

Service Composition Execution Optimization based on State Transition Matrix For Cloud Computing

Sheng LIU, Gang Xiong, Hongxia Zhao, Xisong Dong

State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China;
sheng.liu, gang.xiong, hongxia.zhao, xisong.dong@ia.ac.cn

Abstract—It is difficult to select a composite service with the lowest actual executing cost using the existing methods for QoS-aware service composition in cloud computing. By analyzing the dynamic execution process of composite service with state transition matrix, this paper proposes a new QoS aware optimal service composition method. In view of the effect of composite services reliability on the composite service performance, the method regards the cost averaged for one time of successful execution of a composite as its actual executing cost, and then selects the composite services with the aim of minimizing the composite service execution cost. The simulation result shows that the proposed method is superior to other methods in execution cost.

Keywords- *cloud computing; service composition; quality of service(QoS); state transition matrix*

I. INTRODUCTION

In Cloud computing, everything is treated as a service, e.g. SaaS (Software as a Service), PaaS (Platform as a Service) and IaaS (Infrastructure as a Service) [1]. Business Process Execution Language for Web Services(BPEL4WS), Web Services Description Language (WSDL), Simple Object Access Protocol (SOAP), Universal Description, Discovery, and Integration (UDDI) and are widely used for service composition, description, registration, and binding, respectively[2-4]. Web service providers register web services which are defined by WSDL on cloud computing platform through an UDDI registry. Then web service requestors find the cloud platform for the needed web services via UDDI and invoke those services under SOAP. Multiple web services need to be configured together to form a web service composition which is able to accomplish complex business function to satisfy service requestors' functional and nonfunctional requirements.

The nonfunctional requirements include cost, time, reliability, availability which are so-called quality of service (QoS). One of the web service QoS study is the composition of web services according to QoS requirements. Web service composition is often accomplished by the Business Process Execution Language for Web Services (BPEL4WS for short) [4]. The QoS of the composed service is determined by the QoS of its underlying component services. Zeng et al describe

Jianshi Yao

Unit 61541, The Chinese People's Liberation Army,
Beijing, 100094, China
Yaojianshi@163.com

the aggregation functions for the computation of the QoS through component services in detail in [11]. It is easy to imagine that there may be many feasible web service compositions that can satisfy the same functional requirement but take different QoS attributes. Therefore, the QoS of the resulting composite service is a determinant factor to meet different requirements of different users. There are two kinds of web service quality optimization methods, i.e., local and global optimizations. Ardagna and Pernici [7] discriminate global and local QoS constraints.

The local optimization is done at a task level, i.e., choosing for each task the web service with the best QoS[13]. No doubt that the service composition is with the best QoS if all component services are with the best ones. But the composition cost is so high that the service requestors are unwilling to accept. Global optimization is done at a process level where services are selected for each task to obtain the optimal global quality [7, 11].

Zeng et al discusses liner programming method of QoS aware web service composition [6], D.A. presents a service composition optimization method with service cost constraints [5]. A service composition method based on Generic Algorithm is described in [7]. Xiong et al bring a service selection method to improve executing success rate by describing web service composition with Petri Net [8]. A service composition method to improve composition reliability is shown in [9].

The execution order of each component service relies on its position in the composition. A service composition is multiple services which are arranged for executing. In many cases, when a component service in a service composition modeled with BPEL fails to execute, the BPEL process needs to restart from the first component service [8]. And the service requestor must pay for the component services which succeeded in executing before the failed one. It is not hard to imagine the afterward executing component services have greater effects on the QoS of service composition than the previous ones. The execution sequence of each component service is key factor of the QoS of the service composition. As far as we know, there are no studies on service composition optimization which take execution orders of component services into account.

In this paper we discuss optimal service composition concerning the execution sequences of the component services. Firstly we analyze the dynamic execution process of service composition using state transition matrix. Then according to state transition matrix, a service composition optimization method is brought forward. The simulation result shows that our method is superior to existing ones.

The rest of this paper is organized as follows: Section II introduces the definition and description of the QoS of web service and state transition matrix. Section III analyzes the dynamic executing process of service composition. The aggregation function to compute the quality attributes for a service composition is provided in this section. Section IV shows a real example to certify the validity of the analysis results and methodology. Finally, Section V concludes this paper.

