JACET

J. ADV COMP ENG TECHNOL, 6(2) Spring 2020

A procedure for Web Service Selection Using WS-Policy Semantic Matching

Maryam Amiri Kamalabad[1], Farhad Mardukhi[2], Naser Nematbakhsh[3]
1,3- Faculty of Computer Engineering, Najafabad Branch, Islamic Azad University, Isfahan, IRAN.
(m.amiri@sco.iaun.ac.ir)
2- Computer Engineering and Information Technology Group, Faculty of Engineering, Razi University, Kermanshah, IRAN

Received (2019-12-27) Accepted (2020-04-12)

Abstract: *In general, Policy-based approaches play an important role in the management of web services, for instance, in the choice of semantic web service and quality of services (QoS) in particular. The present research work illustrates a procedure for the web service selection among functionality similar web services based on WS-Policy semantic matching. In this study, the procedure of WS-Policy publishing in the UDDI registry was also described. The approach, which is used to represent the policies, is thus represented as semantic trees, and in this representation, measurable quality attributes are considered; and the certain matching operations are used to identify the similarity match via match function or similarity distance function. The illustration of semantic concepts and rules during policy matching, which is not possible by using a mere semantic concept, leads to better web service matches. The proposed approach has been validated through various tests that can evaluate the similarity of large and arbitrary sets of measurable quality attributes. We also compared the proposed procedure with the other ones. The proposed procedure for web service choose, which uses WS-Policy semantic matching, can be more effective to solve different problems like selection, composition, and substitution of services.*

Keywords: *Ontologies, Rule, Semantic Matching, Service Selection, UDDI, WS-Policy*

How to cite this article:
Maryam Amiri Kamalabad, Farhad Mardukhi, Naser Nematbakhsh. A procedure for Web Service Selection Using WS-Policy Semantic Matching. J. ADV COMP ENG TECHNOL, 6(2) Spring 2020 : 91-106

I. INTRODUCTION

Web service semantic matching will reflex the similarity degree between different concepts among ontologies, rules, and similarity criteria in order to find suitable semantic web services. These services accommodate the requirements of a requester that can be a user, program, or another service among numerous advertised web services[1].

Service Oriented Architecture (SOA) can compose multiple services with various functional properties and make a composed service with a larger function to perform a larger unit of work (tasks). But each of them has different non-functional properties (ex. different data values of response time), so non-functional properties play a major role in all web service-related tasks, especially in determining the success or failure of the composed service [2], [3].

Thus, monitoring and controlling the quality properties of web services are quite of essence to establish an ensured proper behavior and expected QoS stability level. In addition, the successful integration of the large service-oriented architectures in distributed, heterogeneous and dynamic environments depends on supporting quality of service (QoS), management operations of web services at runtime, e.g. QoS-based service selection, QoS negotiation, adaptive web services composition and policy management

[4].

Policies are used to express non-functional properties like the quality of service requirements. They can be applied in different phases of the web service life cycle including design, deployment or runtime [5]. Policies may consist of one or more assertions defining how web services should work and determine the web service behavior. They are associated with a variety of resources like a certain service or endpoint through using the Web Service Description Language (WSDL) that describes functional properties of a service or using other mechanisms defined in a WS-Policy attachment [4].

To select a service, the user should consider the policies; otherwise, he would encounter numerous services with similar functions, many of which may not be matched with the requester's policies and consequently not meeting its QoS requirements [6].

Another issue rising here is that matching the requester's business policies with web service policies is done syntactically; therefore, no perfect matching is achieved and lack of semantic features causes no compatibility matching between the QoS policies regardless of their being equivalent and compatible. Thus, the importance of semantic matching becomes apparent and can be achieved by using semantic web technologies such as ontologies and added semantic information. Moreover, the semantic policies facilitate service negotiation, improve policy monitoring and enable more accurate intersections compared to syntactic approaches [7].

In Service-Oriented Architecture (SOA), web service flexibility, scalability, dynamic selection and composition of services are of significance. Semantic web technologies transform web from passive state to positive state and lead to dynamic management of services, particularly substituting current services in composite services by a new service for instance, a service having better and more compatible quality attributes, dynamically and automatically at run time [8].

Semantic matching algorithms were introduced to focus on matching the large-size or large-number ontologies. During the matching process, it is necessary to exchange information; therefore, a correspondence must be found between different concepts to select the best one. It is impossible to perform this manually; especially in dynamic environments such as the semantic web. Hence, there is a need for automatic matching algorithms [9].

This paper is the extension of the previous published paper [10]. The present paper provides a comparison between the present research work and some related studies and suggests more motivating examples so as to underline the need and usefulness of rules in the matching process and explains the proposed algorithm with a motivating example using transforming rules and explains the time complexity of the proposed algorithm and evaluates it with more tests. Also importance of the critrion of similarity considered in our work through test has been presented. This paper is extended through explaining a method for publishing WS-policies in the UDDI registry and a procedure for selecting web service based on proposed WS-Policy semantic matching algorithm and also importance of the proposed algorithm in the web service select procedure has been explained. Moreover theoretical defining semantic distance that was as the future work of previous published paper and the algorithm using semantic distance has been presented.

The rest of the paper is organized as follows. Section 2 provides relevant research studies. Section 3 describes the semantic web and SWRL. Section 4 exemplifies the proposed approach. Section 5 describes the evaluation of the approach. Section 6 presents a new tuple for publishing WS-Policies in UDDI registry and how to use it to select appropriate service. We also compare issues of this section with other related works. Finally, the conclusion comes in section 7 and future research work is proposed.

II. RELEVANT RESEARCH STUDIES

Some studies (e.g. Bellur & Kulkarni, 2007; Plebani & Pernici, 2009) proposed an approach to match the web service functional properties based on comparing the service profiles assessing the similarity among the profiles in accordance with the concept matching the bipartite graph. Khanam and Yong (2016) have proposed a scheme for semantic web service discovery that utilized a combination of similarity-based method with WSDL specification and ontology using the concept matching the bipartite graph. In

Jiang (2013), the semantic web service functional properties matching algorithm was suggested based on the semantic distance.

Brahim et al. (2012) have presented a semantic approach for specifying and matching the two web service security policies (WS-SP) using an OWL-DL ontology and the execution of SWRL rules. Speiser (2010) proposed an approach that meets matching security requirements and capabilities based on their actual meaning by description and semantic annotations to the WS-Policy documents. Also, he demonstrated that a syntactic–based mechanism of security policies fails to match two compatible policies. OWL language is not sufficient to handle granularity mismatches; however, the OWL reasoners can extract more relationships and knowledge and match them. In Ben Brahim, Chaari, Ben Jemaa and Jmaiel, (2012), authors presented an approach capable of considering several capabilities in combination, matching a single security requirement using the SWRL and resulting in a more flexible matching of security assertions.

