Computer Systems Science & Engineering
DOI:10.32604/csse.2021.014513
Article

Tech Science Press

Functionality Aware Dynamic Composition of Web Services

Mohemmed Sha* **and Abdalla Alameen**

Department of Computer Science, College of Arts and Science, Prince Sattam bin Abdulaziz University, Wadi Al dawaser, 11991, Saudi Arabia
*Corresponding Author: Mohemmed Sha. Email: sahalshas@gmail.com
Received: 25 September 2020; Accepted: 18 October 2020

Abstract: The composition of the web service is a common technique to attain the best results of complex web tasks. The selection of appropriate web services, linking those services in the action flow and attaining the actual functionality of the task are the important factors to be considered. Even though different frameworks and methods have been proposed to dynamically compose web services, each method has its advantage and disadvantage over the other. Most of the methods give much importance to the Quality of Service (QoS) but fail to achieve the actual functionality after composition. This paper proposes a functionality-oriented composition technique for composing web services. Moreover, this method helps reach the extreme functionality of each web service in the composition towards customer satisfaction. Apart from considering the overall QoS of every single service, the non-functional parameters associated with these services are also considered for achieving the expected functionality. Each of these non-functional parameters has a vital role in the functional performance of the web service. The web services that satisfy the non-functional requirements are chosen to form the composition to attain the best performance. The list of services in the proposed composition method is different from the conventional one, which is composed based on the overall QoS. The non-functional parametric values, the QoS of each web service and the overall QoS after composition are evaluated for the proposed method and experimentally analyzed to prove their advantage over the others.

Keywords: Web service; service composition; quality of service; dynamic composition

1 Introduction

The increased interest in cloud-based web application leads to the use of web services more effectively. To achieve a solution for complex web task through web applications lead to dynamic composition of web services. But it is difficult to select an appropriate web service that is to be included in the composition to satisfy a task. Currently, different techniques are used for the composition of web services for better performance. The objective is to reach the actual functionality of the intended task. The functionality of a web service is defined by the provider while publishing it, but in some cases, it will not be the same in

the real-time usage. So, the quality of a web service comes into effect to address the situation. The QoS is always evaluated based on the functional and non-functional parametric values of the web services. The functional parameters are evaluated based on customer feedback, third-party evaluation surveys and other similar methods [1,2]. Evaluating the QoS based on the non-functional parameter is used by both providers and customers to evaluate the performance of the web service. The evaluated QoS of individual web services in the composition will not always reflect the service satisfaction level of the customer. Hence the overall quality of the composed web services is calculated to tackle the situation. Moreover, the performance of each non-functional parameter of the web service is also recorded. It will be more complex when these individual services are from different providers. So, this work proposed a third-party broker-based framework that dynamically selects the web service that will be included in the composition. Here apart from considering the overall performance of the individual web service, the non-functional parameters that are very much needed to satisfy the task are considered for the best composition. Hence, the composition chain formed in the proposed method ensures the functional requirements of the customer.

The entire paper has organized as follows. Section 2 reviews the related works about the frameworks and methodologies used for the composition of web services. Section 3 explains the method used for the dynamic composition of web services and the related experimental results discussed in Section 4. The advantages of this proposed architecture are concluded in Section 5 and followed by references.

2 Literature Review

Web services are ideal choices to implement customer's web-based tasks. Web service composition is achieved through automatic discovery, selection and binding of web services with a variety of functionalities. But it is a challenging task in terms of time limits, correctness, transactional and other related features. Several composition techniques are used to compose web services for the best results. Surveys are conducted to study the advantage of one composition method over the other [3–5]. Sheng et al. [6], in their survey overviewed the life cycle of web service composition and discussed the various standards and platforms used for web service composition. A variety of approaches and frameworks are followed to compose web services for a web task. But all these methods do not result in the expected quality in real-time usage.

D'Mello et al. [7] proposed a broker based dynamic composition architecture in which the broker generated the composition plan. The candidate web services are selected by the broker based on the input/output parameters and preferences from the requester. Several dynamic composition frameworks are also proposed to improve the performance of web service composition. Most of these frameworks are categorized as supervisor aware, agent aware, context-aware and so on [8]. Sun et al. [9] proposed an algorithmic approach that supports the network-based composition of web services.