II. QOS OF WEB SERVICE AND STATUS TRANSITION MATRIX

A. QoS definition for a single web service and a service composition

The component services in a service composition usually execute in sequential, alternative or parallel order[13]. BPEL4WS (Business Process Execution Language for Web Services) [13] is the most popular web service composition language. BPEL4WS model composes the services with sequential, parallel or alternative containers in which the services should execute in corresponding order. This paper employs BPEL4WS to describe web service composition. The influence of the component service logic position on the QoS of the service composition will be discussed in detail. The aggregation functions for the computation of the QoS of service composition are presented.

A service composition described by BPEL4WS is also considered as a service which can be invoked by other services or applications. So single service and service composition are both called service in this paper. The QoS of a service usually includes cost, time, availability, reputation and reliability [16]. The reputation attribute is ignored for it has little relation to the content of the paper. $QoS_{ws} = \{c_{ws}, a_{ws}, r_{ws}, d_{ws}\}$ is often utilized to denote the QoS of web service ws . The meanings of each unit are listed as follow.

c_{ws} is the fee to be paid by a service requestor for invoking the web service ws each time.

a_{ws} is the attribute for evaluating an immediate availability of ws .

r_{ws} is the probability of receiving the right result within the excepted duration time after ws is successfully invoked.

d_{ws} is the duration time from the moment ws is successfully invoked to the moment the result is received.

The service composition described by BPEL4WS(which is called service process in the following) will retry several times and return with failure information if any single component service is unavailable; The service process will exit from executing and restart if any single component service is unreliable[9,11]. At most times the requestors repeat the invocation until the service process executes successfully or give up after retrying too many times.

Here we define cs_{ws} as the average fee to be paid by a service requestor for every successful invoking on the web service ws . The failure cost is taken into account by cs_{ws} .

B. State Transition Diagram and Status Transition Matrix

A Markov system (or Markov process or Markov chain) is a system that can be in one of several (numbered) states, and can pass from one state to another each time step according to fixed probabilities [17].

If a Markov system is in state i , there is a fixed probability p_{ij} of it going into state j the next time step, and p_{ij} is called a transition probability [17].

A Markov system can be illustrated by means of a state transition diagram, which is a diagram showing all the states and transition probabilities (See Fig. 1).

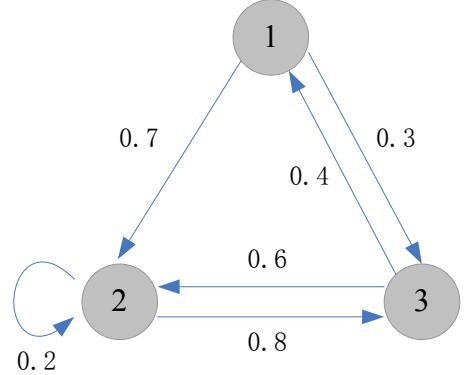


Fig. 1 state transition diagram

The matrix P whose ij th entry is p_{ij} is called the status transition matrix associated with the system. The entries in each row add up to 1. For instance, a 3×3 status transition matrix P would be set up as in the Fig. 2.

$$\begin{array}{c}
 \text{To:} \\
 \begin{array}{ccc}
 1 & 2 & 3
 \end{array} \\
 \text{From:} \\
 \begin{array}{c}
 1 \\
 2 \\
 3
 \end{array}
 \left[\begin{array}{ccc}
 p_{11} & p_{12} & p_{13} \\
 p_{21} & p_{22} & p_{23} \\
 p_{31} & p_{32} & p_{33}
 \end{array} \right]
 \begin{array}{l}
 p_{11} + p_{12} + p_{13} = 1, \\
 p_{21} + p_{22} + p_{23} = 1, \\
 p_{31} + p_{32} + p_{33} = 1.
 \end{array}
 \end{array}$$

Fig. 2 Status transition matrix

III. EXECUTION ANALYSIS AND OPTIMIZATION FOR BPEL4WS WITH STATUS TRANSITION MATRIX

A component service will execute successfully or unsuccessfully after it is invoked. Then we define three statuses which are *ready* (denoted by 1), *successful execution* (denoted by 2) and *unsuccessful execution* (denoted by 3) for web service ws . So we get its status transition matrix:

$$P_{ws} = \begin{bmatrix} 0 & r_{ws} & 1-r_{ws} \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad (1)$$

