On Establishing QoS for Composite Services

Nguyen Van Son, Vo Dinh Hieu

Faculty of Information Technology – University of Engineering and Technology

{sonnv_56,hieuvd}@vnu.edu.vn

Abstract- *Service composition is a process of combining existing atomic services to perform a complex task. To choose atomic services for a composite service, usually both functionality and quality of service (QoS) are considered. The QoS of the composite service obviously depends on QoS of atomic services. In this research, we propose a simple approach for the composite service and atomic services to establishing suitable QoS values. Experiment shows that our approach is feasible and significantly better than the exhaustive approach in performance aspect.*

Keywords-SOA; Web services; service composition; negotiation protocol; QoS

I. INTRODUCTION

Service-oriented architecture (SOA) is a trend in developing IT infrastructures. With this architecture, systems become flexible and they easily respond to requirements from business.

Services are the core of SOA systems. A service provides a function or some functions to build up systems. Services in SOA systems are implemented based on widely used standards and/or protocols such as HTTP and XML. One of the strong points of SOA is the ability to be composed of services. In SOA, new services can be created by composing existing services. Today, the Web service technology is the main technology for implementing SOA and WS-BPEL [1] is the most popular language for service composition.

To illustrate the service composition process, we can take a service for a travel agent as an example (Fig. 1). The Travel Web service (WS) is a composite Web service. This service provides information about hotels, car rental, and weather forecast. This information is obtained from three atomic Web services Hotel WS, Car WS, and Weather WS, respectively.

With the popularity of Web services, there will be the case in that several Web services provide the same functionality but with different quality of service (QoS). For example, there are three Web services W_1, W_2, and W_3 providing information about weather. H_1 and H_2 are services offering information about hotels, meanwhile C_1, C_2, and C_3 are services for car rental information.

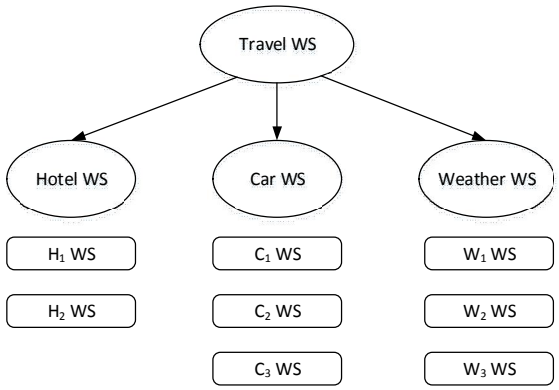

Figure 1. The Travel Web service

These services have different QoS attributes as shown in Table 1. For the sake of simplicity, not all QoS attributes of these services are shown.

Table 1: QoS Attributes of automic services

Service	Price ($)	Duration(ms)	Availability (%)
W_1	1	0.3	99
W_2	0.5	0.5	97
H_1	2	0.4	98
H_2	2.5	0.3	99
C_1	1	0.6	99

With the existing Web services, we have several options for realizing the Travel WS. For example, we may implement Travel WS as the composition of (H_1, C_1, W_1) or (H_2, C_1, W_1), and so on. Each option results in different QoS of the Travel Web service.

This situation originates a challenge for composition processes: how to select suitable atomic Web services with respect to the required QoS of the composite Web service. A possible scenario is that a user sends a QoS requirement to a composite service; the composite service selects atomic services so that the composition will satisfy the requirement. The QoS requirement of the user is also mentioned as the global QoS constraint.

Several works have managed to tackle the mentioned challenge.. In some studies, when QoS data of atomic services is available, an algorithm is exercised to find the optimal solution [2-5]. These studies are classified as service selection studies. In another direction, negotiation processes are established between the composite service and the atomic services for helping two sides to agree on QoS values.

In proposed approaches for QoS negotiation, an important step is to decompose the global constraint into local constraints. These local constraints are then used as the input for the negotiation. The decomposition ensures that the composition (if found) will satisfy the global constraint. However, it is difficult to ensure that a certain local constraint is suitable.

In this paper, we propose an approach to establish suitable QoS attributes between the composite service and the atomic services during the composition. Our approach is in between of service selection and service QoS negotiation. This means our approach is more flexible than approaches proposed in mentioned service selection studies but simpler than negotiation approaches. Our approach does not decompose the global constraint as in almost negotiation approaches. While the decomposition of global constraint is considered as top-down approach, our approach is a bottom-up one.