These approaches follow different frameworks and architectures for composing web services, but the aim is to reach the best QoS. The QoS of composition is calculated based on the functional and non-functional parameters guaranteed by the providers of the service [10–12]. Lee et al. [13] presented an algorithm that can calculate the QoS of the composition based on non-functional factors of semantic web services. These methods take the evaluation measures guaranteed by the provider to calculate the QoS and try to implement the same in real-time. Almost all the approaches used the dynamic composition of web service irrespective of the techniques and algorithms used by Lu et al. [14,15]. Bertoli et al. [16] considered the behavioral description of web services for the automatic composition of web services. Some researchers proposed that the success of the composition is not only based on the QoS evaluated by the non-functional parameters but also the functional characteristics considered to meet the expected results [17,18]. Most of these frameworks consider the overall QoS based on non-functional parameters

to measure the performance of the composition. But from the perspective of the customer, reaching the actual functionality of the task is much more important than the QoS. One non-functional parameter of a web service may affect the entire performance of the composition. So, this method proposes a technique that analyses the functionality of each web service involved in the composition. Therefore, appropriate non-functional parameters that are needed to maintain the functionality are fixed and only those services are considered in the composition.

3 Proposed Composition Methodology

The proposed service composition architecture has shown in Fig. 1. The service requester describes the complex web task with expected QoS, which will be needed for web application development. After getting the request, the broker classifies the task into several sub-tasks so that this will be carried out by the individual web services. Moreover, the broker evaluates and fixes the expected performance of the web services in terms of individual QoS as well as the expected non-functional parametric values such as response time, reliability, availability, success rate and throughput. A pool of services from various providers with expected performance is generated based on these expected measures. The selected services in each group are ranked based on the overall QoS as well as the fixed threshold value of the non-functional parameters. In the next step, a composition chain is formed to decide the flow of control among these services. Based on the availability of the services from the pool of services, the dynamic composition takes place to accommodate the web task. The processes involved in the proposed service composition framework is discussed in the following steps in detail.

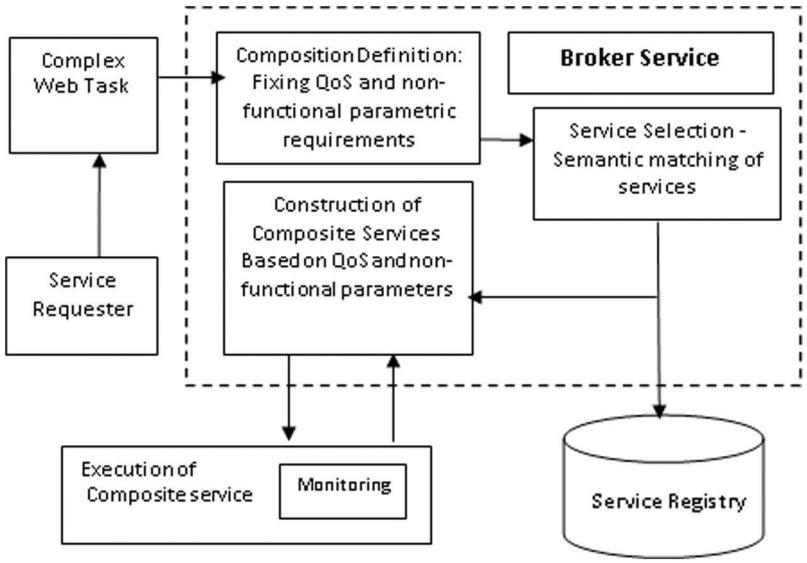

Figure 1: Overview of the proposed hybrid cloud architecture

3.1 Describing the Functionality of the Complex Web Task

The service requester describes the complex web task to be completed along with its functional requirements. This task may contain a series of sub-tasks that may have dependent or independent functionalities. The level of fulfillment of this task is mentioned along with its description. The description of the task from the service requester must be clear and unambiguous with its functional level of satisfaction. This description is forwarded to the broker to assign the best composition of services to activate the task.

For example, Tab. 1 describes the complex web tasks T_1, $T_2 \ldots T_9$ of an airline system that needs the composition of web services for its functionality.

