ISSN:0975-9646

Toward SWSs Discovery: Mapping from WSDL to OWL-S Based on Ontology Search and Standardization Engine

R. Jamgekar#1, N. Sawant*2, P. Bansode#3

Dept. of CSE, SKN Sinhgad College of Engineering solapur, solapur, India

Dept. of CSE, SKN Sinhgad College of Engineering Korti, Pandharpur, Maharashtra

**Dept. of CSE, SKN Sinhgad College of Engineering Korti, Pandharpur, Maharashtra*

Abstract— **Computer easily recognize Semantic Web Services (OWL-S) instead of web services like WSDL. we are going to convert web services to semantic web services so discovering and selecting the services get easier. Ontology repository and standardization engine are basic steps for this conversion. The proposed system presents a distributed Web service discovery architecture This architecture is based on the concept of distributed shared space and intelligent search among a subset of spaces. It allows the publishing of Web service descriptions as well as to submit requests to discover the Web service of user's interests.**

Keywords— **WSDL, OWL, Mapping, OSSE**.

I. Introduction

Semantic Web Services (SWS) [6], are easily recognize by machine. The Semantic Web extend human-readable web to machine-readable form so Computer can search, process, integrate, and present the content of the resources in a semantic way. When we get any service request we can optimize results in Discovery process [7] as well as distinguished by its service repository based on the advertisements of semantic and non-SWS. The advertisement of services improves the speed and quality of the discovery process. The organization of discovery system mainly divided in two steps that is firstly create database, and second one discovery process.

A. Database Creation:-

In Database Creation the semantic services are registered and then follows below steps: The WSDL [8] of the already existing WSs mapped into a semantic one OWL-S [9]. Register all the services whether they belong to WSs or SWSs in the Unclassified Profiles database. Classify these data into clusters to make the discovery easier and faster.

B. Discovery Process:-

In Discovery Process receive the user request of a certain service and then follows below steps: Search into the database for the suitable results. By Ranking the results enhance the user selections. For conversion WSDL to OWL-S redefine the conventional web services using semantic markups. After this process all owl's files are stored in repository and then apply ontology search and standardization engine (OSSE) that helps in the standardization process. OSSE's function is based on searching for a suitable ontology in the ontology repository.

Objective: To Discover any service in web services is time consuming. To overcome this drawback, we introduce a distributed web service discovery architecture this help to reduces the communication overhead and the result obtained is more precise.

Fig.1 steps to deal with any web service (WS):

1. Advertisement aims to publish information about the benefits of the service and how to use it.
2. Discovery aims for finding the list of services that can possibly satisfy the user requirements.
3. Selection witch specify select most suitable WS.
4. Composition integrates the selected WSs into a compound process.
5. Invocation that invokes a single WS or compound Process by providing it with all necessary inputs .

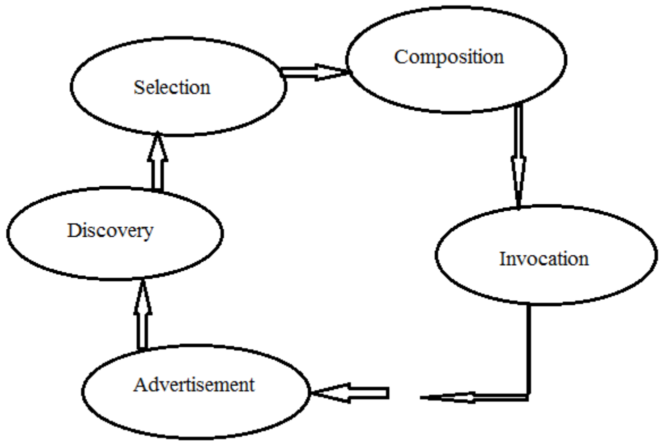

Fig.1 Web services life cycle.

II. Related Work

ASSAM (Automated Semantic Service Annotation with Machine Learning) tool. This tool generate OWL-S file. While using This tool few problems has to face like an organization for the available ontologies. Because when we use two classes this yield huge number of ontologies so organization problem is arise. The tool outputs unordered list So, the choosing class is difficult for any user. This paper is meant about classification of web services. So, ASSAM can be considered as a novel web services classification tool instead of mapping tool.