The rest of the paper is organized as follows. The next section presents related work. After that, we present the models of QoS in service composition. Based on these models we present our approach in section 4. The last section shows experiment results and discusses our proposed approach.

II. RELATED WORK

Since Web service technology has been considered as the main means for realizing service-oriented architecture, many works on Web service QoS negotiation have been introduced [6-10]. However, these studies mainly focus on the negotiation process between a consumer and a Web service (or between their representatives), and leave out the cases of service composition.

In the context of service composition, several works have tackled the issues of QoS negotiation [11-13]. Yan et al. [11] proposed an approach to generate SLA of composite service through negotiation. They proposed a framework based on multi-agent systems in which the service consumer (the composite service) is represented by an agent to negotiate with service providers. A negotiation protocol was proposed. Based on utility function, agents decide activities needed during the negotiation process.

Richter, J., et al. [13] proposed an negotiation approach with the decomposition of the global utility into individual service utilities. These utilities determine the strategy of the negotiation process. In their study, they focused on utility boundaries. Utility boundaries of atomic services are derived from the global utility boundary. They also proposed a mechanism for updating utility boundaries of atomic service during the negotiation process.

Sun et al. [12] presented an QoS negotiation approach based on the decomposition of global constraints to local constraints proposed by Alrifai et al. [3]. During the negotiation process, the proposed values and the local constraints are updated by using Bayes theorem.

As mentioned above, theses studies focused on the decomposition of the global constraint into local constraints before conducting the negotiation process. In our opinion, while the decomposition process takes time, proving the efficiency of this process is difficult. We

have not found any reasonable experiment in the mentioned works.

III. QOS MODEL AND PROBLEM STATEMENT

In this section, we present the model for QoS in service composition. This model is used to describe our approach in the next section. The model presented here is similar to the one proposed by Alrifai and colleagues [3]. The main difference is that our model uses tuples instead of sets for composite services as in their work. The reason is the roles of atomic services during the composition should be differentiated. By using tuples, we utilize the order in tuples to present roles of atomic services.

A. Service Composition

A composite service *CS* performs a complex task $T = (t_1, t_2, ...t_n)$. This complex task includes N simple tasks $t_i, 1 \le i \le n$. We can consider an *abstract* composite service CS is an n-tuple $(S_1, S_2, ..., S_n)$ where $S_i = \{s_1^i, s_2^i, ..., s_m^i\}$ is a service class and all services in this class have the same functionality (t_i). At this point, we have not decided which atomic services are involved in CS. Therefore, CS is considered as an abstract service. Services in a service class are concrete atomic services. They provide the same functionality but different QoS attributes.

A *concrete* composite service cs is modeled as an n-tuple $cs = (s_1, s_2, ..., s_n)$ where s_i is a concrete atomic Web service in class S_i.

There are four models for service composition including sequential model, loop model, parallel model and conditional model. In the sequential model, t_{i+1} will be not performed until t_i completes. The loop model must be defined with a maximum loop count M for iterative task (t_i), so that s_i is not executed for more than M times. A parallel composite service will trigger all tasks to be executed in parallel. On the other hand, the conditional model will trigger only one of the task with a certain probability. A composite service can be developed based on one or several different models. In this paper, we only consider sequential model. As described in a study of Zeng [2], other models can be transformed to sequential models.

B. Quality of Services

1) QoS of Web Services

QoS of a specific Web service s is represented as $Q(s) = (q_1, q_2, ..., q_r)$, where q_k represents the value of the k^{th} QoS attribute. The function $q(s,k)$ can be used to get the value q_k of the service s: $q_k = q(s,k)$.

2) Weights

The importance of each QoS attribute varies between users. A user may consider response time as the most important attribute. Meanwhile, with another user, price of the service is the most crucial factor. We use weights to indicate the relative importance of QoS attributes. The priority of the k^{th} QoS attribute is specified by w_k, where $\sum_{k=1}^{r} |w_k| = 1.0$. The greater $|w_k|$ is, the more importance q_k is. With positive attributes such as availability and performance, $w_k \le 0$. Meanwhile, with negative attributes such as price and execution time, $w_k \ge 0$. These weights are used to calculate the value *Score* for deciding which service is better. This value of a service s is calculated by formula (1).

$$Score(s) = \sum_{k=1}^{r} \bar{q}(s,k) * w_k \qquad (1)$$

$\bar{q}(s,k)$ is the normalized value of k^{th} QoS attribute and $\bar{q}(s,k) = \frac{q(s,k)}{q^{\max}(k)}$ where $q^{\max}(k)$ is the maximum value of the k^{th} QoS attribute of considered services. With this criterion, the service with the lower *Score* is the better.

