Service Discovery using ontology encoding enhanced by similarity of information content

Jiuyun Xu, Ruru Zhang, Kunming Xing
School of Computer & Communication Engineering
China University of Petroleum (Eastern China)
Tsingdao, China
jyxu@upc.edu.cn; justdoitupc@163.com;
xkmb4@126.com

Stephan Reiff-Marganiec
Department of Computer Science
University of Leicester
Leicester, UK
srm13@le.ac.uk

***Abstract*—With the rapid development of web service standards and technology, the number of web services on Internet is increasing rapidly. Consequently, discovering the right service to meet a user's requirements quickly and accurately is crucial for the service community. Many web service discovery methods use web service models with semantic descriptions based on ontologies, allowing to apply logical reasoning to the discovery task. However, requiring logical reasoning can lead to sacrifices in efficiency of web services discovery. To address this problem, this paper proposes a combination of ontology encoding with the similarity of information content approach. We encode the concepts in the ontology in a binary encoding in order to improve the discovery efficiency and then we calculate the semantic similarity of information content between services. Validation of efficiency of the proposed approach is conducted through an experiment using the owls-tc2.0 as benchmark test set. The experimental results show that the proposed method not only can improve the efficiency of service discovery, but also can significantly improve the accuracy of service discovery compared with other discovery methods.**

Keywords-Service Discovery; Ontology Encoding; Information content; Precision and recall

I. INTRODUCTION

Service discovery is one of the key problems in web service research. It is significant because many other problems in this area of research, such as service selection and service composition (see e.g. [19, 20]) are directly based on service discovery; it is also a crucial part of the basic service paradigm, namely "find" in the publish-find-bind paradigm. Web service exist in a dynamic global network with few a-priori agreed conventions, service discovery based on conventional approaches such as keyword based matching at a syntactic level has many limitations. The result of service discovery will be negatively affected as people will use terminology as they feel appropriate, which could lead to incorrect matches and more crucially some possible matches cannot be identified due to language boundaries and different terms being used for the same concept. These challenges necessitate semantic description of web service to obtain more precise descriptions.

The development of semantic web services aims at adding this level and hence improving the precision of service matching. A service matchmaker (or broker) is commonly tasked to judge whether a web service satisfies the needs of a service request, usually eased by an assumption that the service offering and the service request are expressed using the same service description language. Many semantic approaches have been proposed for service discovery, such as OWLS-UDDI matchmaker [15], RACER [10], MAMA [3], WSMO-MX [8], OWLS-MX [6], or OWL-S Discovery [1]. These approaches often fail to identify concepts with similar meaning which are not in a parent-child relation in the concept ontology. A further and crucial shortcoming is that many of these approaches rely heavily on ontology reasoning to unify concepts and hence are relatively slow.

In common with existing work on semantic web services, we use OWL-S to describe web services. OWL-S uses OWL (the Web Ontology Language) to build an upper ontology and describe the properties, capabilities and execution structures related to a web service.

However, our approach improves on efficiency and accuracy of matching by using the following novel contributions:

1. Binary-encoding is used to encode the ontology concepts to support efficient service discovery based on the service functional input and output (IO).
2. Semantic similarity of concepts based on Information Content is used to increase the accuracy of service matching.

The remainder of this paper is arranged as follows: In section 2 related work is introduced. Section 3 provides the overview of the proposed approach and section 4 details the service discovery tool which was implemented; in section 5 we discuss the experimental result and analysis of our method. Finally, we conclude and provide an outline of further research.

II. RELATED WORK

As alluded to earlier, many researchers have proposed web service discovery methods based on semantic web techniques – to present a comprehensive list would be like including a survey paper so we focus on some representative samples. OWLS-MX [6] utilizes a hybrid approach that combines logic based reasoning with approximate matching based on syntactic information retrieval (IR) based similarity computations. Web services are described by OWLS and

semantic reasoning exploits OWLS-DL. During IO semantic matching, it applies five different filters: Exact, Plug-in, Subsumes, Subsumed-by and Nearest-neighbor. Information retrieval (IR) [17] based matching is used when semantic matching fails. [1] presents a new approach combining functional and structural matching for service discovery. The first four filters are using semantic matching techniques , the IR based matching using a structural analysis algorithm will be used if semantic matching fails as fifth filter.

