HAL
open science

QoS testing of service-based applications

Maha Driss, Yassine Jamoussi, Henda Hajjami Ben Ghézala

To cite this version:

Maha Driss, Yassine Jamoussi, Henda Hajjami Ben Ghézala. QoS testing of service-based applications. In Proceedings of the 3rd IEEE International Design and Test Workshop (IDT '08), 2008, Monastir, Tunisia, Tunisia. pp.45-50, 10.1109/IDT.2008.4802463 . inria-00468234

HAL Id: inria-00468234

https://inria.hal.science/inria-00468234v1

Submitted on 30 Mar 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

QoS Testing of Service-Based Applications

Maha Driss[1,2], Yassine Jamoussi[1] and Henda Hajjami Ben Ghézala[1]
[1]National School of Computer Sciences (ENSI)
RIADI-GDL Laboratory
University of Manouba, Tunisia
yassine.jamoussi@ensi.rnu.tn
henda.benghezala@ensi.rnu.tn
[2]IRISA/INRIA, Rennes, France
maha.driss@irisa.fr

Abstract

Web services (WSs) are becoming increasingly popular because of their potential in several application domains including e-Enterprise, e-Business, e-Government, and e-Science. Based on open XML standards, WS technology allows the construction of massively distributed and loosely coupled applications. A service composition mechanism should satisfy not only functional properties but also non-functional Quality of Service (QoS) ones. In this paper, we introduce a discrete-events modeling approach for Service-Based Applications (SBAs). This approach is oriented towards QoS evaluation through simulation.

Keywords: Service-Based Applications, Discrete-Events Simulation, QoS, Evaluation.

1. Introduction

In the past decades, the evolution of software systems observes continuous growth of the complexity levels and the maturation of the networks and protocols. The cross-application integration and interoperability, the flexibility of implementation and reconfiguration became essential requirements for the modern architectural styles. The main challenge is to simplify the integration of software applications, regardless possible heterogeneity of implementing platforms, protocols, and devices. Service-Oriented Architecture (SOA) utilizes services as the fundamental elements in order to deliver low-cost, flexible, and scalable distributed applications [15] [22]. Web Services (WSs) are universally accessible software components advertised, discovered, and invoked over the Web [22]. The key aspect of this emerging paradigm is the use of standard technologies such as: Web Services Description Language (WSDL) [10], Universal Description, Discovery, and Integration (UDDI) [4] and Simple Object Access Protocol (SOAP) [7]. These technologies aim to support the definition of services, their advertisement, and their binding for triggering purposes.

The process-based composition of WSs is gaining a considerable momentum as an approach for the effective integration of distributed, heterogeneous, and autonomous applications [15]. Quality of service (QoS) is an important concern for Service-Based Applications (SBAs). QoS for WSs refers to various non-functional characteristics such as response time, throughput, availability and security [6]. Since many service providers provide similar services with common functionality, different WS processes can be composed satisfying the same user requirement. Users will discriminate these processes based on their QoS [28]. Also, WSs operate autonomously within a highly variable environment: the Web. As a result, their QoS may evolve relatively frequently to the dynamic and unpredictable nature of the execution environment (e.g., network resources, devices characteristics, etc.) [28]. In particular, during an execution of a composite service, the component services involved may change their QoS properties, others may become unavailable, and still others may emerge. For the above reasons, the management and assurance of the quality aspects of SBAs become of utmost importance.

To achieve the desired quality of a SBA, different analytical quality assurance techniques can be applied. The goal of these techniques is to evaluate QoS and to uncover faults in the composed applications after their creation. One of the relevant techniques is testing. The goal of testing is to (systematically) execute services or SBAs in order to uncover failures. During testing, the service or SBA which is tested is fed with concrete inputs and the produced outputs are observed [21] [18] [2]. The observed outputs can deviate from

Authorized licensed use limited to: UR Rennes. Downloaded on March 24, 2009 at 08:20 from IEEE Xplore. Restrictions apply.

the expected outputs with respect to functionality as well as QoS. When the observed outputs deviate from the expected outputs, a failure of the service or the SBA is uncovered.
A special case of testing is simulation. The goal of the simulation is to emulate the conversational behaviour of atomic WSs participating in the composition.
In this paper, we adopt a discrete-events simulation to test WS compositions. We introduce a modeling approach for SBAs. This approach enables analytical description of SBAs and allows QoS predictions in different status and conditions of the execution environment. We define a simple quality model for WSs focusing on essential properties of QoS that play a critical role for the effective management of WSs and that can be measured by simulation technique.
The remainder of the paper is structured as follows: in Section 2, simulation issues are addressed. Experimental results are documented in Section 3. Related research is discussed in Section 4. This paper closes with concluding remarks and future work.

