Int. J. Nonlinear Anal. Appl.
Volume 12, Special Issue, Winter and Spring 2021, 255-263
ISSN: 2008-6822 (electronic)
http://dx.doi.org/10.22075/ijnaa.2021.5116

Introducing test data-set for the QoS-aware web-services discovery and composition

Morteza Khani Dehnoi[a], Saeed Araban[a*]

[a] *Department of Computer Engineering, Faculty of Engineering, Ferdowsi University of Mashhad, Mashhad, Iran.*

(Communicated by Ehsan Kozegar)

Abstract

The QoS-aware web services discovery and composition are considered as two important, open and hot issues in Service-Oriented Architecture (SOA). By definition, web-service discovery is about how to select the best web-service for a role in a workflow among available web-services whereas web-services composition works on developing merely optimum coordination among a number of available web-services to provide a new composed web-service intended to satisfy some users requirements for which a single web service is not (good) enough. The criteria, upon which the web service selection, position, and composition processes are based, may or may not cover the Quality of services (QoS) parameters. The latter case would turn the name of the job into QoS-aware web services discovery and composition. In this article, the QoS-WSC test data-set is introduced for the QoS-aware web-services discovery and composition with regard to specific potentials and needs of the web-services world. In this respect, at first, an interface has been designed to define QoS for the roles attended in a service-oriented architecture. Then a solution, which allows comparison among web-services through the calculation of similarity of the request to amounts of the QoS parameters of available web services, is proposed. The similarity is obtained using the internal multiplication of two vectors of weighted numerical parameters representing request parameters and QoS parameters of available web services. The weights are technically significant coefficients, which can be obtained from the QoS-WSC data-set, which is assembled out of a rule-based integration of two well-known prior data sets in the field of web services.

Keywords: web-service, Service-Oriented Architecture (SOA), data-set, Quality of Service (QoS)

1. Introduction

In Service-Oriented Architecture (SOA), in addition to the functional specification, services are also known by their non-functional attributes usually referred as Quality of Services (QoS) attributes.

*Corresponding Author: Saeed Araban
Email addresses: m.khani@mail.um.ac.ir (Morteza Khani Dehnoi[a]), araban@um.ac.ir (Saeed Araban[a*])

Received: June 2020 *Accepted:* January 2021

The importance of QoS attributes shows itself where among several services that are equal in functionality, the chosen one is the most satisfactory regarding the QoS preferences of the customer. Some researchers have considered QoS and user preferences as two separate categories and believe that some QoS attributes such as trustworthiness are not preferable and the QoS-aware web-services discovery and composition methods should consider them without mentioning the request by the user [25]. Nevertheless, various aspects of QoS should be considered together, and based on the requirements of the service requester. Otherwise, the use of services with improper quality may cause undesirable side effects.

2. Problem statement

In SOA, the selection of services for the roles of a workflow is done often automatically by a software agent. As a result, the human agent does not directly enact any preferences among services applied for roles in architecture. This issue confirms the necessity of the existence of an interface that transfers the non-functional preferences of the human agent to the software agent. The interface should be able to translate the human agent preferences to some applicable intel for the service selection rules, which is carried out by the software agent.

To declare the values of the quality of service from a human agent to a software agent or to evaluate the qualitative properties of a service by a software agent, QoS must be defined formally and in a writable form. In other words, a system is required to have both abilities of quantification of the qualitative data provided by the user in a uniform and standard way, and the provision of a mechanism for comparing two services in terms of quality.

3. Related works

Looking at their general approach, the proposed methods for the stated problem can be separated into two major branches. The first is for the ones, which try to classify services qualitatively. In this way, users can choose among categories (class or cluster) of quality into which the services are put. Each service is registered in one of the predefined classes as soon as it is available to users. In addition, according to some rules, the existing services can switch between existing classes. A few methods of this kind has been presented in [17, 15]. The major areas of failure for these methods are the estimation of the number and proper specification of classes and the rules of class selection for newly arrived services (cold start).

The second branch is owned by the methods based on incorporating the values of the QoS attributes into service packaging. That is, in the service specification (Web Service Definition Language (WSDL) file), in addition to the service interface (specification of service prerequisites and the results) and the functional specification of service (introducing service operations), a section for non-functional (qualitative) service specification is also included. In fact, the methods of this branch are only differing in the way of mapping qualitative data into quantitative data. The mapping, for some methods, turns the complex quality of service just into a single number [4, 3], while for some others, it is mapped to a vector [6**?** , 21]. Few methods can also be found in which the quality of service is mapped to a distribution function. Mapping QoS to a quality vector is most common. The proposed method in the present research falls into this category.

