Research Journal of Applied Sciences, Engineering and Technology 4(8): 987-994, 2012
ISSN: 2040-7467

Submitted: January 10, 2012 Accepted: January 29, 2012 Published: April 15, 2012

Development of an Efficient QoS based Web Services Compositions Mechanism for Semantic Web

R. Bala Krishnan and N.K. Sakthivel
School of Computing, SASTRA University, Thanjavur-613 401, Tamil Nadu, India

Abstract: Web Services are mounting as an inventive mechanism for rendering services to subjective devices over the WWW. As a consequence of the rapid growth of Web Services applications and the plenty of Service Providers, the consumer is facing with the inevitability of selecting the "right" Service Provider. In such a scenario the Quality of Service (QoS) serves as a target to differentiate Service Providers. To select the best Web Services / Service Providers, Ranking and Optimization of Web Service Compositions are challenging areas of research with significant implications for the realization of the "Web of Services" revelation. The "Semantic Web Services" use formal semantic descriptions of Web Service functionality and interface to enable automated reasoning over Web Service Compositions. This study from its experimental results revealed that the existing Semantic Web Services faces a few challenging issues such as poor prediction of best Web Services and optimized Service Providers, which leads to QoS degradation of Semantic Web. To address and overcome these identified issues, this research work is calculating the semantic similarities, utilization of various Web Services and Service Providers. After measuring these parameters, all the Web Services are ranked based on their Utilization. Finally, our proposed technique, selected best Web Services based on their ranking and placed in Web Services Composition. From the experimental results, it is established that our proposed mechanism improves the performance of Semantic Web in terms of Execution Time, Processor Utilization and Memory Management.

Key words: Quality of services/composition, semantic web, service composition, service selection, web services

INTRODUCTION

Web Services are software components designed to provide support interoperable machine-to-machine interaction over a network (Zhang, 2007): With the number increasing of Web Services, Quality-of-Service (QoS) is usually employed for describing non functional characteristics of Web Services (Zeng, 2004). Among different QoS properties of Web Services, some properties are user independent and have identical values for different users (e.g., price, popularity, availability, etc.). The values of the user independent QoS properties are usually offered by Service Providers or by third-party registries (e.g., UDDI). On the other hand, some QoS properties are user dependent and have different values for different users (e.g., response time, invocation failure rate, etc.). Client-Side Web service evaluation requires real-world Web service invocations and encounters the following drawbacks. First, real-world Web service invocations impose costs for the service users and consume resources of the service providers. Some Web Service invocations may even be charged. Second, there may exist too many Web service candidates to be evaluated and some suitable Web Services may not be discovered and included in the evaluation list by the service users. Finally, most service users are not experts on Web service evaluation and the common time-to-market constraints limit an in-depth evaluation of the target Web Services.

However, without sufficient client-side evaluation, accurate values of the user-dependent QoS properties cannot be obtained. Optimal Web service selection and recommendation are thus difficult to achieve. To attack this critical challenge, we propose a collaborative filtering based approach for making personalized QoS value prediction for the service users. Collaborative filtering (Herlocker, 1999): Is the method which automatically predicts values of the current user by collecting information from other similar users or items. Well-known collaborative filtering methods include user-based approaches and item-based approaches.

In this proposed approach, we systematically combine the user-based approach and item-based approach for predicting the QoS values for the current user by employing historical Web Service QoS data from other similar users and similar Web Services. Our approach predicts user dependent QoS values of the target Web Services without requiring real-world Web service

Corresponding Author: R. Bala Krishnan, School of Computing, SASTRA University, Thanjavur-613 401, Tamil Nadu, India

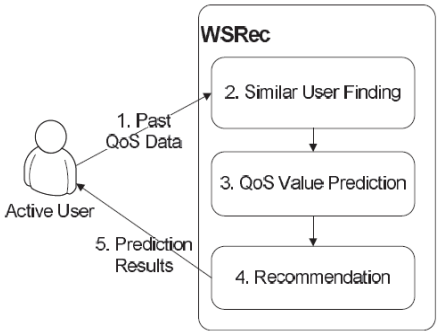

Fig. 1: Procedures of QoS value prediction

invocations. The Web Service QoS values obtained by our approach can be employed by other QoS driven approaches such as Web Service Selection and Fault Tolerant Web Service (Zibin Zheng, 2011).

