I.J. Information Technology and Computer Science, **2014, 09, 24-31**
Published Online August 2014 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2014.09.03

Fast Mapping Algorithm from WSDL to OWL-S

Ashraf B. El-Sisi
Computer Science Dept., Faculty of Computers and Information, Menoufia University, Egypt
Email: ashraf.elsisi@ci.menofia.edu.eg

***Abstract*—** Recently semantic web services represent the most technology developed for machine to machine interaction. The problem of discovering and selecting the most suitable web service represents a challenge for semantic web services. In this paper performance evaluation of mapping algorithm from web services annotations (WSDL) to semantic annotations (OWL-S) based on ontology search engine is presented. During mapping process primitive type remains without change. The complex type are converted to OWL ontology by extracted them and passing to ontology search and standardization process without need of conversion into temporary ontology. The keywords extracted in the linguistic search phase and are extended using word net. The mapping algorithm and its modification are implemented in Java and evaluated by 310 files WSDL. The output results of two algorithms are identical. But the proposed modified algorithm is faster than mapping algorithm.

***Index Terms*—** Semantic Services, WSDL, OWL-S, Ontology Search Engine.

I. Introduction

Semantic web services are a new paradigm for distributed computing [1]. Semantic services are a component of the semantic web. Semantic web uses mark-up that makes data machine-readable. Using semantic web, computers will be able to search, process, integrate and present the content of resources in a meaningful manner [2]. Semantic web services provides Internet that enables one application running on the home computer searches, selects and invokes web services. Web service architecture includes three roles, service provider, service registry and service consumer [3]. The Web Services Description Language (WSDL) is an XML language for describing traditional web services [4]. WSDL can be seen as an interface definition language, defines the interface of the service in terms of its inputs and outputs. WSDL tells the client how to invoke the service, the location of the service, the protocol to be used for invocation, and the format of the messages to be sent to the service. Negotiation process based on Service Level agreement (SLA) between requester and provider must be executed to select the best service [5]. The semantic web allows the representation and exchange of information in a meaningful way, facilitating automated processing of descriptions on the web [6-8]. Annotations on the semantic web express links between information re-sources on the web and connect information resources to formal terminologies; these connective structures are called ontologies [9]. Ontology is a formal specification of a shared conceptualization. The main thread of ontology is the study of entities and their relations. Semantic web service life cycle consists of Advertisement, Discovery, Selection, Composition and Invocation. In the "Advertisement", the srevcie provider publishes the benefits of the service and how to use it [10]. In the "Discovery" the list of available services that satisfy the user query is defined. The "Selection" selects the most suitable web service. The "Composition" process integrates the selected web services into a compound service. In the last step single or compound service is invoked based on preconditions and all the inputs. The OWL-S is the most important semantic web service description language that developed by W3C organization. OWL language describes the properties and functions of Web service ontology. OWL-S use three types of ontology to describe services: services profile, service model, and service grounding. The mapping algorithm [11], is used to converting the web services annotations (WSDL) to semantic annotations (OWL-S). The mapping algorithm is modified to decrease the execution time of algorithm. There are two types of WSDL, the first is primitive type remains without change during mapping. But the other type is complex type that is converted to OWL ontology. The modification is no need conversion into temporary ontology during extract complex type process. The mapping algorithm and proposed modification are implemented and tested with 310 WSDL files that included in OWLS-TC. The output results of mapping algorithm and our modification are the same. But our proposed modification is faster. Both algorithms still use the minimum number of concepts compared to the related work. The organization of this paper as following, section 2 gives an overview about related work of mapping WSDL to OWL-S. In section 3 the description of mapping from WSDL to OWL-S based on ontology search and standardization engine is presented in details. The proposed modification of the mapping from WSDL to OWL-S based on ontology search and standardization engine is described in section 4. Implementation and experimental results of mapping algorithm and modified mapping algorithm are presented in sections 5. Finally, some conclusions and future work are put forward in Section 6.