4. Proposed method

This section started by introducing an interface designed by the author for declaring QoS among the roles involved in a service-oriented architecture. Then a solution for comparing web-services using vector model analysis is proposed. Afterward, two test data-sets in the domain of web-services discovery and composition are introduced from which the QoS-WSC test data-set for QoS-aware web-services discovery and composition has been further integrated.

4.1. Designing an interface for QoS declaration

Quality of Service (QoS) is a combination of several quality properties of a service. Aspects of QoS related to SOA, have been listed below [13], [2]:

1. $Cost(q_c)$: It returns the aspects related to the cost or charge of a service.

2. $Performance(q_{perf})$: It takes the speed of service execution into account. Speed, in addition to performance, may also indicate as response time, throughput rate, and so on. Performance sometimes refers to a combination of the mentioned attributes in which execution rate is involved.

3. $Security(q_{sec})$: A hybrid attribute providing the possibility to implement identification, authorization, authentication, secrecy, non-repudiation, and resistance against certain types of cryptanalytic attacks and other security-related aspects. Further investigation into the issue is outside the scope of this research.

4. $Reputation(q_{re})$: It is a criterion for trustworthy of the service providers. Usually, the reputation of a provider or service depends on the end-users' notes and comments. Reputation can give a reason for the service provider's claims to be believed at the time of service advertisement.

5. $SuccessfulExecutionRate(q_{suc})$: It returns the probability that a request for a particular service gets a response successfully, maximally within the expected time.

6. $Availability(q_{av})$: It refers to the probability that a service is available to hire; the availability of a service.

Suppose that every aspect of QoS for each service can take a value limited to a certain range. Therefore, a six-element vector can illustrate the S_i service in terms of QoS:

$$QoS(S_i) = (q_c^{si}, q_{perf}^{si}, q_{sec}^{si}, q_{re}^{si}, q_{suc}^{si}, q_{av}^{si})$$

We call it the QoS index vector for the Si service. In the software engineering world, each one, has the aforementioned qualitative property that has its formal definition [13] in which the way of mapping the attribute's quality into a numerical value has been considered.

In SOA, the best source for retaining values attributable to the QoS parameters is the Service Level Agreement (SLA) from the WSDL specification of that service.

4.2. Comparison of services using vector model analysis

In the proposed method, the service requester, in addition to describing the requested service as a set of available inputs (pre-conditions) and required outputs (post-conditions), must provide a QoS specification. The specification expresses the non-functional requirements to the service discovery agent. It will hold a maximum and a minimum value for each QoS attribute. In addition, based on the importance of each QoS attribute, the mentioned specification will weight each aspect. In other words, the service requester for the discovery agent must generate three QoS index vectors for maximum values, minimum values, and for importance coefficients separately.

Among the available services, which functionally meet the requester's needs, the best match for the request would primarily require to have all six QoS index vector values within the valid range stated by the requester. Among services having the primary condition, the greatest value obtained from scalar product (SOP) of QoS attributes vector into importance coefficients vector, would determine the chosen one.

This enacts that service discovery agent, when comparing candidate services for a role, should first check for each QoS attribute, the values of QoS index vector related to the candidate service to be less-than-or-equal/greater-than-or-equal to the corresponding values in the maximum/minimum vectors (from the request). Secondly, for each candidate service, the agent must calculate the scalar product of the QoS index vector (of candidate service) in the QoS importance coefficients vector (from the request). Candidate services can then be sorted in ascending-order for prioritization based on the obtained values (i.e. higher SOP value means higher chance).

The latter conceptually embraces that among all services having the first condition, the ones whose balls of quality are most in the bigger baskets of importance form the requester perspective, have a higher chance of being chosen.