Bundled QoS for web service users: To provide accurate QoS value prediction of Web Services without real-world Web service information, we need to collect and analyse the past Web service QoS information from other service users. However, it is difficult to collect Web service QoS information from different service users due to:

- Web Services are launched over the Internet and are launched by various web service granting organizations or service providers
- Services are mostly independent from each other
- The existing Web service architecture does not provide any mechanism for the Web service QoS information sharing (Zibin Zheng, 2011)

The procedures of QoS Value prediction which is stated in the given Fig. 1, which shows the procedures of our users collaborative QoS data collection mechanism, which are given below.

- The Service Broker contributes past Web Services' QoS data to a centralized server WSRec (Web Service Recommender System) (Zheng *et al*., 2009).
- WSRec (Web Service Recommender) selects similar users from the training users for the active user. Training users represent the service users whose QoS values are stored in the WSRec Server.
- WSRec predicts QoS values of Web Services for the active user.
- WSRec makes Web service recommendation based on the predicted QoS values of different Web Services.
- The service user receives the predicted QoS values as well as the recommendation results, which can be employed to assist decision making.

The Semantic Web, where the semantics of information is indicated using machine-processable languages such as Web Ontology Language (OWL), which contains many advantages over the World Wide Web (Smith *et al*., 2004). Information about Web Services can be tagged to explain their functionalities in terms of input parameters, outputs, preconditions and its associated effects. The Semantic Web Services can then be automatically discovered and then composed into more complex services and executed. The process of automating the composition of Web Services by using the semantic technologies is currently a focus of a major research work in the area of Service-Oriented-Computing (Lecue and Mehandjiev, 2011): And the process of optimizing compositions is comparatively rare at today. The proposed approach of optimization uses a combination of functional and nonfunctional considerations. The functional consideration describes that how the functionalities of the constituent services fit together. Other considerations holds the degree to which the composition satisfies the overall goal to be achieved along with the constraints to which the pre- and post conditions are satisfied, etc. In order to find the degree of semantic similarity the concept of semantic link is to be used, which defines the semantic connection between the corresponding pairs of the parameters of Web Services. Web Services composition then will be optimized and ranked using functional and the non functional parameters such as well known Quality-of-Service (Lecue and Mehandjiev, 2011).

LITERATURE REVIEW

In this section, we would like to discuss the various existing Web Services Compositions approaches namely Non-Functional Approach and Semantic Approach. From an initial set of available services, we can define Service Composition as follows:

Definition 1 (web service composition): Web Service Composition aims at selecting and interconnecting Web Services provided by different partners in order to achieve a particular goal. Automating Web Service Composition aims to overcome the problem where no single service can satisfy the goal specified by the service consumer. A number of different approaches have been proposed, including Logic-based, Matchmaking-based, Graph-Theory-based, and AI-Planning-based.

Definition 2 (web service composition optimization): Optimization of Web Service Composition aims at selecting appropriate service components to optimize the overall quality of the composition according to a set of predefined metrics. In this section, we appraise the

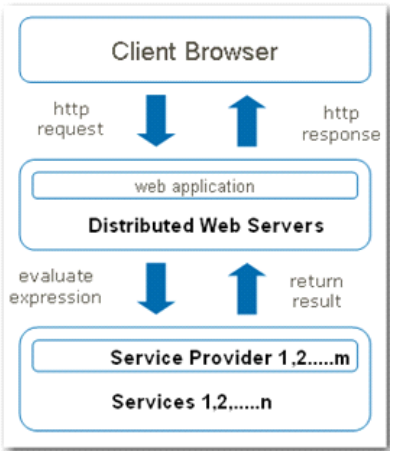

Fig. 2: Existing architecture of distributed web servers

existing approaches to optimizing service composition and classify them according to the following three dimensions:

- First the extent to which an approach considers the non-functional quality of compositions
- Second the extent to which an approach considers the semantic quality of compositions
- Finally the scalability of the approach

Dimension of non-functional approach: The non-functional quality dimension classifies approaches based on their ability to consider non-functional (QoS) properties of compositions. By using semantic type descriptions of services, they only consider optimization in terms of their compatibilities in a composition, and not considering the quality in the granted services. Therefore, such an approach is not able to rank compositions according to some business requirements.