Table 2: QoS Attributes of composite services

QoS attribute	QoS attribute of composite services
Price	$q'(cs,1)=\sum^{N} q(s_i,1)$
Duration	$q'(cs,2)=\sum^{N} q(s_i,2)$
Availability	$q'(cs,3)=\prod^{N} q(s_i,3)$

3) QoS of Composite Web Services

QoS of a composite service $cs=(s_1,s_2,...,s_n)$ is $Q'(cs)$, and

$$Q'(cs)=(q'_1,q'_2,...,q'_r) \quad (2)$$

where $q'_k, 1\le k\le r$ is the k^{th} QoS attribute of cs. It is obviously that this value depends on corresponding values of atomic services forming the composite service. This means q'_k is determined by $q(s_i,k), 1\le i\le n$. Similar to the case of atomic services, q'_k of the composite service cs can be accessed via the function $q'(cs,k)$.

Depending on QoS attribute, $q'(cs,k)$ can be calculated by following formulas shown in Table 2.

In the example of Travel service, if this service is the composition of H_1, C_1, and W_1, $cs = (H_1, C_1, W_1)$ and the QoS attributes of the service are:

$$Price = q'(cs,1) = q(H_1,1)+q(C_1,1)+q(W_1,1) = 1+2+1 = 3$$

$$Duration = q'(cs,2) = q(H_1,2)+q(C_1,2)+q(W_1,2) = 0.4+0.6+0.3 = 1.3$$

$$Availability = q'(cs,3) = q(H_1,3)*q(C_1,3)*q(W_1,3) = 0.98*0.99*0.99 = 0.96$$

As a result, $Q'(cs)=(3,1.3,0.96)$

4) QoS Constraints

When sending a request to a composite Web service, users may provide constraints on QoS of the composite service. We define a constraint on QoS is a tuple $C=(c_1,c_2,...,c_r)$, with c_k is the constraint value for k^{th} QoS attribute. A composite service cs is considered as satisfying the constraint C if and only if $satisfy(Q'(cs),C)$ return *true*.

$$satisfy(Q'(cs),C)=true \Leftrightarrow (q'(cs,k)-c_k)*w_k \le 0, \forall k, 1\le k\le r$$

where w_k is the weight of the QoS attribute *k*.

For the Travel service, the composition of (H_1, C_1, W_1) will satisfy the constraint $C_1=(3.5,1.0,0.96)$ but will not satisfy the constraint $C_2=(3.5,1.0,0.98)$.

5) Decision of an Atomic Service

After being asked by the composite service to provide functionality with QoS values, an atomic service must make a decision and send to the composite service. This decision may be "agree" or "not agree" (accept or reject). Assume that the atomic service intends to provide functionality *t* with QoS values $q=(q_1,q_2,...,q_r)$ and the weights for QoS attributes of this service are $W=(w_1,w_2,...,w_r)$. This service "agrees" with the QoS values $q=(q'_1,q'_2,...,q'_r)$ suggested by the composite service if:

$$\sum_{k=1}^{r} \bar{q}(q',k)*w_k \le \sum_{k=1}^{r} \bar{q}(q,k)*w_k \quad (3)$$

This mean *q'* is better than *q* from the atomic service's perspective.

C. Problem Statement

The problem we intend to address in this paper is that: a user specifies a global constraint $C=(c_1,c_2,...,c_r)$ for a service composition; the abstract composite service $CS=(S_1,S_2,\ ...,S_n)$ interacts with concrete Web services in each class so that a concrete composite service $cs=(s_1,s_2,...,s_n)$ is defined; cs must satisfies C and *Score* values of atomic services are as small as possible.

IV. THE PROCESS FOR ESTABLISHING QOS

This section presents the main contribution of our work which is a process for the composite service and atomic services to establish suitable values for QoS attributes.

In general, the composite service collects QoS information of candidates, calculates, determines expected QoS values, and collects responses of atomic services for further processing. Meanwhile, the atomic services play a minor role with the main activity is to send QoS information to the partner (the composite service) and to decide whether the requested values sent from the partner are accepted or not. In the following sections, we only focus on activities at the composite service. We assume that the composite service has the list of candidate services for each class via some discovery protocol which is not concerned in this paper. Main activities of the process at the composite service side are presented in Fig. 2.