4.3. Introducing two test data-sets in the area of web-service discovery and composition

WSC05 is a test data-set which has been generated and used as the data ground of a test environment to hold the competition for service discovery and composition on the sideline of holding the ICEBE Conference [12] and is regarded as the basis for other test data-sets in this field. It contains 27 service repositories, smallest of which contains 2,101 WSDL files while its largest data repository holds a merge of 8,356 WSDL files to describe the available services. Each WSDL file contains a functional specification of the service (in the form of request and response messages) as well as the specification of the service interface (in the port message format). This test data-set contains 99 service discovery problems in the form of 9 query files and 198 service composition problems in the form of 18 query files. Each query file has a corresponding solution file, which holds all possible solutions for the problems stated there. The solutions are proposed by the BPEL (Business Process Execution Language) standard.

QWS ver2.0 [18], [1] is a test data-set that holds real values of 9 QoS attributes for 2,507 online services. The values of this test data-set during 2008 have been extracted from public UDDI sources. This test data-set has already been used in over 9,000 researches. In this data-set, a QoS index vector containing nine values is retained for each service. The QoS attributes used in this test data-set are described in Table 1.

Table 1: QoS attributes available in QWS ver2.0 data-set [18]

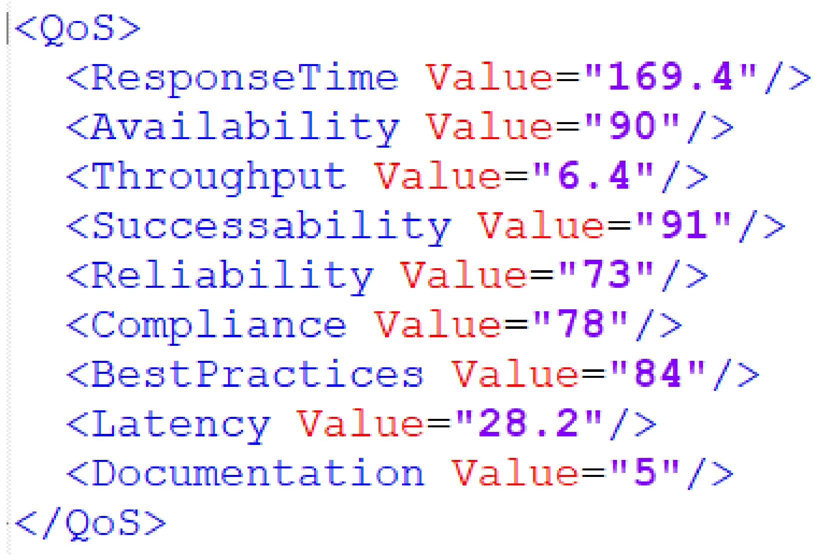
ID	Parameter Name	Description	Units
1	Response Time	Time is taken to send a request and receive a response	Ms
2	Availability	Number of successful invocations/total invocations	%
3	Throughput	Total Number of invocations for a given period of time	invokes/second
4	Successability	Number of response/number of request messages	%
5	Reliability	The ratio of the number of error messages to total messages	%

ID	Parameter Name	Description	Units
6	Compliance	The extent to which a WSDL document follows WSDL specification	%
7	Best Practices	The extent to which a Web service follows WS-I Basic Profile	%
8	Latency	Time is taken by the server to process a given request	Ms
9	Documentation	The measure of documentation (i.e. description tags) in WSDL	%
11	Service Classification	Levels representing service offering qualities (1 through 4)	Classifier
12	Service Name	Name of the Web service	-
13	WSDL Address	Location of the Web Service Definition Language (WSDL) file on the Web	-

4.4. Generating QOS-WSC data-set for QOS-AWARE web-services discovery and composition

By integrating the two test data-sets introduced, the QoS-WSC test data-set as an independent and comprehensive test data-set for the discovery and composition of services is generated with the following order and has been provided for all researchers [14].

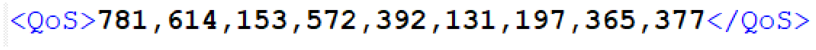
In the QoS-WSC data-set, to generate WSDL files of available services, for each service described in WSC05, a QoS index vector from QWS ver2.0 has randomly been selected and added to the WSDL file of the service in the form of a QoS tag. An example of the generated QoS tag is presented in Figure 1. An example of a WSDL file from the QoS-WSC data-set is available in Appendix 1.



```
<QoS>
   <ResponseTime Value="169.4"/>
   <Availability Value="90"/>
   <Throughput Value="6.4"/>
   <Successability Value="91"/>
   <Reliability Value="73"/>
   <Compliance Value="78"/>
   <BestPractices Value="84"/>
   <Latency Value="28.2"/>
   <Documentation Value="5"/>
</QoS>
```

