An approach for an optimized web service selection based on skyline

[1]Mohamed Ali Bouanaka,[2]Naceredine Zarour

[1] LIRE laboratory, Department of Software Technologies and Information Systems, University of Constantine 2 Constantine, Algeria

[2] LIRE laboratory, Department of Software Technologies and Information Systems, University of Constantine 2 Constantine, Algeria

Abstract

Nowadays, considering Web Services has become one of the hot issues in the area of computer science that makes an ability to collect capabilities and components in a unique interface to fulfil the user's requirements. Sometimes two or more services are discovered in available list of services; therefore, there should be a possibility for selecting the best services from discovered list which can satisfy the user's goal. The selected services should optimize the overall QoS of the composed application, while satisfying all the constraints specified by the client on individual QoS parameters. In this paper, we propose an approach based on the notion of skyline to effectively and efficiently select services for composition and reducing the number of candidate services. At the end of the paper we evaluate our approach experimentally.

Key words: *Web Services selection, Skyline, Service Composition, QoS, Optimization*.

1. Introduction

With the large amount of information stored and manipulated by users, the customer requirements, the diversity of needs, public infrastructure and private information technologies sector want to communicate more easily and without need to focusing on each of their transaction to interpret their various data, they also want to remove the isolation of their systems, for this reason they are more and more multiplatform, multivendor distributed for large scale, which has led to a great development of information systems which become more complex and heterogeneous.

To be adapted to this new situation, technologies enabling communication and data exchange between heterogeneous applications and systems and their distributed environments called Web services have emerged. They aim to ensure interoperability through a standardized presentation of services and a communications protocol standard for structuring exchanged messages between software components. They also provide a specification of publication and localization of services. The particularity of Web services is that they use the Internet as an infrastructure technology for communication between software components. Migration of companies information systems to services (making their services available to other companies as software components) has increased as a result of a rapid increasing of web services. The problem with this increase is how to select the best web service especially if satisfaction of a request requires the intervention of several of them.

Sometimes two or more services are discovered in available list of services. Therefore, there should be a step or process for selecting the best services from discovered list that can satisfy the user's goal. When more than one Web Service which meets functional requirements is available, the Web Service Selection uses some criteria to select the best candidate service. The value of non-functional properties in these matching Web Services may be different, but essentially they should have minimum requirements.
The selection criteria may have an interdependent relationship. A number of methods for decision making are addressed in Web Service Selection because of the complication that exists during the selection process. [1]. Two significant tasks in the process of using services are selection and ranking in which every solution for them is affected directly on description of services. During describing a service, three items have to be considered: behaviour, functional, and non-functional. The non-functional properties of the services are used as criteria for selecting services.

The basic idea in our approach is to make a selection of the best services needed to be composed to have a better composition without having to test all combinations, so to allow compositions to respond to customers' requests in real time even if the number of available web services is important.

In the following sections we will first discuss related studies and works in this field which is web service selection. Section 3, exposes our conceptual methodology and discusses the choices made. Section

4 shows our experimental study followed by a comparative evaluation. Finally, perspectives of the accomplished work are presented in the conclusion of the paper.