The work of Canfora (Canfora *et al*., 2005): is high since it allows quality of compositions to be evaluated using several non functional criteria such as Response Time, Reliability, Security Levels (Krishnan and Sakthivel, 2012), Execution Price, Availability as well as domain dependent attributes.

Dimension of semantic approach: This approach proceeds according to the ability to optimize the service compositions using semantic quality. By increasing the quality of compositions and their connections, the number of mediators that are generated manually, which are required in case of semantic heterogeneity between the exchanged/shared data in the service composition (Lecue and Mehandjiev, 2011).

Most of the non functional quality-based approaches are ranked very low in this dimension since they only take into account precise semantic matches along output-input connections of Web Services. Others do not address semantic evaluation of compositions. At the same time, approaches with low rank on the non-functional quality dimension score very well here (Alrifai and Risse, 2009).

Dimension of scalability approach: This approach is to support compositions with a large number of services is ranked using the Scalability dimension. In this dimension, the GA-based approaches such as (Canfora *et al*., 2008), and rank higher than IP-based services even if suboptimal solutions are reached in some cases.

Pure GA-based methods improve the scalability by experimenting with different parameters (Canfora *et al*., 2008): such as the policy for evolution, population and so on. This problem can also be modelled as a Knapsack problem, wherein Arpinar et al. (Lecue and Mehandjiev, 2011): Proposed stochastic based search and dynamic programming to solve the problem.

Model for web service architecture: The Service Oriented Architecture (SOA) based Web Service Architecture is used to implement both the Non-Functional Approach and Semantic Approach as well.

This Web Service Architecture could be called as Distributed Web Server Model, which is shown in the Fig. 2. The detailed design methodology of this model is discussed below. The Distributed Web Servers are designed and implemented for web applications, which holds the details of Service Providers and their offered Web Services (Canfora *et al*., 2008).

A Service Provider might have numbers of Web Services and at the same time, a particular Web Service could be offered by different Service Providers. To access these types of Web Services from Service Providers, all the requesters need to satisfy a large set of access control rules or policies and each rules/policies specifying the Source Address, Destination Address, Source Port, Destination Port, and Protocols IDs. Initially the Service Seekers (requesters) requests have to be processed from their browsers and all these requests will reach the Distributed Web Servers, which are act as a middleware between the Service Seekers and the Service Providers.

The Distributed Web Servers receive the requests and examines the authorization of the Services Seekers and if authenticated, these requests will be forwarded to the appropriate Service Providers, which will grant the requested services. Due to the multidimensional nature of the rules (including source/destination addresses and ports), the performance of the Web Server degrades as the number of rules increases. Commercially deployed Servers often carry tens of thousands of rules, which are creating performance bottlenecks of Web Servers. For each request, the Server needs to communicate with the Service Providers to get the Service.