II. Related Work

Much research work has been done in mapping WSDL to OWL-S [13-17]. In [13] introduce a mapping tool named Automated Semantic Service Annotation with Machine Learning (ASSAM). ASSAM converts the WSDL file to OWL-S file. ASSAM tool has the many

limitations. ASSAM main topic is machine learning for web service classification. The tool doesn't provide organization for ontologies used. This leads to a large number of concepts. The process of finding the possible list of concepts doesn't depend on the meaning. It depends on text search. List of possible concepts provides for user to choice are not ranked by importance. In [14] conversion from WSDL to OWL-S is done for web service annotation framework. It annotates WSDL descriptions of the services with metadata from relevant ontologies. First it matches concepts of WSDL and OWL. This framework proposes to bridge the gap by converting each of them to a common representation called schema Graph. A schema Graph is simply a graph or tree representation of the XML. Once the schema Graphs have been created, matching algorithms are executed on the graphs in order to determine similarities. Concepts of the schema graph are matched; the concept that has the highest matching score is the best mapping choice. In [15] WSDL2OWL-S provides a transition between WSDL and OWL-S. In this work, conversion of all XSD complex types and generate concepts and properties for each type. This is a blind conversion of XSD types to concepts. Also, there is no dependency or relationships between these concepts. Using this way for conversion, a lot of concepts will be used and semantic web meaning will be missed. In [16] semantic web service discovery using natural language processing techniques has introduced. Only it is match processing between user needs and semantic web service description in Web Service Modeling Ontology (WSMO) formalism.

III. Mapping from WSDL to OWL-S Based on Ontology Search Engine

The solution of the large unrelated concepts set problem is proposed in mapping algorithm [11]. The Local Ontology Repository (LOR) and Ontology Search and Standardization Engine (OSSE) model are used to solving this problem. Description of mapping algorithm, types converter, local ontology repository, and ontology search and standardization engine will be in the following subsections.

A. Mapping algorithm

Mapping algorithm maps WSDL to OWL-S. OWL-S concentrates on the description of functional and non-functional properties of semantic web service. WSDL contains types, operations input and output, service name and binding. WSDL Type must be converted to ontology. Inputs and outputs of WSDL are mapping to input and output in OWL-S. Service name in WSDL became the service name of OWL-S. The mapping algorithm has two phases; automated phase and manual phase. Steps of mapping algorithm are described in Fig. 1. Automated phase parses WSDL service file and extract information fill the required properties that construct the service profile of OWL-S. WSDL does not contain all data required to build OWL-S especially non-functional data. In the manual phase, the results of automated phase are presented to service provider and select most related ontology concepts for each data type.

Input: WSDL file

Output: Service profile file (OWL-S)

First :automatic phase

1-Conversion of WSDL XSD types to OWL ontology.

2-Extract service name and description.

3-Extract service inputs and outputs.

4- Service provider make choice and accept one result

Second :Manual phase

5- Add service preconditions and effects.

6- Add service non-functional properties.

7- Create service profile file (part of OWL-S).

Fig. 1. Mapping Algorithm of WSDL to OWL-S Service Profile.

B. Types converter

WSDL types conversions is the most important step in mapping algorithm, typically XSD types There are two XSD types: primitive (simple) XSD types and complex XSD types. The simple XSD is converted to OWL-S as it is. But the complex one is translated into OWL ontology concepts using type converter in [13]. This method for XSD type's converter generates concepts that are totally unrelated. The generated ontologies with this method lead to missing the meaning of semantic web and ontology. In [11] the author introduce the method based on LOR and OSSE. LOR is a repository designed for storing ontologies, concepts, properties and relationships. OSSE engine searches in LOR for related ontology to find a ranked list of the most related ontologies that already exist. Then, ask the service provider to choose the most suitable ontology concept. If the service provider does not accept any of the already existing ontologies, the system adds this type as a new concept to LOR and relate it to existing ontologies. Then, provide it to the provider as the most related ontology. Type's converter converts WSDL XSD type to related OWL ontology. Steps of types converter described in Fig. 2. The XSD types are extracted one by one from the WSDL file. The first question is "Is this primitive or complex type?". If it is primitive the conversion process does not work and keeps it as it is. On the other hand, if it is a complex type the converter starts a process to find the most suitable OWL ontology. In some rare cases, OSSE fails to find any related ontologies. In this case, the system inserts the temporary ontology to the local ontology repository using the insertion methodology into LOR.

C. Local ontology repository

The main challenge that prevents the semantic web from achieving its goals is the problem of standardization. Standardization is the problem of creating related ontology with truly reasoning .The key point of any standardization research is how to use the ontology concept to define different data types in the system. In 2004, OWL becomes W3C official language [18].