Figure 1: An example of a QoS tag added to the WSDL services file

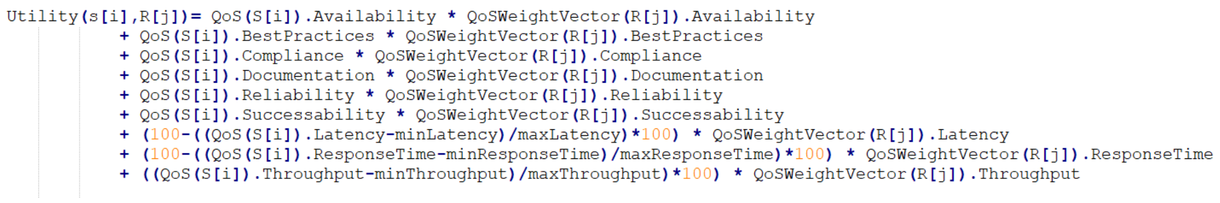
To generate discovery and composition requests in the QoS-WSC data-set, a QoS importance coefficients vector (QoS weight vector) has been added to each request (service discovery or composition) of the WSC05 data-set. Each value of this vector is a random integer number between 0 and 1000. An example of a QoS importance coefficients vector presented in Figure 2. An example of the service composition request file from the QoS-WSC data-set is available in Appendix 2.



```
<QoS>781,614,153,572,392,131,197,365,377</QoS>
```

Figure 2: An example of QoS importance coefficients index vector

To compare the quality of several services (in the process of discovery or composition) the utility (or cost) of the quality of each service should be calculated based on the request (discovery or composition). For this purpose, the inner product of the QoS importance coefficients vector of the request and the QoS index vector of the desired service should be calculated. The equation of Figure 3 shows the calculation process of the utility value of choosing S_i service for R_j request.



```
Utility(s[i],R[j])= QoS(S[i]).Availability * QoSWeightVector(R[j]).Availability
            + QoS(S[i]).BestPractices * QoSWeightVector(R[j]).BestPractices
            + QoS(S[i]).Compliance * QoSWeightVector(R[j]).Compliance
            + QoS(S[i]).Documentation * QoSWeightVector(R[j]).Documentation
            + QoS(S[i]).Reliability * QoSWeightVector(R[j]).Reliability
            + QoS(S[i]).Successability * QoSWeightVector(R[j]).Successability
            + (100-((QoS(S[i]).Latency-minLatency)/maxLatency)*100) * QoSWeightVector(R[j]).Latency
            + (100-((QoS(S[i]).ResponseTime-minResponseTime)/maxResponseTime)*100) * QoSWeightVector(R[j]).ResponseTime
            + ((QoS(S[i]).Throughput-minThroughput)/maxThroughput)*100) * QoSWeightVector(R[j]).Throughput
```

Figure 3: Equation of calculating the utility of $S_i$service selection for R_j request

In the above equation, the QoS $(S[i])$ function returns the values of the QoS index vector of service S_i, and the QoSWeightVector $(R[j])$ function returns the values of QoS importance coefficients vector of the request R_j.

In the QWS ver2.0 data-set, the attributes of Availability, Successfulness, Reliability, Compliance, Best Practices, Documentation, and Throughput are of utility type (which means that higher values for these attributes are desirable). On the other hand, the Latency and Response Time attributes are of cost type (i.e. the lower values they have the more desirable it would be for the service). Also, the values assigned to the attributes of Availability, Successfulness, Reliability, Compliance, Best Practices, and Documentation are numbers of 0 to 100 interval, while the values attributed to Throughput, Latency and Response Time do not have such restriction. Within the calculation of the utility value of choosing S_i service for R_j request (Figure 3), by scaling operation, the values for all attributes are converted to the utility type in the range of 0 to 100.

In the WSC05 data-set, for each service discovery request, all answer services are available (files named as "Solutions"), thus all possible answer services for the corresponding request in the QoS-WSC data-set (whose requests contain the QoS importance coefficients index vector) are also available. Therefore, for each answer service, the qualitative utility can be calculated from the equation of Figure 3, and thus the best (with more utility) service can be selected.

