WS-SIM: A Toolkit for Modeling and Simulation of Service Composition

Fan Zhang
School of Computer Science
Beihang University
Beijing, China
zhangfan_a@act.buaa.edu.cn

***Abstract*—Web service has been rapidly developed in recent years. Web service selection is an important issue in web service composition and lots of service selection algorithms have been presented. As lots of them select an atomic service during runtime, it is not an easy task to evaluate the quality of the composited service which composed of several atomic services. In this work we introduce WS-SIM, a simulation toolkit, to solve this problem. WS-SIM supports modeling and simulating composite services and atomic services in the real world. This system also provides many common service selection algorithms and researchers can custom their own service selection algorithm for a simulation experiment. The quality of composite services can also be generated by our system. Furthermore, to demonstrate suitability of the WS-SIM, in this paper, functionalities of our system are illustrated by a case study. This confirms the usability and the applicability of WS-SIM.**

Keywords-web service; selection; algorithm; simulation

I. INTRODUCTION

Recently, web service providers tend to offer dedicated domain specific atomic services. However, the number of functions that a single web service provided is limited. Composite services, which combine several web services made by others, can provide much more complicated functions. A composite service consists of several tasks and each task can be accomplished by atomic services, which provide the same functionality, rendered by different service providers. Users often prefer to select an ideal one according to their QoS information from alternatives.

Quality of a web service consists of its own properties including time, cost, reliability, and etc. Lots of QoS based web service selection algorithms have been presented [1][2]. Since most of these selection algorithms select service dynamically, it is hardly evaluate the performance of a composite service. Therefore, researchers need a simple framework to do lots of experiments to achieve that objective. However, that confronts several drawbacks: 1) Building a service selection algorithm experiment is time consuming. 2) For the the service is provided for payment, the real world experiment is even more expensive. 3) Algorithm experiment is limited to a few number of available services.

To overcome these problems, we provide WS-SIM, a ready-to-use simulation infrastructure. It can be used for simulating the composite service modeling, atomic service modeling, algorithm creating, and composite service execution. At last, WS-SIM will generate a report which focuses on the performance of this execution for further analysis.

The rest of this paper is organized as follow. Section 2 introduces the system architecture and features in detail. The approach for building a simulation using WS-SIM is presented in Section 3. A use case is discussed in Section 4. Finally, Section 5 gives conclusions and future work.

II. SYSTEM ARCHITECTURE

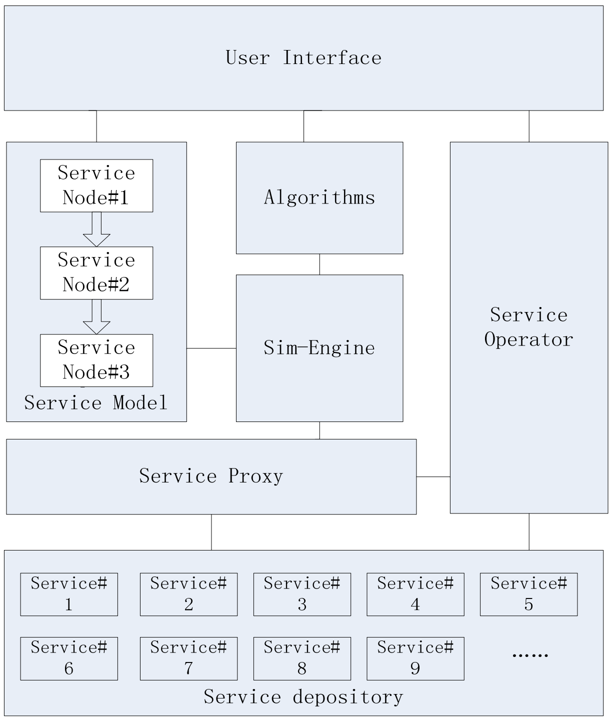

Figure 1. Architecture of WS-SIM

A. System Description

The WS-SIM provides a facility for the simulation of composite services. It can be used to simulate the web service selection and execution procedure of service algorithm in each task. We employ a modular architecture for WS-SIM to leverage existing technologies and manage them as separate components. The abstraction for the WS-

SIM is shown in Figure 1. User Interface Module concerns with presentations and interactions between user and WS-SIM. Composite Service Module focuses on the abstraction of the flow of composite service. It defines internal representations of a composite service. Composite service is represented as BPMN, A standard Business Process Modeling Notation provides the capability of understanding composite service procedures in a graphical notation. Algorithms Module contains several service selection algorithms in common use. It also provides an interface for algorithm extension. Sim-Engine is based on Desmo-J[6], a discrete event driven simulation framework[2][3] in Java. The Sim-Engine focuses on task and service scheduling. It is a very important component of WS-SIM. Service operation Module focuses on atomic service model creation and property setting. User can create atomic service model either by manual or by importing historic log. Service proxy Module concerns with calling atomic service. While Sim-Engine needs to call an atomic service, it will call this module for returning the virtual atomic service. Service depository holds all atomic services which ever be created.