In [21], DAML-S is used as description language for web services. Inheritance relationships between ontology concepts are mainly exploited to realize the IO matching of web services. The result of the matching has been classified to Exact, Plug-in, Subsume, Fail. In [18], authors describe the idea of extracting constraints from the text of natural language description to allow for better matching.

Binary encoding has been very successfully employed in the field of data mining, e.g. [2]. Our use of binary encoding is inspired by the success of this work but presents to the best of our knowledge the first use of the technique in the field of service discovery and selection,

III. THE APPROACH

We will now introduce our approach by first of all providing an overview of the Functional Matching approach – while this is quite similar with existing work, we will use the contributions of the paper (the binary encoding and concept similarity) to refine some steps to increase accuracy and efficiency. The section conlcudes with details of the respective algorithms.

A. Service Functional（IO）Matching

Finding a service to satisfy a user's requirements means at the most basic level that the input and output parameters need to be matched – usually this is referred to as service "functional matching" or "IO matching" and can be on a syntactic or semantic level. We are interested in semantic level matches as they are richer as explained earlier. The assumption is that all input and output parameters are mapped to appropriate concepts belonging to a suitable domain ontology. The relationship between service parameters and request attributes will then be determined by the relationship among the concepts in the domain ontology. A simple ontology is shown in Figure 1.

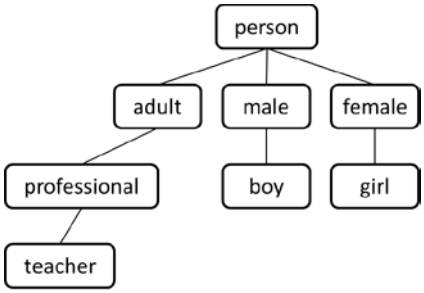

Figure 1: Ontology Concepts

The matches that we consider of interest here are matches between input and output parameters of the service and the respective user requirements. We will need to match the input parameter of the service to the input provided by the request and the output parameter of the service to the output required by the request.

1. *Exact match.* An exact match represents the best case: the service needs the exact input offered and provides the exact output required. Considering this in terms of the ontology, the ontology concepts for the input and output map to the same node as those of the requirements. For example, with reference to the ontology in Figure 1, if the request parameters match to `adult` and `male`, the Service parameters must also be `adult` and `male`.
2. *Plug-in match.* A Plug-in match substitutes a descendant in the ontology for a parent, thus providing something more specific than the exact match, but possibly including some options. For example, if the service's output parameter is `boy`, the request could need `male` or `person`.
3. *Subsumes match.* A Subsumes match is using the inheritance tree in the opposite direction of a Plug-in match – it attempts to substitute an ancestor rather than a descendant.
4. *Fail match.* A so-called Fail match occurs if none of the three levels above lead to a successful match.

B. Functional(IO) Ontology Coding of Service

As we have seen in the previous section, the relationship between two parameters will be decided by the relation of their related concepts in the ontology. Typically methods such as LSC (least specific concept) and LGC (least generic concept) [6] are used to decide the association (ancestor-child, child-ancestor) of two concepts. Due to the number of checks in the ontology hierarchy that need to be made, these methods result in a large overhead and hence inefficient service discovery. Our approach uses a binary-encoding to increase efficiency as all that is needed are some precise matches and then structural relations, binary encodings have been proposed e.g. in [4]. Specifically, we used Binary-encoding to increase efficiency. An overview of Binary-encoding is shown in Figure 2, building on the person ontology from Figure 1.The main gain to be made using bit encoding is in hierarchical comparisons – which are fundamental to Plug-in and Subsumes matches. We can differentiate 3 scenarios for the respective relations of ontology concepts for two services' IO parameters, S1 and S2:

1. the concepts exist in the same domain ontology and the relation is an ancestor--child relation – that is the two concepts exist on a path in the tree (e.g. `boy` and `person`).
2. the concepts exist in the same ontology, but are siblings – such as `male` and `female`.
3. the concepts do not exist in the same ontology.