Also in the WSC05 data-set, for each composition request, all possible answer compositions are available (files named as "Solutions"), thus all possible compositions for the corresponding request in the QoS-WSC data-set (whose requests contain the QoS importance coefficients index vector) are also available. To calculate the aggregated utility of each answer composition, it would be enough, to sum up the utility values for all the services that appeared in the composition. As mentioned before, the utilities are calculated using the equation indicated in Figure 3. Having aggregated utilities of all compositions compared the highest utility value would give the best composition.

For all service discovery and composition requests in the QoS-WSC data-set, the best answers (in terms of QoS) have been calculated and recorded in the form of files each of which named "Best Solutions". Appendix 3 shows the "Best Solutions" file for the composition request in Appendix 2.

5. Conclusions

In this article, the QoS-WSC test data-set is introduced and provided for the researchers, to be used for QoS-aware web-services discovery and composition. Service specification files (WSDL files) of this data-set have been generated by injecting QoS index vectors into service specification files of the WSC05 data-set. The injected QoS index vectors have been randomly picked from the QWS ver2.0 data-set. For generating service discovery and composition requests, the QoS importance coefficients vector has also been added to requests available in the WSC05 data-set. The optimal answer is also calculated and recorded for each request.

It should be noted that the QoS-WSC test data-set, introduced in this article, has been used for the first time in [10].

References

[1] E. Al-Masri and Q. H. Mahmoud, *Investigating web services on the world wide web*, Proc.e 17th Int. Conf. World Wide Web 2008, WWW'08, 2008, pp. 795–804.

[2] S. Araban and L. Sterling, *Quality of service for web services*, WSEAS Trans. Comput. 3 (2004).

[3] P. Bocciarelli and A. D'Ambrogio, *A model-driven method for describing and predicting the reliability of composite services*, Softw. Syst. Model. 10(2) (2011) 265—280.

[4] S. Chattopadhyay, A. Banerjee and N. Banerjee, *A fast and scalable mechanism for web service composition*, ACM Trans. Web, 11(4) (2017) 1—36.

[5] F. Chen, S. Yuan and B. Mu, *User-QoS-based web service clustering for QoS prediction*, Proc. 2015 IEEE Int. Conf. Web Services, ICWS 2015, 2015, pp. 583–590.

[6] A. D'Ambrogio and Andrea, *A Model-driven WSDL Extension for Describing the QoS ofWeb Services*, 2006 IEEE Int. Conf. Web Services (ICWS'06), 2006, pp. 789–796.

[7] A. D'Ambrogio, "Model-Driven Quality Engineering of Service-Based Systems," Springer, Berlin, Heidelberg, 2010, pp. 81–103.

[8] A. D'Ambrogio, *A WSDL extension for performance-enabled description of web services*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2005, vol. 3733 LNCS, pp. 371–381.

[9] A. D'Ambrogio, *A model-driven WSDL extension for describing the QoS of web services*, Proc-ICWS 2006: 2006 IEEE Int. Conf. Web Services, 2006, pp. 789–796.

[10] M. K. Dehnoi and S. Araban, "Automatic qos-aware web services composition based on set-cover problem," Int. J. Nonlinear Anal. Appl., vol. 12, no. 1, pp. 87–109, Feb. 2020.

[11] D. Z. G. Garcia and M. B. F. De Toledo, *Semantics-enriched QoS policies for web service interactions*, ACM International Conference Proceeding Series, 2006, vol. 192, pp. 35–44.

[12] ICEBE 2020." [Online]. Available: https://conferences.computer.org/icebe/2020/index.htm. [Accessed: 16-Feb-2020].

[13] ISO/IEC 13236:1998 Information technology - Quality of service: Framework. 1998.

[14] M. Khani and S. Araban, *QoS-WSC*, Mendeley Data, 2020. [Online]. Available: https://data.mendeley.com/datasets/tjm3jnnj5t/1.

[15] M. K. Dehnoi and M. K. Dehnoi, *Fast fault localization in optical WDM networks*, 2nd International Congress on Technology, Communication and Knowledge, ICTCK 2015, 2016, pp. 332–336.

[16] R. Mohana and D. Dahiya, *Optimized service discovery using QoS based ranking: A fuzzy clustering and particle swarm optimization approach*, Proc. Int. Computer Software and Applications Conference, 2011, pp. 452–457.

