Enhanced Semantic Web Service Discovery Using Machine Learning on Mapped WSMO Services

S. Sandhya [#1], Dr. P. Pabitha [*2], Dr. M. Rajaram [#3]

[#] Department of Computer Technology, Anna University
MIT Campus, Chrompet, Chennai – 600044, India
[1] sandhya.yuvi@gmail.com
[2] pabithap@gmail.com
[*] Anna University
Guindy, Chennai – 600025, India

Abstract—Web services were traditionally defined using the Web Service Description Language (WSDL) as it was the World Wide Web Consortium's standard recommendation for web service description. Later on, with the advent of knowledge representation using ontologies, more and more services were designed using ontological models, more specifically the Sematic Mark-up for Web Ontology Language (OWL-S). And to make use of the huge number of existing web service repositories, a mapping from WSDL to OWL-S was created. But, OWL-S was just an extension of OWL and required additional support such as external Rule Languages in order to bring out the best of its semantic capabilities. Another better ontological knowledge representation model, the Web Service Modeling Ontology (WSMO), is an entirely conceptual model having its own semantic framework for handling web services, but lacks the extensibility of OWL-S. Therefore, a model for mapping WSDL to WSMO is proposed. And thereafter the automated semantic web service discovery mechanism, for WSMO based web services, is to be enhanced using machine learning algorithm.

Keyword- Semantic Web Service, Mapping, WSDL, WSMO

I. INTRODUCTION

Formally, Semantic Web [1] can be described as an extension of the present web, which is able to describe things in a way that computers can understand using Ontologies [2]. Ontologies define the concepts and relationships used to describe and represent an area of knowledge. In terms of web services, they can be used to describe the role of web services, define their characteristics and thus help automate the process of definition, discovery, invocation and composition. Such automated services based on ontologies described using machine understandable technologies are termed as Semantic Web Services (SWS). The semantic web services life cycle can be defined as follows: Description, Discovery, Invocation, Composition, Inter-Operation, Execution and Monitoring the service for future changes.

II. RELATED WORK

A. Web Services

A web service may be simply represented as an entity that comprises of a web bound computational element with a set of input and output parameters. It is in a basic way, a means of exchanging application messages [3] between different systems over the World Wide Web. It mostly consists of three components: a representational format such as WSDL or OWL-S or WSMO, a messaging protocol such as Simple Object Access Protocol (SOAP) to transfer the input requests and the output responses between the end-point systems and finally, a transportation protocol such as Hyper-Text Transfer Protocol (HTTP) on which the web service operates.

B. Web Service Description Language (WSDL)

The Web Services Description Language (WSDL) is an XML-based language that is used for describing the functionality offered by a Web service [4]. A WSDL description of a web service (also referred to as a WSDL file) provides a machine-readable description of how the service can be called, what parameters it expects and what data structures it returns. It thus serves a roughly similar purpose as a Method signature in a programming language.

C. Semantic Mark-up for Web Ontology Language (OWL-S)

OWL-S [5] formerly known as DAML-S is a knowledge representation model made up of several ontologies of services extended from the OWL model for automatic description, discovery, invocation and composition of the web resources. This allows the service providers and consumers to make use of semantic technologies in WSDL files. To make use of a Web service, a software agent needs a computer-interpretable description of the service, and the means by which it is accessed. An important goal for Semantic Web mark-up languages, then, is to establish a framework within which these descriptions are made and shared, which is made possible by OWL-S.

D. Web Service Modeling Ontology (WSMO)

The Web Service Modeling Ontology (WSMO) provides a conceptual framework and a formal language for semantically describing all relevant aspects of Web services [6]. In order to facilitate the automation of discovering, combining and invoking electronic services over the Web. It consists of four main components, which are Ontologies, Web Service descriptions, Mediators and Goals. It uses the Web Service Modeling Language (WSML) as its representational language and has its own in-built reasoning engines.

E. OWLS-S and WSMO – A Comparison

OWL-S is an extensible model and was basically extended for traditional web services (i.e.) designed to semantically enhance WSDL [7]. It is better suited for Semantic Web Service (SWS) description and discovery. While WSMO is a conceptual model and was created as a standalone ontology model based on Web Service Modeling Framework (WSMF) [8] and has its own set of representations, tools and techniques and is most useful modeling and execution environment of SWS. When it comes to input and output parameters OWL-S does not support entire domain ontologies nor does it support anything other than binary predicates in terms of arity (parameters for logical functions) represented by properties. But WSMO not only allows entire domain ontologies as input/output parameters, but also supports n-ary arity thus making logical reasoning more simple and efficient [9].

WSMO’s goal component can be used to represent any of the client’s output perspectives in different possible manners that the clients may perceive [10]. But OWL-S does not have such flexibility and so can only describe a specific format of the output. This may result in the client not being able to discover what they expect as per their perception, even though their requirements are met by the service. Also, according to Lina Azleny Kamaruddin et al., [10], OWL-S uses only a single modeling element for both the service requestor and the service provider’s service profile. WSMO on the other hand allows creation of multiple service profiles [11] in an easy manner.