2. Discrete-events simulation modeling of SBAs

Simulation is an important testing technique, as it allows users to tune and to evaluate software systems without experiencing the cost of enacting them. In this work, we propose an SBA modeling approach that is oriented towards performance evaluation through discrete-events simulation.
Discrete-events simulation is a kind of qualitative description of a dynamic system whose behaviour is event-driven. This technique is frequently used to analyze and predict the performance of real word systems. Given the evolution of the operation of a system, we can analyze its behaviour and evaluate appropriate quality measures. [26] and [11] are fundamental works about discrete-event systems diagnosis. Discrete-events simulation is suitable to model the behaviour 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 [20] that focuses on modeling distributed applications. We model an SBA as a combination of two types of entities: distributed application and network infrastructure entities. Our simulation model is shown in Figure 1.

2.1. Distributed application modeling

The operation of distributed application 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

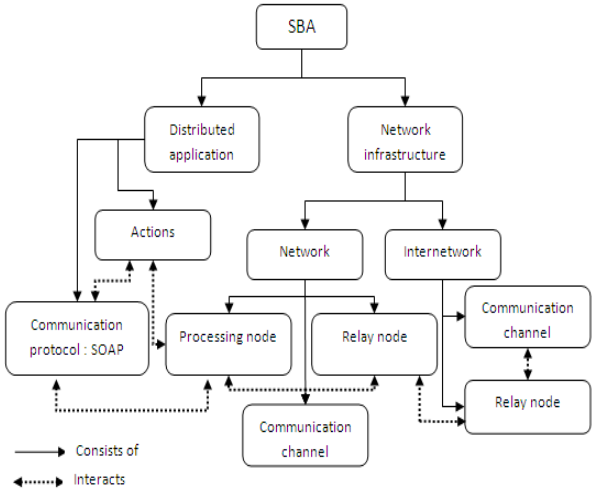

Figure 1. Simulation model for SBAs

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 process 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, read and write, which are characterized respectively by the amount of the stored and retrieved data and the invoked server. Observations and performance analysis of web services applications have proven that SOAP messages are small and simple [9].
Transfer action is used to indicate SOAP messages exchange between processes.
Synchronize action is needed since replication of processes and data is a common practice in such distributed applications. Synchronize action parameters include the process

Authorized licensed use limited to: UR Rennes. Downloaded on March 24, 2009 at 08:20 from IEEE Xplore. Restrictions apply.

replicas that must be synchronized and the amount of transferred data.
To describe the operation of a SBA, we proceed by transforming the process behaviour written in BPEL into discrete-event actions. BPEL is a standard proposed by IBM and Microsoft along with several other companies to model composed Web services [1]. BPEL defines a grammar for describing the behaviour of a WS process. 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 Web service;
- The <reply> activity: is for returning the response to its requestor;
- The <invoke> activity: is for invoking another Web service.

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 transfer and synchronize actions.

2.2. 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 element entity represents 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 processes is performed through exchanging SOAP messages.
Figure 2 illustrates one way of making a remote call using SOAP in OSI network reference model [5].
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 router or switch), till server NIC [5]. 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.

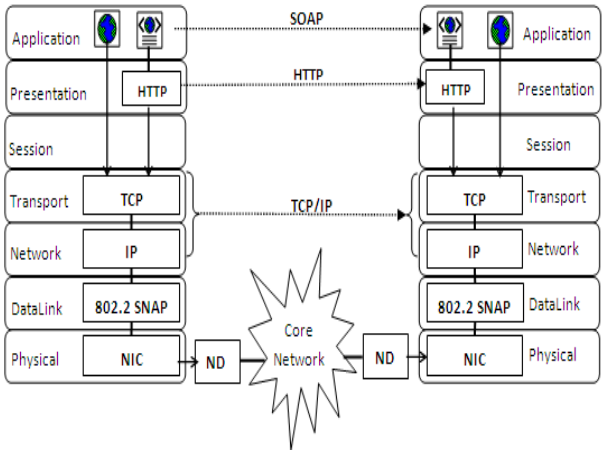

Figure 2. Sending a SOAP request under OSI

2.3. Our simulation tool

We have conducted simulation experiments using NS-2 simulator [13]. 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 in order to implement HTTP, SMTP and SOAP protocols.
Our SBA Simulator (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 3.
User specifies the SBA under study using activity diagrams. 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.