[17] M. A. Moulavi, B. Bahmani, M. Sadeghizadeh, J. A. Nasiri, H. Parvar and M. Naghibzadeh, *DHA-KD: Dynamic hierarchical agent based key distribution in group communication*, Proc. 9th ACIS Int. Conf. Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD 2008 and 2nd Int. Workshop on Advanced Internet Technology and Applications, 2008, pp. 301–306.

[18] *QWS dataset (the qality of service for web services dataset)*, [Online]. Available: https://qwsdata.github.io/. [Accessed: 16-Feb-2020].

[19] T. Rajendran, P. Balasubramanie and R. Cherian, *An efficient WS-QoS broker based architecture for web services selection*, Int. J. Comput. Appl. 1(9) (2010) 79—84.

[20] M. Rathore, M. Rathore and U. Suman, *A quality of service broker based process model for dynamic web service composition*, Proc. 3RD Int. Work. Model. Enterp. Inf. Syst. 7 (2011) 1267–1274.

[21] M. Sadeghizadeh and O. R. Marouzi, Securing cluster-heads in wireless sensor networks by a hybrid intrusion detection system based on data mining, J. Commun. Eng. 8(1) (2019) 1–19.

[22] M. Sadeghizadeh and O. R. Marouzi, *A lightweight intrusion detection system based on specifications to improve security in wireless sensor networks*, J. Commun. Eng. 7(2) (2018) 29-–60.

[23] V. Tewari, U. Thakar and N. Dagdee, *Classifying Web Services based on QoS Parameters using Extended Dataset*, Int. J. Comput. Appl. 74(8) (2013) 33-–36.

[24] A. K. Tripathy, M. R. Patra, M. A. Khan, H. Fatima and P. Swain, *Dynamic web service composition with QoS clustering*, Proc. IEEE Int. Conf. Web Serv. ICWS 2014, (2014) 678—679.

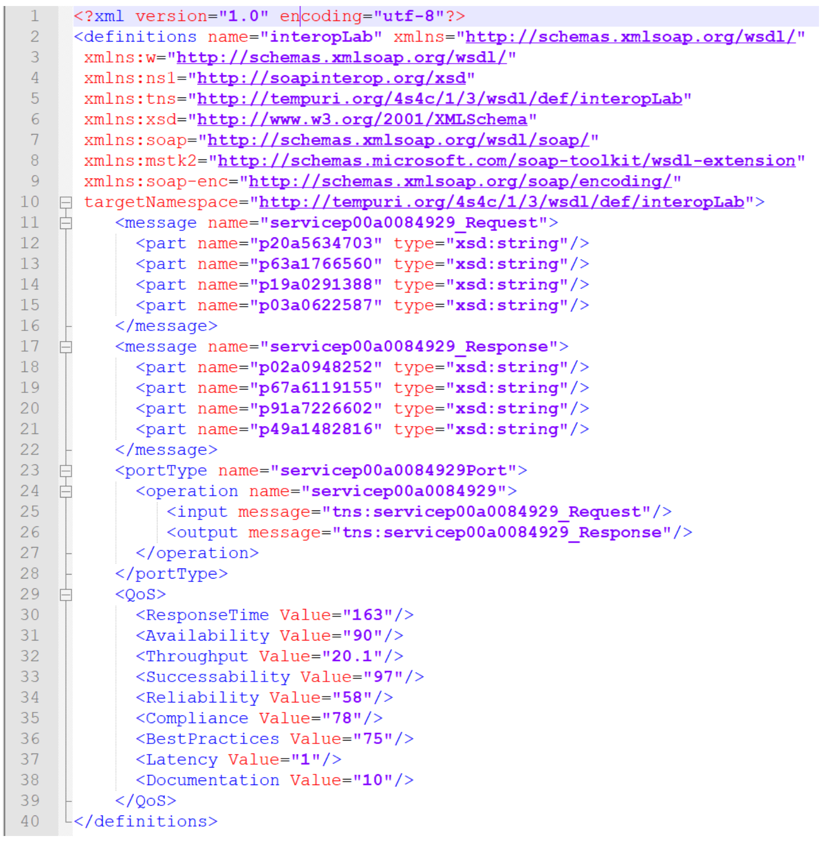
[25] H. Wang, B. Zou, G. Guo, D. Yang and J. Zhang, *Integrating trust with user preference for effective web service composition*, IEEE Trans. Serv. Comput. 10(4) (2017) 574-–588.