Input: WSDL XSD type
Output: OWL ontology
If XSD type is complex **Then**
Create temporary ontology with one concept and many properties.
Call ontology Search and Standardization Engine (OSSE).
If calling of OSSE has result **Then**
Provider selects the most suitable ontology from ordered ontologies.
Else
Insert temporary ontology into Local Ontology Repository
and go to step 3.
End If
Else
Use primitive XSD types.
End If

Fig. 2. Conversion of WSDL XSD types to OWL ontology

Many projects focus on how to collect the largest number of ontologies to construct some ontology libraries. Swoogle can be considered as the best repository for online ontologies as it has catalogued over 10,000 semantic web documents and has indexed metadata about their classes, properties, and the relationships among them. In general, there is a drawback of Swoogle which is; the absence of any clear methodology of how to insert a new ontology into the library or how to update the structure of the already existing ontologies. The reasons of construction of LOR are standardization, definition of methodology of insertion and overcoming the problem of unrelated ontology of the ASSAM [13]. LOR is built based on the object-oriented paradigm which can be considered as one of the most remarkable approaches to describe the ontologies [19-20]. Repository structure as shown in Fig. 3 contains tables for concepts, properties, relations ships and other statistics. LOR uses ontology relations and keywords statistics in OSSE for ontology searching. Searching using OSSE depends on structure search and linguistic search (keywords search). The main target of inserting a new ontology to the repository is to collect two types of data: Data concerning the logical structure and Keywords relevant to the new ontology. These two types of data are used by OSSE. After collecting the required data, they are saved into the repository file and database system. The inserting methodology structure is shown in Fig. 4. Two concurrent processes collect these two types of data: the first process is called "Structure Extraction", during this process, data concerning ontology concepts (classes), properties and relationships are collected. The second process is called "Keywords Extraction" where text-mining techniques will be used to extract the keywords from the whole OWL file. This process consists of five steps:

1) Tokenization: is the process of splitting the text into very simple tokens such as numbers, punctuation and words of different types. In this context, we concentrate on word tokens only.

2) Lemmatization: is the process of reducing inflectional forms or process of reducing word to a common base form.

3) Removing stop words: is the process of removing the extremely common words which are called "stop words" such as (a, an, am, will, he, she, their, … etc).

4) Then, removing any word that appears in the keywords list of the languages used to write the ontology which contains words such as XSD, OWL, RDF, and RDFS.

5) Finally, the term frequency (TF) computed for each word in the list.

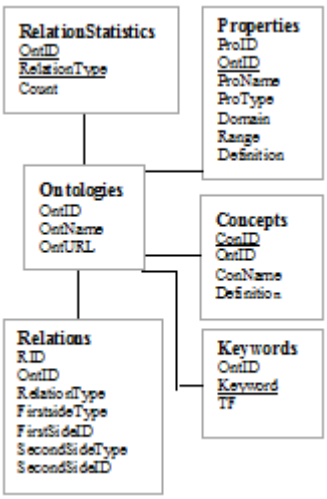

Fig. 3. Structure of repository

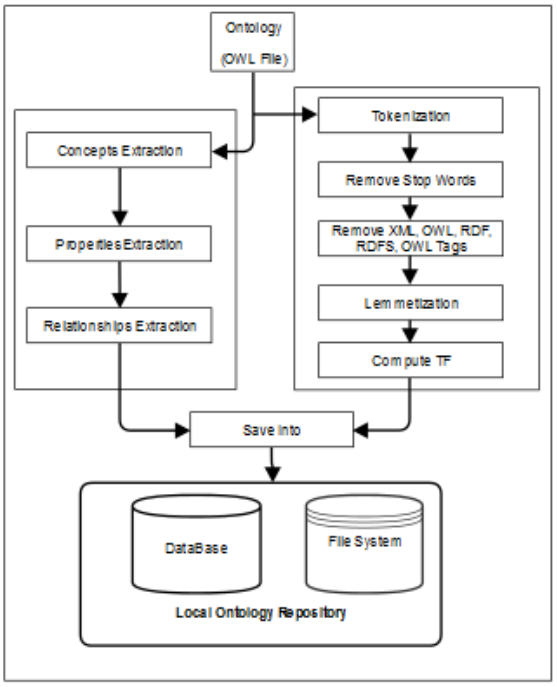

Fig. 4. Inserting new ontology methodology.

D. Ontology Search and Standardization Engine

The main challenge that faces the development of semantic web is the problem of ontology standardization. OSSE engine developed for standardization. OSSE model has three phases; linguistic search, structural refining and statistical ranking. The input is a required concept with certain features (properties) and the output is an ordered

list of possible related ontologies. The concept request is formalized in the form of a temporary OWL ontology that contains one concept with many properties.

