Indian Journal of Science and Technology, Vol 8(16), DOI: 10.17485/ijst/2015/v8i16/55122, July 2015
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Improving Response Time of Web Service Composition based on QoS Properties

Marzieh Karimi*, Faramarz Safi Esfahani and Nasim Noorafza

Department of Computer Engineering, Najafabad Branch, Islamic Azad University, Isfahan, Iran;
karimi.marzieh@gmail.com, faramarz.safi@yahoo.com, n_noorafza@yahoo.com

Abstract

Background: Nowadays, web services are one of the most widely used groups of SOA and service computing. The problem of QoS based selecting a web service dynamically and composing a set of web services to conduct a business task has been investigated in this paper. One of the main objectives of this paper is selecting web services based on non-functional properties and QoS score. **Methods:** In this paper is assumed that there are web services with similar functionality for each task and these web services have different non-functional properties and QoS parameters. To select a web service for eachtask SAW method is used, but this method don't apply SAW method on all of web services. It use user requirements for ranking set of candidate web services and finally apply SAW method on set of candidate web services. **Result:** This method will help to select web services based on QoS score and user requirements. Select web services among many number of web service with this method can use to composition web service and finally will help to optimize response time of web service composition. **Application:** .NET Application with Visual Studio Environment and C# Programming Language.

Keywords: Composition, Quality of Service, Selection, Service Oriented, Web Service

1. Introduction

Web services are modular, comprehensive and reusable software components that based on open XML-based standards to support business-to-business interactions over distributed environments[1].

Web service selection plays an essential role is SOA systems. Web services are considered as self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. QoS-based service selection causes can identify the best component services that satisfy user requirements[2].

One of the most important challenges in Service-oriented Architecture is selecting appropriate services dynamically such that finally have the best composition from services according to business process, policies and non-functional constraints.

Web Service selection is a key component in service-oriented computing. The QoS based web service selection mechanisms plays an essential role in service-oriented architectures, because most of the applications want to use services that accurately meet their requirements[3]. In this paper an algorithm for web service selection is introduced.

Web service selection and discovery system is essential to provide clients with proper results according to their requirements. It is impossible to fulfill this task without considering the ranking relation between thousands of available candidates with similar functionalities. Ranking process is a fundamental step in a Web service selection system, as it integrates the results gathered from previous stages (functional and non-functional matching process) and presents them to the requestors. In this work is focused on the ranking process by considering user's QOS requirements.

* Author for correspondence

2. QOS Properties

The QoS based web service selection mechanisms plays an essential role in service-oriented architectures, because most of the applications want to use services that accurately meet their requirements[3–5]. In this section a number of criteria is presented that should be considered when comparing service selection mechanisms because in this paper is concentrated on approaches that consider non-functional properties.

2.1 Response Time

The maximum time that elapses from the moment that a web service receives a SOAP request until it produces the corresponding SOAP response. It is calculated as

$RT = T1\text{-}T2$

Where T1 = Time at which web service produces soap response.

T2 = Time at which web service receives soap request.

2.2 Execution Time

The execution time measures the expected delay between the moment when a request is sent and the moment when the result are received. It is denoted by

$$q_{du}(s, op) = T_{process}(s, op) + T_{trans}(s, op)$$

2.3 Throughput

The number of Web service requests *R* for an operation *o* that can be processed by a service S within a given period of time.

$$tp(s,op) = \frac{\#R}{timeperiod}(in\ second)$$

Where,

S service within a given period of time.

#R number of web service request.

2.4 Scalability

A Web service that is scalable has the ability to not get overloaded by a massive number of parallel requests. It is calculated as,

$$SC = \frac{t_{rt}}{t_{rt}(throughput)}$$

Where,

$t_{rt}(throughput)$ is the round trip time which is evaluated during the throughput test.

2.5 Reputation

The reputation of a service is the measure of its trustworthiness. It mainly depends on the end users experience of the service. Different end users may have different opinions on the same service. The value of the reputation is defined as the average ranking given to the service by the end users.It is calculated as,

$$q_{rep} = \frac{\sum_{i=1}^{n} R_i}{n}$$

2.6 Availability

Availability of the web service is the probability of the service is accessible. It is calculated using the following expression,

$$q_{av}(s) = \frac{T_i(s)}{n}$$

2.7 Accessibility

It is the capability of serving the Web Service request. The Web service might be available but not accessible because of a high volume of requests. Accessibility can be represented by the following formula:

$$P_{accessability} = \left(1 - \left(\frac{downtime}{uptime}\right)\right)$$

Time is measured in minutes.