[26] J. Wu, L. Chen, Z. Zheng, M. R. Lyu and Z. Wu, *Clustering Web services to facilitate service discovery*, Knowl. Inf. Syst. 38(1) (2014) 207—229.

[27] B. Wu, C. H. Chi and S. Xu, *Service selection model based on QoS reference vector*, Proc IEEE Congress on Services, SERVICES 2007, (2007) 270-–277.

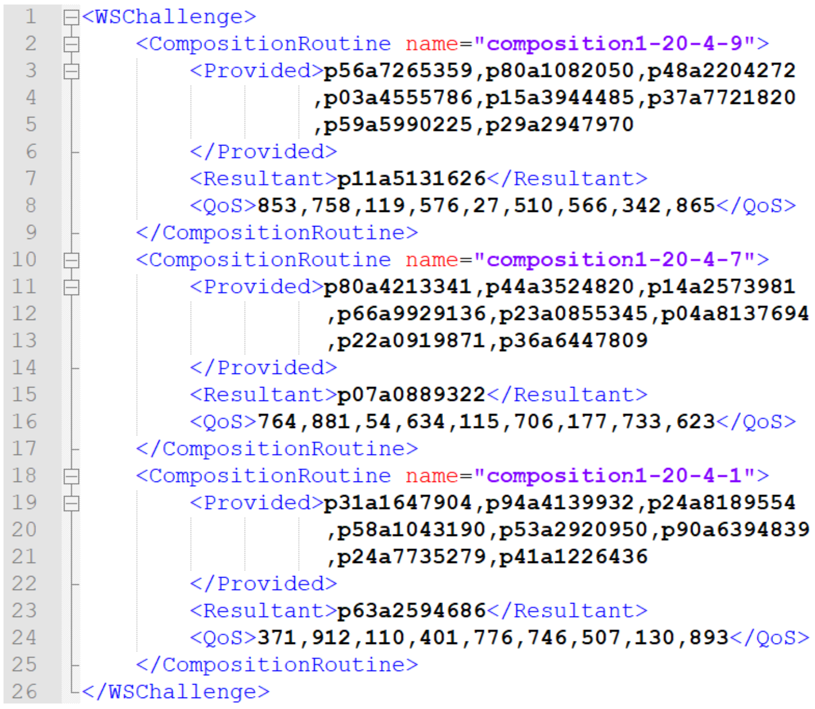
[28] Y. Xia, P. Chen, L. Bao, M. Wang and J. Yang, *A QoS-aware Web service selection algorithm based on clustering*, Proc. IEEE 9th Int. Conf. Web Serv. ICWS 2011, (2011) 428—435.