Harb et al. (2013) also proposed a heuristic algorithm based on greedy technique and genetics for nonfunctional based service matching. Jagtap and Patil (2016) have proposed a mathematical model with the help of Hidden Markov Model to describe the quantifiable metrics regarding quality evaluation by measuring the QoWS of a given web service and to select ideal web service in terms of QoS parameters. In Badr, Abraham, Biennier and Grosan (2008), the authors introduced a weighting coefficient allocated to each of the QoS features; thus, the importance of quality for the service features is specified and a more appropriate service is selected based on user preferences. Li and Horrocks (2003) proposed an approach for matching non-functional attributes. In their approach, the inferences were done only by description logics; however, our approach is more flexible due to using domain rules and ontologies for matching. The rules cannot fully be inferred by using description logics alone so that the rules are expressed by SWRL as a major key in our approach.

In Chan Oh, Woon Yoo, Kil, Lee and Kumara (2007), a semantic web service composition algorithm was suggested to identify the relationships among different parameter types during the service composition process based on flexible parameter matching framework.

Velasco-Olvera, While and Raju (2014) have put forth a web services adaptation process study at policy layer and represented mismatches of nonfunctional like QoS using WS-Policy standard in web service composition. They have also proposed a model to resolve the incompatibilities automatically through adaptors to enhance interoperability in web services. Considering these mismatches in this layer, they have expressed that the compatibility of the two web service policies must be matched with requirement attributes of another service policies. Chaari, Badr and Biennier (2008) developed the WS-Policy specifications by adding different components with regard to measurable quality attributes and presented as an ontological model. Moreover, they defined a general QoS ontology and used domain rules for matching the QoS attributes. They introduced an algorithm to evaluate two sets of measurable attributes according to the compatibility policies of ontological concepts and rules. After this, they explained a pattern for publishing QoS-based Policies in the UDDI registry and illustrated a procedure for selecting web services using QoS Policy matching. Herein, the domain rules and ontological concepts were also used, and our matching algorithm is in a more flexible and extensive manner, and also our selection procedure is more efficient.

Kritikos and Plexousakis (2006) developed a model as several facts for the QoS based on ontological concepts. They introduced a semantic QoS metric matching algorithm inferring the similarity between two different metrics. In the present research, the QoS descriptions were used based on the ontological concepts.

There are some studies in the field of web service selection based on the quality of service such as Fariss et al. (2018) compared Sort Filter Skyline algorithm and Branch-and-Bound Skyline algorithm and proposed that these algorithms can be used to select the best services according to requester's requirements among the same web services functionality. In fact, these algorithms limit the great number of the same web services functionality. Chandra and Niyogi (2019) and Dahan et al. (2019) introduced solutions for we service selection process based on non-functional properties using Artificial Bee Colony algorithm. Singhal et al. (2019) introduced a solution for

microservices discovery and selection based on the quality of microservices using data mining techniques such as association analysis and K-means clustering algorithm. Wang et al. (2017) suggested a two-phase decisions approach that is named ISAT to select web service. In the first phase, they considered some parameters and used convex hull technique and strict rules to obtain reliable web services and search space is limited. In the second phase, the optimal web service is selected among reliable web services.

Briefly, the main differences between our work and the previous relevant papers based on evaluation factors are shown in tabular form (see table I):

TABLE I
MAIN DIFFERENCES BETWEEN PROPOSED WORK AND RELEVANT RESEARCH STUDIES

#	Citation	Evaluating factors considered
1	Bellur & Kulkarni, 2007; Plebani & Pernici, 2009; Khanam and Yong (2016); Jiang (2013);	Matching the web service functional properties [11], [12], [13], [14]
2.	In Chan Oh, Woon Yoo, Kil, Lee and Kumara (2007)	semantic web service composition[22].
3.	Harb *et al.* (2013); Jagtap and Patil (2016); In Badr, Abraham, Biennier and Grosan (2008); Li and Horrocks (2003);	Matching the web service nonfunctional properties[18], [19], [20], [21]
4.	Brahim *et al.* (2012); Speiser (2010); In Ben Brahim, Chaari, Ben Jemaa and Jmaiel, (2012).	Semantic Matching the web service security policies (WS-SP) [15], [16], [15]
6.	Fariss et al. (2018); . Chandra and Niyogi (2019) ;Dahan *et al.* (2019); Singhal *et al.* (2019) ; Wang *et al.* (2017) ;	we service selection process based on Matching non-functional properties[25],[26],[27],[28],[29]
5.	Chaari, Badr and Biennier (2008) ;	Semantic Matching the web service measurable attributes according to the compatibility policies, service selection procedure using this semantic matching [8]
6	Our Proposed work	Semantic Matching measurable attributes as WS-Policy structure, flexible ,extendable, Low probability of failure in matching two WS-Policies , web service selection procedure using the WS-Policy semantic matching ,to compare this procedure with number 5

III. SEMANTIC WEB AND SWRL

Ontology specifies a conceptualization of a specified domain in terms of concepts along with their properties, instances, restrictions and relationships in a machine-understandable manner using a semantic web technology like Web Ontology Language (OWL). It is one of the semantic reasoning standards using description logics and relationships between concepts. In this regard, new knowledge can be inferred and a knowledge base can be provided based on the concept. The OWL can explicitly describe how vocabularies are equivalent or different from each other. It is of benefit when the same concepts are described with different terminologies[1].

Using the OWL reasoning, more relationships can be inferred based on properties such as inversion, symmetry, and transitivity, consequently resulting in the knowledge discovery [16]. For example, the OWL-Q ontology consists of some concepts to describe and model the QoS information and presents the relationships among the QoS properties. Moreover, ontologies of time, currency, and operator were also developed [24].

Semantic Web Rule Language (SWRL) is one of the semantic reasoning standards and is used to express deductive knowledge and improve the OWL language. It is suitable to express combined rules as conjunctive and atomic ones as a logic form of if….then and uses the terms of the OWL concepts [30].

Nowadays, the use of ontology-based systems and in particular adding rules has been increasingly growing. Many the QoS attributes can be expressed by their related metrics. The syntactic matcher and OWL reasoners cannot infer and identify terms indicating the same concept, deposit the fact that the assertions are equivalent. Using ontological concepts and domain rules can make this identification possible. Thus, the rules are used to infer new knowledge and achieve more information about the QoS attributes that cannot be achieved through ontological concepts. The QoS concepts-based domain knowledge is to be completed by domain rules [8], [6]. For example, since delay concept is the same as latency concept in the ontology domain and regarding the domain rules, the result of the computation delay metric and response metric specifies the performance attribute. Therefore, the matching process is correctly performed, and the identification of the relationships between attributes and their metrics would be possible. If the rules are not used, many semantic web capabilities cannot be considered. In this matter, there are many rules that Table II represents some of them [31].