Fig. 3: Initial screen of the proposed SNIFFER server

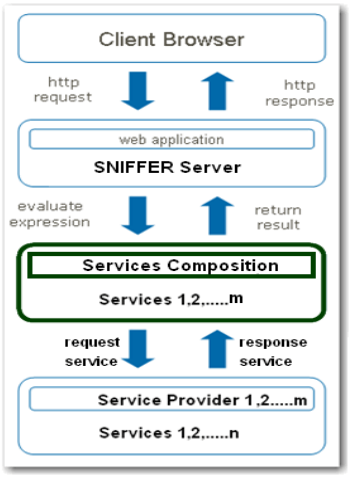

Fig. 4: Proposed architecture of the SNIFFER server

Identified problems and the need for proposed holistic approach: From our literature survey, we have noted that different researchers have been proposed various Web Services approaches including Logic-Based (Rao *et al*., 2006), Matchmaking-Based (Lassila and Dixit, 2004) and GA-QoS-based (Lecue and Mehandjiev, 2011) to improve the performance of Distributed Web Servers. In this section, we have studied Logic-Based, Matchmaking-Based and GA-QoS-based Web Services Composition Approaches thoroughly and observed their drawbacks, which are listed below.

- Inevitability of selecting the "right" service provider because of the Rapid growth of Web Service applications and the great extent of Service Providers
- Poor ranking and optimization of Web Service Compositions
- The process of combining Semantic and Non-Functional criteria.
- Poor prediction accuracy
- Poor authentication

To address these identified problems, we have proposed an efficient QoS based Web Services Composition Mechanism, which addresses the above said issues and it is also achieve higher performance than that of existing Models.

PROPOSED TECHNIQUE

From the previous section, it is observed that various mechanisms have been proposed to improve the performance of the Semantic Web Services. The various identified problems have been listed in the previous section. To overcome these identified problems, this research work has designed an efficient Web Services Composition Architecture, which consists of Secured SNIFFER Server with various users' access policies. The available features of this SNIFFER Server are shown in the Fig. 3.

The proposed Server is named as SNIFFER, which states that the Server is not only offering the features of granting Web Services but also it is analyzing both the users' nature and the Web Services Usages by focusing users authenticity, users request pattern and frequently accessed Services.

It also holds the facility to create Service Providers in which we can load Services and these Service Providers can be created on various locations. We have the facility to remove the lists of the underutilized Web Services and underutilized Service Providers as well. And also this SNIFFER Server Tool has the facility to migrate Web Services from one Service Provider to another. The architecture of the proposed SNIFFER Server is shown in the Fig. 4.

In this Architecture, an efficient technique called Adaptive Services Generator is introduced and implemented through the Adaptive Web Services Selection procedure which is shown in the Fig. 5.

Initially Clients are forwarding various requests for Web Services to the Server application. Clients are having unique identification details and having some Access Control rules which states the policies for each users. i.e. Each user has to satisfy a few policies to access the Services from the Service Providers.

```
Algorithm : Adaptive Web Service Selection

Inputs: Userindex, SerReqName; Indexterms…
Output: fSolbest
begin
     FSOL = { }
          reqServ=FORMATREQUEST(SerReqName;Indexterms…)
if(RIGHTS(Userindex)
   //LookUp on Composition
   if(chkonNWSComposition(reqServ))
     FSOL=GRANT(reqServ)
  else
     for(i=1 to M) //M ➔ Total number of available Service Providers
       for(i=1 to N) //N ➔ Total Number of Services in Each Server
            if(chkonService(reqServ).Available())
                    FSOL=GRANT(reqServ)
            else
                    FSOL=GRANT(Defaultinfo)
            endif
  endif
endif
return Get_Best_Solution(FSOL)
end
```