1) Linguistic search

OSSE uses the text mining techniques and keywords extraction, to extract the most important keywords in the concept request. Steps of linguistic search are described in Fig. 5. This list of keywords (and its related words) is used to search in the local ontology repository to get a list of possible related ontologies. The list of ontologies is arranged based on the keywords in terms of term frequency. For each ontology; the summation of term frequency values of each keyword that belongs to the concept request keywords list and belongs to the ontology at the same time is computed. In some cases, OSSE fails to find any related ontology in the local repository. Therefore, it asks Swoogle for help to find some OWL related ontologies. If there are results for this search, OSSE downloads the top five ontologies. If the service provider accepts any of these downloaded ontologies, the system automatically inserts this ontology to the local ontology repository using the inserting methodology. It is important to note that the inserting process is performed after finishing the process of choosing the suitable concept. In some rare cases, searching in Swoogle returns no results. In this case, OSSE inserts the temporary ontology into the local repository using the inserting methodology.

Input: Temporary ontology

Output: possible ontology list

Keywords extraction from temporary ontology.
Search on these keywords in Local Ontology Repository (LOR).
If searching on LOR has result **then**
  Get list of these ontologies.
  Order this ontologies according to TF and consider it possible ontology list.
Else
  Search for these keywords using Swoogle.
  If search on Swoogle has result **then**
    Download the first five ontology files and consider it possible ontology list
  Else
    Insert temporary ontology into LOR and go to step 1.
  End if
End if

Fig. 5. Linguistic search

2) Structural refining

Structural refining is the second phase in OSSE. It refines the list that is produced by the linguistic search. This refining is performed by using each ontology searching in the list to find any concept related to the required concept. If OSSE does not find any related concept in a particular ontology, this ontology is deleted from the possible ontologies list. If all ontologies in related ontology lists are deleted, the system adds the temporary ontology as the gotten ontology. Structural refining described in Fig. 6. Structure refining phase interested in ordering related ontologies according to concepts relationships.

3) Statistical refining

The choices of the service provider are stored in the "Concepts Mapping History" database. These data are used by OSSE to re-rank the possible ontology list. If there are two ontologies with the same rank in the previous step, OSSE uses this historical data to know the most preferred ontologies for the services providers. Statistical refining phase is described in Fig. 7.

Input: ordered ontology list

Output: reordered ontology list

For each ontology in ordered ontology list **do**
  If there is identical concept **then**
    Reorder ontology list
  Else
    If there is identical super concept **then**
      Reorder ontology list
    Else
      If there is identical sub concept **then**
        Reorder ontology list
      Else
        If there is identical neighbor concept **then**
          Reorder ontology list
        Else
          Eliminate from list
        End If
      End If
    End If
  End If
End For

Fig. 6. Structural Refining

Input: reordered ontology list

Output: ordered ontology list

Get history reordering ontologies from previous concepts mapping history.
Order ontology list according to this history.

Fig. 7. Statistical Ranking

After these three steps, OSSE has an ordered possible ontologies list for the concept request. This list is presented to the service provider, who chooses the most suitable ontology from his point of view. OSSE has three main features that should be tested. The first one is its capability to find the matched ontology for an already existing concept. Second one is its ability to access Swoogle web service to download suitable ontologies for a requested concept that does not belong to the local

repository. Third one is its ability to extract the suitable concept-to-concept relationships. Three experiments are designed to test these features:

Experiment 1:

Goal: Test the linguistic search step of OSSE.

Request for "Car" element as a concept (without creating temp ontology).

Procedure:

1. As we have keyword(s) extend these keywords using WordNet ➔ auto, automobile, machine, motorcar, gondola, rail car.
2. Search "local ontology Repository".

Results: The search yields a list of six possible related ontologies.

Experiment 2:

Goal: Test the usage of external search, i.e. Swoogle web service to download required ontologies.

Preparations: before starting this experiment we delete the six related ontologies for the concept "Car"

Procedure:

1. Request for "Car" element as a concept (without creating temp ontology).
2. As we have keyword(s) extend these keywords using WordNet ➔ auto, automobile, machine, motorcar, gondola, rail car.
3. Search "local ontology Repository".

Results: The search yields no results. So, OSSE uses Swoogle to download the top five ontologies.

The URL of the top ontology is http://morpheus.cs.umbc.edu/aks1/ontosem.owl

Experiment 3:

Goal: Test the ability of OSSE to extract the suitable concept-to-concept relationships.

Procedure:

1. Request for "Address" element as a concept that has properties country, city, street, and building (without creating temp ontology).
2. As we have keyword(s) extend these keywords using WordNet ➔ address, destination, country, state, nation, area, city, metropolis, urban, street, building, edifice, construction.
3. Search "local ontology Repository".
4. Perform the structural refining

Results:

- OSSE linguistic search yields a list to 19 possible related ontologies.
- After structural refining this list contains only 7 ranked ontologies and the top ranked ontology is (portal. owl)
- By manual check this ontology has a concept called "address" and has 9 properties (Area, building, city-or-village, country, number, postcode, pretty-label, region and street).