QoS Attributes	Rules
Response time	Execution time +Network time
Response time	process time +Transfer time +Latency time
Response time	Total time taken – requested time
Integrity	Number of successful transacrions /total number of transactions
Accessibility	number of Ack message/number of Requested message
Accessibility	number of Acknowledgements received /total number of requests sent
Interoperability	Total number of enviroments the web service runs / total number of possible enviroments that can be used
Successability	number of response message/number of Requested message
Availability	Uptime/(Up time +Down time)
Availability	1-Down time /measured time
Availability	success invocation /total invocation
Reliability	1-Probability Of Failure
Reliability	1-(system failure rate +process failure rate)
Cost	enactment cost + realization cost
Reliability	The number of success request / the number of request
Reliability	success access rate
availability	The total number of success access rate / access rate
Efficiency	Real throughput /theoretical throughput
Max Throughput	Max(number of requests processed by service provider in measured tim)/measured time
Accuracy	1-#failed requests / #total requests
Reputation	Sum of the end user's ranks on a service's reputation/the number of times service grads
performance	Response time + latency
Access rate	The sum of success access rate ,failure access rate and bounce access rate
bounce access rate	Total number of bounced web services/ Total number of bounced web services +Total number of invoked web services
success access rate	Successful execution/called for execution
success access rate	1- failure access rate
failure access rate	Failed web services/ (failed web services +successfully invoked web services+ bounced web services

The rules can be applied by the inference engines like JESS, through which new knowledge can be inferred and added to the concept-based knowledge base. The inferred information can be used separate from the asserted information . As a result, a rich domain which could not be created with description logic is created using ontologies and rules [30].

IV. A SEMANTIC APPROACH USING WS-POLICY

1. WS-Policy and Policy Compatibility Evaluation

There are various web service non-functional properties which especially contain the QoS attributes. They vary from one domain to another. In order to manage non-functional properties, a structured presentation is required. Quality properties can be described and displayed using the WS-Policy. There are different XML-based structures of WS-policy, facilitating interoperability between organizations which are established as the standard. In this structure, its associated assertion is defined before defining an attribute or its associated metrics. An assertion specifies a property of a behavior and determines the domain of each attribute which is identified by a qualified name (Qnames consist of namespace URI and local part). In this structure, one or more assertions are grouped by ALL elements and constitute one or more alternatives. Assertions act as a conjunction in an alternative. Alternatives are grouped by An Exactly element and act as disjunction [32].

According to the structure, when two policies are matched with each other at least two alternatives are compatible with each other. In syntactical matching, two alternatives are compatible or equivalent if, there exists an equivalent assertion for each assertion in both alternatives. This would necessitate one-to-one matching and the matching process would be time- consuming and complex. In the semantic matching with considering assertions as a requirement or capability, the matching process would be simpler and more specified [6].

Two QoS policy alternatives A and B, CAset and RAset, are respectively defined as capability assertions and requirement assertions of an alternative. The compatibility between two alternatives is defined as [5]:

$$(A \equiv B) \Leftrightarrow (\forall CA \in CAset(A), \exists RA \in RAset(B) S.TCA \bot RA) \wedge (\forall CA \in CASet(B), \exists RA \in RAset(B) S.TCA \bot RA) \quad (1)$$

2. Semantic Tree

In the first step, we introduce the structure of policy (WS-Policy) as a semantic tree instead of an XML format. Each node expresses a concept. Here, the first node takes policy concept and accordingly the structure is related to an OR element. This operator is related to one or more ALL elements and each element is related to one or more assertions and each assertion is associated with an attribute or its relevant metrics (Figure 1). Each node has specifications to allocate different roles to each. For example, consider the following specifications:

Type: A group of keywords are semantically

relevant to each other under a name which is known and called type. It determines the type of concepts like policy, operator, assertion and attribute.

Keyword: Each type has instances. Each instance is called a keyword. It determines the instance of concept. Examples include RTassertion and response time.

The semantic tree has associated ontologies. In fact, there exists a knowledge base in the background of the semantic tree, according to concepts and rules.

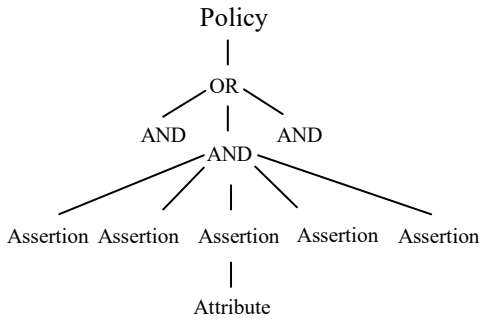


Fig 1. A Semantic Tree of Basic Structure of WS-Policy

3. Flexible Semantic Matching

In the semantic matching approach, a similarity criterion is considered and is usually defined in a range varying from zero to one. Hence, non-functional properties can be categorized into two general categories: qualitative and quantitative [3].

The semantic similarity of the QoS attributes focuses on numerical values. Therefore, we consider the domain measurable quality attributes for matching policies (e.g. reliability, availability, response time, performance, stability, accuracy, capacity, robustness, and cost, scalability, throughput, efficiency, accessibility, successability, reputation, consistency and delivery time).

In Real- time systems, both the computation results and time spent to achieve theses results are of paramount importance.; The similarity criterion is considered here is the difference between the data value of two similar attributes. Based on the data value difference (d) of two attributes, a value from the interval [0, 1] is assigned and it determines the degree of similarity. Values in Table III and Table IV were considered for this kind of systems.

TABLE III
EXAMPLES OF QoS SIMILARITY

similarity	Availability Reliability (%)	Response time (ms)	Successibility (%)
1	d=<0.001	d=< 2	d=< 1
0.8	0.001<d<=0.01	2<d<=4	1<d<=2
0.6	0.01<d<=0.1	4<d<=6	2<d<=3
0.4	0. 1<d<=0.9	6<d<=8	3<d<=4
0.2	0.9<d	8<d	4<d

TABLE IV
EXAMPLES OF QoS SIMILARITY

similarity	Latency (ms)	throughput	Best practices	Documentation
1	d=< 0.05	d=< 0.5	d=< 0.5	d = 1
0.8	0.05<d<=0.1	0. 5<d<=1	0. 5<d<=1	d =2
0.6	0.1<d<=0.2	1<d<=1.5	1<d<=1.5	d =3
0.4	0.2<d<=0.4	1.5<d<=2	1.5<d<=2	d =4
0.2	0.4<d	2<d	2<d	5 <=d

Matching two semantic trees is done based on generic Boolean function match (p1, p2) that determines whether the two parameters of p1 and p2 match [22]. Formally:

Definition 1 (match): A Boolean function, match (p1, p2), returns True

If: p1.type = p2.type and p1.keyword = p2.keyword

OR

In some cases If: p1.type = p2.type and p1. Keyword != p 2. Keyword)

In our approach is If : p1.type = p2.type and p1. Keyword = capability and p 2. Keyword = requirement

When the Boolean function "match (p1,p2)" returns True, it is said that the parameter p1 matches parameter p2. It is called flexible parameter matching.

As already mentioned in the previous paper published on the section of future studies [25], the present approach can be also adopted based on the semantic distance function of between Concepts, Dis (c1.c2), which determines whether the two concepts, namely c1 and c2, are similar to each other [33]. In this paper, we introduce it theoretically.

Definition 2 Dis (c1, c2): the shortest path of all chain relationships between two concepts C1 and C2 in the ontology. The semantic similarity distance formula is described in [33].

The Function Dis (c1, c2) returns values between 0 and ∞. The shorter the distance, the greater the similarity between the two concepts c1.