Fig. 5: Adaptive web services selection specification

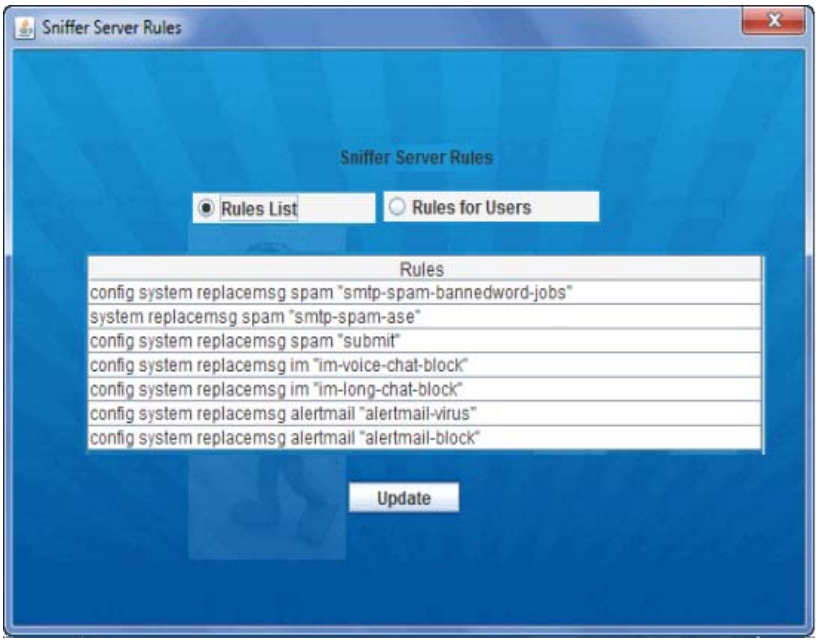

Fig. 6: The rules list of the SNIFFER server

The policies for the users' access are dynamic which can be altered by using the Rules List provision which is shown in the Fig. 7. After receiving the client request for a Web Service, the Server will verify the users privileges and if the requester is an authorized person, then the request will be processed. Otherwise this system will reply to the concern requester as invalid user/invalid request.

If the requester is a valid user, the Server will fetch the requested Service from the Web Services Composition (WSC) bundle, which contains all the best recently used Services with various QoS. If the requested Service doesn't exist in WSC, the Server will search the best Service from the right Service Provider and forward the same to the requester (Service Seeker).

After receiving the client request for a Web Service, the Server will verify the users privileges and if the requester is an authorized person, then the request will be processed. Otherwise this system will reply to the concern requester as invalid user/invalid request. If the

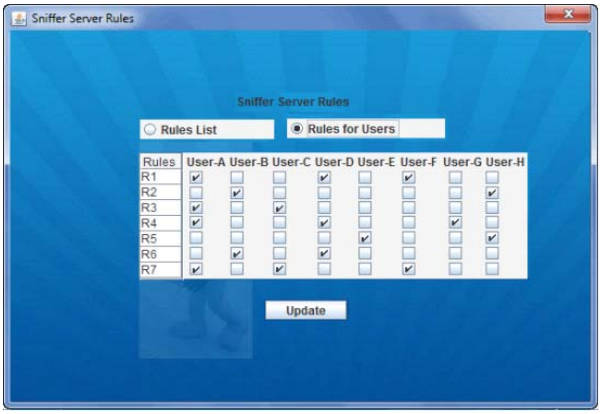

Fig. 7: The rules setter of the SNIFFER server

requester is a valid user, the Server will fetch the requested Service from the Web Services Composition (WSC) bundle, which contains all the best recently used Services with various QoS.

If the requested Service doesn't exist in WSC, the Server will search the best Service from the right Service Provider and forward the same to the requester (Service Seeker). This smart WSC mechanism is improving the Performance of Semantic Web.

The Proposed SNIFFER Server has two special features namely,

- Policy Optimization technique
- Services Composition Optimization technique

Policy optimization technique (minimize the maximizing rule set of SNIFFER server): This Policy Optimization Technique which is used to minimize the maximizing rules set (Krishnan and Sakthivel, 2012): Of SNIFFER Server. It has various rules as shown in the Fig. 6.

The Policy Optimizer is premeditated as a Smart and Heuristic Mechanism, which is used to perform the following activities. They are:

- Early Parallel Rejection Rule (EPRR)
- Parallel Matching of Policies

It is used to view all the available rules and rules which are assigned to users are shown in the Fig. 6 and Fig. 7 respectively. And also various required policies can be assigned to any users dynamically.

As shown in the Fig. 3, this tool has the facility to Block the unauthorized users based on their domain names, Port Blocking and Port Modification, Rules Checking and Rules Reduction. This model for Rules Reduction is working efficiently not only in the situation where all the rules are common for users but also it works well if some of the rules between the users are different.