Table 1: Web task description of airline system

Task	Functionality	Overall level of satisfaction
T_1	User accounts—Registration and creation of user profile/login as guest	Reliability – 90%
T_2	Checking availability—Making reservations/Blocking/ Confirmation	Response time—98%
T_3	Confirm ticket	Response time and Reliability—100%
T_4	Reschedule ticket	Response time and Reliability—100%
T_5	Cancellation ticket	Response time—85%
T_6	Update profile	Usability—60%
T_7	View ticket status	Response time—80%
T_8	Query flight details	Response time—80%
T_9	Telephone access	Reliability, Response time—95%

Let us consider the web task T_3, which is used to confirm a ticket that involves various subtasks to complete its process. Several web services are involved in the composition and the requester must have to describe the expected level of satisfaction in terms of functional and non-functional parameters. Tab. 2 describes the functionality as well as the level of satisfaction for each individual sub-task for the main task T_3.

Table 2: Satisfaction level for sub-tasks

Task	Functionality	Level of satisfaction
T_3F_1	Checking the availability	Reliability—95%
T_3F_2	Booking the Ticket	Response—98%
T_3F_3	Finance Service	Response and Reliability—100%
T_3F_4	Confirmation Services (Email & SMS)	Response time and Reliability—95%

The remaining steps in this proposed composition framework will be carried out by the broker and the respective feedback will be given to the service requester to evaluate the performance. This process involves the following steps:

- Describing the complex web task with overall expected QoS.
- Defining, classifying the complex web task into subtasks so that each subtask will be carried out by the atomic web service.
- Fixing the non-functional limits and individual QoS of services.

3.2 Fixing Minimum Expected QoS and Non-Functional Parameters

The services needed to compose are selected based on their functionality. This combination is a group of services from different providers and the local web services used within the environment.

Let T_1, T_2……T_n are the web tasks that the web application needs to be solved by composing suitable web services.

3.2.1 Defining and Evaluating the QoS of the Composite Services

The composition of web service is an important step because the formation and success of the compositions are mainly based on understanding the functionality of each sub-tasks involved in the main task. The process of defining a composite service involves fixing the interaction between the service components, representing the flow of data among the services, supporting the transactional features and its correctness [19]. It is difficult to assure the performance of composite service because of the difficulty in fixing runtime errors. Ensuring the correctness of the composite service is also important to reach the functional requirements [20].

The main non-functional parameters P_1, P_2, P_3, P_4 and P_5 that influence the functionality of the web service are considered to evaluate the QoS for individual services are as follows:

P_1—Response time, P_2—Availability, P_3—Throughput, P_4—Successibility, P_5—Reliability

The equations for defining the task, web services, QoS and composition of web services are as follows

The task to accomplish the user requirement is defined as:

$T_i\ (WS_1, WS_2, \ldots\ldots\ WS_n)$

Here, each WS_i is a web service that satisfies a functionality in the flow of the web tasks described by the service requester.

Each atomic web service involved in the combination is a tuple and will be defined as

WS_i (Id, Functional Type, Input Parameters, Output Parameters, QoS, Expected Non-Functional Parameters)

Here, the 'Id' is the identification number of the service. The QoS is evaluated from the non-functional parameter guaranteed by the providers.

The web services with similar functionality are grouped based on QoS and the non-functional minimum expected values. Only those services satisfy these two criteria are ranked and maintained in groups.

The Quality of Service (QoS) related to the task in each group is defined as:

$QoS(WSi(p_1, p_2, \ldots\ldots p_n)),\ 0 < p_j < 1$

where p_j is the non-functional parametric values.

The QoS of each web service in these categories can be calculated as follows:

$$QoS(WSi) = \sum_{j=1}^{m} Wi.Pj \tag{1}$$

where, $1 \leq j \leq m$ and W_j weight assigned to each non-functional parameter.

After evaluating the QoS, the services are ranked categorically and stored in the service repository.

The composition of services to satisfy the complex task can be defined as:

$C_k(WS_1, WS_2, WS_3, \ldots\ldots WS_n,\ QoS(C_k))$

The overall QoS of the composition is

$$QoS(Ck) = \sum_{j=1}^{n} QoS(Gi.WSj) \tag{2}$$

for all j ($1 \leq j \leq m$)

Here C_k evaluated for all possible 'n' compositions. G_j is the categorized group of services for the subtasks.