So far, there are many different approaches and frameworks that have been developed in order to web service selection. In this paper, some of approaches are discussed that is QoS-based web service selection. The aim of this selection is to give a summary of these approaches. This is achieved by creating a table, listing all of frameworks that have been discussed in this paper and listing some of the most important features that were identified to categories the selection frameworks (Table 1).

In[6] authors present an overall service selection and ranking framework which firstly classify candidate

web services to different QoS levels respect to user's QoS requirements and preferences with an Associative Classification algorithm and then rank the most qualified candidate services based on their functional quality through semantic matching. This algorithm consists of two phases. In the first phase, authors use a classification data mining algorithm to classify web service candidates into different QoS levels respect to the defined QoS constraints form the user and using the result of this classification to define a utility value for each of the service candidates. In the second phase they focus on composing the best services of each task and more specifically on their functional level that aims to selecting and interconnecting web services by means of their semantic connections.

In[7–9] authors propose an architecture that makes an automatic selection of best service provider that is based on mixed context and QoS ontology for a given set of parameters of QoS. The key idea is to rely on multi-dimensional QoS. In this paper goal is to find the best provider of e-service that responds to a request for service. To achieve that, the following steps are required:

- Submit the query with terms and values of quality without and within their context.
- Compare the qualities of provider services with the qualities of request.
- Select the best provider service.

In the last step, to select the best provider, they first compute the matching degree of published qualities and required qualities for each service without using the context of quality. Second, they make use of the context of quality and compare the two cases[10].

In[11,12] authors propose an integrated service query framework that facilitates users in accessing their desired services. The framework incorporates a service query model and a two-phase optimization strategy. The query model defines service communities that are used to organize the large and heterogeneous service space. The service communities allow users to use declarative queries to retrieve their desired services without worrying about the underlying technical details. The two-phase optimization strategy automatically generates feasible service execution plans and selects the plan with the best user-desired quality. In particular, they present an evolutionary algorithm that is able to "co-evolve" multiple feasible execution plans simultaneously and allows them to compete with each other to generate the best plan.

Proposed work in[13,14] is a Web Service (WS) discovery model in which the functional and nonfunctional requirements are taken into account during service discovery. The proposed infrastructure includes a set of services and tools to support an integrated WS discovery and selection solution. A mechanism is introduced that supports three different functional policies. It is able to take advantage of quality information located into a Web Service description language description file that might be located in a proprietary universal description, discovery and integration registry server or in an independent URL. Moreover, it implements a database supported WS intermediary (Broker) that it is also possible to store Quality of Service (QoS) information for WSs. A selection module is also presented that delivers the WS that maximizes the value of QoS characteristics among others with the same functionality.

The seven QoS parameters considered in[3,15,16] are execution time, response time, throughput, scalability, reputation, accessibility and availability. The objective of the framework is to provide QoS based Semantic Web Service Selection. The framework consists of four major components namely OWL-S converter, Semantic Repository, QoS Broker and Matchmaker. OWL-S converter converts the syntactically described web service into a semantic web service. The Semantic repository contains the advertisements of the web services in OWL-S format. QoS broker assigns and stores the rank of all the accessed services based on the Web service Relevancy factor. Matchmaker identifies a set of services that satisfies the client's functional and non-functional requirements.

In[17,18] authors propose a method for automatic selection of the most relevant service for composition based on non-functional properties and the user's context. In doing this they also propose a method of obtaining and evaluating non-functional aspects.

The complexity of business processes and the dynamic nature of the co-operations make it difficult for the business modeler to select appropriate services, manage the compositions efficiently and understand requirements within a dynamic context correctly. In this paper they present the service management layer developed as part of the in Context project which is aimed at addressing the above issue, in particular considering that a service's suitability depends largely on the user's context. They will focus on a specific aspect of this management layer: namely the service lookup and relevance ranking. What

is special about this lookup is that in addition to the functional aspects of a service non-functional aspects are considered both when looking up a service as well as when finding the most suitable service[18].

The in Context platform provides means of integrating services to support collaborative teams. In that sense it is a quite a complex structure and not all of it is relevant for this paper.