III. IMPLEMENTATION

OSSE IMPLEMENTAION:-

It does not make any sense to find the same definition for a certain concept repeated more than one time in the system. This does not include the case when there are different definitions for the same concept and each of them adds new information. For instance, one ontology might define the class "Animal" equivalent to "Living thing AND NOT Human." Another ontology might define the class "Animal" as "the Union of Herbivores AND Carnivores." Both definitions are correct, since they highlight different aspects or properties for the class animal, and they cannot be considered mutually exclusive. It is illogic to define a concept isolated from the already existing concepts definitions. In other words, it is better from standardization point of view—to try to find a relation between the new concept definition and the already existing ones. This OSSE engine takes in consideration as mentioned before. The input is a concept with certain properties and the output is an ordered list of ontologies. The name of the required concept is the primary supplied information used by OSSE. The engine main stages are described below:

A. Linguistic search:

OSSE uses the text mining techniques to extract keywords in the concept request. Then, it tries to find the synonyms and related words to expand our list by the aid of WordNet which includes over 30,000 word. This list of keywords is used to search in the local ontologies repository to get a list of related ontologies. The list of ontologies is arranged based on the keywords that they contain 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.

This summation represents a measure of the degree of the ontology linguistic relevance (OLR). Degree of OLR can be

calculated by where NCK is the total number of concept request keywords. The function getTF() is used to get the previously computed TF value of the keyword (K) in the tested ontology. The TF value is computed for each ontology keyword during the process of new ontology insertion. Due to large number of threshold value for OLR.

$$\text{OLR}=\sum_{i=0}^{i=\text{NCK}} \text{getTF}(ki)$$

In some cases, OSSE fails to find any related ontology in the local repository. Then, it asks Swoogle for help to find some OWL. OSSE downloads the top five ontologies. If the service provider accepts any of these downloaded ontologies, the system automatically inserts ontology to the local ontology repository using the inserting methodology. This process grantees that our ontology repository is extended to satisfy the service providers needs without changing the features of our repository. Downloading and inserting the selected ontology in the repository is a process that consumes undetermined time. This time depends on the downloading speed and the degree of ontology complexity. 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 with no results. OSSE inserts the temporary ontology into the local repository using the inserting methodology. A long list of related ontologies is expected.

B. Structural refining:

OSSE refines the list produced by the linguistic search. This refining is performed by searching in each ontology 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. "Data concerning the logical structure" which are collected using the inserting methodology, are considered to be the base of the structural refining. The ontology is checked to answer four serial questions that correspond to the four possible concept-to-concept relationship as shown in Fig. 3. When we have a "yes" answer to any question of them, OSSE stops the checking process of the ontology and assigns a rank value for it. "Identical" relation has the highest ranking value followed by "Super," "Sub," and "Neighbor" relations in order from highest to lowest ranking value. After completing the process of checking the ontologies list, OSSE reorders the ontologies list according to the computed ranking value.

C. Statistical refining:

Concepts Mapping History database. These data are used to rerank the possible ontologies list. If there are two ontologies with the same rank, OSSE uses historical data to know the most preferred ontologies for the services providers. 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. His choice recorded in the "Concepts Mapping History" database. The process of ontology editing causes some changes in the database and the file system of the local ontologies repository.

OSSE depends on local ontology repository to retrieve information about already exiting ontologies. This dependency makes OSSE work in fast and accurate manner. OSSE has three main features:

1] capability to find the matched ontology for an already existing concept.

2] Ability to access Swoogle web service to download suitable ontologies for a requested concept that does not belong to the local repository.

3] Ability to extract the suitable concept to concept relationships.

