A Distributed Framework for Semantic Web services Automatic Discovery and Composition Based on Matching of OWL-S

Adel Boukhadra, Karima Benatchba, Amar Balla
National School of Computer Science, ESI
Oued-Smar, Algeria

Abstract

The Web services technology is now widely used in support of interoperability between distributed and heterogeneous applications. With the evolution of Semantic Web services (SWs) within organizations and their uses in a large-scale, composition of services in a distributed setting has become a real challenge. A mechanism for automatic composition of SWs according to their functionality is necessary, since the features are the most important thing that partners seek. In this paper, we describe a distributed approach to automatic composition of SWs, which supports the complexity of both SWs and the task at hand. The approach is based on the notion of alignment of OWL-S, in order to achieve semantic interoperability in a heterogeneous and distributed architecture, to ensure correct operation of Web services provided, and meet the needs and users' preferences. The objective is to optimize the time of the completion of the automatic composition of SWs, and to overcome the problem of communication within and between SWs and network saturation.

1. Introduction

Web services are as stateless software entities, betting provided by suppliers on the Internet and invoked by clients (users or other Web services). The architecture and Web services technology define a set of specifications for WSDL[1], the publication UDDI[2] and communication SOAP[3] Web services that between promote interaction in an open, heterogeneous and versatile is the Web **[2] [12].**

The main interests of SWs add semantics to Web services. They rely totally on the Semantic Web languages, especially on ontologies. They are the next generation of Web technologies for the integration of distributed and heterogeneous applications. They are reserved for automating various tasks using Web services (discovery, composition, invocation, substitution, etc...).

The ontology alignment is a very promising avenue to enable semantic interoperability. It is the heart of this interoperability. The goal of alignment is to make connections, or semantic correspondences between entities belonging to heterogeneous ontologies to enable semantic interoperability in a distributed and heterogeneous. The ontology alignment based on the calculation of similarity measures **[6].** The assessment of similarity between concepts in ontology is a known problem in many areas. There are different measures of similarity, categorized according to the techniques used (Terminology, Structural, Linguistics, Extensional, Semantics, etc...) **[7].**

The composition of SWs is the process of building new SWs value from two or more SWs and already published on the Web. The SWs consists of an automatic way according to their functionality, availability and contingencies that may occur during the composition process **[17] [18]**.

In this paper, we investigate the use of technical alignment of OWL-S, an approach to provide a collaborative mechanism for the discovery and automatic composition of SWs, in a purely distributed and heterogeneous architecture. Our approach to automatic SWs composition that we propose is based on processes (services) atomic. An atomic process is specified by a class, data inputs and outputs, preconditions necessary for its invocation and resulting effects after his execution.

The rest of the paper is organized as follows. Section 2 focuses on the work on the field of automatic and distributed composition of SWs, in Section 3; we present the problem with the objectives. Subsequently, in Section 4, describes in detail our approach and presents our main contributions. Then, in Section 5, we explain our Framework. In Section 6 illustrates the application of our approach through experimentation and evaluation, and we end with a conclusion and give some perspectives in Section 7.

2. Related work

In recent years, research in the field of Web services has been highly developed. Much of this research was devoted to the composition of SWs **[5] [13] [17] [18] [19]**.

Many solutions for the automatic composition have been proposed in the literature. These solutions can be classified into two basic categories: service composition based interfaces and composition of services oriented conversation **[2]**. The first assumes that services are described as a list of operations independent of one another, it is subdivided into two

[1] **WSDL: Web Services Description Language.**
[2] **UDDI: Universal Description Discovery and Integration.**
[3] **SOAP: Simple Object Access Protocol.**

categories: chaining services **[5], [17], [20]**, and selection of services run by conversations **[20], [21]** depending on whether the task is specified with or without combination of conversations. The second assumes that the services have complex behaviors. It is divided into three categories: selection of conversation led by a goal **[17]**, the integration of conversations led by a goal **[5]**, and integration of conversation led by the conversation **[12].**