So, if there are not any similarities between the two concepts, this function approaches ∞

and if the two concepts have the same semantic similar to each other, then there would not be any distance between them and therefore the function returns a value of 0.

4. QoS-based Policy Matching

We have proposed an algorithm, which can evaluate the degree of similarity between the two web service policies based on measurable quality attributes [10]. With a motivating example, we have illustrated this and then made a comparison between the proposed approach and another approach from related work in measurable quality attributes similarity [8].

For instance, a provider provides an execution time of 4 seconds, network time of 1 seconds, and also a set of QoS containing:{Availability=92.78% ,Successability=90%, Reliability=87.56%}. On the other hand, a requester may require a response time of 5 seconds or a set of QoS containing: { Availability=92.7%, Successability=92%, Reliability=87.56% }, as shown in figure 2.

```
</wsp: policy>
  <wsp: exactly One>
        <wsp:all> Execution time = 4000 milliseconds
      Network time = 1000milli seconds
       <wsp: all>
       <wsp: all>
            Availability=92.78,Successability=90, Reliability=87.56
          <wsp: all>
    <wsp: exactly One>
</wsp: policy>
```

(a). Provider capability in WS-Policy

```
</wsp: policy>
  <wsp: exactly One>
      <wsp: all>
           Response time =5000 milliseconds
      <wsp: all>
      <wsp: all>
          Availability=92.7, Successability=92, Reliability=87.56
       <wsp: all>
    <wsp: exactly One>
</wsp: policy>
```

(b.) Consumer requirement in WS-Policy

Fig 2. A sample of the WS-Policy matching problem

The algorithm contains six principal functions that act recursively. Each function is defined as follows:

Function (1) "Get Similarity Policy": To compare the QoS properties described in two WS-Policy documents, first each WS-Policy document is represented as two semantic trees. One of them expresses *requirement* assertions of a policy and the other expresses *capability* assertions. In the next step, the second function is called for matching the two trees. Function (2) would represent their similarity in a range varying from zero to one. Additionally, the first function calls the second function for matching the other trees and according to the compatibility of policies, it would normalize the obtained values and finally return the similarity degree of policies in the interval [0, 1], as shown in figure 3.

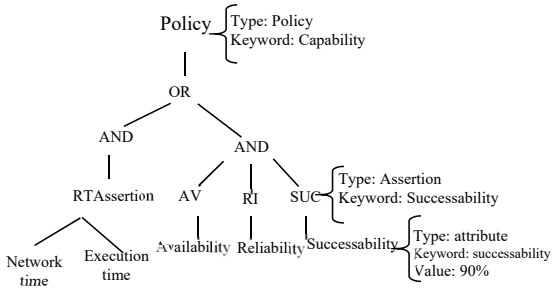
The value zero indicates a lack of similarity between the two policies and value one indicates that two policies have the highest degree of similarity.

Algorithm1. WS-Policy Semantic Matching Algorithm

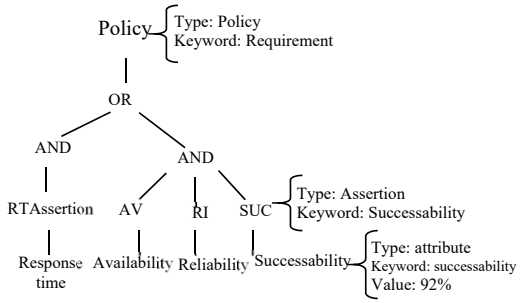
```
1.  Float ontology. get similarity policy (p1,p2)
2.  {T1=Make a tree  for   Cset(p1);
3.   T2=Make a tree  for   Rset(p2);
4.   T3=Make a tree  for   Cset(p2);
5.   T4=Make a tree  for   Rset(p1);
6.   y= similarity match (Tree T1, Tree T2)
7.   x= similarity match (Tree T3, Tree T4)
          IF(y==0||x==0) then Sim=0
8.   else
9.       Sim = (y+x)/2
10.  Return   (Sim);
11. }
```

Fig 3. WS-Policy Semantic Matching Algorithm

The semantic trees of the user requirement and the provider capability pertinent to our example are illustrated in figures 4.



(a). A Semantic tree of the provider capability based on Basic Structure of WS-Policy



(b). A Semantic tree of the user requirement based on Basic Structure of WS-Policy

Fig 4. A sample of the WS-Policy matching problem

Function (2) "Similarity Match": As shown in figure 5, this is to take one Semantic tree of the provider capability from concept WS-Policy and one Semantic tree of the provider capability from concept WS-Policy as input. The correspondence between the concepts and their matching concept in the semantic trees is done hierarchically. Based on the definition of function match (p1, p2) with the same type of concepts at the first level, if the keywords of the first and the second concepts are defined as capability and requirement respectively, this function will call the next function for matching the concepts at the next level. Otherwise, the matching process will not be conducted. In our example, types are equal to "policy" and keywords are equal to capability and requirement respectively. Therefore, Function (3) "Similarity Node" is called.

Function (3) "Similarity Node": This takes one operator "exactly one" from semantic tree 1 and one operator "exactly one" from semantic tree 2 as input. Since the types of concepts are similar, according to the concept of operators "ALL" in the next level, a matrix will be first created with the same dimension number of sets in the two trees and then, it calls the next function for each of the sets. Next, the degree of similarity for the sets is returned and stored in the matrix. Finally, with respect to the two things, (one concept of operator "exactly one" and other the higher the normalized value is, the better level the QoS is), the maximum values in the matrix would be considered as the degree of similarity for two trees.

Function (4) "Match Assertion": This is to take each of assertion concepts from the sets belonging to semantic tree 1 and each of assertion concepts from the sets belonging to semantic tree 2 as input. If the concepts are of the same type and provided that the keywords of concepts are also the same according to the compatibility of the two sets, the next function will be called for the next level trees. In our example, assertions belonging to semantic tree 1 and tree 2 are RT Assertion, RI, AV, and SUC. As long as concept types are equal to the assertion, keywords of concepts are consequently checked. As one instance, since keywords of an assersion type are equal to RT Assertions, the function (5) will be called and it returns the degree of attributes similarity. If all assertions of the first set satisfy assertions of the second set, a normalized degree of similarity is out of function 4. Otherwise, it would be zero.

Function (5) "Match Child": This takes as input two concepts from semantic trees. If the types of concepts are the same, which in our example are "attributes", then semantic reasoning will be performed and transformation rules will be used; because an attribute can be expressed as its associated metrics. If new knowledge is inferred, then each of them will be represented as separate semantic trees. In the next step, when the keywords of concepts are the same, conversions must be done if the dimensions are different. Finally, the specification of values determines the data value difference between the two attributes and calls for the next function. As an instance in tree 2, the attribute "response time" is expressed as metrics of network time and execution time. Hence, new knowledge is inferred named response time since keywords of concepts in trees are equal to response time and function "sim1" is called.