The Binary-encoding follows a topological sort over the ontological domain space starting from the root concepts to the leaf concepts and assigns binary values to the nodes, reflecting the association among the nodes. Thus, in Figure 2 , person, a root node is coded 000001 while all children will be coded 000011, 000101 and 000111, from right to left, the least significant bit string being the same as the parent to capture inheritance. Ancestral relations can now be computed by fast Boolean operation on the binary strings using OR and AND to efficiently determine Plug-in and Subsumes Matches. For example, to check whether boy is a child node of male, the bit strings are composed with AND and the result should match the suspected parent coding: 000101 (male) and 001101 (boy) will indeed be shown as parent and child as expected. OR will allow to check for a child – the resulting string will match the suspected child's encoding if the relation is indeed a child relation. . If the two strings satisfy neither of the parent child relations tested using AND and OR, they will be sibling nodes (assuming that we are comparing values from the same ontology). [Note that by combining parents' encodings using OR this can also express multiple inheritance].

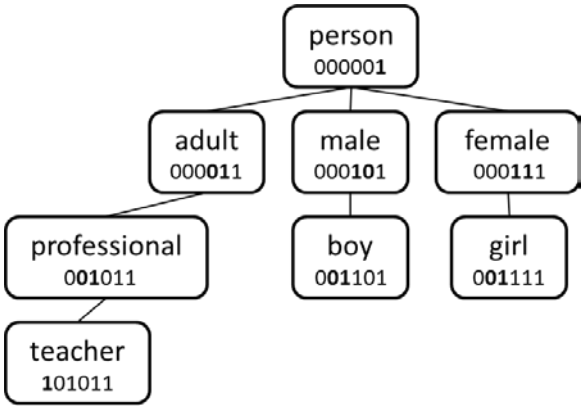

Figure 2: Binary encoding

We will see in the evaluation that the binary encoding allows for very fast comparisons of ancestral relations while also allowing to detect possible sibling relations. However, sibling relations will produce a Fail Match in the classification shown in the previous A. We will attempt to resolve some of the fail matches using our second contribution, the calculation of the similarity of Information concepts.

C. Calculation of Semantic Similarity of Concepts Based On Information Content

Semantic similarity is commonly used to measure the similarity among documents and terms. It is widely used in research domains such as information retrieval and natural language processing. Here we will apply it to compute the similarity between two concepts to improve on service selection.

We use the Information Content (IC) computational model [5], which is not only concerned with subsets of nodes, but also the distance of concepts to ensuring that we obtain suitably precise IC. Such as formula (1.1):

$$IC(w) = k(1-\lg(hypc(w)+1)/\lg(node_{\max})) \\ +(1-k)(\lg(deep(w))/\lg(deep_{\max})) \quad (1.1)$$

Formula 1.1 determines the IC of a specific concept *w*. *hypc(w)* provides the number of sub-concept of a given concept *w*, *deep(w)* returns the depth of concept *w* in the ontology, $node_{max}$ denotes the total number of concepts in the ontology, $deep_{max}$ is the maximal depth of the ontology, *k* is a factor used to adjust the relation between weights and depth -- we use equal weights, hence *k=0.5* in this paper.
[21] presented an approach to calculate similarity. Both information content and the position of two concepts in the ontology are taken into consideration as follows:

$$sim(w_1, w_2) = 1 - k \times \lg(len(w_1, w_2))/ \\ \lg(2 \times \max(depth(w))) - (1-k) \times \quad (1.2) \\ (IC(w_1) + IC(w_2) - 2 \times IC(lso(w_1, w_2)))/2$$

w_1 and w_2 are two different concepts, $len(w_1, w_2)$ is the distance between w_1 and w_2 , k is a weight factor, which can be adapted manually, we are using 0.5 here. $len(w_1, w_2)$ denotes the least common parent node of w_1 and w_2. $\max(depth(w))_{w \subset onto}$ is the maximal depth in the ontology.