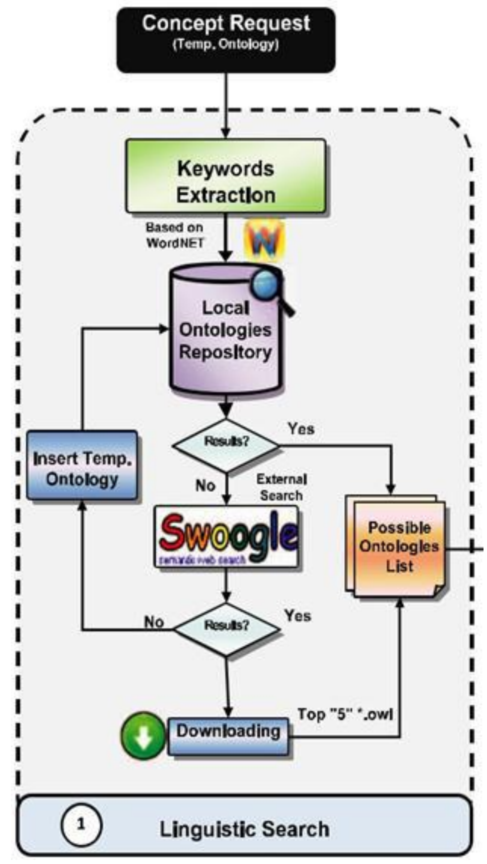

Fig. 2. Linguistic of the OSSE.

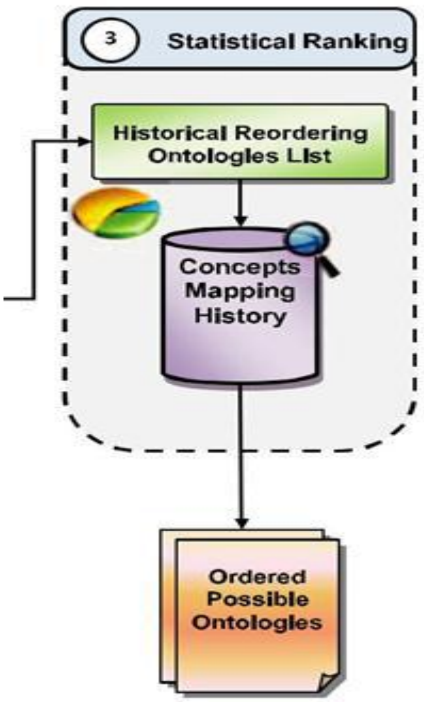
3
Statistical Ranking
Historical Reordering Ontologies List
Concepts Mapping History
Ordered Possible Ontologies

Fig.4. Statistical of the OSSE.