IV. Proposed Modification

The The proposed modification is concerning to types converter of the mapping algorithm in [11]. After checking of WSDL complex types, extracted type and its properties without creating temporary ontology with one concept and many properties can be used. Proposed modification is described in Fig. 8. Modification of types converter will reflect in modification of OSSE model first phase which is linguistic search. Modification of Linguistic search is clarified in Fig. 9. Modification in this phase of search reduces time spends on creating temporary ontology, extracting concept and its properties from this OWL file.

Input: WSDL XSD type
Output: OWL ontology
If XSD type is complex **Then**
Extract type and its properties.
Call ontology Search and Standardization Engine (OSSE).
If calling of OSSE has result **Then**
Provider selects the most suitable ontology from ordered ontologies.
Else
Insert temporary ontology into Local Ontology Repository
and go to step 3.
End If
Else
Use primitive XSD types.
End If

Fig. 8. Proposed Modification of Types Converter

Input: Temporary ontology
Output: possible ontology list
Search on these keywords in Local Ontology Repository (LOR).
If searching on LOR has result **then**
Get list of these ontologies.
Order this ontologies according to TF and consider it possible ontology list.
Else
Search for these keywords using Swoogle.
If search on Swoogle has result **then**
Download the first five ontology files and consider it possible ontology
list
Else
Insert temporary ontology into LOR and go to step 1.
End if
End if

Fig. 9. Proposed Modification of Linguistic Search

V. Implementation and Experimental Results

The mapping algorithm based OSSE and proposed modification of mapping algorithm based OSSE are implemented by using java programming, under windows 7, running on Intel core i3 M330 2.13GHz. OWL ontology files stored on a file system. Relationships of concepts stored in a database designed for this purpose. For testing purposes, we use some of WSDL files to evaluate modification results. These WSDL files belongs to OWLS-TC [11], used in studying the behavior of

mapping algorithm and modified mapping algorithm. For preliminary results we validate the modified mapping algorithm by using 310 WSDL files which are included in OWLS-TC. The complete conversion of 310 WSDL to OWL-S was done automatically. But we record 31 measurement for complete conversion, where the sequence of our measurements are (10, 20, 30 . . . 310).

```
<? xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns="http://127.0.0.1/wsdl/Hotel"
..... XML Namespace
targetNamespace="http://127.0.0.1/wsdl/Hotel">
<wsdl:types>
<xsd:schema version="OWLS2WSDL Sun Jun 07 19:36:17 CEST 2009"
targetNamespace="http://127.0.0.1/wsdl/Hotel"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:annotation>
<xsd:documentation source="Translation (OWL2XSD-SimpleType) of
http://127.0.0.1/ontology/travel.owl#Hotel"/>
</xsd:annotation>
<xsd:element name="Hotel" type="HotelType"/>
<xsd:simpleType name="HotelType">
<xsd:restriction base="xsd:string"/>
</xsd:simpleType>
</xsd:schema>
</wsdl:types>
<wsdl:message name="get_HOTELResponse">
<wsdl:part name="_HOTEL" type="tns:HotelType">
</wsdl:part>
</wsdl:message>
<wsdl:message name="get_HOTELRequest">
</wsdl:message>
<wsdl:portType name="HotelSoap">
<wsdl:operation name="get_HOTEL">
<wsdl:input message="tns:get_HOTELRequest">
</wsdl:input>
<wsdl:output message="tns:get_HOTELResponse">
</wsdl:output>
</wsdl:operation>
</wsdl:portType>
..... Bonding and Port.....
</wsdl:service>
</wsdl:definitions>
```

Fig. 10. Input WSDL file in both cases (hotel service)

By using the same WSDL input file for mapping algorithm based OSSE and modified mapping algorithm shown in Fig. 10. We present one of these WSDL files (HotelService.wsdl). The function of this service is to get hotel. We get the output file for each case before modification (mapping algorithm) and after modification (modified mapping algorithm) as shown in Fig. 11 and Fig. 12 respectively. We find that the output is similar in both cases. Fig. 13 shows the comparison between WSDL2OWL-S tool [15] and the modified mapping algorithm. Fig. 13 presents number of concepts used in each number of services. We notice that number of concepts increased in both cases of modified algorithm and WSDL2OWL-S. But number of concepts increased with high rate in WSDL2OWL-S. For example, in case number of services equal (110), the average number of concepts become 310 in case of WSDL2OWL-S, and in case of modified algorithm average number of concepts less than 100. It is important to notice that discovery process affected with increasing number of concepts. Where number of concepts increase in one system and less in another system, this means that the second system contains related concepts. In the case of related concepts, this means no attention about relationships between the added concepts and the already existing. Adding unrelated concepts lead to the main drawback of mapping mechanism of discovery process. So during discovery process, matching becomes faster and accurate when number of concepts decreased and related in a meaningful way. So our modification and original mapping algorithm based on OSSE are using minimum number of concepts compared to WSDL2OWL-S.