3. Problem and Objective

Proliferation in the number of Web services, resulting in complexity in their research and their composition **[2] [12].** Actually, it is not always easy to find services that mate with user's queries, in an open and dynamic, such as Internet, there is increasingly a need for automatic composition of services. This Web service composition satisfying the query is a growing need today **[16].**

WSDL descriptions are limited to problems related to connectivity and functional properties. They cannot, by themselves, guarantee the success of the composition of Web services. This imposes the need to add the semantic description of Web services as an element in their specifications **[11].** To resolve this problem, the idea is to enrich the descriptions of Web services by other information understandable by machines. The interface description of a Web service can be supplemented with OWL-S.

The lack of an appropriate composition circulated presents a problem which may limit the potential of SWs, in terms of efficiency and scalability. In addition, the distributed nature of SWs should be considered in the design of such a mechanism. The current trend for the automatic composition of SWs aims to enable semantic interoperability between SWs. There are other ways to automatically composition the SWs, such as workflows, the calculation of situations, but it is currently planning one of the most suitable and most studied by the community of this area **[17] [18] [19].**

The objective of our work is to develop an approach to address some aspects related to problems of cooperation in the process of composition of SWs in a distributed and heterogeneous environment. For this, we propose to use the techniques of alignment of OWL-S as part of the automatic planning to address the problems described above. Indeed, the composition management of ontology alignment in OWL-S, will minimize false responses, and significantly improves the overall quality of results.

4. Proposed architecture

In this paper, we propose a distributed architecture for defining, the automatic discovery and composition of SWs. In our approach, we propose to create a set of machines distributed over a network the Internet. The key element of this architecture is our framework **DA5DCSWS** (**A Distributed Architecture for Semantic Web Services Discovery and Composition**), allowing the implementation of automatic discovery and composition of SWs. This framework is installed on different machines on the network. In this architecture, each machine implements a set of SWs described semantically in OWL-S. Figure 1 shows the architecture of our approach.

The main objective of our approach is to provide a purely decentralized scenario that supports the distributed composition of SWs. From a purely distributed and heterogeneous network, we call that a SWs particle responds to a request. This service may consist of several services from different machines in a distributed and heterogeneous. The goal is to create a collaborative workspace between different machines participating in the realization of a common goal. In this approach, all machines have the same roles. These machines act as directories of their kind.

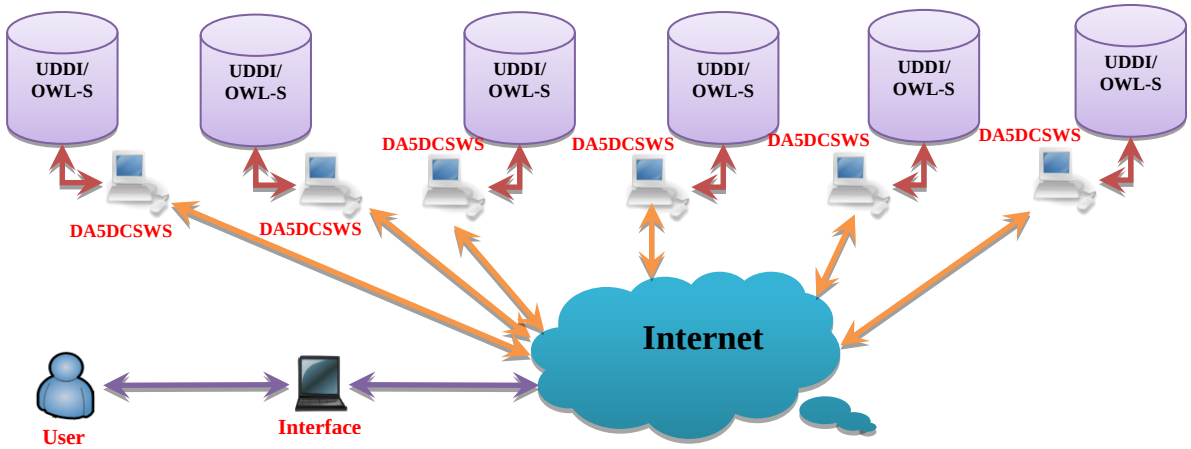

Figure 1. Distributed Architecture for SWs Discovery and Composition.