2. Related work

The problem of QoS-based web service selection and composition has received a lot of attention during the last years. In [2] and [3], authors proposed approaches on policy languages. Coding in policy language or in a QoS policy model is used for defining the non-functional requirements. The policy based designs the QoS policy model as a textual document. Preferences and non-functional limitations of the service requestor are shown in the content of the policy model. For showing the non-functional criteria's relations, a matrix is used, also it is applied for their aggregation [4]. The problem with these approaches is that only a limited number of non-functional properties is accepted. In addition, it is difficult for users to understand the matrix aggregation function, because of the complexity that it have. A based user feedback technique is treated in [11]. Authors propose an extensible QoS computation model for a QoS fair management. Nevertheless, the problem of QoS-based composition is not addressed. In [12] Zeng at al, focuses on a dynamic and a quality-driven selection of services. They use to find the best and the optimal selection of component services a mixed linear programming techniques. A similar work is proposed by Ardagnaet al. [14]. In this work authors extend a linear programming model to include local constraints. Linear programming methods suffer from poor scalability when the size of the problem is big, because of the exponential time complexity of the applied search algorithms [15]. In [16], heuristic algorithms are used to efficiently find a near-to-optimal solution. The authors propose two models for the QoS-based service composition problem: (a) a combinatorial model and (b) a graph model. A heuristic algorithm is introduced for each model. The time complexity of the heuristic algorithm for the combinatorial model (WS HEU) is polynomial, whereas the complexity of the heuristic algorithm for the graph model (MCSP-K) is exponential. In [17], a method for semantic Web service composition is presented. The proposed idea is based on Genetic Algorithms and using both semantic links between I/O parameters and QoS attributes. Despite the significant improvement of these algorithms compared to exact solutions, both algorithms do not scale with respect to the number of candidate web services, and hence are not suitable for real-time service composition. In [29] authors proposed a QoS-based web service selection approach. The approach adopts genetic algorithm to select the most suitable web service with user's QoS requests. Another approaches based on semantic web service are addressed to define non-functional models for selection of web services, such as WSMO [5], OWL-S[6], and SAWSDL[7]. In [8] [9] and [10], approaches based on UDDI extensions are proposed. For example in [9], authors added a Quality broker as an additional component in the SOA architecture between repository service requestor and UDDI. The problem with this approach is that only three non-functional properties are addressed. Another defects noticed, is that these kind of methods are based on a limited number of criteria because, it is not extensible for any new service quality.

The proposed Skyline based method in this paper is complementary to these solutions as it can be used as a pre-processing step to prune non-interesting candidate services even if the their number is huge and hence to reduce the computation time of the applied selection algorithm (even if number of constraints is big).

3. The proposed approach

The issue here is how to make a selection of a big number of services that provide the same functionality, but with different qualities, the composition becomes a problem of decision making on the selection of services, taking into account the functional and non-functional needs. According to the SOA paradigm, the composition of the applications is carried out as abstract processes composed of a set of abstract services. Then, at the time of implementation for each abstract service, a real web service is selected and used. This ensures a low coupling and flexibility in the design. The parameters of Quality of Service (QoS) (for example the reactivity, availability, throughput ...) play a major role in determining the success or failure of the composition. A composition of web services based on QoS aims to find the best combination of web services that satisfies a client request. Do an exhaustive search to find the best combination that meets a certain level of composition is not practical in this scenario, because the number of possible combinations can be quite large, depending on the number of sub-tasks that composes the process and the number of alternative services (with the same functionality) for each subtask. In our work, we address this problem using dominance relationships between web services basing on their quality of service attributes. We focus only on web services that belong to the Skyline set [18], i.e. which are not dominated by any other service with an equivalent functionality. These services are good candidates for composition.

We propose a method with two-steps to select services for an optimal composite web service. The web services are classified according to their functional cores to reduce the search time. We assume a set **S** of classes of services, which classify all available web services according to their functionality. Each service class **Sj** contains all services that provide the same

functionality, but differ in terms of non-functional potential properties (QoS). Service providers provide the same web service with different quality levels: different response times, different prices...etc. The first step consists on selecting a subset of services, only the best ones. This first selection function is called Skyline. We used two algorithms adapted specially for this problem which are: Branch and Bound and Block Nested Loop. Different combinations of services of each class must be taken into consideration. However, all services are not candidates for the final solution. The basic idea in our approach is to perform a function on the Skyline services of each category to distinguish between services being candidates for a possible composition, and those who cannot possibly be part of the final solution can actually be pruned to reduce the search space [19].