3.2.2 Fixing of Minimum Expected QoS and Functionality

After defining the composite service, the next task is to identify the set of web services that need to include the flow of composition based on its functionality. There may be several services are available that satisfy the specific functionality. The minimum expected QoS and level of non-functional parameters are fixed based on the functional requirements of the web service in the composition. These values above the minimum threshold can satisfy the functionality of the web service involved in the composition. Tab. 3 listed the minimum expected overall QoS and non-functional parametric values fixed for future comparison. The data used in this research are collected from the QoS data set provided by the development platform "GitHub" [21].

Table 3: Expected QoS non-functional parametric values

Task	Service category & function	Expected QoS (0–1)	Expected values of non-functional parameters
TF_1	**User Registration**—The API allows developers to integrate their applications with the API service, enabling their users to get and manage customers' account information.	0.817	$P_2 = 0.915$ $P_5 = 0.935$
TF_2	**Booking**—This API returns various booking data for availability, bookings of Air Ticket	0.955	$P_2 = 0.945$ $P_4 = 0.955$
TF_3	**Payment**—The Accounts API allows users to create applications with built-in account management features.	0.975	$P_1 = 0.955$ $P_2 = 0.945$ $P_4 = 0.975$
TF_4	**Confirmation**—The API allows developers to integrate messaging services into their applications, enabling users to send SMS messages and check their status from the applications.	0.935	$P_1 = 0.925$ $P_4 = 0.945$

3.3 Composition Algorithm for Web Services

After defining the composite service, it is necessary to discover the appropriate services to compose. The discovery is based on the matching between the requirement and description of the service. Here semantic matching is proposed to improve the automation of service discovery. Normally, the discovered services are identical in functionality and with different QoS. The QoS of individual web services is evaluated as in Eq. (1). This research proposes to consider the specific non-functional parametric values along with the QoS of each atomic service involved in the flow to attaining the actual functionality of the composition. The overall QoS and non-functional parametric values are compared with the fixed value and those services satisfy this condition are inserted into a stack. These web services are sorted based on the QoS and included in the composition chain based on its availability. The process generator ranks all these

compositions as per the evaluated QoS. The best composition will be selected from this pool of services and executed to complete the task. The proposed algorithm to compose the web service is as follows:

Composition of Web Services (List of services in each task)

```
{
  for each task
  {
     If (Overall QoS > fixed QoS & non-functional parametric values > fixed values)
     {
        Add the web service in the stack of the task
     }
   Sort the stack based on QoS
   if (stack(top) available)
   {
      Add this service in the composition chain
      Record the QoS up to this composition
   }
   else
   {
      Move to the next topmost element in the stack
}
}
```

4 Experimental Results

In this dynamic composition framework, the services that satisfy the expected measures are ranked and grouped into a specific category based on their functionality. For example, an airline reservation tasks may contain functionalities such as user registration, booking, payment and confirmation. The number of services considered in each group is the broker's choice; here top four services are taken into consideration for composition. Tab. 4 shows the services that are categorized into groups based on their ranking.

Table 4: List of services for composition

Task	Service category	Services	Provider	QoS (0–1)	Non-functional expected value (0–1)
T_3F_1	User Registration	$T_3F_1WS_1$—Act-On Account	Act on software	0.898	$P_2 = 0.917$, $P_5 = 0.938$
		$T_3F_1WS_2$—GoSquared Account	GoSquared	0.866	$P_2 = 0.934$, $P_5 = 0.922$
		$T_3F_1WS_3$—Spredfast Account	Spredfast	0.855	$P_2 = 0.945$, $P_5 = 0.936$
		$T_3F_1WS_4$—Amazon Cloud Drive Account	Amazon	0.832	$P_2 = 0.92$, $P_5 = 0.937$

(Continued)

Table 4 (continued).