Also, we evaluate modified algorithm by computing execution time spend in mapping algorithm and modified mapping algorithm. For preliminary results we use 25 WSDL file that included in OWLS-TC. Then average time spends in mapping algorithm and modified mapping algorithm is recorded for each file. Average time for mapping algorithm and modified mapping algorithm is clarified in Fig. 14. For example in WSDL file 15, average time for mapping algorithm 900 milliseconds.

And in case of modified mapping algorithm average time is 290 milliseconds. In case of modified mapping algorithm time decreases leading to improved performance in general.

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
<rdf:RDF >
<owl:Ontology rdf:about="">
.....  ontology imports ......
</owl:Ontology>
<service:Service rdf:ID="_HOTEL_SERVICE">
<service:presents rdf:resource="#_HOTEL_PROFILE"/>
<service:describedBy rdf:resource="#_HOTEL_PROCESS_MODEL"/>
<service:supports rdf:resource="#_HOTEL_GROUNDING"/>
</service:Service>
<profile:Profile rdf:ID="_HOTEL_PROFILE">
<service:isPresentedBy rdf:resource="#_HOTEL_SERVICE"/>
<profile:serviceName xml:lang="en">
WorldwideHotelInfoService
</profile:serviceName>
<profile:textDescription xml:lang="en">
This service returns information of all famous hotels of the world.
</profile:textDescription>
<profile:hasOutput rdf:resource="#_HOTEL"/>
<profile:has_process rdf:resource="_HOTEL_PROCESS" /></profile:Profile>
<process:ProcessModel rdf:ID="_HOTEL_PROCESS_MODEL">
<service:describes rdf:resource="#_HOTEL_SERVICE"/>
<process:hasProcess rdf:resource="#_HOTEL_PROCESS"/>
</process:ProcessModel>
<process:AtomicProcess rdf:ID="_HOTEL_PROCESS">
<process:hasOutput rdf:resource="#_HOTEL"/>
</process:AtomicProcess>
<process:Output  rdf:ID="_HOTEL">
<process:parameterType rdf:resource="http://127.0.0.1/ontology/travel.owl#Hotel" />
 <rdfs:label></rdfs:label>
</process:Output>
<grounding:WsdlGrounding rdf:ID="_HOTEL_GROUNDING">
<service:supportedBy rdf:resource="#_HOTEL_SERVICE"/>
</grounding:WsdlGrounding>
</rdf:RDF>
```

Fig. 11. Service Profile of Mapping Algorithm

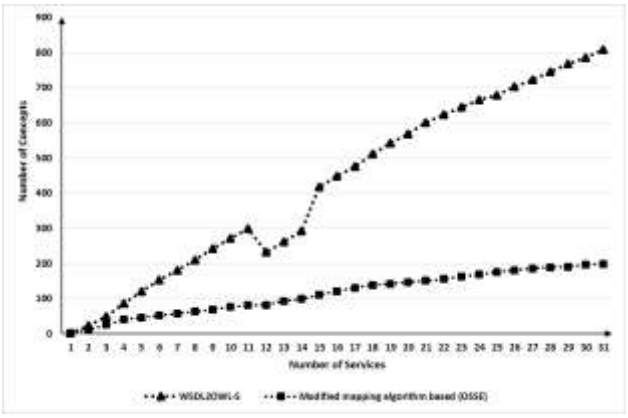

Fig. 13. Number of Concepts for WSDL2OWL-S and Modified Algorithm

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
<rdf:RDF >
<owl:Ontology rdf:about="">
..... ontology imports ......
</owl:Ontology>
<service:Service rdf:ID="_HOTEL_SERVICE">
<service:presents rdf:resource="#_HOTEL_PROFILE"/>
<service:describedBy rdf:resource="#_HOTEL_PROCESS_MODEL"/>
<service:supports rdf:resource="#_HOTEL_GROUNDING"/>
</service:Service>
<profile:Profile rdf:ID="_HOTEL_PROFILE">
<service:isPresentedBy rdf:resource="#_HOTEL_SERVICE"/>
<profile:serviceName xml:lang="en">
WorldwideHotelInfoService
</profile:serviceName>
<profile:textDescription xml:lang="en">
This service returns information of all famous hotels of the world.
</profile:textDescription>
<profile:hasOutput rdf:resource="#_HOTEL"/>
<profile:has_process rdf:resource="_HOTEL_PROCESS" /></profile:Profile>
<process:ProcessModel rdf:ID="_HOTEL_PROCESS_MODEL">
<service:describes rdf:resource="#_HOTEL_SERVICE"/>
<process:hasProcess rdf:resource="#_HOTEL_PROCESS"/>
</process:ProcessModel>
<process:AtomicProcess rdf:ID="_HOTEL_PROCESS">
<process:hasOutput rdf:resource="#_HOTEL"/>
</process:AtomicProcess>
<process:Output rdf:ID="_HOTEL">
<process:parameterType rdf:resource="http://127.0.0.1/ontology/travel.owl#Hotel" />
 <rdfs:label></rdfs:label>
</process:Output>
<grounding:WsdlGrounding rdf:ID="_HOTEL_GROUNDING">
<service:supportedBy rdf:resource="#_HOTEL_SERVICE"/>
</grounding:WsdlGrounding>
</rdf:RDF>
```