Since OWL-S is an extensible model, it does not have any native Rule Languages [12] and has to borrow from external Rule Languages which can lead to inefficient rule handling and un-decidability. The OWL-S language has 3 levels of expressiveness while WSMO has 5 levels. The higher the level of expressiveness, the ontology is able to express more in-depth meaning but becomes very complex to understand, since WSMO uses more expressiveness it is supposedly more complex and involves extra cost. But it makes use of Logical Language based Reasoning techniques as a positive trade-off for the costs involved.

The OWL-S is based on the highly popular but very limited Resource Description Format (RDF) and uses Description Logic (DL) to enrich the web services with semantics [5]. And WSMO is based on Frame Logic and Logic Programming (LP). The syntax of LP is almost an Object Oriented (or Ontological) notational variant of normal Logic Programming [6]. Hence it can easily represent ontologies by describing concepts, relations and inference rules.

Finally, OWL is designed as an extensible model. Hence it can easily interact with existing ontologies, data and technologies available from the non-semantic web services. Since WSMO is an independent model, it cannot re-use domain ontologies developed in diverse domains, thus not allowing the service providers to model the diverse concepts related to services. This implies a strong need for mapping to WSMO technology from different other technologies.

From the study of the above papers it is inferred that WSMO clearly has a very good advantage over OWL-S, as it has very good interoperability with existing web service technologies, and provides support for ontology reuse. While the mapping between WSDL and OWL-S [13] serves as an advantage for OWL-S, it still doesn’t cover WSMO’s positive features and also doesn’t have a dedicated ontology editor, as it uses only the popular but generic Protégé [14] ontology editor. WSMO can be a better option because of the following: High support for rules at design and execution level with effective editing facilities using the Web Service Modeling Toolkit (WSMT) [15] and the possibilities of exploiting several mature and matching technologies developed for OWL ontologies.

Therefore a mapping between WSDL and WSMO will substantially enhance the usage of WSMO and its list of extensive features with respect to legacy services as well.

III. PROPOSED SYSTEM

The proposed system maps the existing WSDL based non-semantic web service documents/files to a structure corresponding to the WSMO architecture. This is done by analyzing and processing each and every individual entity of the WSDL documents. These may be either a representational entity that describes the information contained within it. Or it may be the parameter-names/types supplied by the service provider regarding the web service. It can also include the actual atomic values for any of those parameters. Once processed, these entities are mapped to the corresponding ontological elements of WSMO architecture and are then written to a file with the .wsml extension, which is the representational language of WSMO.

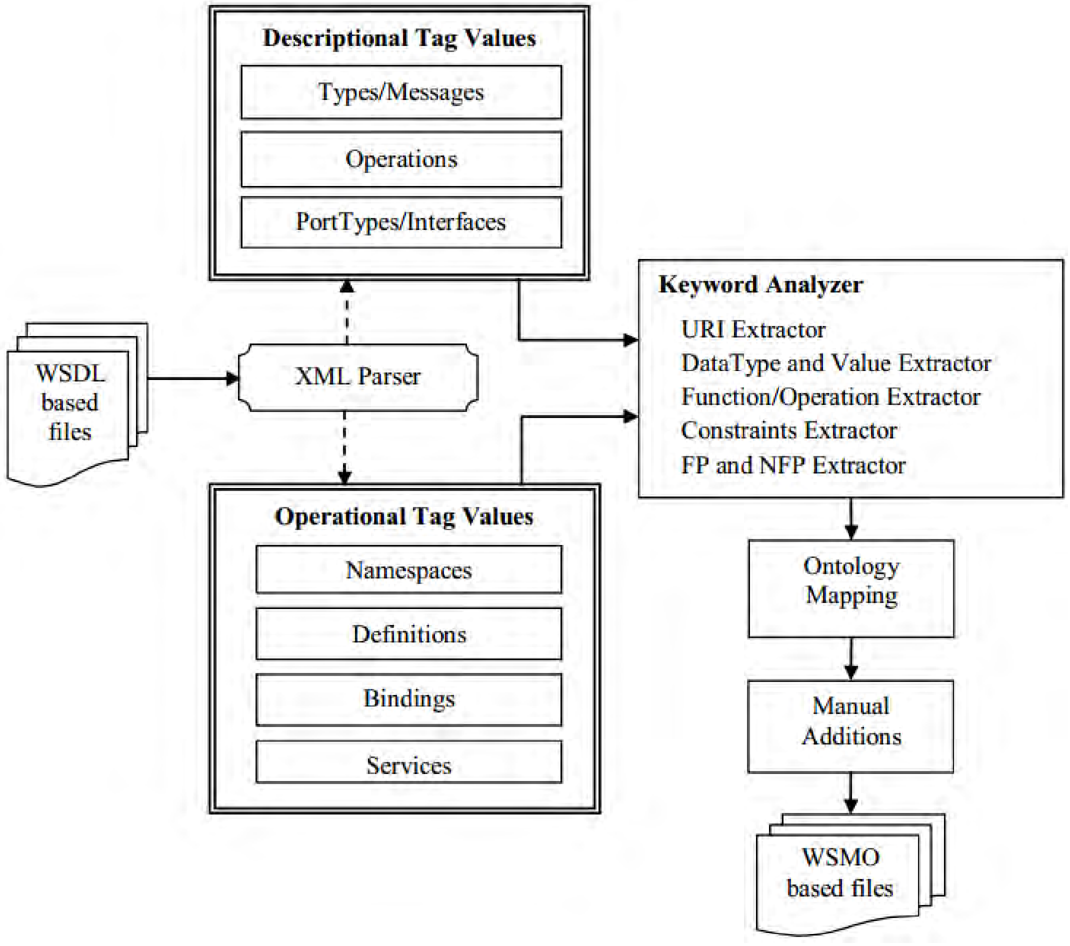


Fig. 1. Proposed System Architecture

In the next phase of this work, these mapped services will be subjected to a machine learning based, enhanced SWS discovery mechanism. Along with other native WSMO based services as well. This discovery process, with various combinations of these two sets of mapped and native SWS documents, allows us to test the performance of the enhanced discovery mechanism for WSMO. The efficiency of the WSDL-WSMO mapping can also be compared by using the native service based discovery engine results as the benchmark.