They maintain a list which records information on adjacent machines. A machine is the server and client specific services in other machines. Since there is no central directory for the other machines in the network, a machine must cooperate to transmit his application to a machine he knows of. If the purpose defined in the query is not satisfied, then the request is forwarded to another machine known. The request will be forwarded to the next. In the event that a machine has a positive answer, a message is returned to the client.

The Web service can be run on the platform of the machine that holds and the result is sent to the applicant Web service. A machine that has just received a query will run locally, if there are no positive results, it will pass the request to its neighbor. If all machines fail, the request will be forwarded to the next machine where this process is repeated until all machines will be consulted and be given a positive result. There may be no positive results found in this case the result will be sent by the last machine will be accessed negative. In this first stage of our work, we describe a strategy for discovery and composition of SWs. This phase is to provide a distributed algorithm for the discovery and composition of SWs. The main idea is to present a core algorithm that searches for SWs distributed on all network machines to compose new personalized services. This algorithm ensures the progress of the operation of the discovery in the network. Each machine runs this algorithm, when it receives a user request for discovery of SWs. We must first define the following concepts:

1. SWs is a 5-tuple (Input, Output, Precondition, Result and TextDescription),
2. SWs is a compound locally constructed from a set of SWs of the same machine.

Our main algorithm is defined as:

Algorithm 1 : Algorithm Principal () ;

1. **Begin**
2. A machine receives a user query such as **"Search the SWs: (Input, Output, Precondition, Result** and **TextDescription)"**.
3. **Discovery_Alignment ();** *// To discover a local SWs in this machine which responds to the query.*
4. **If** (There is a local SWs) **Then**: Send the Favorable Response;
   Go to End;
5. **If** (There is not a local SWs) **Then**: **Composition_Alignment ();**
6. **If** (It is possible to compose a composite SWs locally) **Then** : Send the Favorable Response;
   Go to End;
7. **If** (There is not a local Composition) **Then**: Send the request to another machine, to discover other missing SWs.
8. **End.**

The operation of SWs discovery made in step 3 of Algorithm 1 (Discovery_Alignment ()) is detailed in the following section, similarly to the operation of composition of SWs (Composition_Alignment ()) which is presented in step 5.

5. Operating system

In our work, and after presenting the strategy of discovery and automatic composition of SWs in a purely distributed and heterogeneous network, which presents an interesting part of which are proposed framework and different architectures. We will present our **DA5DCSWS** framework allowing the implementation of this strategy. The key element of this strategy is our distributed **DA5DCSWS** framework (A Distributed Architecture for Semantic Web Services Discovery and Composition), which implements the discovery and automatic composition of SWs. The overall architecture is shown in Figure 2. We detail the different components of this Framework, as well as all possible interactions in the following:

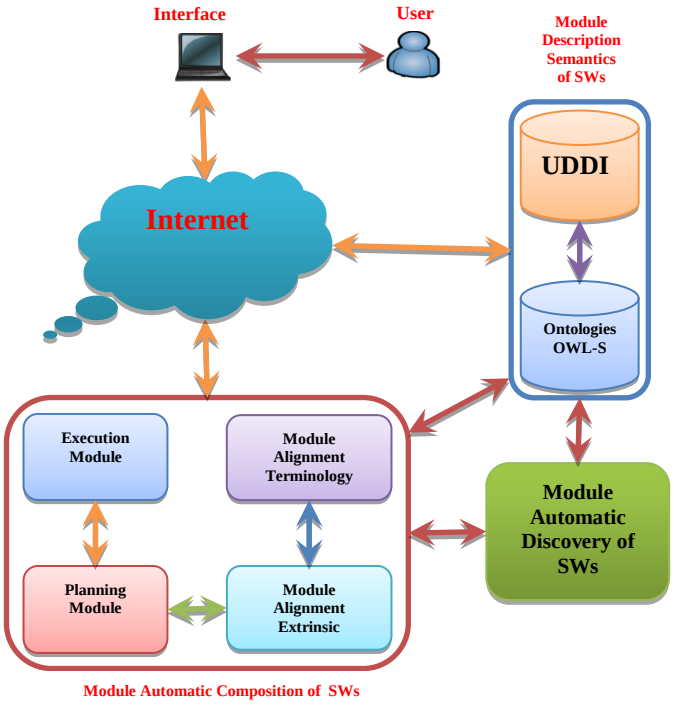

Figure 2. Architecture of the Framework DA5DCSWS.

5.1. User Interface

The user has at its friendly interface, enabling it to formulate a query according to their preferences through the following parameters: Input, Output, Precondition, Result and TextDescription. These parameters are represented by empty fields in an interface, and are entered by the user. They represent the user request and a SWs models, the same way, the OWL-S which are used for automatic discovery and composition, are modeled through the same parameters. The User Interface is considered the first window system to the world of users; it represents the visible part of the Framework. It is responsible for communication between the system and the various users. It provides tools for presenting the user requests the transmission of the request to all machines and displays the results of the architecture to users. It also takes care of sending the query from the user to all machines in the network.

We distinguish two categories of parameters, the parameters that are short strings and those strings that are long. The first category contains the parameters: Input, and Output, which normally exist as a word, term, or at most an expression of few words. The second category includes parameters: Precondition, Result and TextDescription, which is often in the form of a phrase, sentence, or even a paragraph.

This module gives as result of relevant information which is represented through the previous settings, and is used in the module discovery and automatic composition of SWs.

5.2. Module Description Semantics of SWs

This layer manages the semantic descriptions of SWs. The user can use a generator and OWL-S ontology to generate local descriptions OWL-S of the various SWs. The local ontology contains a comprehensive development of different areas of SWs. It can be developed locally according to Central OWL ontology, using a dedicated environment such as Protégé-OWL. In other cases, the user can directly download one or more domain ontologies at each machine, from the OWL ontology based power plant.

In this case, the designer can use a description OWL-S as a template to develop SWs. In the second case, the designer can generate descriptions using OWL-S the generator and the basis of local ontology.

The generator uses the OWL-S description of the syntactic WSDL Web service and domain ontology to generate semantic description OWL-S. In our work, we used the tool wsdl2owl-s to implement this component.

Our Framework allows us to use the same semantic description language (OWL-S) using the same ontology language (OWL), and using the same concepts of maximum fields. These ontologies are used implicitly in the automatic discovery module with the parameters of the user request, based on the alignment of ontologies and similarity measures.

5.3. Module Automatic Discovery of SWs

This module receives the user interface concepts that constitute the above parameters to satisfy the request, and proceeds to the discovery of SWs that can satisfy this request in an automatic manner.

In this context, we propose an algorithm for automatic discovery of SWs, which is based entirely on two technical alignments of OWL-S, the technical terminology and technique extrinsic **[1].** The current trend of most alignment systems of ontologies is to combine the different techniques alignments of OWL-S **[8].**

The algorithm performs calculations of similarity measure based on common characteristics of both concepts to produce a similarity between these concepts. Calculations of similarity measure are divided into two categories depending on the nature of the concept of characteristic exploited **[1].**

A machine that has just received a request will discover locally, if there are no positive results, it will pass the request to its neighbor. At the end of this module, we get a list of candidates for SWs composition. A list of SWs cannot meet the needs of the user, but the composition of SWS, which is charged to the process module Automatic SWs Composition.

5.4. Module Composition Discovery of SWs

Our work concerns particularly the problem of automatic composition of SWs in a distributed, heterogeneous and dynamic. The objective is to find an intelligent solution to automate the composition of

distributed SWs. For this, we propose to use the techniques of alignment of OWL-S as part of automatic scheduling. It is in this context, we propose an algorithm (see Algorithm 2) automatic composition of SWs, which is based on two techniques alignments of OWL-S: the technical terminology and technique extrinsic.