A BPEL process is composed of *<receive>*, *<assign>*, *<reply>*, *<invoke>* and other basic BPEL elements. They are organized according to certain execution order by *<sequence>*, *<flow>* and other BPEL containers. For no fee happens during the executions of basic BPEL elements except *<invoke>*, we only take *<invoke>* into account. Since a BPEL process is nested by *<sequence>*, *<flow>* and other BPEL containers, we can analyze BPEL process by analyzing the execution processes of them. Here only *<sequence>* and *<flow>* are considered because they are frequently used in modeling a BPEL process.

A. Execution Analysis for *<sequence>* container

Supposing that a *<sequence>* container seq contains 3 web services which are denoted by $ws1, ws2$ and $ws3$ (see Fig. 3), its execution statuses are listed on Table 1.

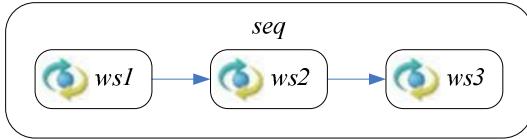


Fig.3 The structure of the *<sequence>* container seq

TABLE I. THE STATUSES OF THE *<SEQUENCE>* CONTAINER SEQ

| No. | Description |
|-----|---------------------------------------------------------------------|
| 1 | BPEL process is ready |
| 2 | $ws1$ is successfully executed |
| 3 | $ws1$ fails to execute |
| 4 | $ws2$ is successfully executed |
| 5 | $ws2$ fails to execute |
| 6 | $ws3$ is successfully executed (seq is successfully executed) |
| 7 | $ws3$ fails to execute |

So we get the status transition diagram of seq displayed in Fig.4. The 1st status is initial execution status of seq and the 3rd, 5th, 6th and 7th statuses are the final one. The execution steps from the 1st status to the 6th form a successful execution process. The successful execution process is uncertain because

each component service is probable to execute successfully or not.

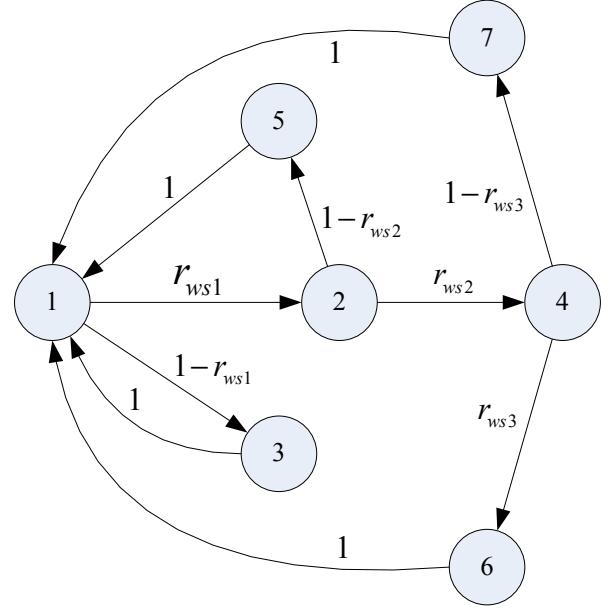


Fig. 4 Status Transition Diagram of seq

According to Formula (1), the status transition matrix is

$$P_{seq} = \begin{bmatrix} 0 & r_{ws1} & 1-r_{ws1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & r_{ws2} & 1-r_{ws2} & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & r_{ws3} \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (2).$$

The fee to be paid to the providers of the 3 services is variable because the execution process is uncertain. Thus we calculate the average value of the fee and let it be one of the QoS attributes for a *<sequence>* container. Let cs_{seq} denote the average cost of seq for each successful execution. Let $p_{s1 \rightarrow s2}$ and $c_{s1 \rightarrow s2}$ denote the probability and cost from the Status $s1$ to Status $s2$ respectively during the execution. It is not hard to get:

$$cs_{seq} = \left(\frac{p_{1 \rightarrow 3} * c_{1 \rightarrow 3} + p_{1 \rightarrow 5} * c_{1 \rightarrow 5} + p_{1 \rightarrow 6} * c_{1 \rightarrow 6} + p_{1 \rightarrow 7} * c_{1 \rightarrow 7}}{p_{1 \rightarrow 6}} \right) \quad (3).$$