The system design consists of the following components: XML Parser, which parses the tags into their Functional and Operational sections respectively. The Keyword Analyzer, Ontology Mapping Engine and the Manual Approval module. In Figure 2, with regards to the manual approval process, the service provider's choice for manual approval, will override any other conditions.

1. If the service providers specify that, all mapped services are to be checked manually, then, every mapped service will be checked.
2. If the service providers specify that, no manual check is to be performed, then, there will no checking at all, even if there are errors/inconsistencies/in-sufficient data in the mapped services.
3. If the service providers specify that, a check is done based on the conditions given below, then, those conditions are checked. And if the services satisfy even one of them, manual intervention is requested.
   a) If the number of goals are zero or less than required, after the service has been mapped, manual addition of the goals can be requested.

b) If there are no values for certain non-functional properties, then fill them using manual intervention.

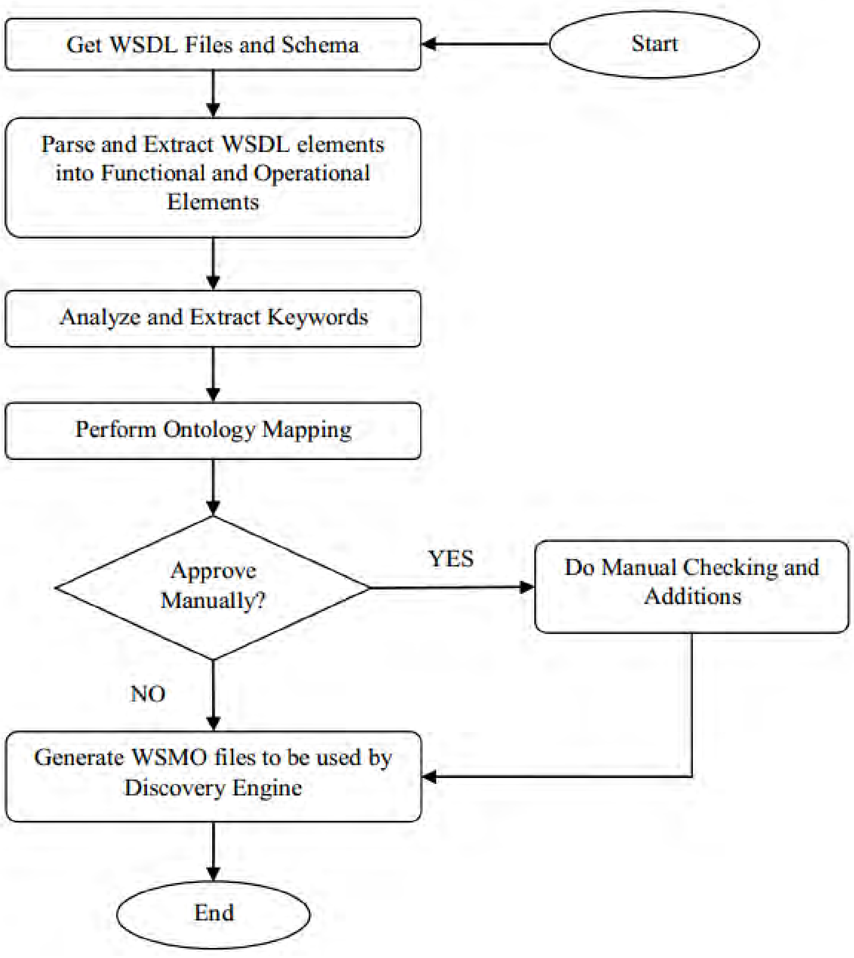

Start
Get WSDL Files and Schema
Parse and Extract WSDL elements into Functional and Operational Elements
Analyze and Extract Keywords
Perform Ontology Mapping
Approve Manually?
YES
Do Manual Checking and Additions
NO
Generate WSMO files to be used by Discovery Engine
End

Fig. 2. Flow diagram of the Mapping Phase

F. Algorithm for Mapping

```
INPUT: WSDL Files
OUTPUT: WSMO Files
BEGIN
    While (more WSDL files available)
        Read and parse the WSDL elements
        Separate functional/descriptional elements as $E_D$
        Separate operational elements as $E_O$
        Extract keywords $K_{1 \text{ to } n}$ in $E_D$ and $E_O$
        For each k in $K_{1 \text{ to } n}$
            Map k to its corresponding WSMO ontology $W_k$
        End for
        If (manual check required is true)
            Ask for input from service provider
        End if
        Generate WSMO file from ontology mappings
    End While
END
```

IV. Modules and Components

A. XML Parser

This module does the job of reading the several XML based WSDL documents and parsing the WSDL tags individually. So that, the entities represented by these tags, can be stored separately, based on their purpose in describing the web service. This is achieved by utilizing the tag elements, attributes and their values that are obtained from the parser. Any parser available in the public domain can be used for this purpose. Since the categorization can be easily done on the parser's output.