Order	Src ip	Dst ip	Dst port	Action
1	10.0.0.5	20.0.0.1	20-21	accept
2	10.0.0.5	20.0.0.1	21-22	accept
3	10.0.0.5	20.0.0.1	23-25	deny
4	10.0.0.0/30	20.0.0.1	80	deny
5	10.0.0.0/30	20.0.0.1/30	80	deny
6	10.0.0.0/24	20.0.0.1/24	any	accept
7	any	any	any	deny

Fig. 8: The list of arrived requests from various sources and the action status

It checks and finalizes the common rules applicable for all the users and the common rules will be checked once and all.

Services composition optimizer: This Services Composition Optimizer is intended as an effective mechanism, which regularly monitors the usage of Services and from the gathered information it synchronized with the service handler and it periodically rebuilds the Web Services Composition. This Services Composition Optimizer monitors the usages of each Web Service periodically and also it maintains the Services Usage Level for each Web Service. If this Optimizer identifies that any one of the Web Services falls below the threshold level of usage, this Web Service will be removed from this Composition. This approach improves the performance of Web Server.

PERFORMANCE ANALYSIS

The Proposed Web Services Composition based SNIFFER Server Software is implemented and studied thoroughly and the performance of this proposed system is analyzed with the existing GA-QoS-based System in terms of Execution and Processing Time, CPU and Memory usage and vulnerability level. From the results, our proposed work performs well as compared with the existing approach. Figure 8 shows that our work rejected a few unauthorized users/requests based on our policies.

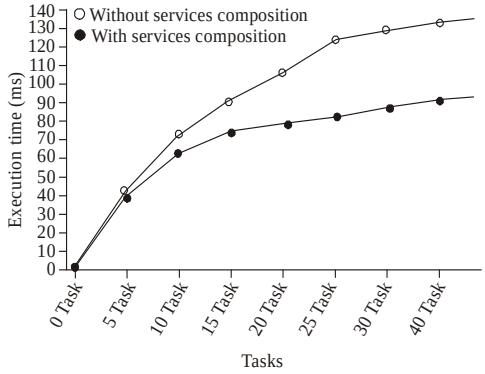

Fig. 9: Evolution of the web services composition

Tasks	Execution time(ms)	
	Without composition	With composition
0	0	0
5	44	40
10	75	65
15	94	77
20	110	81
25	129	85
30	134	90
40	140	95

Fig. 10: Execution time vs tasks

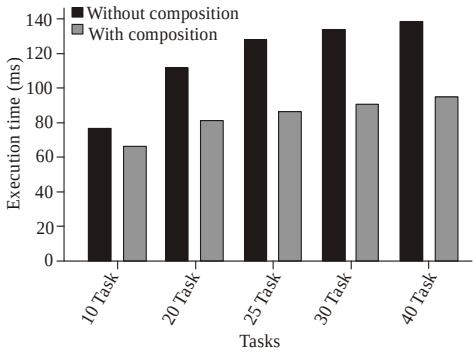

Fig. 11: Evolution of the web services composition execution time for various tasks

EXPERIMENTAL RESULTS

We analyze the performance of our proposed approach by the following parameters.

- Execution Time With and Without Web Services Composition
- Response Time With and Without Web Services Composition and
- CPU Usage With and Without Web Services Composition

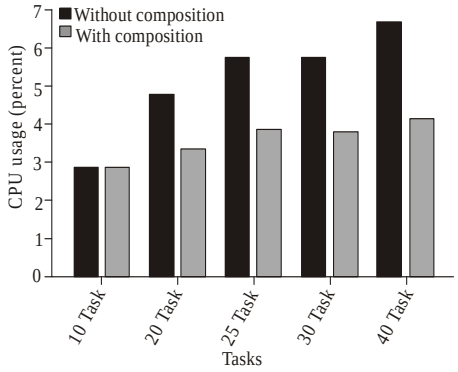

Fig. 12: Comparison of the CPU usage and tasks

From the Fig. 9, it is observed that our proposed work finds the best Web Services for requests with minimum execution time as compared with that of existing work. The execution time to handle more numbers of Tasks are tabulated and listed in the Fig. 10. i.e. The Web Services Composition approach reduces the request handling time which improves the overall System performance.