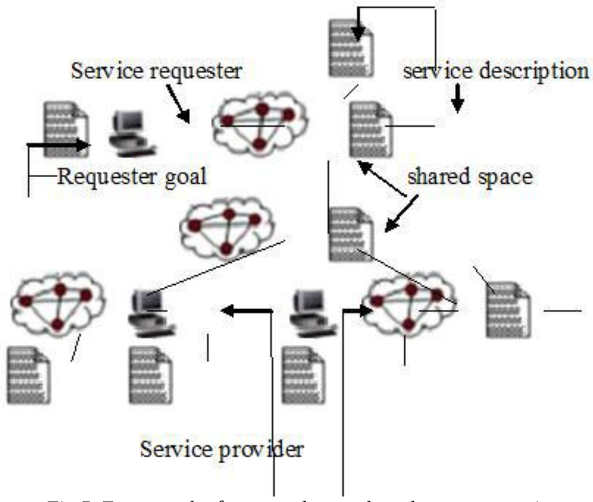

Fig.5. Framework of proposed space-based system overview

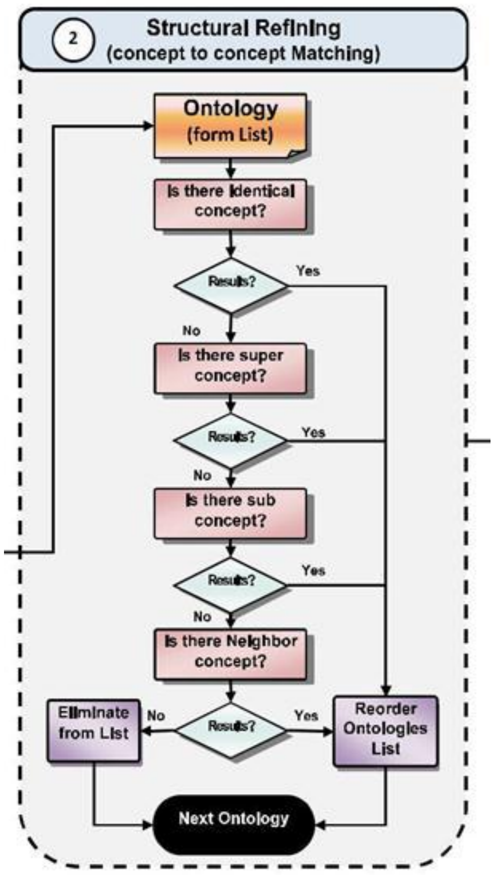

Fig. 3. Structural of the OSSE.

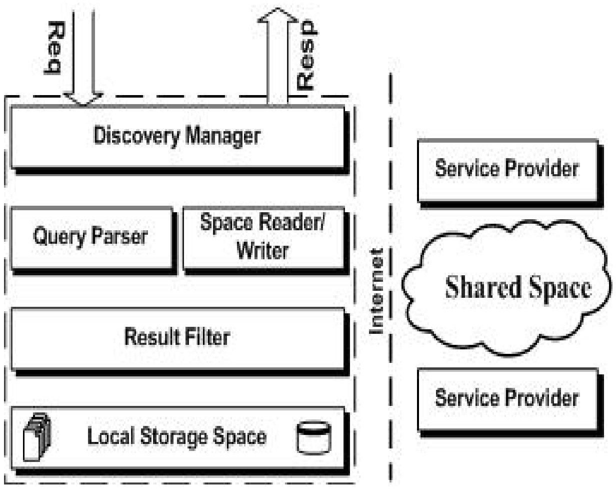

Figure 6. Discovery architecture

Distributed Web Services Implementation:
The problem of standardization in the development: -
This architecture allow users to create a new virtual shared space, or use an existing one if available. Messages can be written to or read from the space. Any user (either a service provider or a requester) can have access to the information available on the space (subject to security policy).Thus, when looking for aWeb service, a request can be sent to the virtual shared space instead of sending to individual Web service description repositories. The added advantage of this approach is that it reduces the communication overhead and the result obtained are more precise. The detailed architecture and its components are described in next section.

Component Descriptions
Based on the above mentioned architecture concepts the architecture components and their interactions are introduced
in this section. Figure 2 depicts the proposed architecture. These components are loosely coupled and are pluggable. This allows replacement of existing implementations over time with alternative or more expressive implementations as well as new components.

Discovery Manager: Discovery manager is a gateway to distributed Web service discovery and provides access interfaces serving as a point of interaction. It receives requests from the user and returns response to the user. When a request is received, it schedules a job for query parser.

Query Parser: Query parser is responsible for parsing each incoming requests. It determines the requests type (e.g., read, write, take) and forwards them to space reader or space writer.

Space Reader/Writer: The job of finding the available spaces, writing messages to and reading messages from a space is handled by space reader/writer component. It consist of two sub-components, reader and writer. Reader finds and reads messages from the shared space and writer writes messages to the spaces. It is the job of the writer to keep status of requests to the local storage space.

Matchmaker: The concrete matchmaking between the requester goal and available Web service descriptions is done by the matchmaker component. It receives the set of Triples from the space reader, obtains goal descriptions from the local storage space and performs matchmaking between them.

Local Storage Space: The local storage space is used to store the intermediary data produced by the distributed Web service discovery system. It is also used for storing interface descriptions of the internal components and information about shared spaces

Interfaces For the distributed Web service discovery architecture to work properly interfaces has to be defined and implemented. In the following the interfaces of main components shown in figure 6 are described and their implementation approach is discussed. The interfaces presented here are the basic required interfaces, and they can be extended to support more advanced operations without hindering the core idea of the architecture.