Function (6) "Sim 1": Additionally, a function is defined for each of QoS attributes to consider various numerical ranges for different data values, assign a value from the interval [0, 1] for each of them and determine and return the similarity

degree of the two attributes according to the data value difference (d), which is as function's input. A single case example of the response time attribute is shown in Figure 5. Since (d) is smaller than 2, the function returns as output value 1. It means that the QoS level is acceptable.

```
1.  Function similarity Match (Tree T1,Tree T2)    //
    Trees Matching
2.  {  Node1= get root(T1)  ;
3.    Node2= get root (T2)  ;
4.    IF( ( node1.Type == node2 .Type) &&
        (node1.keyword == Capability)&&
        (node2.keyword == requirement) then  {
5.          For child do
6.             Return  similarity node
    (node1,node2)}};
7.  Function similarity node (node1,node2)    //
    Operator Matching
8.  { IF ( node1.Type == node2 .Type) then
9.     {  l1=degree (node1)//number Child of  node1
10.      l2=degree(node2)//number Child  of  node2
11.      For all of child do{
12.           For (i=0 ; i< l1; i++){
13.                For (j=0; j<l2; j++)
14.                     For all child do{
15.                          Matrix [i][j]= Match
    assertion       (node1(i), node2(j) ;}}}
16.     Return Max matrix [i][j]
17.   }};
18. Function Match assertion (assertion list1[],
    assertion list2[]) // Assertion Matching
    {
19. If (assertion list1 [].type == assertion list2 []. Type)
    then
20.   {   find=true ; k=0 ; z=0;
21.     While (find) and (k< length (assertion list1[]))
22.      {   g=0 ; Find1= false;
23.          While (not find1) and ( g< length
    (assertion list2[])
24.          {
25.            IF  (assertion list1[k].keyword ==
    assertion list2[g].keyword)   then
26.              { for all of Child do // leafs
27.                   s= match child (node1,node2)
28.                   If (s==0) then { g++
29.                   else
30.                       {z= s+z;Find1 = true ; k++}}
31.            Else
32.        g++
33.          }
34.        If( g == length (assertion list2[]) then
35.         { find = false; z=0};
36.    } };
37.   z=z/length (assertion list1[])
38.   Return(z);
39. }};
```

```
40. Function match child (node1,node2)// Attributes
    Matching

    {
41.  IF ( node1.Type== node2 .Type) then{
42.    As1=transforming rule( assertion list1[k]);
43.    As2= transforming rule( assertion list1[g]);
44.    If  As1!=null then{T5= make a tree for As1 and
    node1=get root(T5)};
45.    If  As2!= null then{T6= make a tree for  As2
    and  node2=get root(T6)};
46.    IF node1.keyword == node2. Keyword   Then
47.      { If (node1.unit)!=(node2.unit)  then  convert
    unit(node1,node2)
48.       d = |node1.value- node2.value|
49.       Select case node1.keyword
50.          Case reliability
51.             s=sim1(d)
52.          Case availability
53.             s=sim2(d)
54.       end select
55.       Return (s);
56.    Else
57.    Return (0)}};
```

```
1.  Function sim1 (d) {
2.  If   d=< 2        then   s=1
3.  If   2<d<=4       then   s=0.8
4.  If   4<d<=6       then   s=0.6
5.  If   6<d<=8       then   s=0.4
```

Fig 5. Functions of the WS-Policy Semantic Matching Algorithm

As was mentioned previously, we can use semantic distance function Dis (C1, C2) [26] for Matching of the proposed approach. In the present QoS-Based policy Algorithm, lines 4, 8, 19, 25, 41 and 46 would be substituted as follows:

Line 4 in Function similarity Match is replaced with **IF** Dis (node1. Type, node2. Type) == 0 && Dis (node1.keyword, Capability) == 0 && Dis (node2.keyword, requirement) == 0 **then**

Lines 8, 41 are replaced with **IF** Dis (node1. Type, node2. Type) == 0 **then**

Line 19 is replaced with **IF** Dis (assertion list1 [].type, assertion list2 []. Type) == 0 **then**

Line 25 is replaced with **IF** Dis (assertion list1 [k].keyword, assertion list2 [g].keyword) == 0 **then**

Lines 46 is replaced with **IF** Dis (node1. keyword, node2. keyword) == 0 **then**

5. Complexity Analysis of the Proposed Algorithm

The time of complexity of the matching QoS-based policy algorithm is computed with the inclusion of several tasks as follows:

"Get similarity policy" functionality: the time complexity of this function depends on the "similarity mach" functionality. The "similarity node" functionality is done twice: once for capability assertion concepts in the first WS-Policy with requirement assertion concepts in the second WS-Policy, once for capability assertion concepts in the second WS-Policy with requirement assertion concepts in the first WS-Policy.

Similarity mach functionality: the time complexity of this function depends on the "similarity node" functionality

"Similarity node" functionality: l1 denotes the number of AND operators in one tree and l2 denotes the number of AND operators in another tree. This function calls the "Mach assertion" functionality l1* l2 times. Time complexity of the search of a maximum value in the created matrix is bounded by O (l1*l2).

"Match assertion" functionality: assertion list1 denotes the number of assertions in one set in one tree and assertion list2 denotes the number of assertions in one set in another tree. K denotes the number of attributes in one assertion in one tree and g denotes the number of attributes in one assertion in another tree. This function calls the "mach child" function for two assertion list1 and assertion list2 concepts assertion list1*assertion list2 *k*g times.

"Mach child" functionality: this function O (1) time is executed.

Hence the total time complexity of the matching QoS-based policy algorithm is simplified as: l1*l2* assertion list1*assertion list2 *k*g.

The time complexity of this algorithm is polynomial, and considering the worst case, where l1=l2= assertion list1=assertion list2 =k=g= N, the time complexity of the proposed algorithm is bounded by O (N6).

6. Comparison between the proposed approach and other related approach

Chaari, Badr, and Biennier (2008) did not consider the WS-Policy structure and they introduced an algorithm to compare only one set of provider measurable capability attributes and one set of consumer measurable requirement attributes according to the compatibility policies of ontological concepts. Moreover, the algorithm result was expressed as false that signals a lack of compatibility and the similarity degree is equal to 0 or as true that means the two sets are compatibility and similarity degree that is equal to 1, while our proposed approach considers the WS-Policy structure and our algorithm compares and calculates two WS-Policy structures according to the compatibility policies of ontological concepts. As a result, we can calculate similarity degree and enormous measurable QoS, therefore, our work is way more extensible. Besides, similarity degree is expressed in the interval [0, 1] that value 1 means the QoS level is highly acceptable and value 0 is not acceptable. Since our work enjoys more flexibility and with respect to its high extensibility, the algorithm can affect real-time systems and high probability can find web service with acceptable service quality level.

V. EVALUATION

To evaluate the proposed algorithm obtaining from the flexible parameter matching framework; several experiments were performed using different data sets. It was shown that the proposed approach can evaluate the degree of similarity

in large and various sets of measurable QoS attributes for the requester and provider.

In all experiments, let us assumedly consider several measurable QoS attributes of provider and requester-considered as capability (C) and requirement (R) to generate a large number of WS-Policies as the experimental test input data sets. Some of them are shown in tables IV, V, VI, VII, VIII, IX, X and 11. Furthermore, Table 2 and Table 3 represent the degrees of similarity according to data value difference (d) of each attribute.