B. *Features*

Algorithm module embedded several popular service selection algorithms, this feature facilitates comparing different algorithms in a specific case. In addition, WS-SIM also provides an algorithm interface for algorithm extension or customizing.

Service Operator is used for generating virtual atomic services. An atomic service model is a set of QoS (time, cost, reliability, etc). Atomic service can be created through importing execute log which have been collected. When researchers model a large number of atomic services which existing in the real world, it is too hard to finish such a trivial work manually. Modeling atomic services by importing log is an elegant way to solve this problem. Besides, it also allows to model atomic services which never invoked by local user. This enable user to create some services that never exist or cannot be found in the real world, so that researchers can test an algorithm's performance in some extreme situations.

Sim-Engine is based on Discrete-Event Simulation. It does not need to invoke a real atomic service, so it can be executed without network connections. Meanwhile, the invocation of a virtual atomic service can be finished in an instant. Compared with an experiment through real invocation, executing a simulation experiment makes more productive. Moreover, WS-SIM do all these simulations locally, which leads to a cost reduction.

At the end of the experiment, statistics of all the operations can be recorded. The data of execution logged is extremely important for the algorithm analysis.

III. BUILDING SIMULATION WITH WS-SIM

To simulate algorithms execution using the WS-SIM. Following steps should be done:

1) *Composite service modeling:* A composite service model with BPMN is created.

2) *Importing algorithm:* The algorithm to be evaluated should be specified by users. It can be done either by selecting an existing algorithm which WS-SIM provided or customizing an algorithm that compatible with the algorithm customization interface.

3) *Atomic service modeling:* Enough virtual atomic services should be created in WS-SIM. For achieving this objective, QoS information of each atomic service should be told. WS-SIM provides a function of QoS evaluation from the system log, that facilitates the creation of virtual atomic services. Besides, by taking elastic into consideration, creating virtual atomic services manually is allowed.

4) *Experiment execution:* After several steps above has been completed and the number of execution has been specified, by researchers a simulation experiment can be started.

5) *Log analysis:* Log would be generated during the experiment. It contains a lot of knowledge for the algorithm execution, which is important for Researchers to do performance analysis.

IV. CASE STUDY

This section describes how a composite service simulation experiment is handled by using WS-SIM.

Firstly, a composite service model should be given for the simulation experiment. Figure 2 illustrates an example for a composite model.

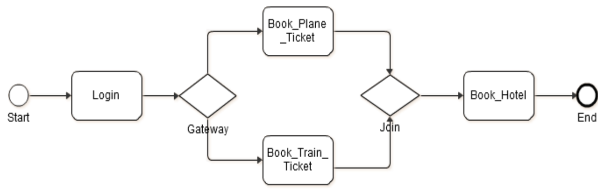

Figure 2. A Experimental Scenario

The composite service, describing a process of trip planning on a travel website, comprises three steps. The first step is login. The second step is choosing transport for the trip. Here we set 70% of customers by train and 30% by plane on the Gateway task. The last step is booking a hotel.

Secondly, we define a service selection algorithm to be evaluated. We focus on a local optimal algorithm. Here we consider three QoSs, i.e. time, cost and reliability. An overall QoS of the selected service can be formulated as:

$$\begin{aligned} P_i = {} & \lambda_{time}(Time_{avg} - Time_i)/(Time_{\max} - Time_{\min}) \\ & + \lambda_{\cos t}(Cost_{avg} - Cost_i)/(Cost_{\max} - Cost_{\min}) \\ & + \lambda_{rel}(\mathrm{Re}\,l_i - \mathrm{Re}\,l_{avg})/(\mathrm{Re}\,l_{\max} - \mathrm{Re}\,l_{\min}) \end{aligned}$$

where Time is the round trip delay of an atomic service invocation from client side; Cost is the expense user need to pay for his invoking to the service provider; Rel represents the reliability of a atomic service; λ means the weight of a QoS. The objective of WS-SIM is to find an atomic service which satisfied $P_m = P_{\max}$. In this case, for finding a better decision, we test the results when $(\lambda_{time}, \lambda_{\cos t}, \lambda_{rel})$ is set to be different values.

Thirdly, virtual atomic services should be created. This can be done using the wizard dialog, in which the range of each QoS, the number of virtual services are needed, as shown in Figure 2. Then WS-SIM generates virtual atomic services automatically.