Authorized licensed use limited to: UR Rennes. Downloaded on March 24, 2009 at 08:20 from IEEE Xplore. Restrictions apply.

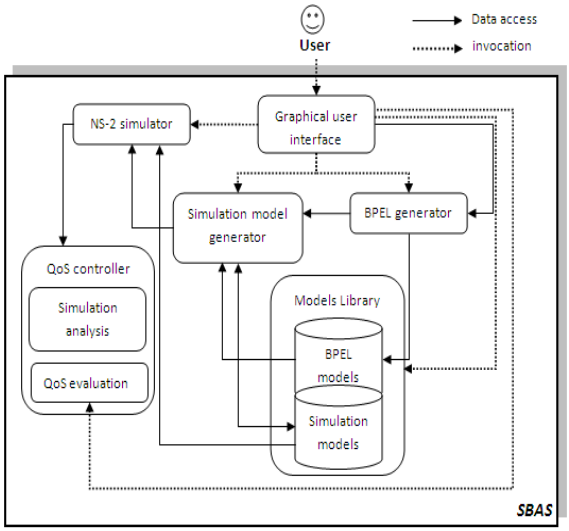

Figure 3. SBAS architecture

3. Experimentation

As a running example, we use the Travel Planner (TP) composite service, whose BPEL code is sketched in Figure 4, while Figure 5 illustrates the corresponding activity diagram with its abstract services.

```
...
<sequence>
        <while condition="condition=trigger" ...>
                <flow ...>
                        <sequence>
                                <invoke ...
                                        operation="FlightTicketBooking" .../>
                                <invoke ...
                                        operation="HotelBooking" .../>
                        </sequence>
                        <invoke ...
                                operation="AttractionSearch" .../>
                </flow>
        </while>
<invoke ...
operation="DrivingTimeCalculation" .../>
        <switch ....>
                <case condition="carRental=OK">
                        <invoke ...
                                operation="CarRental" .../>
                </case>
                <otherwise>
                        <invoke ...
                                operation="BikeRental" .../>
                </otherwise>
        </switch>
```

Figure 4. The TP BPEL specification

Simulation experiments were conducted in the testing platform described in Figure 6. This platform consists of two identical 1.86GHz CPUs machines connected via 100Mbps LAN.

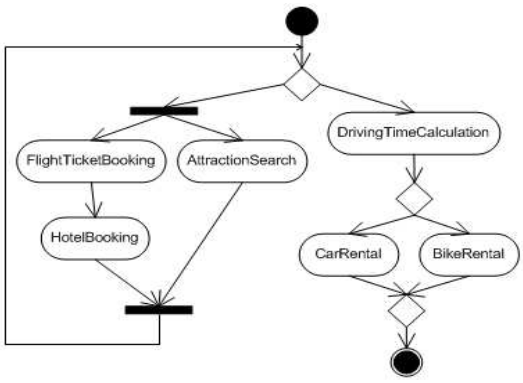

Figure 5. The TP activity diagram

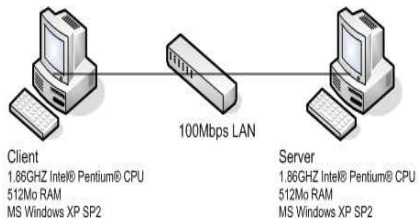

Figure 6. Test platform

In this paper,we focus on evaluating only response time. Other QoS parameters (e.g., reliability, availability and scalability) are described and analyzed in our previous works [12].
Response Time 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:

$$RT(s) = ST(s) + DT(s)$$

- 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.

$$DT(s) = T_{msgTrans}(s) + T_{synch}(s) + T_{app}(s)$$

- $T_{msgTrans}$ represents the total cost to transfer a specific amount of messages over network;
- T_{synch} represents the overhead of the extra synchronization required by protocols;
- T_{app} represents the time spent in business logic at application level.

Authorized licensed use limited to: UR Rennes. Downloaded on March 24, 2009 at 08:20 from IEEE Xplore. Restrictions apply.

For more details about $T_{msgTrans}$, T_{synch} and T_{app} modeling see the work described in [9].
We have performed a set of simulation experiments, and we have considered the average value. Our simulation results are summarized in Figure 7 and we conclude that service "BikeRental" is the fastest and the most scalable one among all tested WSs. "CarRental" service poses time problems so it must be changed.