B. Keyword Analyzer

Based on the WSDL tags, attributes and their values, WSMO specific keywords are extracted using the keyword analyzer, which makes use of WordNet [22] for common language constants classification and a small WSDL elements based repositories for identifying the type of values the keywords represent. The keywords are then mostly classified into one of the following categories. Uniform Resource Identifiers (URI), Data Types and Values, Functions or Operations, Constraints on properties, Functional Properties (FP) and Non-Functional Properties (NFP).

C. Ontology Mapping Engine

This engine takes its input from the keyword analyzer and maps into to the corresponding WSMO element in the required ontological structure. This mapping is done using a customized algorithm. In order to efficiently make use of the existing web services both WSDL 1.1 and WSDL 2.0 specifications are considered when mapping to WSMO. As every entity is represented as an Ontological entity, except for literals and constants, this mapping algorithm is sufficient to generate the set of WSMO based files for further service discovery. But WSDL basically is a non-semantic language and hence it may lack some of the vital information such as the different goal perspectives from the client's point of view. Therefore, before generating the WSMO files a manual check and addition of data is performed, which maybe skipped based on the service provider's preferences thereby, avoiding or, at least reducing the bottleneck created by the human involvement.

D. Semantic Web Service Discovery Engine

This is the ultimate goal of this work, that is, to enhance the semantic web service discovery based on machine learning algorithmic techniques. The discovery process is to be applied on both native WSMO based services and the ones mapped from WSDL. And intend to bridge the gap in discovering technologically older but still useful services. This module is planned to done for the next phase of the work. A literature survey [16, 17] is being performed at this stage in-order to find if a suitable machine learning algorithm exists or if any customization is needed based on an existing algorithm. This is because the algorithm must efficiently discover both mapped and native WSMO services which differ in the amount and quality of semantic data offered..

V. Software Description

A. Eclipse Indigo

Eclipse Indigo is multi-language integrated development environment (IDE) for software development. It comprises of an in-built text editor, several compilers for various languages and also an extensible plug-in system. It can support the development and deployment of web services. And also supports some of the more popular languages such as C/C++, PHP, Java, Python and Android SDK.

B. JAXP Library

The Java API for XML Processing (JAXP) is for processing XML data using applications written in the Java programming language. JAXP leverages the parser standards Simple API for XML Parsing (SAX) and Document Object Model (DOM) so that the developer can choose to parse the data as a stream of events or to build an object representation of it. JAXP also supports the Extensible Style Sheet Language Transformations (XSLT) standard, giving control over the presentation of the data and enabling conversion of the data to other XML documents or to other formats, such as HTML. JAXP also provides namespace support, which allows working with Document Type Definitions (DTDs) that might otherwise have naming conflicts..

VI. IMPLEMENTATION

Implementation was done for the mapping phase. The job of xml parser is to read and parse WSDL tags and separate them as descriptive and operational tags. Input dataset was obtained from OWL-S TC. A Web service repository of WSDL files for testing service retrieval [21]. It is followed by the keyword analyzer module. It utilizes the WSDL keyword's type to determine its appropriate place in the final WSML document. The mapping structure followed in this work is given in Figure 3. Finally the actual conversion process takes place wherein the capability descriptions of the WSML are defined by using the WSDL Port Type elements and the WSDL messages that serve as the input and outputs of the web service.