A. Collect Information

In this step, the composite service acquires QoS information of candidates in each class. This step initiated by the event of arrival of QoS constraints C from a user. After this step, for all service s in service class S_i, $1 \le i \le n$, $Q(s)$ is defined.

B. Establish Expected QoS values

Based on the data provided by services, the composite service establishes expected QoS values for each class. For service class S_i, the expected QoS values at the round j^{th} of the process are $E(i,j)$. $E(i,j)$ is defined as follows.

At the first round, $j=0$, the composite service asks every service candidate in class S_i whether they can provide functionality t_i with QoS values $E(i,0)=\left(e^i_{0,1}, e^i_{0,2}, ..., e^i_{0,r}\right)$ with $e^i_{0,k} = q^{\min}(k,i)$ is the minimum value of the k^{th} QoS value in class S_i. This means

$$v = q^{\min}(k,i) \rightarrow$$
$$(\exists s \in S_i((q(s,k)=v) \wedge (\forall s' \in S_i, s \neq s', q(s',k) \ge v)))$$

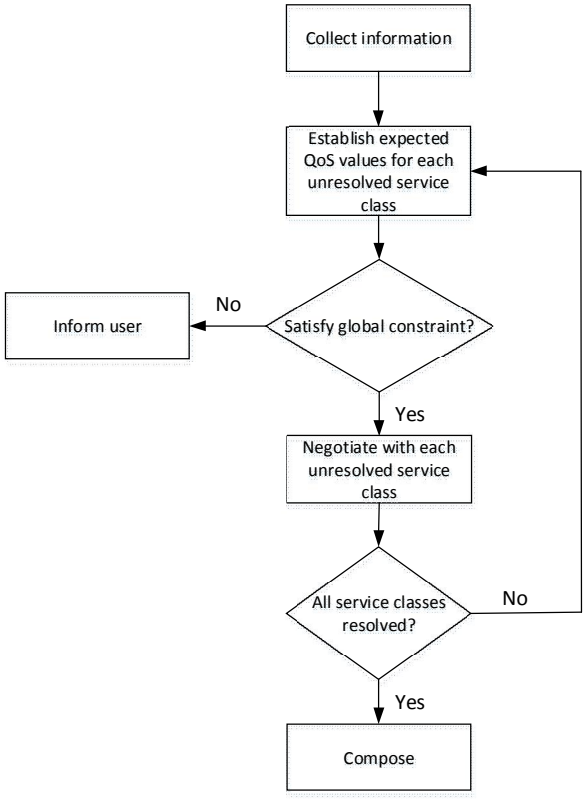

Figure 2. The process for establishing QoS values

If a certain Web service agrees with the QoS values suggested by the composite service, the sub-process for determining QoS for the class S_i finishes.

If after j rounds, there is no service agreeing with QoS values proposed by the composite service, the composite service updates expected QoS values to $E(i,j+1)$ and prepares for the next round $j+1$. $E(i,j+1)$ is determined as follows.

$$E(i,j+1) = NEXT\left(E(i,j)\right) = \left(e^i_{j+1,1}, e^i_{j+1,2}, ..., e^i_{j+1,k}, ..., e^i_{j+1,r}\right)$$

where $\forall l \neq k, 1 \le k, l \le r, e^i_{j+1,l} = e^i_{j,l}$ and $e^i_{j+1,k} = next\left(e^i_{j,k}\right)$

(4)

Here, $next\left(e^i_{j,k}\right)$ is defined as the next smallest value of the k^{th} QoS attribute proposed by all services in the service class S_i. This means

$$next\left(e^i_{j,k}\right) = v \Leftrightarrow$$
$$\exists s \in S_i((q(s,k) = v) \wedge (\forall s' \in S_i, s' \neq s, q(s',k) \geq v > e^i_{i,k}))$$

k is determined by following formula. If the k^{th} QoS attribute is chosen to update, then

$$\forall k' \neq k, 1 \leq k', k \leq r,$$
$$\frac{(next(e^i_{j,k'}) - e^i_{j,k'})}{q^{\max}(i,k')} * w_{k'} \geq \frac{(next(e^i_{j,k}) - e^i_{j,k})}{q^{\max}(i,k)} * w_k \quad (5)$$

where $q^{\max}(i,k)$ and $q^{\max}(i,k')$ are the maximum values of the k^{th} and k'^{th} QoS attributes in class S_i.

C. Check the Global Constraints

After being determined, the expected QoS values are checked with the global constraint provided by the user. As it can be seen from Fig. 2, the assessment is carried out before a new round is conducted. If these expected values do not satisfy the global constraint, the process stops and no solution is found.

