouba. Her interests include Service Oriented Architecture.