For this, there are two types of algorithms:
- Approximate algorithms: used for classes with a large number of services. In our work, we chose as an approximate algorithm Branch and Bound (BB).
- Exact algorithms: Used for classes that contains a limited number of services. As exact algorithm we chose Nested Loop Block (BNL).

In [20], authors made a comparative study between these two classes of algorithms, the results showed that BB is the most performed algorithm followed by BNL. BNL is faster than BB but only in data spaces with a reduced size. However BB is the most efficient algorithm when dealing with huge data spaces.

The second step uses the Skyline services to construct a vector which will remain in main memory. The first step will be executed only one time because it is very costly in time calculation especially if we have several classes with a big number of web services. The second step was added because Web services can be inserted, removed and updated continuously, so it is impossible to perform the first step every time. On the other hand checking if a web service belongs to a set of Skyline services does not require much time because of the particular organization of Skyline services in memory.

3.1 Skyline

Given a set of points in a d-dimensional space, a Skyline function selects points which are dominated by other points. A point $\mathbf{a_i}$ (In our case points are web services and the dimensions are the qualities of services) dominates another point $\mathbf{a_j}$, if $\mathbf{a_i}$ is lower (or higher if we look for maximum) or equal to $\mathbf{a_j}$ in all dimensions and strictly less (above) in at least one dimension. Intuitively, a Skyline function selects the best points or the most interesting points in all dimensions.

In this paper, we define and exploit the dominance relationships between web services based on their quality of service attributes. This function is used to identify and reduce the number of services in a class that are dominated by other services in the same class. Determining Skyline services of a class of service requires pair's comparisons of QoS vectors of web services. This process can be expensive in computation time if the number of services is important. More efficient algorithms have been proposed for calculating Skyline. Therefore, we use two existing methods for determining the Skyline services in offline to accelerate the process of services selection after the client request [19]. Skyline operation aims to reduce data space and reduce calculation time. This data filtering is used to sort the present objects by putting those chosen in the central memory and those rejected to a buffer file.

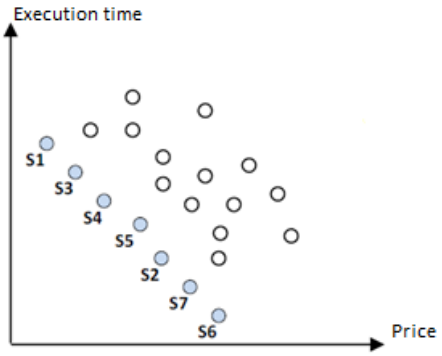

Fig 1. Skyline points (in Blue)

3.2 Utility function

A utility function is used to evaluate the overall multidimensional qualities of a given service, as an example, determine the classification by aggregating the vector of QoS in a single value. For this, we use the technique of Simple Additive Weighting technique [21]. The utility function calculation consists of using the attribute values of QoS to enable a uniform measure of QoS regardless of their units and their ranks. In this technique, each value of QoS is transformed into a value between 0 and 1, by comparing the minimum and the maximum value possible in based on available information about QoS of alternative web services (the other web services that provide the same functionality).
Qmin (j, k) and Qmax (j, k) are respectively the minimum and the maximum values of the *k-th* attribute of QoS of the class **Sj** of web services.

$$Qmin(j,k) = \min_{\forall s \in S_j} q_k(s)$$

$$Qmax(j,k) = \max_{\forall s \in S_j} q_k(s)$$

The function **U** is an aggregate function (Utility function). Now the utility of a web service s belonging to **Sj** is calculated as follows:

$$U(s) = \sum_{k=1}^{r} \frac{Qmax(j,k) - q_k(s)}{Qmax(j,k) - Qmin(j,k)} \cdot w_k$$

and the overall utility of a composite service is computed as follows :

$$U'(CS) = \sum_{k=1}^{r} \frac{Qmax'(k) - q'_k(CS)}{Qmax'(k) - Qmin'(k)} \cdot w_k$$

$wk \in R^{+}_{0}$ and $\sum_{k}^{r} = 1$, the weight of the QoS *qk* represents the user priority, it is like a coefficient. If all attributes have the same priority (a user can prefer time execution than the price of a web service, in this case time weight will be greater than price weight), then all *wk* will be equals : w1 = w2 = w3 =…..= wr.