Test1. In the first experiment, we evaluated the similarity of provider policy and requester policy, each of which containing at most a set of an attribute. Some datasets are shown in Table V and table VI. The experiment was carried out on 25 datasets. For example with regards to the data set (1), similarity value of provider C QoS and user R QoS is obtained 1 and similarity value of user C QoS and provider R QoS is obtained 1 and normalized similarity of policies is obtained 1. As it is shown in Figure 6, the values obtained in experiments are equal to the desired and acceptable values

TABLE V
DATA SETS OF TEST1

Data set	User QoS attributes	
	Capability	*Requirement*
1	-	Response time=133.33
2	throughput=6.3	availability=87
3	latency=11	Reliability=90.85
4	Best practice =84.5	-
5	-	Response time=160
6	documentation=8	Response time=210
7	Reliability=79.999	latency=11

TABLE VI
DATA SETS OF TEST1

Data set	Provider QoS attributes	
	Capability	*Requirement*
١	Response time=135.33	-
٢	availability=86.5	throughput=6
٣	Reliability=89.99	latency=11.33
۴	-	Best practice=84
۵	Response time=155	-
۶	Response time=200	documentation=6
٧	latency=10.67	Reliability=80

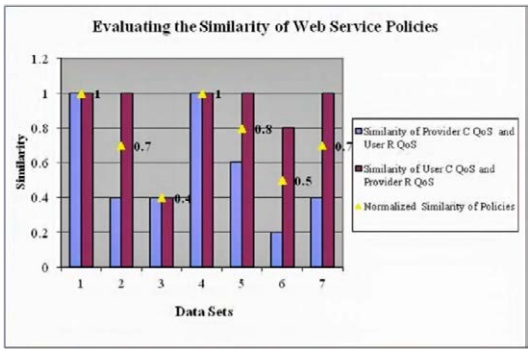


Fig 6. Results of test1

Test2. In the second experiment, each policy contains multiple sets of an attribute. Experiment was performed on 30 data sets. Some datasets are shown in table VII and table VIII. As it is shown in Figure 7, the experiment values are equal to the desired and acceptable values.

TABLE VII
DATA SETS OF TEST2

Data set	Provider QoS attributes	
	Capability	*Requirement*
1	{Response time=102} {succcessibility=97}	{Availability=95.9} {best practice=100}
2	{ succcessibility=95} {Reliability=89.99}	{Response time=133 } {Response time=133.86} {documentation =10}
3	{Reliability=90.75} {Availability=80}	-

TABLE VIII
DATA SETS OF TES

Data set	User QoS attributes	
	Capability	*Requirement*
1	{Availability=96.03} {best practice=99.5}	{Response time=101.5} {successibility=96}
2	{Response time=133.33} {documentation= 8}	{Accuracy=88.5} {Reputation=90 } { succcessibility=97}
3	-	{Availability=80.50}

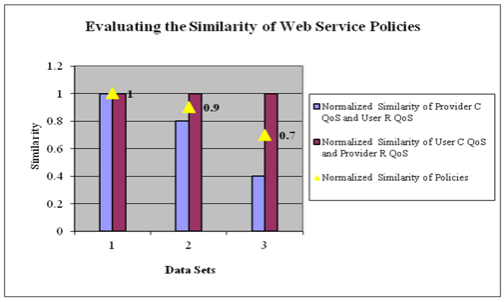


Fig 7. Results of test2

Test3. In the third experiment, each policy contained multiple sets of multi-attributes. Experiment was conducted on 10 data sets. Some are shown in Table IX and table X. As shown in Figure 8, the experiment values are equal to the desired values.

TABLE IX
DATA SETS OF TEST3

Data set	User QoS attributes	
	Capability	*Requirement*
1	{Reliability =98.50 throughput =6.1}	{Availability=98.5 Response time=98.75}
2	{documentation=3}	{Response time=95.5, Best practice =84, throughput=8}
3	-	{Response time=102, Latency=11,successibility=90, Reliability=86.57} {Availability=92.78, Best practice=84}
4	-	{Response time=92.5, Latency=11,Reliability=91.5 } {Response time=100, Latency=11,Reliability=92.99}

TABLE X
DATA SETS OF TEST3

Data set	Provider QoS attributes	
	Capability	*Requirement*
1	{Availability=102.5 Response time=101.5}	{Reliability =95.99 throughput =6.3}
2	{Response time=95.5, Best practice=85, throughput=7.7}	{documentation=8}
3	{Response time=100, Latency=11, successability=90 ,Reliability=87.56} {Availability=92.8, Best practice=84}	-
4	{Response time=95.5, Latency=11,Reliability=92.85}	-

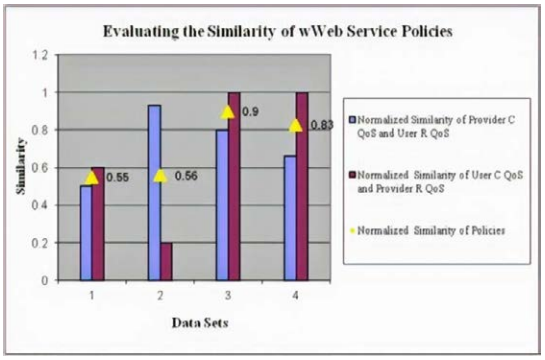


Fig 8. Results of test3

Test4. In the fourth experiment, if there were no similarity between the capability of a policy QoS attributes and the requirement of another policy QoS attributes, then the two policies would not be similar and the obtained value was zero as shown in Figure 9. Its data set is also shown in Table XI and table XII.

TABLE XI
DATA SETS OF TEST4

Data set	User QoS attributes	
	Capability	*Requirement*
1	{best practice=85}	{Response time=105.5 Reliability=99.99}
2	{Response time=95} {Response time=96} {documentation=2}	{successibility=95, Reliability=95.99}

TABLE XII
DATA SETS OF TEST 4

Data set	Provider QoS attributes	
	Capability	*Requirement*
1	{Response time=102, Reliability=99.99}	{throughput=8,latency=8} {latency=9}
2	{successibility=95, latency=10 Reliability=98.50}	{Response time=95} {Response time=90} {documentation=10}

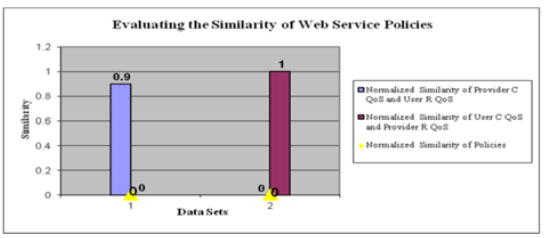


Fig9.Results of test4

Test5. the critrion of similarity considered in our work , the data value difference (d) of the two same attributes, is vital.for this reason, In this experiment, we have compared the results of our proposed algorithm with the result of previous related work from Chaari, Badr, and Biennier (2008) [5]. The experiment was carried out on 25 datasets. For example, with regards to the data set shown in table XIII and table XIV, in the proposed algorithm, the similarity value of provider C QoS and user R QoS is obtained 1 and similarity value of user C QoS and provider R QoS is obtained 1 and normalized similarity of policies is obtained 1. As it is shown in Figure 10, the values obtained in the experiment are equal to the desired and acceptable values. Whereas, in previous algorithm [5] similarity value of provider C QoS and user R QoS is obtained 0 and similarity value of user C QoS and provider R QoS is obtained 1 and normalized similarity of policies is obtained 0. It can be inferred that our work is more and more flexible and also values are equal to the desired and acceptable values.