In[19,20] authors propose a QoS broker based architecture for dynamic web service selection which facilitates the clients to specify the non-functional requirements like QoS along with functional requirements. The paper presents an efficient mechanism for finding the most suitable web service according to the consumer's requirements. The architecture consists of the basic web service model components like the web service provider, web service consumer and the UDDI registry. In addition, UDDI registry has the capability to store QoS information using tModel data structure and a WS QoS Broker component. The WS-QoS Broker assists clients in selecting web services based on a set of QoS parameters. The WS-QoS Broker has four components: Service Publisher[23], Verifier and Certifier, Service Selector[23] and Web Service Storage (WSS)[24]. Broker services may be used to facilitate service registry access. The broker performs the interaction with the UDDI.

QoS-based service selection aims at finding the best component services that satisfy the end-to-end quality requirements. In[22,25] problem is modeled as a multi-dimension multi-choice 0-1 knapsack problem, which is known as NP-hard. Recently published solutions propose using linear programming techniques to solve the problem. However, the poor scalability of linear program solving methods restricts their applicability to small-size problems and renders them inappropriate for dynamic applications with run-time requirements. In this paper, they address this problem and propose a scalable QoS computation approach based on a heuristic algorithm, which decomposes the optimization problem into small sub-problems that can be solved more efficiently than the original problem.

Table 1. Web Services Selection models

Framework	Selection Strategy	Execution Selection	QoS Modeling
Zeng L, et al.[21]	Using classification data mining	Semantic selection	Yes
Dai Y, et al.7	Mixed Context and Quality of ServiceOntology	Automatic selection	Yes
Chatel P, et al.[11]	Using service query and evolutionary algorithm	Automatic selection	Yes
Ying Y, et al.[13]	Using functional and non-functional requirements	Static selection	Yes
Anonymous[3]	QoS based Semantic Web Service Selection	Semantic selection	Yes
Ardagna D, et al.[17]	Using user-context and non-functional requirements	Automatic selection	Yes
Salehie M[19]	Using UDDI and QoS information	Dynamic selection	Yes
Ardagna D[22]	Selection using heuristic algorithm	Dynamic selection	Yes

3. Proposed Algorithm

Suppose there are n services that have similar functional properties and k required QoS attributes determined by user. Based on the proposed method, k ranked lists will be generated according to each attribute. To involve user requirements in the algorithm, query attributes is considered as a sample service S_q and add it to the list of offered services. In this phase, all of the services that is located after the user's requirement will be deleted. Thus the remaining services fulfill user request. Now among these services, a service with the higher score will be selected.

In order to evaluate two attributes fairly, it is necessary to consider their direction or tendency of their values. In other word, if the tendency of the attribute is positive, it means a bigger value is better. On the contrary if the tendency is referred as negative, it means smaller values are preferred. For example for attribute "cost" the smaller value is usually preferred, so the tendency of this parameter is negative, whereas for attribute "availability"

the bigger value indicates a better quality for the specified parameter, so the tendency is positive.

3.1 QoS Normalization

For negative criteria, values are scaled according to (Equation 2). For positive criteria, values are scaled according to (Equation 1)[7].

$$\frac{Q_{i,j} - Q_i^{min}}{Q_i^{max} - Q_j^{min}} \quad (1)$$

$$\frac{Q_i^{max} - Q_{i,j}}{Q_i^{max} - Q_j^{min}} \quad (2)$$

In this paper the values of n QoS attributes of a service S as a vector: $Q_s = (Q_{s1}, Q_{s2}, ..., Q_{sn})$ are modeled and the values of QoS requirements requested by a consumer as a vector $Q_r = (Q_{r1}, Q_{r2}, ..., Q_{rn})$ are considered. The consumer's preferences values are set on each QoS attribute in a vector $p_r = (p_{r1}, p_{r2}, ..., p_{rn})$ where $p_{ri} \in [1, n]$. If the consumer has no preferences for an attribute, n will be considered as the preference value for that specific parameter. We represent the vector of weights assigned to attributes as: $W = (w_1, w_2, ..., w_n)$

Where $\sum_{i=1}^{5} W_j = 1$, $w_i \in (0,1)$. We set p_{rmax} as the maximum value in vector p_r and use the following equations to compute the weight for each attribute:

$$w_i = \frac{p'_{ri}}{\sum_{i=1}^{n} p'_{ri}} \qquad i = 1,2,\ldots,n$$

Where $p'_{ri} = (p_{rmax} + 1) - p_{ri}$

$$Score_{si} = \sqrt{\sum w_i q'^{2}_{i}}$$

Where $w_l \in [0,1]$ and $\sum_{j=1}^{5} w_j = 1$

4. Evaluation

In all of the following experiments different subsets derived from the QWS dataset provided by Al-Masri, and Mahmoud is used[26]. The original dataset includes information of over 2000 web services available on the Web. The dataset includes real data for various QoS attributes such as response time, availability, throughput, successability, reliability, compliance, best practices, latency and documentation. The service name and its WSDL address are also included in the dataset.