Fig. 12. Service Profile of Modified Mapping Algorithm

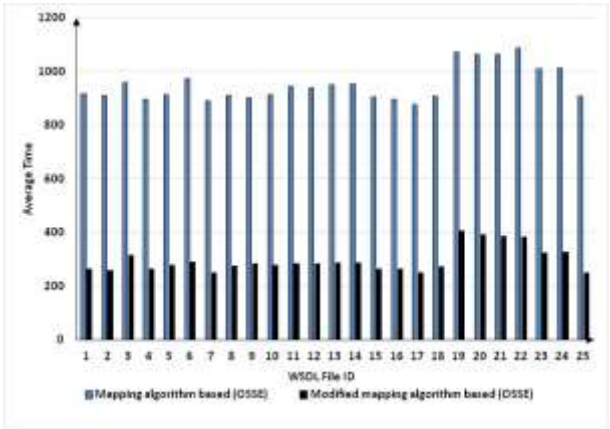

Fig. 14. Time Comparison Before and After Modification of Mapping Algorithm

VI. Conclusion and Future Work

In this paper performance evaluation of mapping algorithm from WSDL to OWL-S based on ontology search engine is presented. Complex type of WSDL must be converted to OWL ontology. The step of keywords extracted from temporary ontology created from conversion algorithm is the same as keywords extracted directly from WSDL complex type. This needs more time and programming effort spent in creating temporary OWL ontology in WSDL conversion to OWL ontology and keywords extraction from this temporary ontology. So this step can be modified, in the conversion of WSDL to OWL ontology, after extracting type and properties from complex type, these keywords passing to first phase in ontology search and standardization engine. Also, in linguistic search first phase in ontology search and standardization engine, concept request and temporary ontology is replaced with this extracted type and its properties. Then, WordNet extends these keywords with its synonyms. Based on our modification, extracting elements of XSD complex types, to create a temporary ontology of this element and its properties not repeated anymore. The extracted elements can be used directly. This will decrease a time for the proposed modification system of mapping and standardization approach. Mapping algorithm and modified algorithm are implemented in Java. Evaluation of proposed modification is done using 310 WSDL files which are included in OWLS-TC v3.0 and mapped using the proposed modified algorithm. Output results of both mapping algorithm and modified mapping algorithm are identical. Also both algorithms still uses minimum number of concepts compared as the related work. But the modified mapping algorithm has less execution time, which will have a good effect on the discovery process. Other formalisms of semantic web services such as WSMO, WSDL-S and SWSF will be considered in future work.

References

[1] F.H. Abanda, J.H.M. Tah, R. Keivani, " Trends in built environment semantic Web applications: Where are we today?", Expert Systems with Applications, 40, pp. 5563–5577, 2013.

[2] H. N. Talantikite, D. Aissani, and N. Boudjlida, "Semantic Annotations for Web Services Discovery and Composition," Computer Standards & Interfaces, vol. 31, pp. 1108-1117, 2009.

[3] Lixin Zhou,"An Approach of Semantic Web Service Discovery", International Conference on Communications and Mobile Computing (CMC), pp. 537 – 540, 2010.