As we have coded concepts in the ontology using Binary-encoding, the values of $deep(w)$, $deep_{\max}$, $lso(w_1, w_2)$, $len(w_1, w_2)$ and $\max(depth(w))_{w \subset onto}$ can be obtained using *OR* and *AND* operations and the analysis of concept coding (0-1). $hypc(w)$ has been calculated during encoding (to know the number of bits required for encoding concepts at a specific level one needs to know how many sibling nodes exist at that level).Overall, the type of calculations that need to be performed is computationally simple and hence the complexity of the calculation has been decreased to increase efficiency of service discovery.

We can further improve efficiency by considering what we know about concepts from the earlier analysis of concept matching:

1. If two concepts w_1 and w_2 are the same, then the value of $IC(w_1)$ and $IC(w_2)$ will be the same. $len(w_1, w_2)$ will obviously be *0*, and $sim(w_1, w_2)$ will equal *1*. As these observations will always hold for identical concepts, there is no need to perform calculations for information content and

similarity and hence we check this case before engaging in calculations. The result is that no IC or similarity calculation will be undertaken for identical concepts.

2. Similarly, If concept w_1 and w_2 are in different domains (that is occur in different reference ontologies) and are not the same then their similarity is *0* and we will again not undertake IC calculations.

So, in conclusion we will only compute similarity measures if we have convinced ourselves that concept w_1 and concept w_2 are in the same domain but differ..

D. The Algorithms

The algorithms encode the diverse aspects of our approach, mainly the functional matching, the encoding and the calculation of semantic similarity based on IC. Overall the approach is guided by a user set threshold that controls that only services scoring higher than the threshold will be returned.

Algorithm 1: Function(IO) Ontology Coding of Service

```
1 function IOEncoding()
2  int countConceptsNode [] = countNodes();
3  ArrayList sortConcepts = topologicalSort();
4  HashMap conceptCoding =
      Encoding(countConceptNode, sortConcepts);
5 end function
```

Algorithm 1 handles the binary encoding. Algorithm 2 encodes the matching itself: it is controlled by a set of service request (`requestSet`) and the threshold value (`a`). The set of services available is available as `serviceSet`, result will be collected in `resultsSet` – which will includes the services matching the requirements based on our approach.

Algorithm 2: Matching Services to requirements.

```
1 function match(set requestSet, a)
2 set  resultSet , failSet
3 HashMap  serviceSet = readConceptCoding();
4 for(int i=0;i< serviceSet.length;i++ )  do
5   for(int j=0;j< requestSet.length;j++) do
6     filterI = matchIO(serviceSet.get(i).getInput,
requestSet.get(j).getInput)
7 filterO=matchIO(serviceSet.get(i).getOutput,
requestSet.get(j).getOutput)
8         If(judgetypes(requestSet.get(j), filterI,
FilterO)) then
9                 resultSet.add(serviceSet.get(i))
10                else
11        simI =  SimIC(serviceSet.get(i).getInput,
requestSet.get(j).getInput)
12   simO = SimIC(serviceSet.get(i).getOutput,
requestSet.get(j).getOutput)
13          if(simI>=a and simO>=a) then
14              resultSet.add(serviceSet.get(i))
15            else
16               failset.add(serviceSet.get(i))
17             end if
18          end if
19     end for
20end for
21end function
```

Algorithms 3 and 4 deal with functional IO matching and IC matching respectively.

Algorithm 3 : Functional IO Matching

```
1 function  matchIO(Service s, Service r)
2   if(exactMatch(r,s))         then       //exact
matching
3      return exact;
4   else if(plug-inMatch(r,s))    then
//plug-in matching
5      return plug-in;
6  else if(subsumes(r,s))       then
//subsumes matching
7       return subsumes;
8  else
9       return false;
8    end if
9   end function
```

Algorithm 4: Calculation of the similarity of concept semantic based on IC

```
1 function matchIC(Service s,Service r)
2    if(identifyConcept(s,r))    then
3       sim(s,r) = 1.0
4    else if(!hasSameField(s,r))   then
5        sim(s,r) = 0.0
6     else sim=calculateSim(s,r)  then
7   end if
8  return sim
9   calculateSim(r,s)
10 hypcr = calculatehypc(r);
11 hypcs = calculatehypc(s);
12 deepr = calculateDeepr(r);
13   deeps = calculateDeeps(s);
14   maxnode = calculateMaxNode();
15   len = calculateLen(r,s);
16   maxDept=calculateMaxdept();
17   sim=
countSim(r,s,hypcr,hypcs,deepr,deeps,maxnode,len,m
axdept);
18   return sim;
19  end calculateSim
20 end function
```