To study the proposed performance is used the QWS Dataset[38], the 2 set case is created that impacting the performance.

- Altering number of attributes (increasing the number of QoS attributes).
- Altering the number of candidate web services. Each set of test cases is solved with proposed algorithm and the LP algorithm[21]. The response time and the value of the objective function of proposed algorithm to LP algorithm are compared.

4.1 Experiment 1: Different Datasets with Different Sizes

In the following scenario the effect of increasing the size of dataset on the performance of each model is studied. To fulfill the task, the applications on different datasets containing 10, 50, 100, 150, 200, 300, 500, 1000, 1500 and 2000 Web services are run. In this experiment three QoS numeric attributes including: response time, availability, and reliability is considered (Table 2).

We measured the execution time of algorithms by running each application 500 times and get the average value of the results. A sample query could be: response time <1000ms, availability >95%, reliability >70%. Then the query vector is set as: (1000, 95, 70). A sample preference vector for this query could be: (1, 3 and 2. Figure 1 shows comparison of computation time between proposed algorithm and SAW algorithm (Figure 1).

Table 2. Results of varying the number of candidate Web Services

Average Execution Time of SAW (ms)	Average Execution Time of Proposed Algorithm (ms)	Candidate Web Services
6	2.5	10
22	3.7	50
30	6.7	100
43	7.1	150
55	7.4	200
78	15.1	300
109	19.5	500
200	40.7	1000
280	60.5	1500
340	90	2000

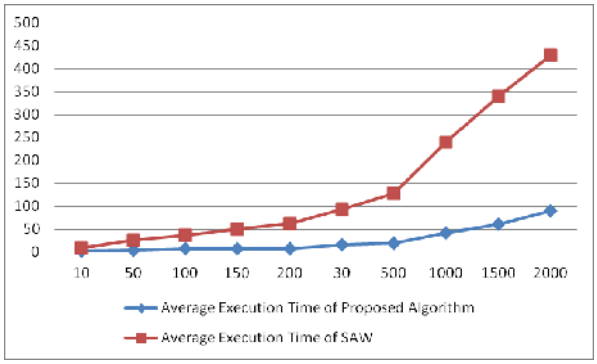

Figure 1. Comparison of computation time.

The selection web services are displayed in following Table (Table 3 and 4).

Table 3. Selection Web Service of varying the number of candidate Web Services

Reliability	Availability	Response Time	Web Service Name	Candidate Web Services
73	87	107	CasUsers	10
#	#	#	#	50
83	96	123	UniquesubsService	100
83	96	123	UniquesubsService	150
83	96	123	UniquesubsService	200
83	96	123	UniquesubsService	300
89	98	190	eUtilsService	500
89	96	149	eUtilsService(2)	1000
89	96	149	eUtilsService(2)	1500
89	96	149	eUtilsService(2)	2000

Table 4. Selection Web Service of 50 number of candidate Web Services

Reliability	Availability	Response Time	Web Service Name	Candidate Web Service with 50 Web Service
83	83	408	VersionService	SAW Algorithm
73	94	124.17	XFormWebService	Proposed Algorithm

4.2 Experiment 2: Varying Number of QoS

To study the impact of increasing the number of QoS attributes on execution time of different algorithms, a different set of experiments with a combination of number of Web services and different number of QoS attributes is did. The two algorithms on datasets with sizes containing 2000 candidates are run. In this section, the performance results on each dataset in a separate table are presented (Tables 5, 6, 7, 8). The first row of each table shows the result when there are three numeric attribute, i.e. response time, availability, and reliability. The user's preference vector is set as (1, 3, 2) and the query is submitted as (response time <= 1000 ms, availability >= 95%, reliability >= 70. The second line shows the results with 6 QoS attributes: response time, availability, reliability, throughput, successability and compliance. In this case the query was submitted as (response time <= 1000, availability >= 95%, throughput >= 20, sucessability >= 95%, reliability >= 70%, compliance >= 85%) and consumer's preference vector is set as: (1, 1, 3, 3, 2, 3). The last row shows the execution time of each algorithm based on 9 QoS attributes: response time, availability, reliability, throughput, successability, compliance, best practice, latency and documentation. The query and preference vector in this case are (response time <= 1000, availability >= 95%, throughput >= 20, sucessability >= 95%, reliability >= 70%, compliance >= 80%, best practice >= 50%, latency <= 50ms, Documentation >= 50) and consumer's preference vector is set as (1, 1, 3, 3, 2, 3, 2, 2, 2) respectively. The average execution time was computed over 500 runs. (Table 4) represents the performance of the algorithms on a dataset including 2000 Web services. In the following tables the comparison of computation time and selection web service with varying the number of Quality of Service is displayed.