Task	Service category	Services	Provider	QoS (0–1)	Non-functional expected value (0–1)
T_3F_2	Booking	$T_3F_2WS_1$—Max Booking	Max Booking	0.977	$P_2 = 0.952$, $P_4 = 0.965$
		$T_3F_2WS_2$—OnSched Online Booking	OnSched Online Booking	0.975	$P_2 = 0.964$, $P_4 = 0.952$
		$T_3F_2WS_3$—Bookeo	Bookeo	0.965	$P_2 = 0.955$, $P_4 = 0.956$
		$T_3F_2WS_4$—DotTransfers	DotTransfers	0.964	$P_2 = 0.951$, $P_4 = 0.957$
T_3F_3	Payment	$T_3F_2WS_1$—Max Booking	Max Booking	0.977	$P_2 = 0.952$, $P_4 = 0.965$
		$T_3F_2WS_2$—OnSched Online Booking	OnSched Online Booking	0.975	$P_2 = 0.964$, $P_4 = 0.952$
		$T_3F_2WS_3$—Bookeo	Bookeo	0.965	$P_2 = 0.955$, $P_4 = 0.956$
		$T_3F_2WS_4$—DotTransfers	DotTransfers	0.964	$P_2 = 0.951$, $P_4 = 0.957$
T_3F_4	Confirmation	$T_3F_4WS_1$—Sinch Messaging	Sinch	0.962	$P_2 = 0.932$, $P_5 = 0.955$
		$T3F_4WS_2$—TeleSign SMS Verify	TeleSign	0.961	$P_2 = 0.934$, $P_5 = 0.952$
		$T_3F_4WS_3$—Bandwidth	Bandwidth	0.953	$P_2 = 0.945$, $P_5 = 0.946$
		$T_3F_4WS_4$—Papa Texts	Papa	0.952	$P_2 = 0.931$, $P_5 = 0.947$

The proposed composition algorithm generates all possible combinations of these web services from each group that can be used to develop the composition chains. The QoS of these compositions is evaluated based on Eq. (2) and ranked in a list. The top desirable numbers of compositions that reach the expectation of the requester are selected by the broker for real-time usage. Here top 10 compositions are listed in Tab. 5.

Table 5: Selected top web service compositions

Task	Services involved in the composition flow	Overall QoS of the composition
C_1	$T_3F_1WS_1 \rightarrow T_3F_2WS_1 \rightarrow T_3F_3WS_1 \rightarrow T_3F_4WS_1$	0.9575
C_2	$T_3F_1WS_1 \rightarrow T_3F_2WS_1 \rightarrow T_3F_3WS_1 \_\rightarrow T_3F_4WS_2$	0.95725
C_3	$T_3F_1WS_1 \rightarrow T_3F_2WS_2 \rightarrow T_3F_3WS_1 \rightarrow T_3F_4WS_1$	0.957
C_4	$T_3F_1WS_1 \rightarrow T_3F_2WS_2 \rightarrow T_3F_3WS_1 \rightarrow T_3F_4WS_2$	0.95675
C_5	$T_3F_1WS_1 \rightarrow T_3F_2WS_1 \_\rightarrow T_3F_3WS_2 \rightarrow T_3F_4WS_1$	0.95575
C_6	$T_3F_1WS_1 \rightarrow T_3F_2WS_1 \rightarrow T_3F_3WS_2 \rightarrow T_3F_4WS_2$	0.9555
C_7	$T_3F_1WS_1 \rightarrow T_3F_2WS_1 \rightarrow T_3F_3WS_1 \rightarrow T_3F_4WS_3$	0.95525
C_8	$T_3F_1WS_1 \rightarrow T_3F_2WS_2 \rightarrow T_3F_3WS_2 \rightarrow T_3F_4WS_1$	0.95525
C_9	$T_3F_1WS_1 \rightarrow T_3F_2WS_1 \rightarrow T_3F_3WS_3 \rightarrow T_3F_4WS_1$	0.955
C_{10}	$T_3F_1WS_1 \rightarrow T_3F_2WS_1 \rightarrow T_3F_3WS_1 \rightarrow T_3F_4WS_4$	0.955

The following Fig. 2 represents the QoS of top compositions along with functionality ranges each web service involved in the composition. From the diagram, the QoS web services compositions are up to the expectation of the requester. The individual level of parametric values is also maintained to reach the actual functionality of the task.