According to Formula (2) and Formula (3), we get:

$$cs_{seq} = \left(\frac{r_{ws1}(1-r_{ws2}) * c_{ws1} + r_{ws1}r_{ws2}(1-r_{ws3}) * (c_{ws1} + c_{ws2}) + r_{ws1}r_{ws2}r_{ws3} * (c_{ws1} + c_{ws2} + c_{ws3})}{r_{ws1}r_{ws2}r_{ws3}} \right) \quad (4).$$

B. Execution Analysis for *<flow>* container

Supposing that a *<flow>* container *flo* contains 3 web services which are denoted by *ws1*, *ws2* and *ws3* (see Fig. 4), its execution statuses are listed on Table 2.

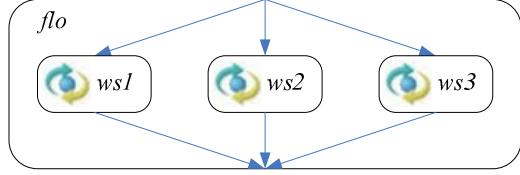


Fig.4 The structure of the *<flow>* container *flo*

TABLE II. THE STATUSES OF THE *<FLOW>* CONTAINER *FLO*

| No. | Description |
|-----|---------------------------------------------------------------------------------------------------------|
| 1 | BPEL process is ready |
| 2 | <i>ws1</i> fails, <i>ws2</i> fails, <i>ws3</i> fails |
| 3 | <i>ws1</i> succeeds, <i>ws2</i> fails, <i>ws3</i> fails |
| 4 | <i>ws1</i> succeeds, <i>ws2</i> succeeds, <i>ws3</i> fails |
| 5 | <i>ws1</i> fails, <i>ws2</i> succeeds, <i>ws3</i> fails |
| 6 | <i>ws1</i> fails, <i>ws2</i> succeeds, <i>ws3</i> succeeds |
| 7 | <i>ws1</i> fails, <i>ws2</i> fails, <i>ws3</i> succeeds |
| 8 | <i>ws1</i> succeeds, <i>ws2</i> fails, <i>ws3</i> succeeds |
| 9 | <i>ws1</i> succeeds, <i>ws2</i> succeeds, <i>ws3</i> succeeds (<i>flo</i> is successfully executed) |

So we get the status transition diagram of *flo* shown in Fig.5. The execution steps from the 1st status to the 9th form a successful execution process.