Table 5. Results of varying the number of Quality of Services

Average Execution Time of SAW Algorithm (ms)	Average Execution Time of Proposed Algorithm (ms)	QoS Properties
340	90	3
358	24	6
375	11.7	9

Table 6. Selection Web Service with 3 number of Quality of Services

Reliability	Availability	Response Time	Web Service Name	Algorithm	Number of QoS
89	96	149	eUtilsService(2)	SAW	3
89	96	149	eUtilsService(2)	Proposed	3

Table 7. Selection Web Service with 6 number of Quality of Services

Compliance	Reliability	Throughput	Successability	Availability	Response Time	Web Service Name	Algorithm	Number of QoS
100	80	36.3	100	99	80	Service1	Proposed	6
78	80	13.6	96	89	165	HelloService	SAW	6

Table 8. Selection Web Service with 9 number of Quality of Services

Number of QoS	Algorithm	Web Service Name	Response Time	Availability	Reliability	Throughput	Successability	Compliance	Best Practice	Latency	Documentation
9	Proposed	getJoke	207	99	80	29	100	100	87	2	95
9	SAW	getJoke	207	99	80	29	100	100	87	2	95

5. Conclusion

An approach is proposed to solve the QOS-aware Web Service selection problem. For this, an algorithm is presented based on QoS properties which reveal that this selection is extremely fast and leads to results that are very close to the optimal solution.

6. References

1. W3C Working Group: Web Services Architecture. 2012 May. Available from: http://www.w3.org/TR/ws-arch/
2. Dustdar S, Schreiner W. A survey on web services composition. International Journal of Web and Grid Services. 2005; 1:1–30.
3. WS-Diamond Team. WS-DIAMOND: Web Services - DiAgnosability, MONitoring and Diagnosis. MIT press; 2009. p. 213–39.
4. Rajendran T, Balasubramanie P, Cherian R, An Efficient WS-QOS Broker Based Architecture for Web Services Selection: ICJA Journal, 2010. Available from: http://www.ijcaonline.org/archives/number9/194-333, http://journaldatabase.info/articles/efficient_ws-qos_broker_based.html, http://www.researchgate.net/publication/265033497_QoS_Based_Efficient_Web_Service_Selection
5. Vadivelou G, IIavarasan E, Manoharan R, Praveen P. A QoS based web service selection through delegation. International Journal of Scientific and Engineering Research. May 2011; 2(5).
6. Makhlughian M, Hashemi M, Rastegari Y, Pejman E. Web Service Selection based on ranking of qos using associative classification; 2012.
7. Dai Y, et al. QOS-Driven self-healing web service composition based on performance prediction. Journal of Computer Science and Technology. 2009 Mar; 24:250–61.
8. Keskes N. Context of qos In Web Service Selection. American Journal of Engineering Research (AJER); 2013.
9. Keskes N, Lehireche A. Web services selection based on mixed context and quality of service ontology. Computer and Information Science. 2011 May; 4(3).
10. Wenjuan L, et al. A framework to improve adaptability in web service composition. 2nd International Conference on Computer Engineering and Technology (ICCET); Chengdu. 2010.
11. Chatel P, et al. QOS-based late-binding of service invocations in adaptive business processes. Proceedings of the 2010 IEEE International Conference on Web Services; 2010. p. 227–34.
12. Yu Q, Rege M, Bouguettaya A, Medjahed B, Ouzzani M. A two-phase framework for quality-aware Web service selection. Service Oriented Computing and Applications. 2010 Jun; 4(2):63–79. Available from: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.225.1648
13. Ying Y, et al. A Self-healing composite Web service model. Proceedings of the IEEE Asia-Pacific Services Computing Conference (APSCC); 2009. p. 307–12.
14. Diamadopoulou V, Makris CH, Panagis Y. Internet and Multimedia Technologies Research Unit. Available from: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.75.617
15. Maheswari S, Karpagam GR. QoS based efficient web service selection. European Journal of Scientific Research. 2011; 66(3):428–40. ISSN: 1450-216X. Available from: http://www.researchgate.net/publication/265033497_QoS_Based_Efficient_Web_Service_Selection
16. Achkoski J, Trakovik V. Metrics for service availability and service reliability in service-oriented intelligence information system. IEEE Conference on ICT Innovations; 2012. Available from: http://www.researchgate.net/publication/236886122_Metrics_for_Service_Availability_and_Service_Reliability
17. Ardagna D, et al. PAWS: A Framework for Executing Adaptive Web-Service Processes. IEEE Conference on Software; 2007. p. 39–46.
18. Reiff-Marganiec S , Yu H, Tilly M. Service selection based on non-functional properties. International Conference on