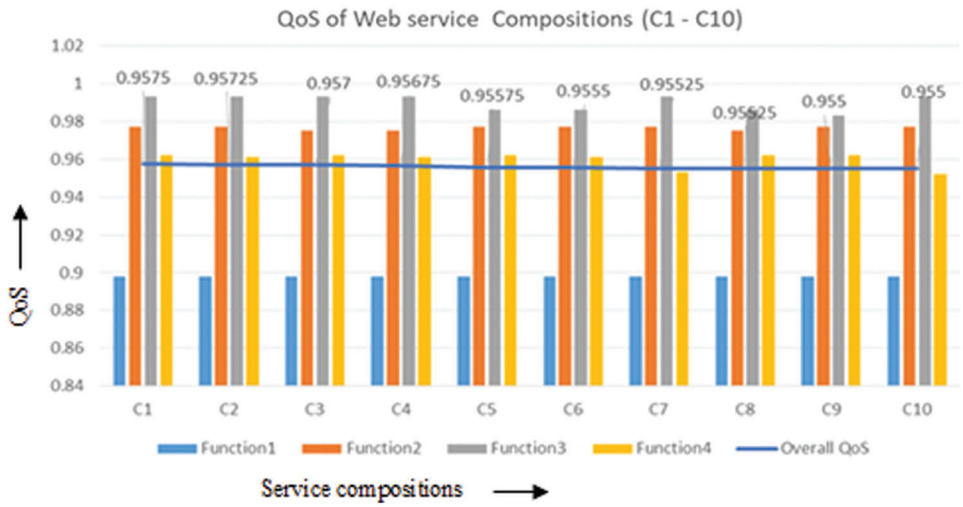

Figure 2: Functional performance and QoS of web service compositions (C1–C10)

The following Tab.6 compares the quality of compositions by both proposed and general methods. Both methods result in almost similar QoS values, but the proposed one has a slight advantage over the other. The general methods will not result in constant QoS for all compositions. But in terms of functional performance, the proposed services are selected based on non-functional parameters that influence the functionality of the task. So, these compositions give the best customer satisfaction compared to the general method.

Table 6: Comparison of QoS in proposed and general method

Compositions	QoS-proposed method	QoS-general method
C_1	0.9575	0.957
C_2	0.95725	0.94925
C_3	0.957	0.95525
C_4	0.95675	0.949
C_5	0.95575	0.9475
C_6	0.9555	0.95575
C_7	0.95525	0.95225
C_8	0.95525	0.967
C_9	0.955	0.95475
C_{10}	0.955	0.95675

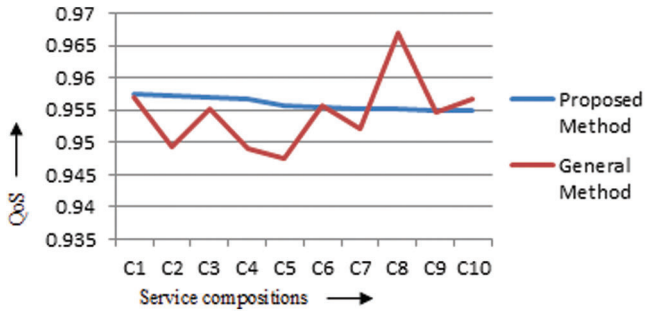

Figure 3: QoS comparison of proposed and general methods

These results are also graphically shown in Fig. 3. The proposed methods give comparatively constant QoS in each composition. Also, the QoS values are up to the mark to reach the actual functionality of the specific task.

5 Conclusion

The composition of web service is an important technique to satisfy the customer's complex web tasks. Choosing a suitable service for composition is vital to achieving the real functionality of the web task. A low-performance service may collapse the entire workflow of the composition. To reach the actual functionality of the web service, apart from considering the QoS, the non-functional parameters that are most needed for the functionality also used in this research. Based on the request from the customer, the broker considers the QoS and non-functional parametric values for the desired functionality to select the web service. These services are selected from the pool of similar services and ranked based on the proposed criteria. This method supports the dynamic composition of web services and helps to achieve the best result during real-time usage. The performance measures of non-functional parameters of the web services are recorded in real-time and the web service that satisfies the expectations will be included in the group so that it will be added to the composition in the future. In a cloud-based setup, the composition of services is more common and available monitoring and management tools from cloud providers can enhance the quality of web service composition.

Acknowledgement: The authors M. Sha and A. Alameen are thankful to the Deanship of Scientific Research, Prince Sattam Bin Abdulaziz University, KSA for supporting this work.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] X. Wu, C. Chen and H. Huang, "A Survey on web service composition: From service description, automatic process generation to process evaluation," *International Journal of Digital Content Technology and its Applications*, vol. 6, no. 17, pp. 273–281, 2012.