IV. SERVICE DISCOVERY TOOL

To illustrate the functionality of our method, we have implemented a prototype as a service discovery tool. Users can describe service using an OWLS file, the tool accepts the requirements file in the "request" textbox (either by typing or through a more interactive dialogue triggered by "add request"). In the Figure 3 the request “book_price_service.owls” has been added. The other field

in the tool accepts a file with service descriptions.. Users can control which types of matches (Exact, plug-in, subsumes and sibling) are considered – ideally one considers all of course. Once matching (functional and IC based) are completed, result are shown as in Figure 4. The format of the result is <service name, level of semantic matching> or <service name, similarity>, such as for example:

```
Book_authorprice_service.owls(Exact)
Monograph_price_service.owls(SUBSUMES)
Book_taxedpriceprice_service.owls:0.93704654689298
89
```

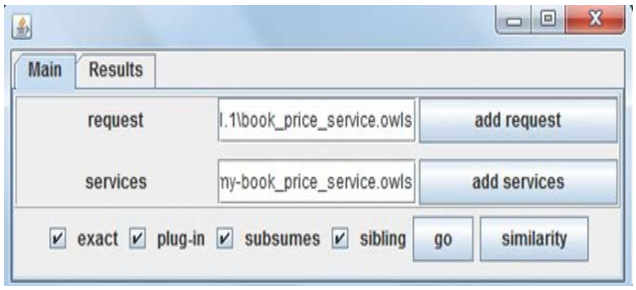

Figure 3: Input for a service request

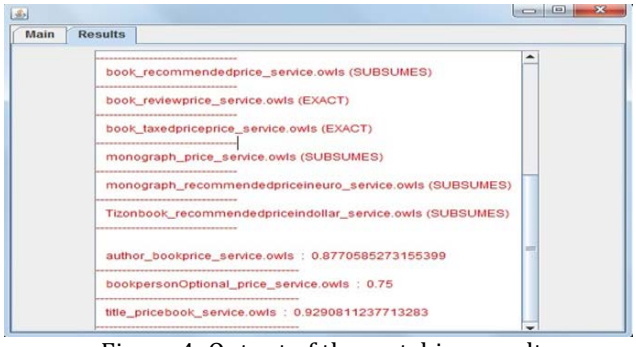

Figure 4: Output of the matching results

I. Experimental Result and Analysis

Our implementation was completed in Java 1.6. We used the API (http://www.mindswap.org/2004/owl-s) for developed by University of Maryland was for our analysis. And Xampp®(http://en.wikipedia.org/wiki/XAMPP) was used as local service host. We used OWLS-TC as test set and the services come from seven different areas: education, medical, care, food, travel, communication, economy, and weaponry.

We analyzed three aspects: precision, recall and average response time to gain an insight into the accuracy and efficiency of the approach. Considering why these three form good evaluation criteria, we can say that fast responses are obviously desirable, precision captures how good the match of the found services is with respect to the user's needs (or very informally how many bad candidates are selected) and recall shows how many of the services that should have been identified were actually found (or very informally how many good candidates were missed out).

Given a service request $Q(Inputs, Outputs)$ to which there are n relevant services related to, our prototype will retrieve m services for the query result, but only k of them retrieved which is related to the service query $Q(Inputs, Outputs)$ ($k \le n$).