5.4.1. Module Alignment Terminology of SWs. Calculations of terminological similarities that are based on comparing the labels of entities (words, strings or texts), they are used to calculate the value of the similarity of text features, such as names, labels, comments, descriptions.

Our algorithm applies both methods in parallel to the similarity of terminology: the method that is based on the strings for the following parameters (Input, and Output), and another method that is based on tokens for parameters: Precondition, Result and Text Description.

This module is implemented as a function Similarity_Terminology (Word1, Word2) in Algorithm 2. He takes care of calculating a similarity measure for the concepts of input parameters and output, between queries with the SWs using the Jaro Winkler metric **[15].** There are several measures calculating the value of similarity or distance between two strings in the literature such as the Jaccard similarity, the Hamming distance, the Levenshtein distance, the distance of Jaro. But in our approach, we use the Jaro-Winkler metric that is based on the number and order of common characters between two terms. This metric gives the best result in general, in most cases, although the scope may be undetermined, and the speed of computing a similarity measure is faster **[4].**

In these algorithm 2 parameters: Precondition, Result and TextDescription are often described as a long text including phrases, sentences or even paragraphs. For this reason, the similarity measures designed to deal with short strings, such as Jaccard, Hamming, and Jaro are no longer appropriate. Instead, we propose a measure that is based on a hybrid method to compare strings of characters long. The hybrid method cut both channels into several shorter chains **[8].**

5.4.2. Module Alignment Terminology of SWs. Calculations of similarities extrinsic utilizing essentially productive and expressive properties of natural language, for example, morphological or syntactic. This module is implemented as a function Similarity_Extrinsic (Word1, Word2) in Algorithm 2. It takes care of calculating the similarity measures using the extrinsic resource WordNet.

There are several methods to calculate semantic links in WordNet, among these methods; we used the Jiang-Conrath measure of a mixed approach that combines the two approaches, one based on the distance and the other on the information content.

The principle of this similarity measure is to consider the shortest path between two concepts in the ontology, and weight those links from their semantic weight, by combining the information content of the lowest common ancestor of those concepts. The weight of semantic links takes into account the information content of concepts **[9].** This distance measure is probably the currently most used and most effective way to determine the semantic proximity between two concepts **[3].**

During the calculation of similarity between the concepts of a parameter of ontology with the same parameter of ontology, we obtain values of partial similarities that are calculated by standardized measures. There are several strategies in the literature to gather partial similarity values, and in our case, we take the maximum similarity value.

5.4.3. Planning Module. Arrange a Web service with inputs and outputs is very similar to a problem of automatic planning, and this is to find the correct order in the Web services automatic composition of SWs. Indeed, the role of planning is to find a sequence of actions, or plan to, from an initial state, reaching a goal state, expressed by the user.

This module deals with the determination and organization of tasks to be performed by the SWs, each of which can execute one or more tasks in a dialing plan. Planning is an essential module. Indeed, this is what determines who will participate in the SWs composition. Moreover, it determines the scheduling of SWs in the present composition.

The construction of a dialing plan is based on Algorithm 2, we have shown previously to find the similarity between the parameters of two different OWL-S. That is to say, the dialing plan starts from the first SWs as its parameters (Input, Precondition, Result and TextDesciption) are semantically similar with the same parameters of the user query, and then the output parameter of the first SWS, is semantically similar to the Input of another Web service. Then, the parameters (Input, Precondition, Result and TextDesciption) of this second Web service are semantically similar with the same parameters of another Web service. This process is continued until the last Web service, such as the Output parameter is semantically similar to the Output parameter of the user request.

At the end of this algorithm, we finally get a plan or several plans for automatic SWs composition. This process of building a dialing plan will be repeated several times to find other ways of automatic composition until we cannot construct a complete plan.

5.4.4. Execution Module. This module manages the execution of user-selected SWs. It offers the user the opportunity to follow in detail the process of executing SWs, and that by displaying the progress and interim and final results. And also the problems that occurred during the invocation.
Our Framework provides a graphical interface that facilitates the understanding of these data and their interpretation.