Service-Oriented Computing (ICSOC); 2009. p. 128–38. Available from: https://lra.le.ac.uk/bitstream/2381/4715/3/NFPSLA09.pdf.txt
19. Salehie M, Tahvildari L. Self-adaptive software: Landscape and research challenges. ACM Transactions on Autonomous and Adaptive Systems. 2009; 4:1–42.
20. Rajendran T, Balasubramanie P. An efficient architecture for agent-based dynamic web service discovery with QoS. Journal of Theoretical and Applied Information Technology; Islamabad Pakistan. 2010 May; 15(2).
21. Zeng L, et al. QOS-Aware Middleware for Web Services Composition: IEEE Trans Softw Eng. 2004; 30:311–27.
22. Ardagna D, Mirandola R. Per-flow optimal service selection for Web services based processes. Journal of Systems and Software. 2010; 83:1512–23.
23. Ardagna D, et al. A service-based framework for flexible business processes. IEEE Conference on Software; 2011. p. 61–7.
24. Bianculli D, et al. Automated dynamic maintenance of composite services based on service reputation. Proceedings of the 5th International Conference on Service-Oriented Computing (ICSOC'07); Vienna, Austria. 2007. p. 449–55.
25. Alrifai M, Risse T, Dolog P, Nejdl W. A scalable approach for QoS-based web service selection. International Conference on Service-Oriented Computing (ICSOC); 2008. Available from: http://link.springer.com/chapter/10.1007/978-3-642-01247-1_20, http://www.sciencedirect.com/science/journal/15708268/1
26. Al-Masri E, Mahmoud QH. Investigating web services on the world wide web. 2008.
27. W3C Working Group: QOS for Web Services: Requirements and Possible Approaches. 2012 May. Available from: http://www.w3c.or.kr/kr-office/TR/2003/ws-QOS/
28. Hwang SY, et al. A probabilistic approach to modeling and estimating the QOS of web-services-based workflows. Information Sciences. 2007; 177:5484–550.
29. Cardoso J, et al. Quality of service for workflows and Web service processes: Web Semantics Science, Services and Agents on the World Wide Web. 2004; 1:281–308.
30. Cardellini V, et al. MOSES: A framework for QOS driven runtime adaptation of service-oriented systems. IEEE Transactions on Software Engineering. 2011.
31. Erradi A, Maheshwari P. Dynamic binding framework for adaptive web services. Proceedings of the 2008 Third International Conference on Internet and Web Applications and Services; 2008. p. 162–7. Available from: http://www.scirp.org/journal/PaperInformation.aspx?paperID=22407
32. Canfora G, et al. A framework for QOS-aware binding and re-binding of composite web services. The Journal of Systems and Software. 2008; 81:1754–69.
33. Calinescu R, et al. Dynamic QOS Management and Optimization in Service-based Systems: IEEE Trans Softw Eng. 2011; 37:387–409.
34. Cardoso J, Sheth A, Miller J, Arnold J, Kochut K. Quality of service for workflows and web service processes. Web Semantics: Science, Services and Agents on the World Wide Web. 2004; 1(3):281–308.
35. Aggarwal R, Verma K, Miller J, Milnor W. Constraint driven web service composition in METEOR-S. IEEE International Conference on Services Computing (SCC'2004); Shanghai, China. 2004.
36. Cardoso J. Quality of Service and Semantic Composition of Workflows [PhD Dissertation]. Athens, GA, USA: Department of Computer Science, University of Georgia; 2002.
37. Huang AC, Steenkiste P. Building self-configuring services using service-specific knowledge. Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing; 2004. p. 45–54.