TABLE XIII
DATA SETS OF TEST 5

Data set	Provider QoS attributes	
	Capability	***Requirement***
Proposed research	{Response time=102 , latency=10, Reliability=99}	{throughput=8, latency=8} {latency=9}
Previous research	{Response time=102 ,latency=10,Reliability=99}	{throughput=8, latency=8} {latency=9}

TABLE XIV
DATA SETS OF TEST 5

Data set	Provider QoS attributes	
	Capability	***Requirement***
Proposed research	{Response time=102 , latency=10, Reliability=99}	{throughput=8, latency=8} {latency=9}
Previous research	{Response time=102 ,latency=10,Reliability=99}	{throughput=8, latency=8} {latency=9}

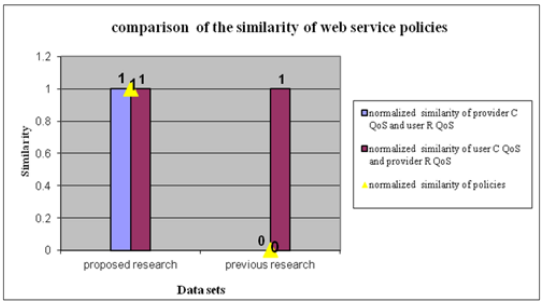


Fig 10. Results of test5

VI. A Method for Publishing WS-Policy and a Procedure for Selection Web Service

A challenge that is still faced in the selection process of a web service is that in the discovery process of service one deals with lots of similar web services functionality that each of them contains different nonfunctional properties. This should be addressed considering their nonfunctional properties. Another challenge is applying these nonfunctional properties to enhance the selection of services. To address this challenge a structure is required to represent these properties. To represent nonfunctional properties as lots of sets with the desired and arbitrary number of qualities of service and also as various nonfunctional properties, the standard form of the WS-Policy can be appropriate.

1. A method for Publishing the WS-Policy

The tModel field is used to publish WS-Policy []. We define three fields as a tuple named tModel, tModel =< WS-Policy ID, WS-Policy URL, S-ID>, this tuple is used to publish WS-policy in the UDDI registry.

Each WS-policy has a unique code that is named as WS-Policy ID and is defined as a distinct file, WS-Policy URL will refer to the WS-Policy XML file, and finally, each web service has a unique code that is named S-ID.

2. A procedure for Selecting Web Services Using WS-Policy semantic matching algorithm

We illustrate our procedure through a simple scenario of web service substitution. We need to

replace the similar web service both functionality and non-functionality.In our research, we would only focus on the service nonfunctional selection phase.

First of all, the Selection Engine (SE) sends an initial request message with the unique Code of WS1(S-ID) that will be substituted to the UDDI registry. The UDDI registry does some tasks: finding tModel, finding services, and finding WS-Policies. Acceptable WS-Policy is attached to the WS1.The policy matcher receives WS-Policies and according to WS-Policies stored in tM files, and WS-Policy attached to the WS1 acts to match WS-Policies. Figure 11 shows a glimpse of the procedure of web service selection.

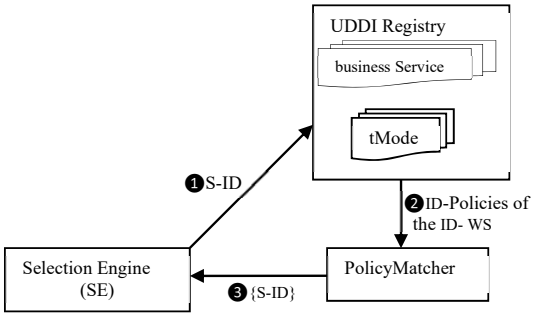


Fig11. Exchanging messages between different entities

After this, the Selection Engine (SE) constructs a table that is named the Selection Table, ST = (STi, 1<=i<=n), each its row belongs to an STi, which includes a particular service, a WS-Policy and, a value. Values can be between 0 and 1 that are an indicator of the similarity degree. The SE chooses the service with the maximum value. The table is shown in Figure 12.

S_1	WS-Policy$_1$	0.8
S_2	WS-Policy$_2$	1
S_n	WS-Policy$_n$	0

Fig12. Example of the selection table

3. Comparison of the proposed web service procedure with the other related web service procedure

There are four advantages in the proposed procedure in compression with another as follows:

1. In the proposed solution to publish WS-Policy in the UDDI registry, a new tuple is defined as tModel that needs three fields, while in the other solution, the defined tuple, tModel, needs four fields. Because in the proposed solution, WS-Policy standard form is used,as known in this form, all of attributes that requester needs are as separated sets and no longer quality attributes of requester need to be searched and grouped as the sets . Therefore, less information is stored.

2. In the proposed solution, three components cooperate in selecting an appropriate web service, while in the other solution, four components cooperate in selecting an appropriate web service.

3. In the proposed solution, three request-reply messages among the components are transmitted, while in the other related search, five request-reply messages among the components are transmitted. Therefore, the number of messages is less.

4. In the proposed solution, a web service with acceptable QoS so more likely is finally selected, and the real-time system can continue its tasks, while in the other solution, the service selection is limit. It is possible that service with requester quality attributes cannot be accessible, and the service cannot be selected, and the system cannot continue its tasks. However, there is web service with acceptable quality attributes of the requester.

VII. Conclusions and Future Work

Quantitative attributes are one of the most important subsets of non-functional properties that can be presented as policies using the WS-Policy standard document. This immense range of attributes accommodates the quality requirements of the service. Syntactic–based policy matching limits the selection of suitable services. A Semantic tree is a powerful data structure which can be used to represent the concepts. Flexible parameter matching framework is a significant factor in evaluating the similarity of web service policies. An approach was proposed for matching measurable quality parameters according to the WS-Policy document using semantic trees, ontologies and rules with regard

to flexible semantic matching via match function or semantic distance function. It is possible to compare the measurable qualitative attributes of service arbitrarily in terms of both number and type of attributes, and to calculate the consistency of their attributes and their semantic similarity, and also, according to the semantic similarity criteria of the attributes, the system is so much likely to be accessible. Not only these, with fewer messages, fewer components, and less stored information, can be selected acceptable quality attributes Therefore, the proposed procedure is potentially efficient when choosing a service, controlling measurable quality attributes of the composed services and replacing a suitable web service. In Future studies, we will test the present mechanism with large and various sets of QoS attributes based on tree matching using Edit distance