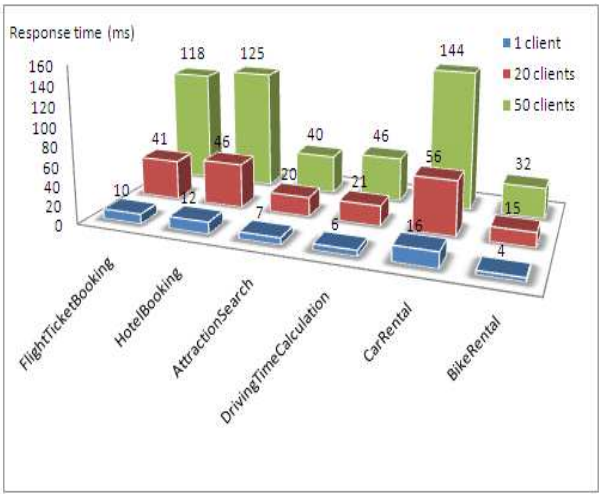

Figure 7. Simulation results

4. Related work

To achieve the desired quality of an SBA, different analytical quality assurance techniques can be employed. The goal of these techniques is to evaluate QoS and to uncover faults in the SBA after it has been created.
The review of relevant literature in the field of quality assurance for SBAs shows three major classes of analytical techniques and methods: (i) Testing, the goal of which is to (systematically) execute services or SBAs with predefined inputs in order to uncover failures, (ii) Monitoring, which observes services or SBAs as well as their context during their current execution, (iii) Static analysis, the aim of which is to systematically examine (without execution) an artifact (e.g., a service specification) in order to determine certain properties or to ascertain that some predefined properties are met.
These classes have been proposed in the software quality assurance literature (e.g., see [21] and [18]) and have been used in a recent overview of quality assurance approaches for SBAs [2]. There have been numerous efforts by leading research groups to use monitoring and static analysis to evaluate QoS and to discover problems in SBA execution. Relevant works in the area of SBAs monitoring are: [23] [6] [3] [27], etc. Static analysis is adopted in [25] [14] [24], etc.
In this work, testing technique is used for analyzing the efficiency of SBAs. We adopt simulation approach to test SBAs. Simulation allows us to predict system performance in different status and load conditions of the execution environment. The predicted results are used to provide feedback on the efficiency of the system. Simulating Web processes for performance evaluation is a research area with little previous work. Works in simulation that are the closest to ours are described by [19] [8] and [16].
[19] proposes a model-theoretic semantics as well as distributed operational semantics that can be used for simulation, validation, verification, automated composition and enactment of DAML-S-described SBAs. To provide a fully 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 verification problems are simulated and analyzed: reachability, liveness and existence of deadlocks.
Use of simulation for WSs is described also in [8]. This work focused 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 simulation model that can be processed by the JSIM simulation environment. In this work, Chandrasekaran et al. have enhanced WSFL to include QoS estimates obtained by performing simulation tests. Only the QoS response time dimension is supported by the implementation of JSIM [8].
In contrast to our approach, both [19] and [8] don't consider execution environment information in simulation model.
[16] presents a framework whose aim is to support the development of self-optimizing, predictive and autonomic systems for WS architectures. It adopts a simulation-based methodology, which allows predicting system QoS in different status and load conditions.
In contrast to [19] and [8], this work considers execution environment information in simulation model.
This work focuses on simulating only atomic WSs. Also it proposes only one possible QoS optimization that is response time minimization. Enhancements are needed to simulate SBAs and to add more optimization rules for QoS parameters.

5. Conclusion

We have presented a discrete-events simulation approach for testing SBAs.
We have contributed in the definition of a simulation model for SBAs. The proposed model is composed of two types of entities: distributed application and network infrastructure.

Authorized licensed use limited to: UR Rennes. Downloaded on March 24, 2009 at 08:20 from IEEE Xplore. Restrictions apply.

Both are described in terms of their elementary components. Distributed application is described by a set of primitive actions: Processing, Request, Write, Read, Transfer and Synchronize. Network infrastructure is considered as a collection of individual networks and internetworks, exchanging messages.

Our modeling approach is oriented towards QoS evaluation. A simulation tool has been constructed for this purpose. To show the effectiveness of our approach, we have conducted a set of simulation experiments.

One possible extension of our work is the support of dynamic adaptations of SBAs. This requires specific testing and analysis techniques to verify the adaptation behaviour.

References

[1] Business process execution language for web services. *Online: http://www.ibm.com/developerworks/library/specification/ws-bpel/*, 2003.

[2] L. Baresi and E. DiNitto. Test and analysis of web services. *Springer-Verlag GmbH*, 2007.

[3] L. Baresi and S. Guinea. Towards dynamic monitoring of ws-bpel processes. *Third International Conference of Service-Oriented Computing - ICSOC*, pages 269–282, 2005.