From our results, we also revealed that our proposed system achieves best performance for both low volume of requests and heavy volume of requests as well. i.e. In our proposed work, the execution time is almost same for different levels of requests. This is because of Web Services Composition Mechanism. Whereas, in the existing system, the execution time is increased as requests increases, which is shown in the Fig. 11.

Through this proposed Web Services Composition approach, the CPU usage of the Web Server is better than that of the Server without having the Web Services Composition mechanism. The Server which is having the Web Services Composition feature provides best performance and hence through this approach the performance of the Semantic Web is improved considerably with required QoS which is demonstrated in the Fig. 12.

From the Fig. 12, it is observed that the CPU usage of this proposed system is considerably less as compared with existing approach and hence this Web Server could handle more number of requests.

CONCLUSION

Our proposed work demonstrates that the existing Semantic Web Services without Web Services Composition (WSC) degrades the performance of Semantic Web. To address this major issue, we have introduced the Web Services Composition mechanism which is the wonderful approach and it provides more benefits for Semantic Web Services. From our experimental results, our work performs better as compared with the existing system in terms of:

- Handling more numbers of requests
- CPU usage
- Execution Time, and
- Vulnerability Level.

REFERENCES

Zibin Zheng, H.M., M.R. Lyu and I. King, 2011. QoS-aware web service recommendation collaborative filtering. IEEE Transact. Serv. Comput., 4(2).

Zhang, L.J., J. Zhang and H. Cai, 2007. Services Computing. Springer and Tsinghua University,

Zeng, L., B. Benatallah, A.H. Ngu, M. Dumas, J. Kalagnanam and H. Chang, 2004. Qos-aware middleware for web services composition. IEEE Trans. Software Eng., 30(5): 311-327.

Herlocker, J.L., J.A. Konstan, A. Borchers and J. Riedl, 1999. An algorithmic framework for performing collaborative filtering. Proceedings of the 22nd International ACM SIGIR Conferance Research and Development in Information Retrieval, (SIGIR '99), pp: 230-237.

Zheng, Z., H. Ma, M.R. Lyu and I. King, 2009. Wsrec: A collaborativefiltering based web service recommender system. Proceeding of the Seventh International Conferance Web Services (ICWS '09), pp: 437-444.

Smith, M.K., C. Welty and D.L. McGuinness, 2004. Owl Web Ontology Language Guide. W3C Recommendation, W3C.

Lecue, F. and N. Mehandjiev, 2011. Seeking quality of web service composition in a semantic dimension. IEEE Transact. Knowledge Data Eng., 23(6).

Canfora, G., M. Di Penta, R. Esposito and M.L. Villani, 2005. An approach for qos-aware service composition based on genetic algorithms. Proc. Genet. Evolut. Computat. Conf., pp: 1069-1075.

Alrifai, M. and T. Risse, 2009. Combining global optimization with local selection for efficient qos-aware service composition, Proceedings of the International Conferance World Wide Web, (ICWWW'2009), pp: 881-890.

Canfora, G., M. Di Penta, R. Esposito and M.L. Villani, 2008. A framework for qos-aware binding and re-binding of composite web services. J. Syst. Software, 81(10): 1754-1769.

Krishnan, R.B. and N.K. Sakthivel, 2012. Development of smart firewall load balancing framework for multiple firewalls with an efficient heuristic firewall rule set. J. Theoret. Appl. Inform. Technol., 32(2).

Rao, J., P. Ku¨ ngas and M. Matskin, 2006. Composition of semantic webservices using linear logic theorem proving. Informat. Syst., 31(4/5): 340-360.

Lassila, O. and S. Dixit, 2004. Interleaving discovery and composition for simple workflows. Proceedings of the Semantic Web Services, (SWS' 2004.), AAAI Spring Symp Series, pp: 22-26.