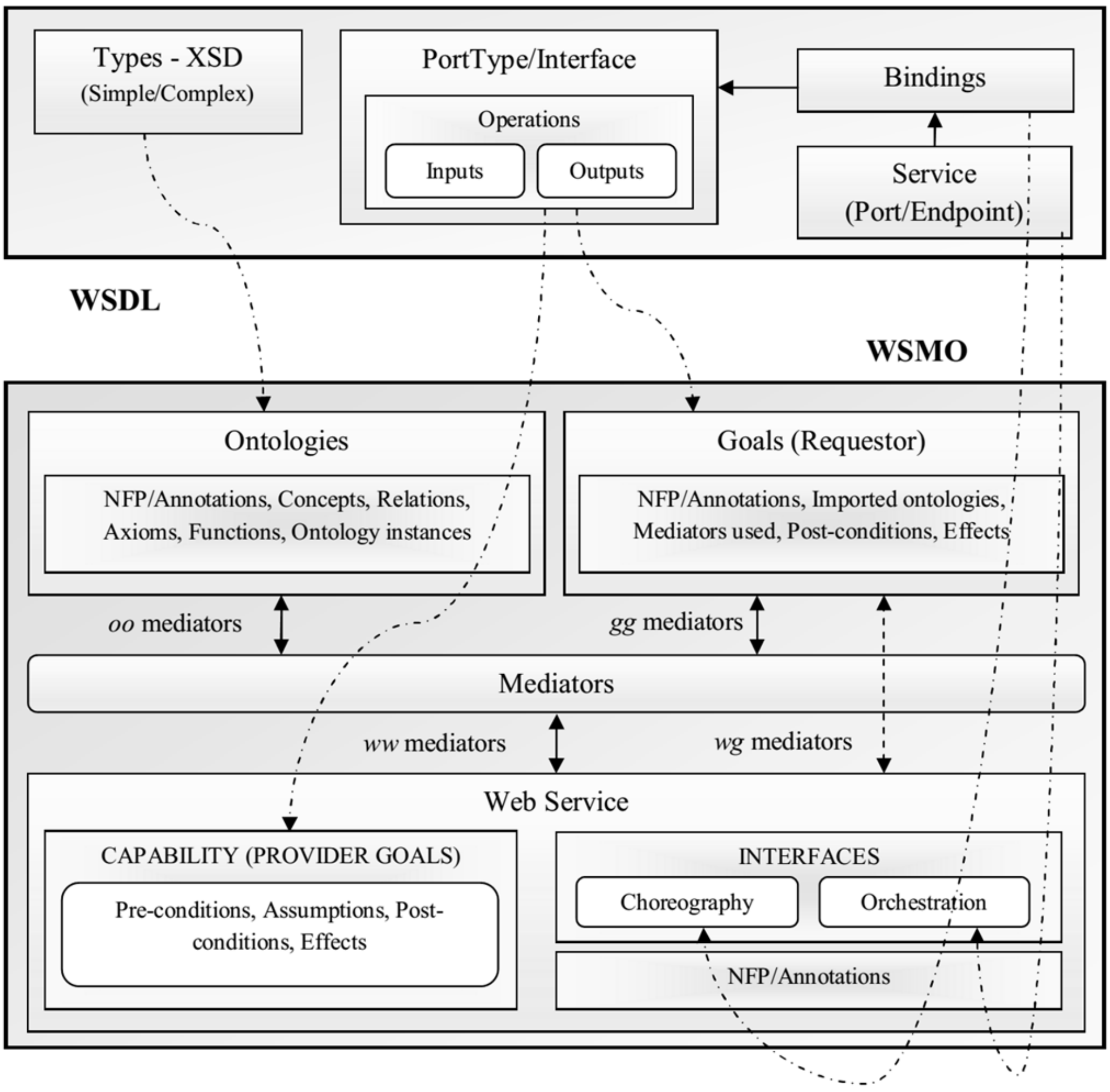
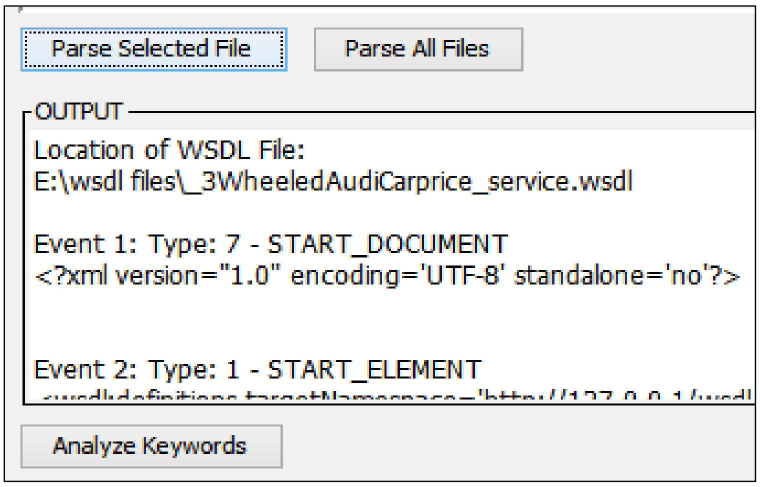


Fig. 3. Mapping Structure of the Proposed System

VII. Results and Discussion

After the WSDL file is passed on to the Parser, it extracts all the individual elements in the file which are then processed by our program and classified into different types of elements based on their properties. A sample screenshot of a processed WSDL file is given in Figure 4.