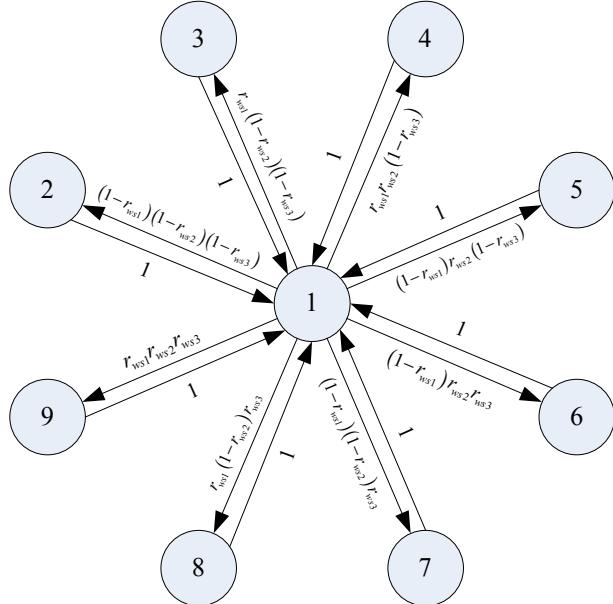


Fig. 5 Status Transition Diagram of *flo*

According to Formula (1), the transpose matrix of status transition matrix is

$$P'_{flo} = \begin{bmatrix} 0 & 1 & 1 & \dots & 1 \\ (1-r_{ws1})(1-r_{ws2})(1-r_{ws3}) & 0 & 0 & \dots & 0 \\ r_{ws1}(1-r_{ws2})(1-r_{ws3}) & 0 & 0 & \dots & 0 \\ r_{ws1}r_{ws2}(1-r_{ws3}) & 0 & 0 & \dots & 0 \\ (1-r_{ws1})r_{ws2}(1-r_{ws3}) & 0 & 0 & \dots & 0 \\ (1-r_{ws1})r_{ws2}r_{ws3} & 0 & 0 & \dots & 0 \\ (1-r_{ws1})(1-r_{ws2})r_{ws3} & 0 & 0 & \dots & 0 \\ r_{ws1}(1-r_{ws2})r_{ws3} & 0 & 0 & \dots & 0 \\ r_{ws1}r_{ws2}r_{ws3} & 0 & 0 & \dots & 0 \end{bmatrix} \underbrace{8 \text{ columns}}_{(5)}.$$

The execution process is variable. So the fee to be paid for the 3 services during each execution process is different from one another. Then the average value of the fee is computed and considered as one of the QoS attributes for a *<flow>* container. Let cs_{flo} stand for the average cost of *flo* for each successful execution. Let $p_{s1 \rightarrow s2}$ and $c_{s1 \rightarrow s2}$ denote the probability and cost from the Status *s1* to Status *s2* respectively during the execution. It is not hard to get:

$$cs_{flo} = \sum_{i=2}^9 p_{1 \rightarrow i} * c_{1 \rightarrow i} / p_{1 \rightarrow 9} \quad (6).$$

According to Formula (5) and Formula (6), we get:

$$cs_{flo} = \frac{\begin{aligned} &r_{ws1}(1-r_{ws2})(1-r_{ws3})c_{ws1} \\ &+ r_{ws1}r_{ws2}(1-r_{ws3})(c_{ws1} + c_{ws2}) \\ &+ (1-r_{ws1})r_{ws2}r_{ws3}(c_{ws2} + c_{ws3}) \\ &+ (1-r_{ws1})(1-r_{ws2})r_{ws3}c_{ws3} \\ &+ r_{ws1}(1-r_{ws2})r_{ws3}(r_{ws1} + r_{ws3}) \\ &+ r_{ws1}r_{ws2}r_{ws3}(c_{ws1} + c_{ws2} + c_{ws3}) \end{aligned}}{r_{ws1}r_{ws2}r_{ws3}} \quad (7).$$

C. The comparision between STM based Service Composition QoS Computation with Other Methods

The attribute c_{ws} of the QoS or c_{ws} / r_{ws} is used as the cost at which the service requestor invokes the service *ws* in most papers [2-10]. When the service is a single one it is true. But most services in the internet are composite ones. We will prove that the attribute cs_{ws} is the real service invoking cost by experiments. Only according to cs_{ws} of the composite service *ws* can we get the optimal service composition for *ws*.

The QoS values of component services in a *<sequence>* container *seq* shown in Fig. 3 are listed in Table III.

TABLE III. THE QOS VALUES OF SERVICES IN FIG. 3

| | <i>c</i> | <i>r</i> | <i>d</i> |
|------------|----------|----------|----------|
| <i>ws1</i> | 8 | 0.9 | 11 |
| <i>ws2</i> | 9 | 0.9 | 7 |
| <i>ws3</i> | 12 | 0.8 | 20 |

By existing methods, the execution cost of *seq* is:

$$c_{\text{seq}} = (c_{\text{ws1}} + c_{\text{ws2}} + c_{\text{ws3}}) / (r_{\text{ws1}} r_{\text{ws2}} r_{\text{ws3}}) = 44.75.$$

By our method, the average cost of *seq* for each successful execution is:

$$cs_{\text{seq}} = 34.36.$$

We also build a computer program to simulate the execution of *seq*. The simulation result of execution cost is 34.358. It is obvious that our method is nearer to the simulation result.

IV. SERVICE COMPOSITION OPTIMIZATION BASED ON STM

The BPEL4WS process shown in Fig. 6 is a distributed traffic simulation process. Because the traffic model is too huge to be simulated by one computer, we divide it into 3 parts and simulate them on 3 computers respectively. The simulation result is calculated when the simulation finishes. 5 services should be invoked during the simulation process. Each service has 4 candidates which are functionally same but carry different QoS attributes. The QoS values are all listed in Table IV in which can_i(i=1,2,3,4) denotes the *i*th candidate of corresponding service node. Only *c* and *r* of the service QoS are taken into account here.