6. Experimentation and Evaluation

To test and verify the feasibility of our approach to automatic and distributed composition of SWs, a first version of a **DA5DCSWS** framework has been implemented in Java. Regarding the different similarity measures which are implemented in our architecture (Jaro-Winkler, Jiang-Conrath, and Hybrid), we used the Java API SIMPACK (Similarity Package) which is a comprehensive library that contains all measures important similarity.

Our **DA5DCSWS** framework is designed on top of a distributed infrastructure using the JXTA[4] 2.5 platform specifications. The reason we chose the JXTA 2.5 platform relies on the project to create a distributed platform that allows building simple, easy and effective set of machines for distributed applications to any device that could be a machine.

We use two API to WordNet 2.0; the API has JWNL feature extraction for each lemma the list of its corresponding synsets in WordNet and the ontology API JWordNetSim to measure the similarity between synsets in WordNet. And manipulate OWL-S ontologies, we used OWL-S API provides a Java API for accessing programs to create, read, write and execute OWL-S. The data used in our experiments are OWLS-TC[5] version 4.0. This database includes 1160 Semantic Web Services in OWL-S version 1.1.

The experimentation step consists to define the parameters to verify and to evaluate the performances of our framework **DA5DCSWS** by flooding the network and the scalability that is analyzed in terms of the total number of machines grows in a large scale network. The performances are devoted to measure:

- The total number of a request submitted to a machine necessary for network operating.
- The total number of requests produced, the number of requests forwarded, and the number of transmitted results on the network.

For experimentation purposes and because our query language is inspired from the parameters of OWL-S for helping the users, a query benchmark is created. The benchmark contains a set of queries, with different intends that are executed over the OWL-S. For every query, we computed, manually, the set of the relevant solutions for evaluating precision and recall:

$$\text{Precision} = \frac{I}{N} ; \text{Recall} = \frac{I}{R} \quad (1)$$

Where, I is the number of relevant solutions returned, N is the number of solutions in relevant solutions, and R is the number of solutions found by the system. Usually, Precision and Recall scores are not discussed in isolation. Instead, they are combined into a single measure, such as the F-measure, which is defined as follow:

$$\text{F} - \text{measure} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad (2)$$

The precision is the proportion of correct matches among all those returned by the system. This reflects the accuracy of the system. More value precision, the higher the noise in the output is reduced, and therefore the quality of the result is impressive. The recall is the proportion of correct matches returned by the system among all those who are correct (also including correspondence correct undetected). The recall measures the efficiency of a system. More value return, the higher the result covers all correct matches. In the following we explain the experimentation results.

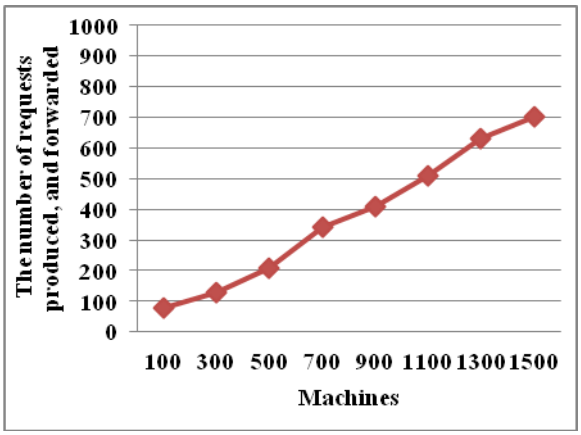

Figure 3. Evaluation the Performances vs Machines.

The experimentation has been performed with the number of machines that varies in the range [100-1500], with increments of 200. Figure 3 shows the performances of our framework **DA5DCSWS** in terms of the number of machines. We can see that, as the number of machines grows, the performances quite linearly.