3.3 QoS calculation of composite web services

The value of a QoS of a composite service is determined by the QoS values of its sub services and the structure of the used composition (for example sequential, parallel, conditional and / or loops). Here, we focus on the sequential composition model. Other models can be reduced or transformed into sequential model [22]. The QoS vector of a composite service CS = {s1, ..., sn} is defined as follows:

Qcs = {q_1 (cs), ..., q_r (cs) or q} in which q_i(cs) is the estimated value of the attribute ***i*** of QoS and can be calculated by aggregating the corresponding values of the sub services. Typical functions of QoS aggregation are addition, multiplication, and minimum. Examples are given in Table 1.

Table 1: Examples of QoS aggregation functions

Type	Examples	Function
summation	response time, price	$q'(CS) = \sum_{j=1}^{n} q(s_j)$
	reputation	$q'(CS) = 1/n \sum_{j=1}^{n} q(s_j)$
multiplication	availability, reliability	$q'(CS) = \prod_{j=1}^{n} q(s_j)$
minimum	throughput	$q'(CS) = \min_{j=1}^{n} q(s_j)$

In what follows, we present the second step of our approach, which is a method of selecting the representative Skyline services to remedy to the situation where the number of Skyline web services of a certain class remains too large and cannot be treated effectively. The main challenge raised is how to organize a set of Skyline services that represent the best compromise of all QoS parameters, so it will be possible to find a solution that satisfies all constraints and also has a high utility score. The main idea is to put all Skyline services in a sorted vector. Services will be sorted according to their utility function.