[4] T. Bellwood, S. Capell, L. Clement, J. Colgrave, M. J. Dovey, D. Feygin, A. H. R. Kochman, P. Macias, M. Novotny, M. Paolucci, C. von Riegen, T. Rogers, K. Sycara, P. Wenzel, and Z. Wu. Universal description, discovery and integration specification (uddi) 3.0.2. *Online: http://uddi.org/pubs/uddi-v3.0.2-20041019.htm*.

[5] D. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall, 1992.

[6] D. Bianculli and C. Ghezzi. Monitoring conversational web services. *IW-SOSWE '07: 2nd international workshop on Service oriented software engineering*, pages 15–21, 2007.

[7] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winer. Simple object access protocol (soap) 1.1. *Online: http://www.w3.org/TR/SOAP/*, 2001.

[8] 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*, 13(2), 2003.

[9] S. Chen, B. Yan, J. Zic, R. Liu, and A. Ng. Evaluation and modeling of web services performances. *in ICWS '06: Proceedings of the IEEE International Conference on Web Services.*, 2006.

[10] R. Chinnici, J.-J. Moreau, A. Ryman, and S. Weerawarana. Web services description language (wsdl) version 2.0: Core language. *Online: http://www.w3.org/TR/wsdl20/*.

[11] M. O. Cordier and S. Thi'ebaux. Event-based diagnosis for evolutive systems. *In Proceedings of the Fifth International workshop on Principles of diagnosis(DX'94)*, pages 64–69, 1994.

[12] M. Driss, Y. Jamoussi, J.-M. Jézéquel, and H. H. B. Ghézala. A discrete-events simulation approach for evaluation of service-based applications. *ECOWS'08 : Proceedings of the 6th European Conference on Web Services*, 2008.

[13] K. Fall and K. Varadhan. The *ns* manual. *Online: http://www.isi.edu/nsnam/ns/doc/ns-doc.pdf*.

[14] H. Foster, S. Uchitel, J. Magee, and J. Kramer. Ltsa-ws: A tool for model-based verification of web service compositions and choreography. *in ICSE '06: Proceedings of the 28th International Conference on Software Engineering*, pages 771–774, 2006.

[15] M. Huhns and M. P. Singh. Service-oriented computing: Key concepts and principles. *IEEE Internet Computing*, 9(1):75–81, 2005.

[16] E. Mancini, U. Villano, M. Rak, and R. Torella. A simulation-based framework for autonomic web services. *ICPADS '05: Proceedings of the 11th International Conference on Parallel and Distributed Systems - Workshops (ICPADS'05)*, pages 433–437, 2005.

[17] D. A. Menascé. Qos issues in web services. *IEEE Internet Computing*, 6(6):72–75, 2002.

[18] G. J. Myers. *Art of Software Testing*. John Wiley & Sons, Inc., New York, NY, USA, 1979.

[19] S. Narayanan and S. A. McIlraith. Simulation, verification and automated composition of web services. *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 77–88, 2002.

[20] M. Nikolaidou and D. Angnostopoulos. An application-oriented approach for distributed system modeling and simulation. *ICDCS'01: Proceedings of the The 21st International Conference on Distributed Computing Systems*, page 165, 2001.

[21] L. J. Osterweil. Strategic directions in software quality. *ACM Comput. Surv.*, 28(4):738–750, 1996.

[22] M. P. Papazoglou. Service -oriented computing: Concepts, characteristics and directions. *WISE '03: Proceedings of the Fourth International Conference on Web Information Systems Engineering*, page 3, 2003.

[23] M. Pistore and P. Traverso. Assumption-based composition and monitoring of web services. *Test and Analysis of Web Services*, pages 307–335, 2007.

[24] M. Rouached, O. Perrin, and C. Godart. Towards formal verification of web service composition. *Business Process Management*, pages 257–273, 2006.

[25] G. Salan, L. Bordeaux, and M. Schaerf. Describing and reasoningon web services using process algebra. *in ICWS '04: Proceedings of the IEEE International Conference on Web Services*, page 43, 2004.

[26] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 40(9):1555–1575, 1995.

[27] G. Spanoudakis and K. Mahbub. Requirements monitoring for service-based systems: Towards a framework based on event calculus. *9th IEEE International Conference on Automated Software Engineering (ASE 2004)*, pages 379–384, 2004.

[28] L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. Qos-aware middleware for web services composition. *IEEE Trans. Softw. Eng.*, 30(5):311–327, 2004.

Authorized licensed use limited to: UR Rennes. Downloaded on March 24, 2009 at 08:20 from IEEE Xplore. Restrictions apply.