IV. RESULTS

we use many WSDL files as a case study to monitor the mapping process (specially the automatic phase) and to evaluate its results. Some of these files belong to OWLS-TC [28] which provides the WSDL file and its corresponding OWLS file. This collection is used to compare the results of our mapping process with those already included within the collection. The rest of the WSDL files belong to real web services which help us to study the behavior of our algorithm in a practical environment.

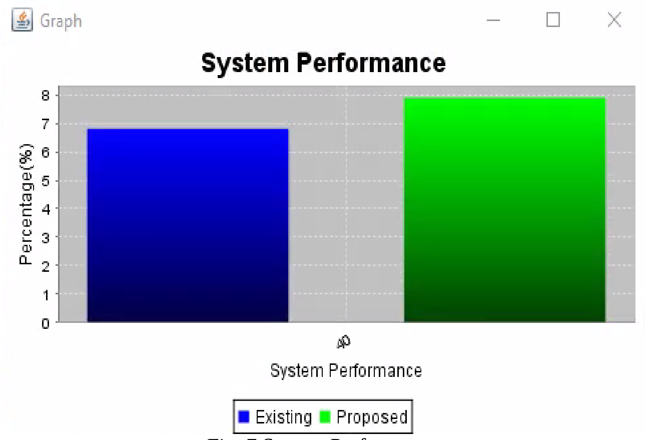

Fig. 7 System Performance

comparison between the WSDL2OWL-S tool [18] and the proposed mapping algorithm. The figure presents the relation between the number of concepts generated by the system and the number of registered services. We can obviously note that the number of used concepts will be increased when the number of services increases in both cases. But, in the case of SDL2OWL-S tool the number of concepts increase with very high rate when compared to the case of our proposed algorithm. For example, when the number of registered services becomes 1,000, the average number of concepts per service is 2.66 in case of WSDL2OWL-S and 0.4 in case of the proposed algorithm. So, the proposed algorithm is more scalable than WSDL2OWL-S. It is important to state that the performance of the discovery process is negatively affected when the number of concepts defined in the system increases. That is because the study of inputs/outputs matching between the request and the available services is a major task for any discovery process. There is no doubt that this matching become faster and more accurate when the total number of concepts which defines the inputs and outputs types become smaller. So, the proposed algorithm will be better than WSDL2OWL-S from the discovery point of view.