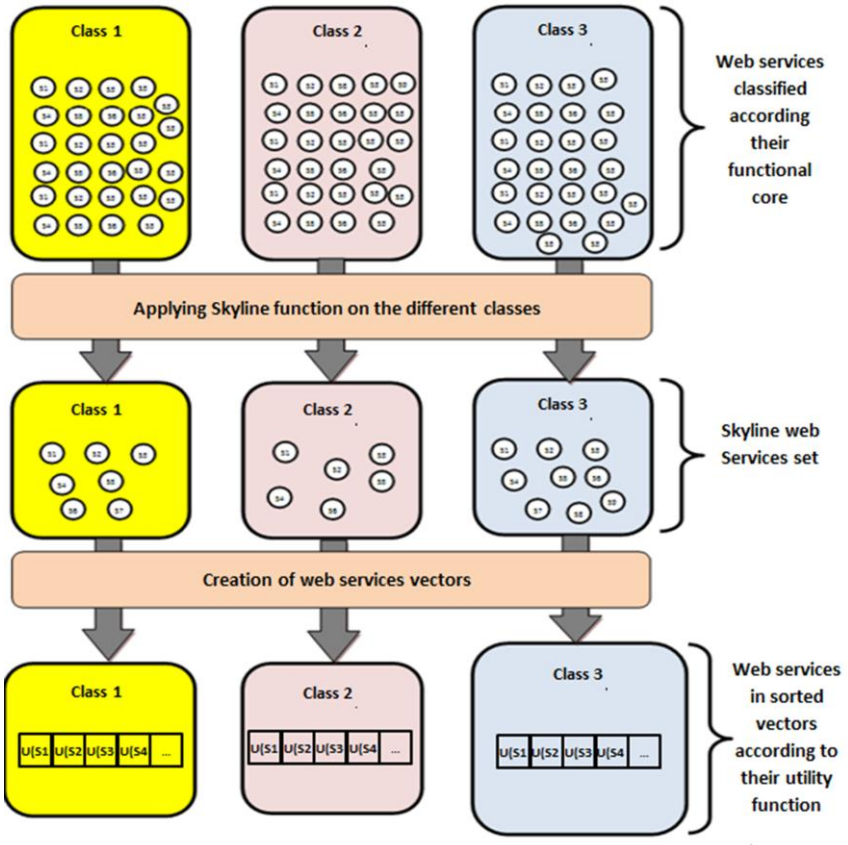

Fig 2. Web services process selection for a composition

At runtime, when a request for a web service composition is treated, the system starts with the first element (the first element contains the service with the highest value of the utility function).

To resume the proposed approach (Figure 2):
- Web services are classified in classes with the same functionality.
- A Skyline function is applied on these classes to select only the best candidates for the future composition.
- Two algorithms are used for Skyline selection: Branch and Bound, and Best Nested Loop. The first algorithm is an approximate method used for large data space. The second one is known for its exactitude but only in small data spaces. We took profits of these two qualities and we used BB when the class size is important to avoid long calculation time. And we used BNL with classes which have a limited size because we are sure that execution time remains acceptable with an optimal an extract results.
- After Skyline sets are created, real-time update, composition, insertion… will be possible.

4. Experimental study

In this section, we present an experimental evaluation of our approach, taking into account the measurement of the effectiveness in terms of execution time. We started our experience by creating databases with different dimensions according to QoS attributes with a measure of their values (Utility function), and also by increasing the number of services of each dimension in order to test our approach with a larger number of services and different distributions.
Firstly, we worked on a database having as a dimension: execution time and price, with a price value between [0,100] (MU), and time between [0,1] (MU). We filled the database with 10000 services, with different values for each attribute discussed before. Two algorithms which are well known with their limitations are adapted to our approach so it is up to us to customize them for our case like the specification of the size of the endpoints in the R-tree (the data space is organize with a specific method using R-trees technique [23]). The generated results of the used two algorithms (BNL and BB) will be put in sorted vector that contains only Skyline web services; each element contains the utility value of a service, and other information that allows its identification. Figure 3 is a graphical representation of the results of the first step. The red points represent Skyline services in the opposite of the black ones. We can notice that the most representative services represent a very limited number, what facilitates the real time web services composition. Tables 2 and 3 below contain the execution time of the two algorithms BB and BNL with of the number of web services. It should be noted that this test was carried out on a computer with an Intel (R) Core (TM) i3 M 370 2.40 GHz and 4GB RAM

Fig 3. Graphical representation of Skyline web services

These results confirm the comparative study done in [20]. We notice that the time taken by the BB algorithm is significantly lower than time taken by BNL. But it does not preclude the use of BNL in some cases where the initial number of web services is not significant (neglected execution time and maximum efficiency).

Table 2. Execution time of BNL algorithm in 2d data space

Number of web services	*Execution time (in ms)*
100	73
1000	76
5000	1480
10000	2068

Table 3. Execution time of BB algorithm in 2d data space

Number of web services	*Execution time (in ms)*
100	1
1000	2
5000	4
10000	6

Computation time taken by the system during the creation of QoS sorted vectors in the second of our approach, are listed in the table 4. Because the system uses only Skyline services, we notice that the computation time is almost equals to zero (some milliseconds).

Table 4. Execution time to create Skyline vectors

Number of Skyline web services	*Execution time (in ms)*
10	≈0
20	≈0
50	≈0
100	8
200	9
500	10
1000	12
2000	13

Table 5. Comparison of some proposed QoS based selection methods

	Criteria				Overall result
Approaches	Performance	Hierarchical Properties	Automatic	Scalability	
Improve Protocol [9]	Low	No	Average	Low	Weak
Semantic [6][25]	Average	Yes	Average		Good
Policy [3][26]	Low	No	Low	Low	Weak
Trust and Reputation [27]	Average	Yes	Low		Average
Preference [13][28][29]	N/A	No	Low	Low	Weak
Our approach	High	No	High	High	Very Good

The results of this experimental study show a significant gain in terms of performance compared to existing approaches. The same thing is noticed for the management of web services i.e. when the system receives a new web services, or when a web service leaves the system, the required time for updating web services classes is also almost equals to zero, thanks to the selected data structure, what results a faster update.