To better demonstrate the effectiveness of our framework **DA5DCSWS**, a performance analysis has been performed to measure the variation in the execution time. The execution time has been

[4] **http://java.net/projects/jxta**
[5] **http://www.semWebcentral.org/**

measured upon the number of invokes that our framework, reflecting the number of discovering and composing SWs. This helps demonstrate the scalability of our framework. The execution time has been measured using The Eclipse Test and Performance Tools Platform[6].

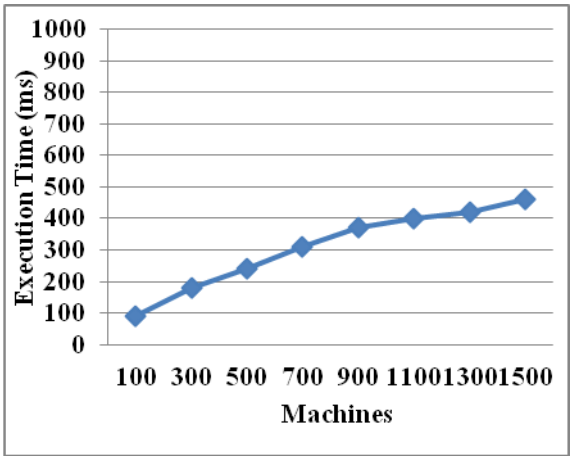

Figure 4. Performance Analysis vs Execution Time.

The graph in Figure 4 summarizes some of the performance test results. We notice that the execution time increases linearly with the increasing number of machines.

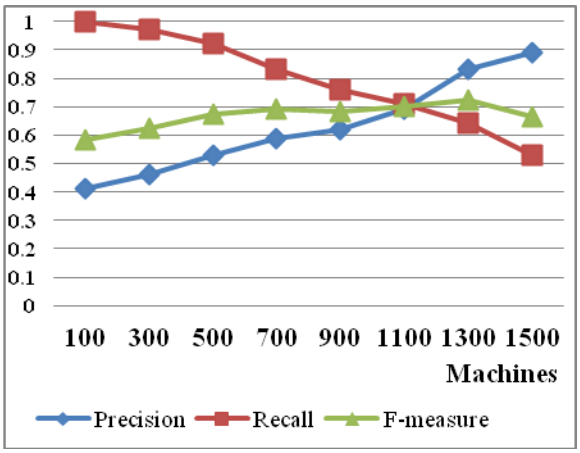

Figure 5. Precision, Recall and F-measure vs Machines.

Figure 5 describes the corresponding curves to the precision, recall and F-measure statistics obtained by applying our framework **DA5DCSWS** on the chosen SWs for different machines values. On the other hand, the recall decreases, and the precision increases smoothly as the number of machines grows. Figure 5 shows that best results of our framework are obtained on a network with 1100 machines, which correspond to the topmost value of the F-measure curve. Following these results, and other experiments and previous evaluations, all results obtained by our framework **DA5DCSWS** are encouraging. However, the performance measure of our framework shows that the results returned by our proposed framework are quality and good.

[6] **TPTP: http://www.eclipse.org/tptp/**

7. Conclusion and future work

In this paper, we presented a purely decentralized mechanism that supports the automatic discovery and composition and scalable runtime environments for distributed SWs. This mechanism is implemented as a distributed **DA5DCSWS** Framework. In this approach, we defined a strategy based on the alignment of OWL-S for the discovery and composition of SWs.

Our solution has two main characteristics: First, the request is routed to all machines on the network without knowing any of them. The second, the input and output of machines on the network is made without any requirements on other machines. Which will allow to design a composition model of a large positive, satisfying two criteria that are important: the accuracy and completeness.

To demonstrate the feasibility of our approach, we think it will be more appropriate to evaluate the proposed solution in a larger network of machines.

Such an assessment allows for tests measuring machine performance before and after using our prototype, as it allows for providing statistics on response time and bandwidth usage. We also intend in the near future to enrich our approach by using optimization techniques such as heuristics and Meta heuristics to select the best candidates SWs in terms of quality of service after the stage of automatic discovery. This work can be completed by the introduction of a formal semantics for verification of a composition.