Appendix



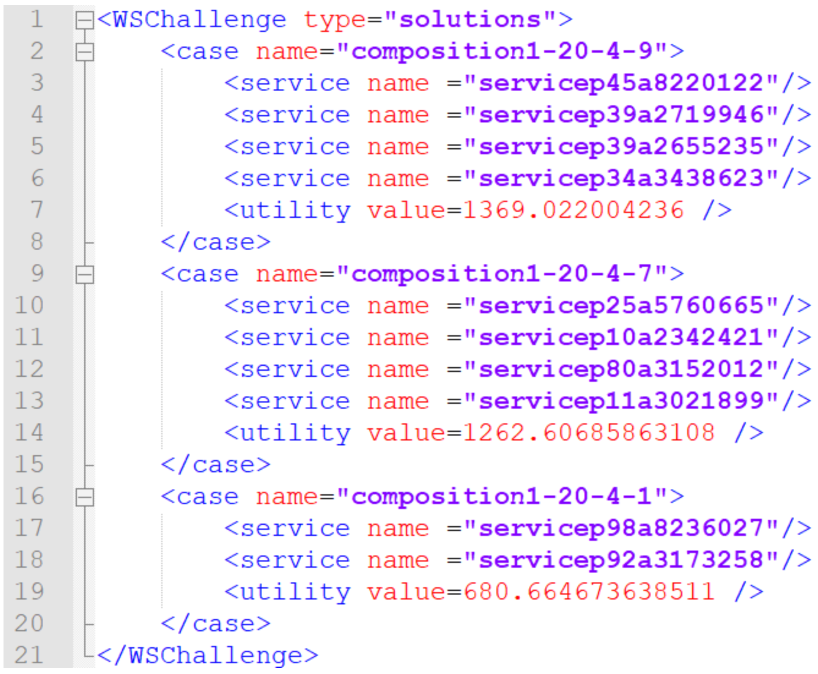
```xml
<?xml version="1.0" encoding="utf-8"?>
<definitions name="interopLab" xmlns="http://schemas.xmlsoap.org/wsdl/"
 xmlns:w="http://schemas.xmlsoap.org/wsdl/"
 xmlns:ns1="http://soapinterop.org/xsd"
 xmlns:tns="http://tempuri.org/4s4c/1/3/wsdl/def/interopLab"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:mstk2="http://schemas.microsoft.com/soap-toolkit/wsdl-extension"
 xmlns:soap-enc="http://schemas.xmlsoap.org/soap/encoding/"
 targetNamespace="http://tempuri.org/4s4c/1/3/wsdl/def/interopLab">
    <message name="servicep00a0084929_Request">
      <part name="p20a5634703" type="xsd:string"/>
      <part name="p63a1766560" type="xsd:string"/>
      <part name="p19a0291388" type="xsd:string"/>
      <part name="p03a0622587" type="xsd:string"/>
    </message>
    <message name="servicep00a0084929_Response">
      <part name="p02a0948252" type="xsd:string"/>
      <part name="p67a6119155" type="xsd:string"/>
      <part name="p91a7226602" type="xsd:string"/>
      <part name="p49a1482816" type="xsd:string"/>
    </message>
    <portType name="servicep00a0084929Port">
      <operation name="servicep00a0084929">
         <input message="tns:servicep00a0084929_Request"/>
         <output message="tns:servicep00a0084929_Response"/>
      </operation>
    </portType>
    <QoS>
      <ResponseTime Value="163"/>
      <Availability Value="90"/>
      <Throughput Value="20.1"/>
      <Successability Value="97"/>
      <Reliability Value="58"/>
      <Compliance Value="78"/>
      <BestPractices Value="75"/>
      <Latency Value="1"/>
      <Documentation Value="10"/>
    </QoS>
</definitions>
```

Appendix 1. A sample of WSDL file from the QoS-WSC data-set



```xml
<WSChallenge>
    <CompositionRoutine name="composition1-20-4-9">
        <Provided>p56a7265359,p80a1082050,p48a2204272
                ,p03a4555786,p15a3944485,p37a7721820
                ,p59a5990225,p29a2947970
        </Provided>
        <Resultant>p11a5131626</Resultant>
        <QoS>853,758,119,576,27,510,566,342,865</QoS>
    </CompositionRoutine>
    <CompositionRoutine name="composition1-20-4-7">
        <Provided>p80a4213341,p44a3524820,p14a2573981
                 ,p66a9929136,p23a0855345,p04a8137694
                 ,p22a0919871,p36a6447809
        </Provided>
        <Resultant>p07a0889322</Resultant>
        <QoS>764,881,54,634,115,706,177,733,623</QoS>
    </CompositionRoutine>
    <CompositionRoutine name="composition1-20-4-1">
        <Provided>p31a1647904,p94a4139932,p24a8189554
                 ,p58a1043190,p53a2920950,p90a6394839
                 ,p24a7735279,p41a1226436
        </Provided>
        <Resultant>p63a2594686</Resultant>
        <QoS>371,912,110,401,776,746,507,130,893</QoS>
    </CompositionRoutine>
</WSChallenge>
```

Appendix 2. A Sample of the file of service composition request from the QoS-WSC data-set



```xml
<WSChallenge type="solutions">
    <case name="composition1-20-4-9">
        <service name ="servicep45a8220122"/>
        <service name ="servicep39a2719946"/>
        <service name ="servicep39a2655235"/>
        <service name ="servicep34a3438623"/>
        <utility value=1369.022004236 />
    </case>
    <case name="composition1-20-4-7">
        <service name ="servicep25a5760665"/>
        <service name ="servicep10a2342421"/>
        <service name ="servicep80a3152012"/>
        <service name ="servicep11a3021899"/>
        <utility value=1262.60685863108 />
    </case>
    <case name="composition1-20-4-1">
        <service name ="servicep98a8236027"/>
        <service name ="servicep92a3173258"/>
        <utility value=680.664673638511 />
    </case>
</WSChallenge>
```

Appendix 3. The "Best Solutions" file for the composition request file Shown in Appendix 2