5. Comparative evaluation

The comparison of Web Service Selection approaches that is described in section 2 is based on the following criteria:

5.1 Performance

Performance refers to time needed for web service selection and composition.

5.2 Hierarchical Properties

Ordering properties in an hierarchical structure is meaningful when most interested non-functional properties are at a lower level. Also, this is helpful to sort the properties and group them together, for example by their broader or their domain. While privacy and security are both safety aspects, speed and quality properties are performance aspects. By using this structure, users can show their preferences in the higher level and offerings will be represented by service providers, in useful detail. Therefore, new criteria will be added to mechanism of ranking, also a benefit is provided by considering aggregation of results into final score of ranking.

5.3 Automatic

Last step of service selection is performed by a human; it means, in a registry, user find appropriate services and decides to choose one of them. Providing fully automatic processes for service selection, is still one of the important things for researchers especially for service selection based on non-functional criteria. The essential thing in automatic service selection is in the last step when a service is available service designer specify data for it and user can specify the requirement but in performing service selection human doesn't participate. A situation that needs automation is the aggregation function selection; and the other one is for specific criteria, evaluation functions selection.

5.4 Scalability

Scalability is, once an approach considers lots of properties and also many ranking processes are occurred concurrently. Obviously in this situation the important thing is the accuracy of the result. Therefore by increasing the number of criteria or properties in selection mechanism the accuracy should not be affected.

Table 5 shows a comparison based on these criteria. We notice that the proposed approach is better than many existing solution in literature, in terms of performance, scalability…

6. Conclusion

In this paper we presented an effective method with two steps to make web services composition possible in real time, even if the number of the initial set is important. In the first step, we identify the Skyline services in terms of their QoS values to select representative services for a composition.

To deal with cases where the size of the Skyline is still large compared to the initial dataset, in the second step we organized services in sorted vectors according to their utility function. These vectors will remains in main memory for a quick access. When a client request arrives, only these vectors are used to make a limited

number of combinations to find the best composition. The required time in this case is very limited as we showed it in the experimental study.

Finally, the results of the experimental evaluation indicate a significant performance gain in comparison to existing approaches. Our experiments have shown also that the performance of our skyline-based methods is affected by the difficulty of the composition problem, in terms of the number of the specified QoS constraints.

In the future work, we plan to develop a method for estimating the difficulty of each composition problem. We intend to use this method to deal with other problems like composition problem in cooperative information systems.

References

[1] G. Manish, S. Rajendra, and M. Shrikant, Web Service Selection Based on Analytical Network Process Approach,in Proceedings of the 2008 IEEE Asia-Pacific Services Computing Conference. 2008, IEEE Computer Society.

[2] Y. Liu, A.H.H. Ngu, and L. Zeng. QoS computation and policing in dynamic web service selection. 2004. New York, NY, United states: Association for Computing Machinery.

[3] H. Janicke and M. Solanki. Policy-driven service discovery. in 2nd European Young Researchers Workshop on Service Oriented Computing. 2007.

[4] H.Q. Yu and S. Reiff-Marganiec, Non-functional Property based service selection: A survey and classification of approaches. 2008, Sun SITE Central Europe.

[5] D. Fensel, M. Kerrigan, and M. Zaremba, Implementing Semantic Web Services. 2008, Berlin: Springer.

[6] U.S. Manikrao and T.V. Prabhakar. Dynamic selection of web services with recommendation system. 2005. Seoul, Korea, Republic of: Inst. of Elec. and Elec. Eng. Computer Society.