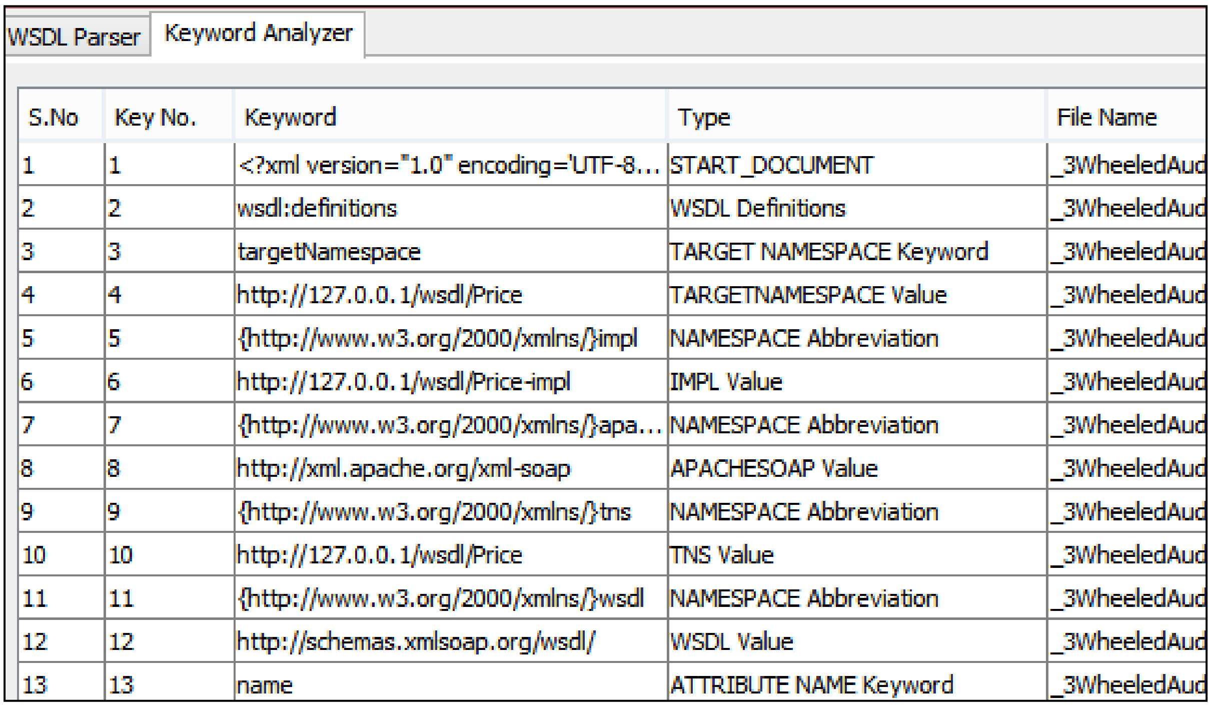
```
Location of WSDL File:
E:\wsdl files\_3WheeledAudiCarprice_service.wsdl

Event 1: Type: 7 - START_DOCUMENT
<?xml version="1.0" encoding='UTF-8' standalone='no'?>


Event 2: Type: 1 - START_ELEMENT
```

Fig. 4. Parsed WSDL File

These parsed files are then analyzed for keywords and the keywords are grouped into operational and descriptional keywords which will aid in the framing of WSML files based on these keyword types. Finally the keywords are used to find the WSML equivalent of the WSDL elements and are placed in their appropriate positions as shown in Figure 5. A sample of the final mapped WSML document can be seen in Figure 6.



S.No	Key No.	Keyword	Type	File Name
1	1	<?xml version="1.0" encoding='UTF-8...	START_DOCUMENT	_3WheeledAud
2	2	wsdl:definitions	WSDL Definitions	_3WheeledAud
3	3	targetNamespace	TARGET NAMESPACE Keyword	_3WheeledAud
4	4	http://127.0.0.1/wsdl/Price	TARGETNAMESPACE Value	_3WheeledAud
5	5	{http://www.w3.org/2000/xmlns/}impl	NAMESPACE Abbreviation	_3WheeledAud
6	6	http://127.0.0.1/wsdl/Price-impl	IMPL Value	_3WheeledAud
7	7	{http://www.w3.org/2000/xmlns/}apa...	NAMESPACE Abbreviation	_3WheeledAud
8	8	http://xml.apache.org/xml-soap	APACHESOAP Value	_3WheeledAud
9	9	{http://www.w3.org/2000/xmlns/}tns	NAMESPACE Abbreviation	_3WheeledAud
10	10	http://127.0.0.1/wsdl/Price	TNS Value	_3WheeledAud
11	11	{http://www.w3.org/2000/xmlns/}wsdl	NAMESPACE Abbreviation	_3WheeledAud
12	12	http://schemas.xmlsoap.org/wsdl/	WSDL Value	_3WheeledAud
13	13	name	ATTRIBUTE NAME Keyword	_3WheeledAud

Fig. 5. List of WSDL Keywords after analysis

Since we perform the mapping only for discovery purpose, the only part of the WSDL files that we are interested in, is its capability section as shown in Figure 3. Since the Goals of requestor will be designed by developers at the client end and thee interface section is need only when the mapped WSML services are to invoked, executed or used in composition of web services. Though ontologies are useful by their presence, they are not very essential to the actual service discovery. Hence the formation of ontology from WSDL Type elements is given a lower priority now and is planned for the next phase of this work.

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-rule"
namespace { _"http://namespaces.snowboard_info.com#",
          esxsd _"http://schemas.snowboard_info.com/EndorsementSearch.xsd#",
          soap _"http://schemas.xmlsoap.org/wsdl/soap/#",
          es _"http://www.snowboard_info.com/EndorsementSearch.wsdl#",
          discovery _"http://wiki.wsmx.org/index.php?title=DiscoveryOntology#,
          dc _"http://purl.org/dc/elements/1.1#"
 }

webService EndorsementSearchService
dc#description hasValue "snowboarding_info.com Endorsement Service"

capability GetEndorsingBoarderPortType
nonFunctionalProperties
discovery#discoveryStrategy hasValue discovery#HeavyweightDiscovery
discovery#discoveryStrategy hasValue discovery#NoPreFilter
endNonFunctionalProperties