Service Discovery Precision (P) is the fraction of the k service retrieved that are relevant to the number m of the user's service need. The formula is as follows:

$$P = \frac{|\{relevant\_Services\} \cap \{retrieved\_Services\}|}{|\{retrieved\_Services\}|} = \frac{k}{m}$$

Service Discovery Recall (R) is the fraction of the services that are relevant to the query that are successfully retrieved. The formula is as follows:

$$R = \frac{|\{relevant\_Services\} \cap \{retrieved\_Services\}|}{|\{relevant\_Services\}|} = \frac{k}{n}$$

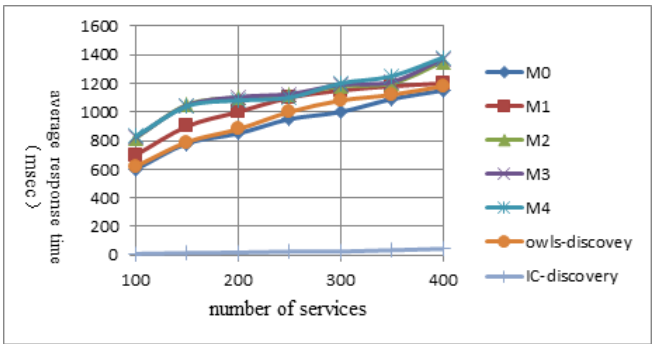

Figure 5: Comparison of average reply time between IC-discovery and other discovery methods

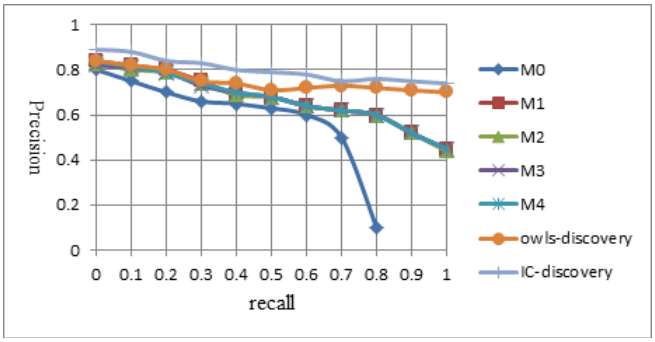

Figure 6: Comparison of Precision and Recall between IC-discovery and discovery methods

Figure 5 and Figure 6 illustrate the average response time, precision and recall for different service discovery matchers using the same test set. In these two figures, M0-M4 are using mixed semantic methods in OWLS-MX [6]. Specifically, M0 is only using an I/O based matching method, M4 is the best available matching method in

OWLS-MX. OWLS-Discovery [1] adapts a combined IO semantic match and structural analysis which first matches based on I/O and if this fails, structural analysis based on a synonyms dictionary is used to improve accuracy. Our novel IC-Discovery method presented in this paper provides significantly improved results in terms of response time and hence will scale much better for larger numbers of services. The main reason for this significant improvement is that the binary encoding used to judge relation between two concepts using AND and OR presents a very significant improvement over reasoning based approaches.

We also find that our IC-Discovery method is presenting better results than OWLS-MX and OWLS-Discovery (as shown in Figure 6). Single semantics-based web service discovery is not working well, as expected, for example M4 will identify some services which M0 cannot find using simple similarity-based methods. OWLS-Discovery will find more services as it can reduce the Fail Match cases by applying the synonyms dictionary. Since our method uses IC-based similarity it produces a more comprehensive comparison of concepts and hence allows us to achieve better results – and thanks to the Binary encoding this does not come at a cost to the efficiency.

I. CONCLUSION AND FUTURE WORK

This paper proposes a hybrid approach for web service discovery based on functional service aspects that uses a combination a binary encoding of ontology concepts and calculation of semantic similarity based on Information Content (IC). Experimental evaluation confirms that this allows for very significantly faster service discovery with a higher accuracy.

In future work, we will consider a more fine grained approach to matching service parameters to client requirements to evaluate the approach against more complex service interfaces and demands. We will also explore extending the encoding and information content calculations to non-functional properties of services.

A further piece of future work will include a comparison to tree –based encoding techniques (e.g. [9, 13, 16]) and a more detailed study of the advantages and disadvantages of IC (e.g. [14]) vs. reasoning based encoding techniques (e.g. [7, 12].

REFERENCES

[1] Amorim, R., Claro, D. B., Lopes, D., Albers, P., Andrade, A. Improving Web service discovery by a functional and structural approach, Web Services (ICWS), 2011 IEEE International Conference on . pp. 411-418 , 2011.

[2] Ayres, J., Gehrke, J., Yiu, T. and Flannick, J. "Sequential Pattern Mining using A Bitmap Representation", Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 429-435, ACM, 2002.