[7] M. Klusch and P. Kapahne. Semantic Web Service Selection with SAWSDL-MX. in Second International Workshop on Service Matchmaking and Resource Retrieval in the Semantic Web. 2008. Germany.

[8] M. Zuo, S. Wang, and B. Wu. Research on web services selection model based on AHP. 2008. Beijing, China: Inst. of Elec. and Elec. Eng. Computer Society.

[9] Y.-J. Seo, H.-Y. Jeong, and Y.-J. Song. A study on web services selection method based on the negotiation through quality broker: A MAUT-based approach. 2005. Hangzhou, China: Springer Verlag.

[10] E. Al-Masri and Q.H. Mahmoud. Discovering the best web service: A neural network-based solution. 2009. San Antonio, TX, United states: Institute of Electrical and Electronics Engineers Inc.

[11] Y. Liu, A. H. H. Ngu, and L. Zeng. Qos computation and policing in dynamic web service selection. In WWW (Alt. Track Papers & Posters), pages 66–73, 2004.

[12] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Z. Sheng. Quality driven web servicescomposition. In WWW, pages 411–421, 2003.

[13] E.M. Maximilien and M.P. Singh, A framework and ontology for dynamic Web services selection. IEEE Internet Computing, 2004. 8(5): p. 84-93.

[14] D. Ardagna and B. Pernici. Adaptive service composition in flexible processes. IEEE Trans. Software Eng., 33(6):369–384, 2007.

[15] Maros. Computational Techniques of the Simplex Method. Springer, 2003.

[16] T. Yu, Y. Zhang, and K.-J. Lin. Efficient algorithms for web services selection with end-to-end qos constraints. ACM Trans. on the Web, 1(1), 2007.

[17] F. L´ecu´e. Optimizing qos-aware semantic web service composition. In ISWC, pages 375–391, 2009.

[18] S. B¨orzs¨onyi, D. Kossmann, and K. Stocker. The skyline operator. In ICDE, pages 421–430, 2001.

[19] Mohammad Alrifai , Thomass Risse, Article “Combining Global Optimization with Local Selection for Efficient QoS-aware Service Composition » . International World Wide Web Conference Committee (IW3C2) 2010

[20] Marios Kokkodis. Implementation Of Skyline Query Algorithms 2003

[21] K. . P. Yoon and C.-L. Hwang. Multiple Attribute Decision Making: An Introduction (Quantitative Applications in the Social Sciences). Sage Publications, 1995.

[22] J. Cardoso, A. P. Sheth, J. A. Miller, J. Arnold, and K. Kochut. Quality of service for workflows and web service processes. J. Web Sem., 1(3):281–308, 2004.

[23] GUTTMAN, A. 1984. R-trees: A dynamic index structure for spatial searching. In Proceedings of the ACM Conference on the Management of Data (SIGMOD; Boston, MA, June 18–21). 47– 57.

[24] D. Fensel, M. Kerrigan, and M. Zaremba, Implementing Semantic Web Services. 2008, Berlin: Springer.

[25] Y. Liu, A.H.H. Ngu, and L. Zeng. QoS computation and policing in dynamic web service selection. 2004. New York, NY, United states: Association for Computing Machinery.

[26] Y. Wang and J. Vassileva. A review on trust and reputation for web service selection. 2007. Toronto, ON, Canada: Institute of Electrical and Electronics Engineers Inc.

[27] S. Lamparter, et al. Preference-based selection of highly configurable web services. 2007. Banff, AB, Canada: Association for Computing Machinery.

[28] C. Schropfer, et al. Introducing preferences over NFPs into service selection in SOA. 2009. Vienna, Austria: Springer Verlag.

[29] Guofeng Chang QoS-Based Web Service Selection Approach. Software Engineering and Knowledge Engineering: Theory and Practice Advances in Intelligent and Soft Computing Volume 115, 2012, pp 887-892