sharedVariables ?x
precondition GetEndorsingBoarderRequest
          definedBy
                    ?x memberOf GetEndorsingBoarder
postcondition GetEndorsingBoarderResponse
          definedBy
                    ?x memberOf GetEndorsingBoarderResponse
```

Fig. 6. WSML file after mapping

The mediators on the other hand are used to resolve inconsistencies only and in no way hinders the service discovery process, unless there are heavy inconsistencies in the mapping process. In such a case the mediation process alone can be daunting task and hence it has been excluded as out of scope for this work.

The performance measures used for testing the efficiency of the proposed system is derived by comparing the measures of existing vs. proposed system. The amount of time and resources spent in the overall proposed system process for obtaining the same evaluation results that can be obtained using the existing system needs to be compared. The efficiency of the WSDL-WSMO mapping can also be compared by using the native service based discovery engine results as the benchmark. Such parameters include,

- The amount of time and system resources like memory, processing speed, storage used for processing the WSDL input file.
- The time and resources spent by both humans and computers for adding the necessary semantic details to the service.
- And finally, the time required for obtaining the correct results from the service discovery process and the accuracy of the services discovered with respect to the user's requirements.

Evaluation Parameters for measuring the performance of the proposed system are discussed below: The ultimate result of the proposed system is the resultant semantic WSML services discovered for the given user request. Therefore, the accuracy of the relevant results returned is the major parameter for performance measure. This parameter value may be obtained by testing the system against known queries and known correct results. So, the performance gain factor PG of the proposed system is given by,

$$Performance\ Gain\ Factor = \frac{A_{\mathrm{PS}}}{A_{\mathrm{ES}}} \qquad \text{(Eq. 1)}$$

In the above formula, the value A_{ES} represents the accuracy of the existing system, while A_{PS} represents the accuracy of the proposed system. The accuracy of both the systems can be defined in general as,

$$Accuracy\ (A) = \frac{R_r}{R_t} \quad \text{(Eq. 2)}$$

Where, R_r is given by number of results returned correctly and R_t is the total number of results that were returned in response to the user query. The next major parameter is the overall Execution Time (T), which is given by,

$$T = T_a + T_p + T_r \quad \text{(Eq. 3)}$$

(i.e.) Time T is the sum of Access time of WSDL from repository (T_a), Processing time for getting semantics T_p and Retrieval time of WSMO files from repository T_r

Apart from the above parameters, the storage size in the Memory and Hard Drive before and after processing and the amount of CPU utilized during the process also affect the performance [14] and should have a value less than the conventional, manual conversion systems.

The Figure 7 shows a comparison of the percentage of relevant result results returned using the service discovery mechanism available in the WSMT toolkit under the WSML-rule scheme. Since our proposed system's enhanced service discovery module is a work in progress, this existing discovery mechanism was used as an intermediary benchmark.

The dataset consisted of traditionally created native WSMO services for the first iteration and the mapped WSMO services for the second iteration. Finally both these sets of services were taken together as the source repository of services to which the discovery mechanism was applied. The queries used in all three scenarios were same and all available WSDL and WSMO services under the required domains were utilized.

From the graph in Figure 7, it can be clearly seen that there is a remarkable correlation between the overall results returned in all the three scenarios except for queries 2 and 3, which reflects in the overall performance of the proposed scheme as well for those same queries.

But the major inference to be obtained from this evaluation is that the number of relevant services that are discovered for each query is increased substantially, thereby providing more choice in the selection of desired services for the end user by making full use of the otherwise ignored WSDL or WSMO services.

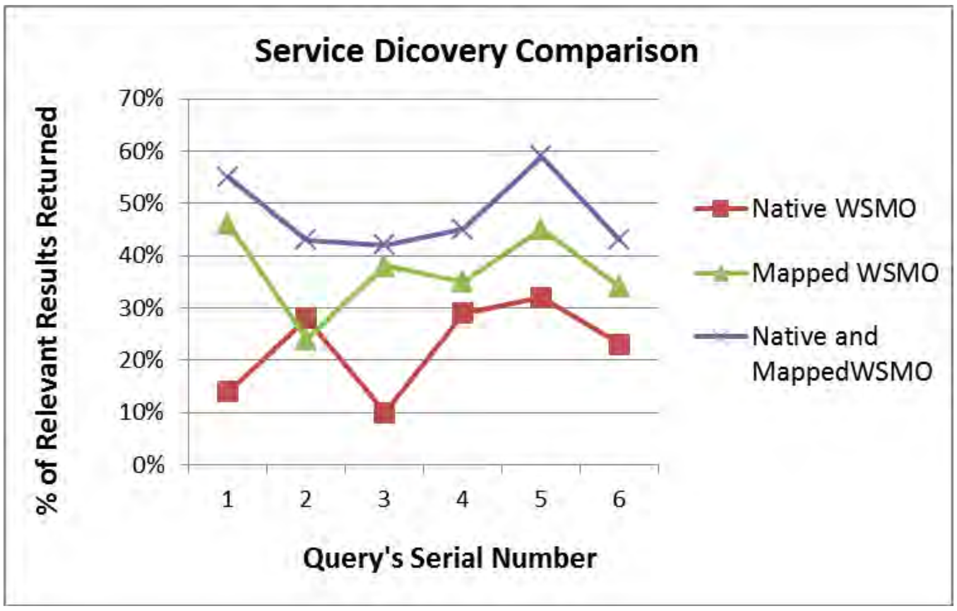


Fig. 7. Comparison between Mapped and Unmapped Service Discovery

VIII. CONCLUSION