[2] S. Chattopadhyay and A. Banerjee, "QoS-aware automatic web service composition with multiple objectives," *ACM Transaction on the Web*, vol. 14, no. 3, pp. 1–38, 2020.

[3] F. Seghir, A. Khababa and F. Semchedine, "An interval-based multi-objective artificial bee colony algorithm for solving the web service composition under uncertain QoS," *Journal of Supercomputing*, vol. 75, no. 1, pp. 5622–5666, 2019.

[4] R. W. Feenstra, M. Janssen and R. W. Wagenaar, "Evaluating web service composition methods: The need for including multi-actor elements," *Electronic Journal of e-Government*, vol. 5, no. 2, pp. 153–164, 2007.

[5] A. L. Lemos, F. Daniel and B. Benatallah, "Web service composition: A survey of techniques and tools," *ACM Computing Surveys*, vol. 48, no. 3, pp. 33–49, 2015.

[6] Q. Z. Sheng, X. Qiao, A. V. Vasilakos, C. Szabo, S. Bourne *et al.,* "Web services composition: A decade's overview," *Information Sciences*, vol. 14, no. 280, pp. 218–238, 2014.

[7] D. A. D'Mello and V. S. Ananthanarayana, "Dynamic web service composition based on operation flow semantics," *International Journal of Computer Applications*, vol. 1, no. 26, pp. 975–987, 2010.

[8] F. Atampore, J. Dingel and K. Rudie, "Supervisor aware service composition framework: An implementation and evaluation," in *Proc. WODES*, Sorrento Coast, Italy, pp. 277–284, 2018.

[9] W. Sun, X. Zhang, Y. Yuan and T. Han, "Context-aware web service composition framework based on agent," in *Proc. ICITA*, Chengdu, China, pp. 30–34, 2013.

[10] S. Chhun, K. Malang, C. Cherifi, N. Moalla and Y. Ouzrout, "A web service composition framework based on centrality and community structure," in *Proc. SITIS*, Bangkok, Thailand, pp. 70–78, 2015.

[11] C. F. Lin, R. K. Sheu, Y. S. Chang and S. M. Yuan, "A relaxable service selection algorithm for QoS-based web service composition," *Information and Software Technology*, vol. 53, no. 12, pp. 1370–1381, 2011.

[12] J. M. Ko, C. O. Kim and I. H. Kwon, "Quality of service-oriented web service composition algorithm and planning architecture," *Journal of Systems and Software*, vol. 81, no. 11, pp. 2079–2090, 2008.

[13] D. Lee, J. Kwon, S. Lee, S. Park and B. Hong, "Scalable and efficient web services composition based on a relational database," *Journal of Systems and Software*, vol. 84, no. 12, pp. 2139–2155, 2011.

[14] Y. Lu, Z. Gao and K. Chen, "A dynamic composition algorithm of semantic web service based on QoS," in *Proc. ICFN*, Sanya, Hainan, China, pp. 354–356, 2010.

[15] F. Lécué, E. Silva and L. F. Pires, "A framework for dynamic web services composition," *Emerging Web Services Technology*, vol. 2, no. 1, pp. 59–75, 2014.

[16] P. Bertoli, M. Pistore and P. Traverso, "Automated composition of web services via planning in asynchronous domains," *Artificial Intelligence: An International Journal*, vol. 174, no. 3, pp. 316–326, 2010.

[17] D. Berardi, D. Calvanese, G. DeGiacomo and M. Mecella, "Automatic service composition based on behavioural descriptions," *International Journal of Cooperative Information Systems*, vol. 14, no. 4, pp. 333–376, 2005.

[18] H. Xia, S. Deng and Z. Wu, "Research and design of web services dynamic composition method," *Computer Engineering & Design*, vol. 28, no. 6, pp. 1334–1337, 2007.

[19] L. Tang, X. Huai and S. Li, "A method of dynamic service composition based on context negotiation," *Journal of Computer Research and Development*, vol. 45, no. 11, pp. 1902–1910, 2005.

[20] Z. Chen, W. Zhou and W. Wang, "The research of personalized web service composition based on context," *Application of Electronic Technique*, vol. 5, no. 4, pp. 124–126, 2007.

[21] W. S. Dream-Dataset, *Built for developers—Web service QoS dataset*, 2020. [Online]. Available: https://github.com/wsdream/wsdream-dataset.