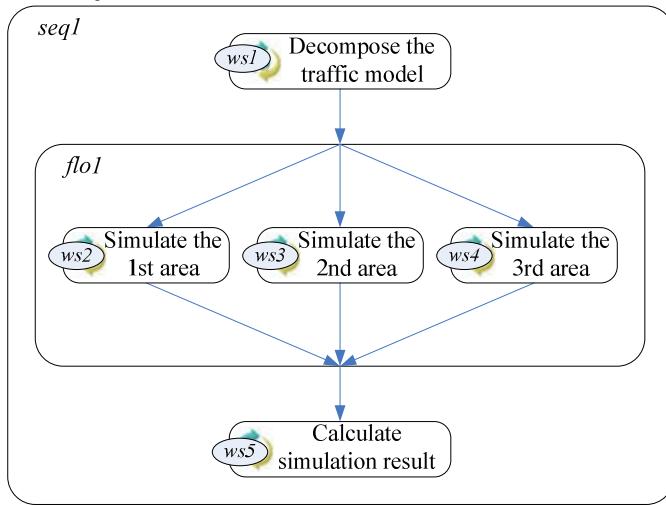


Fig. 6 The BPEL4WS for distributed traffic simulation

TABLE IV. THE QOS VALUES OF ALL CANDIDATE SERVICES

| | can_1 | | can_2 | | can_3 | | can_4 | |
|------------|-------|------|-------|------|-------|------|-------|------|
| | c | r | c | r | c | r | c | r |
| <i>ws1</i> | 5 | 0.82 | 6 | 0.93 | 4.5 | 0.76 | 5.6 | 0.88 |
| <i>ws2</i> | 6 | 0.81 | 6.7 | 0.89 | 5.5 | 0.76 | 7.5 | 0.94 |

| | | | | | | | | |
|------------|-----|------|-----|------|-----|------|-----|------|
| <i>ws3</i> | 6.2 | 0.82 | 6.6 | 0.87 | 5.7 | 0.79 | 7.3 | 0.92 |
| <i>ws4</i> | 5.8 | 0.79 | 6.1 | 0.81 | 6.8 | 0.9 | 7.6 | 0.96 |
| <i>ws5</i> | 4.5 | 0.81 | 5.6 | 0.92 | 4.0 | 0.75 | 5.1 | 0.87 |

Two service compositions with minimized cost are selected according to existing method and our method separately.

Then the simulation is done to show the real costs of the service compositions obtained by the two methods. The results are compared in Fig. 7. It is obvious that our method is superior to existing method.

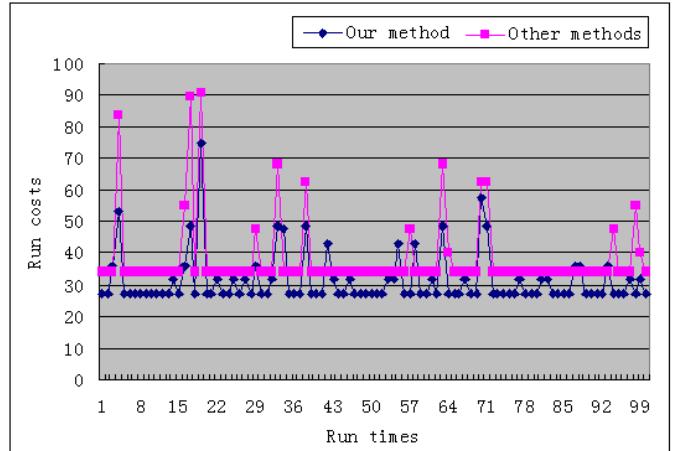


Fig. 7 Running costs comparison of service compositions obtained by our method and other methods

V. CONCLUSION

This paper uses state transition matrix to analyze the dynamic execution process of service composition described by BPEL4WS. A method is proposed to select optimal service composition according to the cost that the service requester should pay for each successful execution. We compare our method with other methods by simulation which shows our method is distinctly superior to existing methods. For some reasons, only the *<sequence>* container and *<flow>* container of BPEL4WS model are mentioned in the paper. Other containers will be discussed in future explorations.