[4] Matjaz B. Jurica, Ana Sasa, Bostjan Brumen, , Ivan Rozman, "WSDL and UDDI extensions for version support in web services", The Journal of Systems and Software, 82, pp. 1326–1343, 2009.

[5] M. Brian Blakea, David J. Cummings, Ajay Bansal, Srividya Kona Bansal, "Workflow composition of service level agreements for web services", Decision Support Systems 53, pp. 234–244, 2012.

[6] D. Fensel and M. Kerrigan, Modeling Semantic Web Services: The Web Service Modeling Language: Springer, 2008.

[7] B. Fazzinga, G. Gianforme, G. Gottlob, and T. Lukasiewicz, "Semantic Web Search Based on Ontological Conjunctive Queries," Web Semantics: Science, Services and Agents on the World Wide Web, vol. 9, pp. 453-473, 2011.

[8] Nikolaos Loutas, Vassilios Peristeras, Konstantinos Tarabanis, "Towards a reference service model for the Web of Services", Data & Knowledge Engineering, 70, pp. 753–774, 2011.

[9] E. Hyvönen, K. Viljanen, J. Tuominen, and K. Seppälä, "Building a National Semantic Web Ontology and Ontology Service Infrastructure–the FinnONTO Approach," in The Semantic Web: Research and Applications, ed: Springer, 2008, pp. 95-109.

[10] John Domingue, Liliana Cabral, Stefania Galizia, Vlad Tanasescu, Alessio Gugliotta, Barry Norton, Carlos Pedrinaci, "IRS-III: A broker-based approach to semantic Web services", Web Semantics: Science, Services and Agents on the World Wide Web, 6 pp.109–132, 2008.

[11] T. Farrag, A. Saleh, and H. Ali, "Towards SWSs Discovery: Mapping from WSDL to OWL-S Based on Ontology Search and Standardization Engine", IEEE Transactions on Knowledge and Data Engineering, vol. 25, no. 5, pp. 1135-1147, May 2013.
[12] Gao Ting; Wang Haiyang; Zheng Naihui; Li Fei, "An Improved Way to Facilitate Composition-Oriented Semantic Service Discovery", International Conference on Computer Engineering and Technology, pp. 156 – 160, 2009.
[13] A. Heß, E. Johnston, and N. Kushmerick, "Assam: A tool for semi-automatically annotating semantic web services," in The Semantic Web–ISWC 2004, ed: Springer, 2004, pp. 320-334.
[14] A. A. Patil, S. A. Oundhakar, A. P. Sheth, and K. Verma, "Meteor-s web service annotation framework," in Proceedings of the 13th international conference on World Wide Web, 2004, pp. 553-562.
[15] M. Paolucci, N. Srinivasan, K. P. Sycara, and T. Nishimura, "Towards a Semantic Choreography of Web Services: from WSDL to DAML-S," in ICWS, 2003, pp. 22-26.
[16] J. Sangers, F. Frasincar, F. Hogenboom, and V. Chepegin, "Semantic Web Service Discovery Using Natural Language Processing Techniques," Expert Systems with Applications, Vol. 40, Issue 11, pp. 4660-4671, 2013.
[17] Montserrat Batet, David Isern, Aida Valls, David Sánchez, "Ontology-based semantic similarity: A new feature-based approach", Expert Systems with Applications 39, pp. 718–7728, 2012.
[18] Oghabi G., Bentahar J., Benharref, A., "On the Verification of Behavioral and Probabilistic Web Services Using Transformation", International Conference on Web Services (ICWS), pp. 548 – 555, 2011.
[19] Guanghui Yang, Junkang Feng, "Database Semantic Interoperability based on Information Flow Theory and Formal Concept Analysis", I.J. Information Technology and Computer Science, Issue 7, pp. 33-42, 2012.
[20] Vishal Jain, Mayank Singh, "Ontology Based Information Retrieval in Semantic Web: A Survey", I.J. Information Technology and Computer Science, Issue 10, pp. 62-69, 2013.

Author's Profiles

Ashraf B. El-Sisi received the B.Sc. and M.Sc. in Electronic Engineering and Computer Science Engineering from Menoufia University, Faculty of Electronic Engineering in 1989 and 1995, respectively and received his PhD in Computer Engineering & Control from Zagazig University, Faculty of Engineering in 2001. His current research interest includes cloud computing, privacy preserving data mining, and intelligent systems.

How to cite this paper: Ashraf B. El-Sisi,"Fast Mapping Algorithm from WSDL to OWL-S", International Journal of Information Technology and Computer Science(IJITCS), vol.6, no.9, pp.24-31, 2014. DOI: 10.5815/ijitcs.2014.09.03