The assessment is similar to the case in that we assume that in the each class S_i, there is virtual service s^*_i providing functionality t_i with QoS $Q(s^*_i) = E(i,j)$. In this case, the virtual composite service is cs^* and $cs^* = \left(s^*_1, s^*_2, ..., s^*_n\right)$.

As mentioned in the section III.B.4, the expected QoS values satisfies the constraint $C = \left(c_1, c_2, ..., c_r\right)$ if

$$(q'(cs,k) - c_k) * w_k \leq 0, \forall k, 1 \leq k \leq r \quad (6)$$

D. Negotiate with Unresolved Service Classes

At round j of the process, the expected QoS values $E(i,j)$ of the service class S_i will be sent to all candidates in the class. If a certain service in the class accepts the proposal, the service is selected for composing and class S_i is marked as "resolved" class at round j and no further action for the class is needed.

V. EXPERIMENT AND DISCUSSION

In this section, we present experiment result and discuss several points of the proposed approach. The experiments were conducted on a PC with Intel Core i5 Quad 2.5 GHz CPU and 6GB RAM. Running time is determined from the point the first expected QoS values is established to the end of the process (*"Inform user"* or "*Compose"* in Figure 2).

We implemented our approach and compared with the exhaustive approach. We assume that each atomic service has three QoS attributes. The composite service performs a complex task which includes *n* element tasks. Each task has a corresponding service class which includes $m = 10$ candidates. Values of QoS attributes of services are randomly generated based on predefined values. If the predefined QoS values are $AQ = (Q_1, Q_2, Q_3)$, where Q_k is the predefined QoS value of the k^{th} $(1 \leq k \leq 3)$ attribute, the generated value of the k^{th} attribute should be in the range $[Q_k - 0.2 * Q_k, Q_k + 0.2 * Q_k]$. The k^{th} attribute value of global constraint is $n * rand(Q_k - 0.2 * Q_k,\ Q_k + 0.2 * Q_k)$, where the $rand(a,b)$ function is used to get a random value in the range [*a, b*]. The generation of weights is implemented in a similar method based on predefined weights $AW = (W_1, W_2, W_3)$. We changed the number of service classes from 2 to 20. The result of the experiment is shown in Table 3. The exhaustive approach is implemented as following. With *m* candidates in each class, we form m^r possibilities of QoS values and these values are used as requested QoS. This means the composite service asks services in a class to provide $Q = \left(q_1, q_2, ..., q_r\right)$ where q_k is previously proposed by a certain service in the class.

The term "rounds" refers to loops from step *"Establishing expected QoS values for each unresolved service class"* to *"All service classes resolved?"* (in Figure 2), for both approaches.

In Table 3, we can see that the exhaustive approach is better than our approach in term of finding agreement

between the composite service and atomic services (10 vs. 8). However, regarding time spent for the process, the experiment result shows that our approach is superior to the exhaustive approach.

In our implementation, the QoS establishment process is carried out concurrently between the composite service and atomic services. We assume that each Web service has a QoS component. The QoS establishment process is accomplished by the QoS component of composite service and those of atomic services. The QoS component of the composite service may create a sub-component for each class of services.

Before interacting with services in a specific class, the corresponding sub-component of the composite service must check the satisfaction of the QoS values which are used to request with the global constraint. Hence, the proposed approach can assures that the output of the establishment process always satisfies the global constraint.

In approaches proposed by others, the decomposition of the global constraint into local constraints is an important step. This means the global constraint is always satisfied. However, it is difficult to ensure when the best solution is reached to and when the negotiation process should stop. In our work, *Score* values of expected QoS values increase in each round. Therefore, the first solution is the best one.

In this paper, we let the composite service side to generate the expected values of QoS and the atomic services only simply accept or reject the offer. This is a drawback of the proposed approach. Integration of counter-offers proposed by atomic services is considered in a future work.

VI. CONCLUSIONS

Service composition provides a means for building value-added services. Selecting atomic services for a composition with the consideration of QoS is a challenge. In this paper, we have proposed a simple approach for composite service to interact with atomic services to establish QoS of functionality to be provided.