ACKNOWLEDGMENT

The authors would like to thank Dr. ZHU Feng-hua and Dr. CHEN Song-hang for their help in providing the case study. The authors would also like to thank the colleagues of State Key Lab of Management and Control for Complex Systems.

REFERENCES

- [1] S. Marston, Z. Li, S. Bandyopadhyay, J. H. Zhang, and A. Ghalsasi, "Cloud computing — The business perspective" *Decision Support Systems*, vol. 51, no. 1, pp. 176-189, April 2011.
- [2] P. C. Xiong, Y. S. Fan, and M. C. Zhou, "Web Service Configuration Under Multiple Quality-of-Service Attributes," *IEEE Transactions on Automation Science and Engineering*, vol. 6, no. 2, pp. 311-321, April 2009.
- [3] D. Kyriazis, K. Tserpes, A. Menychtas, A. Litke, and T. Varvarigou, "An innovative workflow mapping mechanism for Grids in the frame of Quality of Service," *Future Generation Computer Systems*, vol. 24, no. 6, pp. 498–511, June 2008.

- [4] R. Chinnici, M. Gudgin, J. J. Moreau, and S. Weerawarana, "Web Services Description Language (WSDL) Ver. 1.2," *World Wide Web Consortium, 2002*. [Online]. Available: www.w3.org/TR/wsdl12/.
- [5] N. Zhang, F. Y. Wang, F. H. Zhu, D. B. Zhao; and S. M. Tang, "DynaCAS: Computational Experiments and Decision Support for ITS", *IEEE Intelligent Systems*, vol.23, no.6, pp.19-23, Nov./Dec. 2008.
- [6] C. Rudolph, N. Kuntze, and Z. Velikova, "Secure Web Service Workflow Execution," *Electronic Notes in Theoretical Computer Science*, vol.236, no.2, pp.33-46, 2009.
- [7] D. Ardagna, and B. Pernici, "Global and local QoS constraints guarantee in Web service selection," in *Proc. 2005 IEEE Int. Conf. Web Services*, Orlando, FL, pp. 805–806, Jul. 2005.
- [8] P. C. Xiong , Y. S. Fan, and M. C. Zhou, "A Petri Net Approach to Analysis and Composition of Web Services", *IEEE Transactions on SYSTEMS,MAN,AND CYBERNETICS* vol.40, no.2, pp.376-387, Mar. 2010
- [9] S. Nurcan. "Analysis and design of co-operative work processes: a framework," *Information and Software Technology*, vol.40, no.3, pp.143- 156, 1998.
- [10] D. A. Menasce, "QoS issues in Web services," *IEEE Internet Computing*, vol. 6, no. 6, pp. 72–75, Nov./Dec. 2006.
- [11] L. Z. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-aware middleware for Web services composition," *IEEE Transaction Software Engineering*, vol. 30, no. 5, pp. 311–327, May 2004.
- [12] H. P. Lin and Y. S. Fan, "Scheduling method based on workflow management technology," *J. Tsinghua Univ. (Sci. and Tech.)*, vol.43, no.3, pp.402-405, 2003.
- [13] C. L. Huang, C. C. Lo, K. M. Chao, and M. Younas, "Reaching consensus: A moderated fuzzy web services discovery method," *Information and Software Technology*, vol. 48, no. 6, pp. 410–423, June 2006.
- [14] F. Y. Wang, "Agent-Based Control for Networked Traffic Management Systems", *IEEE Intelligent Systems*, vol.20, no.5, pp.92-96, Sep./Oct 2005.
- [15] J. N. Cao, C. Chan, K. Chan. "Workflow analysis for web publishing using a stage-activity process model," *The Journal of Systems and Software*, vol.76, no. 3, pp.221–235, June 2005.
- [16] S. Liu, Y.S. Fan, and H.P. Lin. "Dwelling time probability density distribution of instances in a workflow model," *Computers & Industrial Engineering*, vol 57, no. 3, pp. 874-879, Oct. 2009.
- [17] R. N. Riggins and W. B. Ribbens, "Designed Inputs for Detection and Isolation of Failures in the State Transition Matrices of Dynamic Systems," *IEEE Transactions on Control Systems Technology*, vol. 5, no. 2, MAR 1997.