8. References

[1] Boukhadra, A., Amrouche, H., Benatchba, K., Hidouci, W. K., Balla. A. (2010) Une approche sémantique pour la découverte automatique des services Web sémantiques. Workshop of Web Services, WWS'10, CERIST, Algeria.

[2] Baligand. F. (2008) Une Approche Déclarative pour la Gestion de la Qualité de Service dans les Compositions de Services. PhD Thesis, University of Nantes, France.

[3] Budanitsky. A., Hirst. G. (2006) Evaluating wordnet based measures of semantic distance. Computational Linguistics, 32(1), pages 13-47.

[4] Cohen, W., Ravikumar, P., Fienberg. S. (2003) A Comparison of String Distance Metrics for Name Matching Tasks. In Proceedings of KDD Workshop on Data Cleaning and Object Consolidation.

[5] Emonet. R. (2009) Semantic Description of Services and Service Factories for Ambient Intelligence. PhD Thesis University of Grenoble, France.

[6] Euzenat, J., Valtchev. P. (2004) Similarity-based ontology alignment in OWL-Lite. In Proceedings of 15th ECAI, Valencia, Espagne.

[7] Euzenat, J., Bach, T.L., Barrasa, J., Bouquet, P., Bo,

J.D., Dieng-Kuntz, R., Ehrig, M., Hauswirth, M., Jarrar, M., Lara, R., Maynard, D., Napoli, A., Stamou, G., Stuckenschmidt, H., Shvaiko, P., Acker; S.V. et Zaihrayeu, I. (2004) Stat of the art on ontology alignment, IST Knowledge Web NoE, Knowledge Web NoE.

[8] Euzenat, J., Shvaiko. P. (2007) Ontology Matching. Edition Springer, Berlin Heidelberg.

[9] Jiang, J., Conrath. D. (1997) Semantic similarity based on corpus statistics and lexicalterminology. In Proceedings of the International Conference on Computational Linguistics, RoclingX.

[10] Kreger. H. (2001) Web Service Conceptual Architecture, IBM Software Group.

[11] Lausen, H., Innsbruck. D. (2007) Semantic Annotations for WSDL and XML Schema, Édition Springer.

[12] Lopez-Velasco. C. (2008) Sélection et composition de services Web pour la génération d'applications adaptées au contexte d'utilisation. PhD Thesis University of Joseph Fourier, France.

[13] Ponge, J. (2008) Model-based Analysis of Time aware Web Services Interactions. PhD Thesis, University of Blaise Pascal France.

[14] Seco, N., Veale, T., Hayes. J. (2004) An intrinsic information content metric for semantic similarity in Wordnet. In Proceedings of ECAI'2004, the 16th European Conference on Artificial Intelligence, Valence, Espagne.

[15] Winkler, W. E. (2004) The state of record linkage and current research problems. Statistics of Income Division, Internal Revenue Service Publication.

[16] Yildiz, U. (2008) Décentralisation des procédés métiers qualité de services et confidentialité. PhD Thesis, University of Henri Poincaré France.

[17] Abi Lahoud, E. (2010) Composition dynamique de services application à la conception et au développement de systèmes d'information dans un environnement distribué. PhD Thesis, University of Bourgogne France.

[18] Reza Motahari-Nezhad, H., Saint-Paul, R., Casati, F.,Benatallah, B. (2011) Event correlation for process discovery from Web service interaction logs, Springer Verlag New York.

[19] C. Ba, M. Halfeld Ferrari, Martin A. Musicante. (2011) PEWS platform: a Web services composition environment. WEWST '11: Proceedings of the 6th International Workshop on Enhanced Web Service Technologies.

[20] Martinez, E., Lespérance, Y. (2004) Web service composition as a planning task: Experiments using knowledge-based planning. In: Proc. of the 14th International Conference on automated Planning and Scheduling (ICAPS 2004), Whistler, BC, Canada.

[21] Benatallah, B., M.Dumas, and Sheng, Q. Z. (2003) the self-serv environment for Web services composition. In IEEE Internet Computing.