Newer implementations of semantic web services are not replacing the huge repositories of existing web services. Also, re-writing such as huge number of services to incorporate semantics is a resource consuming and redundant task. Therefore, several mapping methodologies are required. The mapping of WSDL to WSMO is proposed here.

Further, the discovery engines used now has room for improvement in several areas. These include: the way the services are organized and stored, the method of search and matchmaking [15] involved and the querying mechanisms involved [16]. So, an enhanced machine learning based Bi-clustering algorithm is used for clustering the services. Also an ordinal classification system has been used to provide the best possible results for the native and the mapped semantic web services

REFERENCES

[1] Bruijn J. de, C. Bussler, J. Domingue, D. Fensel, M. Hepp, and U. Keller, “Web Service Modeling Ontology (WSMO)”, W3C Member (Submission 3), 2005.
[2] Breitman, K. K. and Casanova M. A., “Semantic Web: Concepts, Technologies and Applications”, Springer, pp. 1-327, 2006.
[3] Booth D., Haas H. et al., “Web Services Architecture Document”, World Wide Web Consortium (W3C), Working Group Note. 11, pp. 7-8, 2004.
[4] Booth D., Kevin Liu C., “Web Services Description Language (WSDL) 2.0 Primer”, World Wide Web Consortium (W3C), Recommendation 26, pp.1-2, 2007.
[5] Martin D., Burstein M., Hobbs J., Lassila O., McDermott D., McIlraith S., Narayanan S., Paolucci M., Parsia B., Payne T., Sirin E., Srinivasan N. and Sycara K., “OWL-S: Mark-up for Web Services”, W3C Member (submission), 2004.
[6] Fensel D., Lausen H., Polleres A., de Bruijn J., Stollberg M., Roman D., Domingue J., “Enabling Semantic Web Services: The Web Service Modeling Ontology (WSMO)”, Springer, 2006.
[7] Panziera L., Palmonari M., Comerio M., and de Paoli F., “WSML or OWL: A lesson learned by addressing NFP-based Selection of Semantic Web Services”, in Proc. of NFPSLAM-SOC Workshop, 2010.
[8] Balzer S., Liebig T. and Wagner M., “Pitfalls of OWL-S: A Practical Semantic Web Use case”, in Proc. of the 2nd International Conferences on Service Oriented Computing, pp.289-298, ACM, 2004.
[9] Kamaruddin L. Azleny, Shen J and Beydoun G., “Evaluating Usage of WSMO and OWL-S in Semantic Web Services”, in Proc. of the Eighth Asia-Pacific Conference on Conceptual Modelling, APCCM, pp. 53-58, 2012.
[10] Martin D., Burstein M., Hobbs J., Lassila O., McDermott D., McIlraith S., Narayanan S., Paolucci M., Parsia B., Payne T., Sirin E., Srinivasan N. and Sycara K., “OWL-S: Mark-up for Web Services”, W3C Member (submission), Section 4, 2004.
[11] Farrag T. Ahmed, Saleh A. Ibrahim and Ali H. Arafat, “Towards SWSs Discovery: Mapping from WSDL to OWL-S based on Ontology Search and Standardization Engine (OSSE)”, IEEE Transactions on Knowledge and Data Engineering, Vol. 25, No. 5, pp.1135-1147, 2013.
[12] Gruninger M., Bodenreider O., Olken F., Obrst L. and Yim P., “Ontology Summit 2007 - Ontology, Taxonomy, Folksonomy: Understanding the Distinctions”, Applied Ontology, Vol. 3, No. 3, pp.191-200, 2008.
[13] Fensel D. and Bussler C., “The Web Service Modeling Framework (WSMF)”, Electronic Commerce Research and Applications, Vol. 1, No. 2, pp. 113-137, 2002.
[14] Fensel D., Kerrigan M. and Michal Z., “Implementing Semantic Web Services, the SESA Framework”, Springer, 2008.
[15] Mick K. and Mocan A., “Web Service Modeling Toolkit (WSMT)”, Ontology Management Working Group OMWG–Deliverable 9, 2005.
[16] Giuseppe Pirrò, Paolo Trunfio, Domenico Talia, Paolo Missier, and Carole Goble, “Ergot: A Semantic-Based System for Service Discovery in Distributed Infrastructures”, in Proc. of 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid), pp. 263-272, 2010.
[17] Naveen Kumar S., Pabitha P. and Mansoor Ahamed A. K., “Web Service Discovery Based on Semantic Description”, in Proc. of International Conference on Cloud & Ubiquitous Computing & Emerging Technologies (CUBE), pp. 199-203, 2013.
[18] Stefan Dietze, Alessio Gugliotta, and John Domingue, “Exploiting Metrics for Similarity-Based Semantic Web Service Discovery”, In Proc. of International Conference on Web Services”, ICWS, pp. 327-334, 2009.
[19] Zhao Dexin, Wenjie Li and Degan Zhang, “Web Service Matchmaking Based on Linguistic Variables”, Physics Procedia Vol. 33, pp. 236-243, 2012.
[20] Martin Junghans, Sudhir Agarwal and Rudi Studer, “Towards Practical Semantic Web Service Discovery”, The Semantic Web: Research and Applications, pp.15-29, Springer Berlin Heidelberg, 2010.
[21] Matthias Klusch, Mahboob Alam Khalid, Patrick Kapahnke, Benedikt Fries and Martin Vasileski, “OWLS-TC: OWL-S Service Retrieval Test Collection Version 4.0”, Available: http://projects.semwebcentral.org/projects/owls-tc
[22] Princeton University "About WordNet." WordNet. Princeton University. 2010. Available at: http://wordnet.princeton.edu