[3] Colucci, S., Noia, T. D., Sciascio, E. D., Mongiello, M., Donini, F. M.. Concept abduction and contraction for semantic-based discovery of matches and negotiation spaces in an e-marketplace , In Proc. 6th Int Conference on Electronic Commerce (ICEC 2004), pp. 41-50.ACM Press, 2004.

[4] Dasgupta, S., Bhat, S., Lee, Y. Taxonomic Clustering and Query Matching for Efficient Service Discovery. Web Services (ICWS), 2011 IEEE International Conference on .pp. 363-370 , 2011

[5] Jay, J., Conrath, D. W. Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy, Proceedings of International Conference Research on Computational Linguistics, pp:19-33, 1997.

[6] Klusch, M., Fries, B., Sycara, K. Automated semantic web service discovery with OWLS-MX, Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems. pp. 915-922. ACM, 2006.

[7] Klusch, M., Fries, B., and Sycara, K.. OWLS-MX: A hybrid semantic Web service matchmaker for OWL-S services. Journal of Web Semantics, 7(2):121-133, 2009.

[8] Klusch, M., Kapahnke, P., Kaufer, F. Evaluation of wsml service retrieval with wsmo-mx , IEEE Computer Society, pp. 401–408, 2008.

[9] Krall, A., Vitek, J. and Horspool, N. "Near Optimal Hierarchical Encoding of Types," Proc. 11th European Conf. Object Oriented Programming, pp. 128—145, Springer, 1997.

[10] Li, L., Horrocks, I. A software framework for matchmaking Based on semantic web technology . In: International Journal of Electronic Commerce , pp. 39-60 , 2003.

[11] Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., ... Sycara, K. OWL-S:Semantic Markup for Web Services, available: http://www.daml.org/services/owls/1.0/owl-s.html , November 2003.

[12] Paolucci, M., Kawamura, T., Payne, T. R., Sycara, K. Semantic Matching of Web Services Capabilities, The Semantic Web Conference (ISWC 2002). pp. 333-347, 2002.

[13] Preuveneers, D. and Berbers, Y. Prime Numbers Considered Useful: Ontology Encoding for Efficient Subsumption Testing tech. report CW464, Dept. of Computer Science, Katholieke Univ. Leuven, 2006.

[14] Preuveneers, D. and Berbers, Y. "Encoding Semantic Awareness in Resource-Constrained Devices," IEEE Intelligent Systems, vol. 23, no. 2, pp. 26-33, March/April, 2008.

[15] Sycara, K., Paolucci, M., Ankolekar, A., Srinivasan, N. Automated discovery, interaction and composition of Semantic Web services. Web Semantics: Science, Services and Agents on the World Wide Web, 2003. 1(1): p. 27-46.

[16] van Bommel, M.F. and Beck, T.J. "Incremental Encoding of Multiple Inheritance Hierarchies," Proc. 8th Int'l Conf. Information and Knowledge Management (CIKM 99), 1999.

[17] van Rijsbergen, C. J. Information Retrieval, 1979.

[18] Wei, D., Wang, T., Wang, J., Chen, Y. Extracting Semantic Constraint from Description Text for Semantic Web Service Discovery, Proceedings of the 7th International Conference on The Semantic Web2008, Springer-Verlag: Karlsruhe, Germany. p. 146-161.

[19] Xu, J., Chen, K., Reiff-Marganiec, S. Using Markov Decision Process Model with Logic Scoring of Prefence Model to Optimize HTN Web Service Composition, International Journal of Web Service Research 8(2): 53-73 (2011).

[20] Yu, H. Q., Reiff-Marganiec, S. Automated Context-Aware Service Selection for Collaborative Systems, in Proceedings of the 21st International Conference on Advanced Information Systems Engineering 2009, Springer-Verlag: Amsterdam, The Netherlands. P. 261-274.

[21] Zhou, Z., Wang, Y., Gu, J. A Novel Method of Extracting Domain Ontology Based on WordNet , In Computer Science and Software Engineering, 2008 International Conference on . Vol. 4, pp. 376-381,2008