REFERENCES

1. Khanam, S.A. and H.Y. Youn, A Web Service Discovery Scheme Based on Structural and Semantic Similarity. J. Inf. Sci. Eng., 2016. 32(1): p. 153-176; Available from: https://pdfs.semanticscholar.org/6913/bd2f608246972ae38da1a856bea49e8457f7.pdf.
2. Velasco-Olvera M., W.D., Raju P., Web Services Adaptation at Policy Layer. International Journal of Multimedia and Image Processing(ijmip), 2014. 4(3/4): p. 226-233; Available from: https://infonomics-society.org/wp-content/uploads/ijmip/published-papers/volume-4-2014/Web-Services-Adaptation-at-Policy-Layer.pdf.
3. F. Hadjila. QoS-Aware Service Selection based on Genetic Algorithm. in Proceedings of CIIA'11. 2011. Saida Algeria: Citeseer.
4. Badidi, E. and L. Esmahi, A Scalable framework for Policy-based QoS management in SOA Environments. Journal of Software, 2011. 6(4): p. 544-553; Available from: http://www.jsoftware.us/vol6/jsw0604-4.pdf.
5. Mukhi, N.K. and P. Plebani. Supporting policy-driven behaviors in web services: experiences and issues. in Proceedings of the 2nd international conference on Service oriented computing. 2004.
6. Chaari, S., et al., Framework for web service selection based on non-functional properties. 2008: p. 94-109; Available from: https://hal.archives-ouvertes.fr/hal-00348511/.
7. Oldham, N., et al. Semantic WS-agreement partner selection. in Proceedings of the 15th international conference on World Wide Web. 2006.
8. Chaari, S., Y. Badr, and F. Biennier. Enhancing web service selection by QoS-based ontology and WS-policy. in Proceedings of the 2008 ACM symposium on Applied computing. 2008.
9. Algergawy, A., E. Schallehn, and G. Saake. A sequence-based ontology matching approach. in Proceedings of 18th European Conference on Artificial Intelligence Workshops. 2008.
10. Kamalabad, M.A., et al. Evaluating the similarity of web service policies using flexible parameter matching. in Proceedings of 2012 International Conference on Measurement, Information and Control. 2012. IEEE.
11. Bellur, U. and R. Kulkarni. Improved matchmaking algorithm for semantic web services based on bipartite graph matching. in IEEE international conference on web services (ICWS 2007). 2007. IEEE.
12. Plebani, P. and B. Pernici, URBE: Web service retrieval based on similarity evaluation. IEEE Transactions on Knowledge and data engineering, 2009. 21(11): p. 1629-1642; Available from: https://ieeexplore.ieee.org/abstract/document/4760142.
13. Khanam, S.A. and H.Y. Youn, A Web Service Discovery Scheme Based on Structural and Semantic Similarity. J. Inf. Sci. Eng., 2016. 32(1): p. 153-176; Available from: https://pdfs.semanticscholar.org/6913/bd2f608246972ae38da1a856bea49e8457f7.pdf.
14. Jiang, B. and Z. Luo, A New Algorithm for Semantic Web Service Matching. JSW, 2013. 8(2): p. 351-356.
15. Brahim, M.B., et al. Semantic matching of web services security policies. in 2012 7th International Conference on Risks and Security of Internet and Systems (CRiSIS). 2012. IEEE.
16. Speiser, S. Semantic annotations for ws-policy. in 2010 IEEE International Conference on Web Services. 2010. IEEE.
17. Brahim, M.B., et al. Semantic matching of ws-securitypolicy assertions. in International Conference on Service-Oriented Computing. 2011. Springer.
18. Harb, I., M. Ezz, and H. Farahat, A Heuristic Algorithm For QoS (Non-Functional) Based Service Matching. International Journal of Computer Science Issues (IJCSI), 2013. 10(6): p. 132; Available from: https://search.proquest.com/openview/b034aaa82d6356bd1b8b7d4958f5fde8/1?pq-origsite=gscholar&cbl=55228.
19. Jagtap, M.S. and P. Patil, Ideal Web Service Selection in terms of Response Time and QoS Parameters. 2016; Available from: https://www.academia.edu/download/54528656/IRJET-V3I7195.pdf.
20. Badr, Y., et al. Enhancing web service selection by user preferences of non-functional features. in 2008 4th International Conference on Next Generation Web Services Practices. 2008. IEEE.
21. Li, L. and I. Horrocks. A software framework for matchmaking based on semantic web technology. in the proceedings of the World Wide Web 2003.
22. Oh, S.-C., et al. Semantic web-service discovery and composition using flexible parameter matching. in The 9th

IEEE International Conference on E-Commerce Technology and The 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services (CEC-EEE 2007). 2007. IEEE.

23. Velasco-Olvera, M., D. While, and P. Raju, Web Services Adaptation at Policy Layer. International Journal Multimedia and Image Processing (IJMIP), 2014. 4(3/4): p. 226-233; Available from: http://infonomics-society.org/wp-content/uploads/ijmip/published-papers/volume-4-2014/Web-Services-Adaptation-at-Policy-Layer.pdf.

24. Kritikos, K. and D. Plexousakis. Semantic qos metric matching. in 2006 European Conference on Web Services (ECOWS'06). 2006. IEEE.

25. Fariss, M., H. Asaidi, and M. Bellouki, Comparative study of skyline algorithms for selecting Web Services based on QoS. Procedia Computer Science, 2018. 127: p. 408-415; Available from: https://www.sciencedirect.com/science/article/pii/S1877050918301509.

26. Chandra, M. and R. Niyogi, Web service selection using modified artificial bee colony algorithm. IEEE Access, 2019. 7: p. 88673-88684; Available from: https://ieeexplore.ieee.org/abstract/document/8752353.

27. Dahan, F., H. Mathkour, and M. Arafah, Two-step artificial bee colony algorithm enhancement for QoS-aware Web service selection problem. IEEE Access, 2019. 7: p. 21787-21794; Available from: https://ieeexplore.ieee.org/abstract/document/8625404.

28. Singhal, N., U. Sakthivel, and P. Raj, Efficient Microservices Discovery and Selection Based on QoS Ontology a Data Mining Approach. International Journal of Innovative Technology and Eexploring Engineering(IJITEE), 2019. 8: p. 4.

29. Wang, W., Z. Huang, and L. Wang, ISAT: An intelligent Web service selection approach for improving reliability via two-phase decisions. Information Sciences, 2018. 433: p. 255-273; Available from: https://www.sciencedirect.com/science/article/abs/pii/S0020025517311696.

30. Plinere, D. and A. Borisov, SWRL: Rule acquisition using ontology. Applied Computer Systems, 2009. 40(1): p. 117-122; Available from: https://content.sciendo.com/view/journals/acss/40/1/article-p117.xml.

31. Ramachandran, M. and Z. Mahmood, Requirements engineering for service and cloud computing. 2017: Springer.

32. Bajaj, S., et al., Web services policy framework (WS-Policy). Policy, 2006.

33. Jiang, B. and Z. Luo, A New Algorithm for Semantic Web Service Matching. JSW, 2013. 8(2): p. 351-356.