Table 3. Experiment Results

n	Our approach			Exhaustive approach		
	Time(ms)	Number of rounds	Reach agreement	Time(ms)	Number of rounds	Reach agreement
2	254	13	x	2640	109	x
3	84	2		57	2	
4	228	9	x	2485	92	x
5	224	8		639	22	
6	426	15		3931	128	
7	31			25		
8	421	13	x	3006	102	x
9	267	6		608	15	
10	382	10		343	7	
11	456	12	x	4337	143	x
12	504	15	x	4082	139	x
13	469	10		4816	151	x
14	45			55		
15	420	9		2639	86	x
16	594	13	x	4152	104	x
17	602	13	x	4929	140	x
18	534	9		1738	25	
19	428	6		1752	26	
20	787	16	x	7295	197	x

(More detail can be found at https://www.dropbox.com/s/p4e4gu86b3yr488/data.zip?dl=0)

Our experiment shows that our approach is feasible and better than the exhaustive approach in terms of performance. The plan is to extend our approach to be a negotiation protocol. To do that, we need to consider counter-offers created by atomic services. In addition, we may take into account the relationships between atomic services during the negotiation process.

ACKNOWLEDGEMENT

This research was supported by Vietnam National Foundation for Science and Technology Development (NAFOSTED) grant 102.03-2014.40.

REFERENCES

[1]. OASIS, *Web Services Business Process Execution Language version 2.0,* available from: http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html, accessed

[2]. Zeng, L., et al., *QoS-Aware Middleware for Web Services Composition.* IEEE Trans. Softw. Eng., 2004. **30**(5): p. 311-327.

[3]. Alrifai, M., et al., *A Scalable Approach for QoS-Based Web Service Selection*, in *Service-Oriented Computing --- ICSOC 2008 Workshops*, F. George and L. Winfried, Editors. 2009, Springer-Verlag. p. 190-199.

[4]. Yu, T., Y. Zhang, and K.-J. Lin, *Efficient algorithms for Web services selection with end-to-end QoS constraints.* ACM Trans. Web, 2007. **1**(1): p. 6.

[5]. Alrifai, M., D. Skoutas, and T. Risse, *Selecting skyline services for QoS-based web service composition*, in *Proceedings of the 19th international conference on World wide web*. 2010, ACM: Raleigh, North Carolina, USA. p. 11-20.

[6]. Comuzzi, M. and B. Pernici, *An Architecture for Flexible Web Service QoS Negotiation*, in *Proceedings of the Ninth IEEE International EDOC Enterprise Computing Conference*. 2005, IEEE Computer Society. p. 70-82.

[7]. Zulkernine, F., et al., *A Policy-Based Middleware for Web Services SLA Negotiation*, in *Proceedings of the 2009 IEEE International Conference on Web Services*. 2009, IEEE Computer Society. p. 1043-1050.

[8]. Hung, P.C.K., H. Li, and J.-J. Jeng, *WS-Negotiation: An Overview of Research Issues*, in *Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 1 - Volume 1*. 2004, IEEE Computer Society. p. 10033.2.

[9]. Zheng, X., et al., *Applying Bargaining Game Theory to Web Services Negotiation*, in *Proceedings of the 2010 IEEE International Conference on Services Computing*. 2010, IEEE Computer Society. p. 218-225.

[10]. Mach, W. and E. Schikuta, *A generic negotiation and re-negotiation framework for consumer-provider contracting of web services*, in *Proceedings of the 14th International Conference on Information Integration and Web-based Applications & Services*. 2012, ACM: Bali, Indonesia. p. 348-351.

[11]. Yan, J., et al., *Autonomous service level agreement negotiation for service composition provision.* Future Gener. Comput. Syst., 2007. **23**(6): p. 748-759.

[12]. Sun, S.X., J. Zhao, and H. Wang, *A negotiation based approach for service composition*, in *Proceedings of the 5th international conference on Global Perspectives on Design Science Research*. 2010, Springer-Verlag: St. Gallen, Switzerland. p. 381-393.

[13]. Richter, J., et al., *Utility Decomposition and Surplus Redistribution in Composite SLA Negotiation*, in *Proceedings of the 2010 IEEE International Conference on Services Computing*. 2010, IEEE Computer Society. p. 627-630.

AUTHORS' BIOGRAPHIES

Nguyen Van Son received his B.S. degree from University of Engineering and Technology, Vietnam National University, Hanoi (2015). He is now an assistant teaching at Faculty of Information Technology, University of Engineering and Technology, Vietnam National University, Hanoi. His research interests include Web service and service composition.

Vo Dinh Hieu is working at Faculty of Information Technology, University of Engineering and Technology, VNU Hanoi. He received MSc degree from Leeds University, England and PhD degree from Japan Advanced Institute of Science and Technology. His research interests include Web services, service composition, and service-oriented architectures.