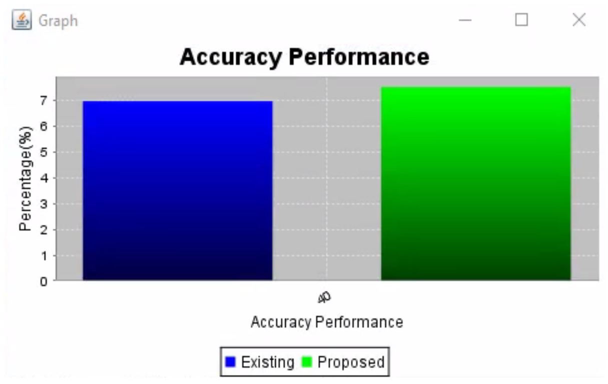

Fig. 8 Accuracy Performance

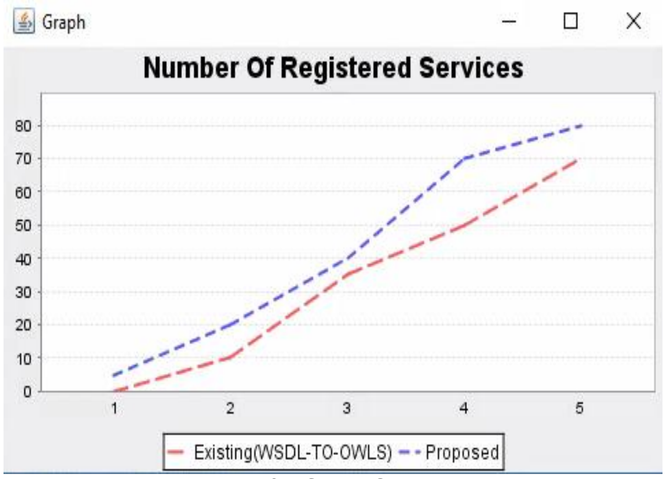

Fig. 9 WSDL to OWL

V. CONCLUSSION

Semantic web services used OWL in a mapping process that helps to convert WSDL files to OWL-S file. Mapping algorithm has backbone of local ontology repository & OSSE. It enhance the results of OSSE by using the structural refining stage SWSs ranking algorithm. This matchmaking process measure the semantic distance between the user requests and the available services. The problems of wasted time, non accurate mapping and absence of any standardization are most important issues that SWS has to deal. A distributed Web service discovery architecture is designed to be reliable, flexible and scalable. It allows the publishing of web service descriptions as well as web service of user's interests. A distributed Web service discovery architecture is built in such a way to reduces the communication overhead and the result obtained is more precise. As part of our next step, we intend to upgrade the current implementation from centralized shared space to Distributed shared space. While doing so, WSMX Web Service Execution Environment) will be used as a test bed platform. In the current implementation, shared space is centralized. It allows to create virtual spaces and query those virtual spaces. As part of our next step, we intend to upgrade the current implementation from centralized shared space to distributed shared space.

ACKNOWLEDGMENT

I would like to express thanks to my project guide Jamgekar R.S. and co-guide Sawant N. M. He helped and supported for this work and No words are sufficient to express my gratitude to my family for their unwavering encouragement.

REFERENCES

[1]. T. A. Farrag , A. I. Saleh , "*Toward SWSs discovery : Mapping from WSDL to OWL-S Based on Ontology Search and Standardization Enginee* " , IEEE Transaction on Knowledge & Data Engg, vol. 25 , No. 5 , May 13.

[2]. D. Martin , M. Burstein, D. Mcdermott, S. Mcilraith , M. Paolucci , K. Sycara , D.L. Mcguinness, E. Sirin , and N. Srinivasan , " *Bringing Semantics to Web Services with OWL-S* " World Wide Web , vol. 10, no. 3, pp. 243-277, Sept. 2007.

[3]. S. Vinotha , J. Vikneshwaran , " *Relevency Based ContentSearch in Semanti Web* ", International Journal of Emerging Trends and Technology in Computer Science(IJETTCS),Vol. 3 , Issue 2 , March – April 2014.

[4]. S. Pradeepha , B.Lakshmipathi , " *Augmenting the SWS Discovery by Categorization of Web Service* " , International Journal of Advanced Research in Computer Science and Tech , Vol .2 Issue Special1 Jan- March 2014.

[5]. J. Praba. & M. A. Hema , " *Semantic Web Service To Support Modeling In Mapping From Web Service Description Language*", International Journal of Computer Science and Engg,Vol. 3, Issue 3, May 2014.

[6]. M . Burstein , C . Bussler , M. Zaremba , T. Finin , M.N. Huhns , M. Paolucci , A.P. Sheth , and S. illiams , " *A Semantic Web Services Architecture* " , IEEE Internet Computing, vol. 3, no. 5, pp. 72-81, Sept./Oct. 2005.

[7]. T .A. Farrag " *A Cluster-Based Semantic Web Services Discovery and Classification* " , Proc . ACME Second Int'l Conf . Advanced Computer Theory and Eng., pp 1825-1834, 2003.

[8]. "*Web Services Description Language (WSDL)*" , W3C Note, 2001.

[9]. "Web Ontology Language for Service OWL-S" , W3C Member Submission, 2004.

[10]. B . Di Martino , " *Semantic Web Services Discovery Based on Structural Ontology match* " , Int'l J. Web &Grid Services , vol. 5 , no. 1, pp. 46-65, 2003.

[11]. M. Paolucci, N. Srinivasan, K. Sycara, and T. ishimura," *Towards a Semantic Choreography of Web Services: From WSDL to DAML-S*", Proc. First Int'l Conf. Web Services (ICWS '03), pp 22-26, June 2003.