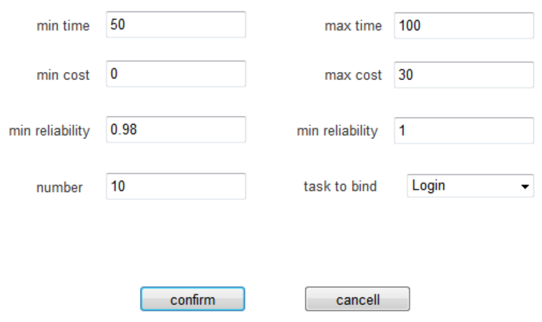
min time: 50
max time: 100
min cost: 0
max cost: 30
min reliability: 0.98
min reliability: 1
number: 10
task to bind: Login

Figure 3 Wizard dialog to create virtual atomic services

TABLE I. RESULT OF SIMULATION

$(\lambda_{time}, \lambda_{\cos t}, \lambda_{rel})$	**Avg time**	**Avg cost**	**Fail**
(1,1,1)	203.23101669230758	16.74118263041123	12
(1,1,5)	215.8521150394736	38.03809606965849	12
(5,1,1)	166.01946777971307	32.78621663801568	24
(1,5,1)	202.81718219675471	16.749077037080742	14
(5,5,1)	180.90571521897797	18.65890239096863	41
(1,5,5)	203.1570196825076	16.743169670390863	11
(5,1,5)	170.58467654065032	31.817158541320236	16

We set the number of execution to 1000 and then start the experiment. WS-SIM will generate Java code for running the simulation experiment automatically. The result is shown in Table 1 and a part of the experiment's trace log is shown in Table 2.

Table 1 shows that higher weight specified to a QoS, better performance the QoS achieves. We also discovered that compared with $(\lambda_{time}, \lambda_{\cos t}, \lambda_{rel})$ =(1,1,1), $(\lambda_{time}, \lambda_{\cos t}, \lambda_{rel})$ =(1,5,1) improves the weight of cost may not help to reduce the total cost. However, the parameters of $(\lambda_{time}, \lambda_{\cos t}, \lambda_{rel})$ set to (5,1,1) with a increase weight of response time, result in a decreased response time, but increases the cost and leads to a higher fail rate. So researchers can balance these factors in the strategy selection.

TABLE II. THE TRACE OF THE SIMULATION PROCESS

Task	**Service_id**	**Time**	**Cost**
Experiment_1#Login	Service_2	58.87699787009054	7.735549956283414
Experiment_1#Book_Train_Ticket	Service_25	62.42806513354333	4.615727269441611
Experiment_1#Book_Hotel	Service_40	48.59782559177801	19.563757581368133
Experiment_2#Login	Service_2	58.880723063403295	7.735549956283414
Experiment_2#Book_Train_Ticket	Service_25	59.86889804956354	4.615727269441611
Experiment_2#Book_Hotel	Service_40	48.91241186116051	19.563757581368133
Experiment_3#Login	Service_2	50.26843888662006	7.735549956283414
Experiment_3#Book_Plane_Ticket	Service_13	68.69390180151235	4.21378871817033
Experiment_3#Book_Hotel	Service_40	48.440282231484446	19.563757581368133
...	...	...	...

V. Conclusion And Future Works

By using the Discrete Event simulation and extending Desmo-J, we present a.simulation toolkit, called WS-SIM, for web service modeling and algorithm simulation. We have introduced the architecture and components of the WS-SIM. We have compared different weights in local optimal selection methods in the case study. The experimental results show the applicability.of our system.

In the future work, we are going to discover a method for generating more accurate virtual atomic services from existed log automatically. We will also aim at exploring more service selection algorithms and adapt these algorithms into WS-SIM.

References

[1] L. Zeng, B. Benatallah, M. Dumas, and J. K. and H. Chang. QoS-Aware Middleware for Web Services Composition. IEEE Trans. on Soft. Eng., May 2004.

[2] Danilo Ardagna and Barbara Pernici. Global and local QoS Guarantee in Web Service Selection. In Proc. Of Business Process Management Workshops, pages 32-46, Sept. 2005.

[3] JA Sokolowski and CM Banks. Modeling and simulation fundamentals: theoretical underpinnings and practical domains. 2010.

[4] V Hlupic and S Robinson. Business process modeling and analysis using discrete-event simulation. 30th conference on Winter simulation, 1998.

[5] Business Process Modeling Notation 2.0, http://www.omg.org/spec/BPMN/

[6] DESMO-J; http://desmoj.sourceforge.net/home.html. (10, 11, 2011)

[7] T. Lechler and B. Page, DESMO-J: An Object Oriented Discrete Simulation Framework in Java, Proceedings of the European Simulation Symposium '99, 1999

[8] V Hlupic. S Robinson, Business process modeling and analysis using discrete-event simulation, the 30th conference on Winter simulation, 1998.

[9] Z. Zheng. H. Ma, M. R. Lyu and I. King. Wsre: A collaborative filtering based web service recommend system. In Proc. 7th Int'l Conf. Web Services(ICWS'09), pages 437-444, 2009

[10] Z. Zheng and M. R. Lyu. Collaborative reliability prediction for service-oriented systems. In Proc. IEEE/ACM 32nd Int'l Conf. Software Engineering(ICSE'10), 2010.