

## Web service discovery approach: application in e-healing domain

Mohamed Halim<sup>1</sup>, Nouha Adadi<sup>1</sup>, Mohammed Berrada<sup>2</sup>, Driss Chenouni<sup>1</sup>

<sup>1</sup>Informatique et Physique Interdisciplinaire (IPI) Laboratory, Higher Normal School, Sidi Mohamed Ben Abdellah University, Fez, Morocco

<sup>2</sup>Intelligence Artificielle, Sciences des données et Systèmes Émergents (IASSE) Laboratory, National School of Applied Sciences, Sidi Mohamed Ben Abdellah University, Fez, Morocco

### Article Info

#### Article history:

Received Jun 29, 2022

Revised Nov 3, 2022

Accepted Nov 29, 2022

#### Keywords:

Matchmaking algorithms

Online hospital problem

Semantic web

Service discovery

Web services

### ABSTRACT

The main objective of today's companies is to cope with rapid changes in the environment. For this, they must ensure the integration and interoperability of their applications. To manage and automate the life cycle of these applications, companies are adopting web services technology. The current problem is that the content of these web services cannot be processed automatically. Only humans can interpret its contents. The semantic web is a new vision of the web that promises to overcome this difficulty. The goal of this technology is to automate the retrieval, assembly, and selection of web services. In this post, we are interested in semantic detection of web services. The main problem is automatically discovering web services on request from clients. Against this background, we first describe the principle of the proposed detection mechanism and then present the designed matchmaking algorithm. Finally, we implement our proposed method. To verify our work, we run tests against various user requests and web service panels. As part of a case study, we consider an online hospital problem. This problem is a typical web service discovery scenario to which the concepts of our method are applied.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



### Corresponding Author:

Mohamed Halim

Informatique et Physique Interdisciplinaire (IPI) Laboratory, Higher Normal School

Sidi Mohamed Ben Abdellah University

Fez, Morocco

Email: mohamed.halim@usmba.ac.ma

## 1. INTRODUCTION

The semantic web [1] has attracted the attention of many researchers in recent years. It's a move to an intelligent web where information is no longer stored, but understood by computers, giving users what they're really looking for. The goal of the semantic web is to make vast amounts of information available, accessible and interpretable by machines, thereby eliminating the difficulties encountered on the web today (finding information, and services) [2].

The standards used by web services do not fully describe the functionality of the service and provide syntax definitions that cannot be understood by programs. Human intervention is required to make sense of the inputs, outputs, constraints, and contexts in which the service can be used. Semantic web technologies have provided a semantic layer to web services. The combination of semantic web and web services has created a new technology called semantic web services [3]. Semantic web services are required to automate the following functions:

- Automatic detection of web services [4]-[6].
- Automatic invocation of web services.
- The composition and automatic interoperability of Web services [7], [8].
- Monitoring the automatic execution of a web service.

Given the large number of different web services published in universal description, discovery, and integration (UDDI) [9] registries, this task becomes a challenge in locating web services. In fact, web service discovery is an emerging research area. Discovery is done first in the UDDI registry and is primarily based on searching the syntactical web services description language (WSDL) description of the web service. However, with the development of semantic web technologies, discovery techniques have become inherently semantic.

This article is concerned with the field of automatic discovery of web services. Indeed, we propose a discovery mechanism of web services guided by a Matchmaking algorithm based on semantic techniques that make discovery a dynamic and automatic filtering mechanism. The layout of this document is as; The second part provides an overview of web service discovery methods. In section 3, we focus on the proposed method. The fourth section is devoted to the results of the work and discussions. Section 5 presents the conclusions and future work.

## 2. THE PROPOSED METHOD

Semantic web service discovery approaches [4] depend, in particular, on the semantic matching method used, also known as "semantic service matchmaking". Current approaches to semantic matching of Web services can be classified into three main classes [5]: logical approaches, non-logical approaches and hybrid approaches. In the rest of this section, we will detail each of these classes and present the works that are interested in these approaches.

### 2.1. Logical approaches

This category of approaches [6], [10] uses the concepts of ontologies and logical rules to check the compatibility between the request and the services by comparing either the input/output (I/O), or the precondition/effect, or both. They are mainly based on semantic matching filters called degree of match (DoM) filters. Filters are determined based on subsumption relations between concepts within an ontology.

### 2.2. Not logical approaches

Not logical semantic matching exploits mechanisms for information retrieval based on different similarity measures such as cosine [11], Jaccard's coefficient [12], structured graph matching methods based on structural similarities [13], methods for calculating the distance of numerical concepts on ontologies like Wu and Palmer [14]. In some works they use a combination of several measures as is the case for the COV4SWS.KOM approach [15] which applies similarity measures from the domain of the semantic relation, namely the metrics of Lin [16] and Resnik [17] and uses a method of aggregating results following the weighted average strategy. We also cite the work of [18] which ensures non-logical semantic matching by combining a method of structural and textual similarity.

### 2.3. Hybrid approaches

Hybrid semantic matching uses a combination of logical and not-logical classes [19]. The idea is to overcome certain limitations of each of these two classes. Klusch and Kapahnke [20] present the OWLS-M hybrid approach where the matching mainly concerns the I/O parameters of the OWL-S services. The approach uses both logical reasoning by defining logical filters and not logical mechanisms by using information retrieval techniques (a threshold of syntactic similarity value). Other works like OWLS-MX3 [21] apply logical and not logical pairing. Another feature of the OWLS-MX3 is that it uses an adaptive approach (a support vector machine (SVM) model) to aggregate the matching results. Klusch *et al.* [22] propose a hybrid approach for semantic annotations for web services description language (WSDL) and extensible markup language (XML) (SAWSDL) services by applying an I/O pairing that includes the not logical COV4SWS approach, and a logical LOG4SWS approach. LOG4SWS was inspired by OWLS-MX3 logic filters. LOG4SWS uses bipartite graphs to find the optimal similarity between request and service descriptions. In addition, it exploits WordNet with a backtracking strategy in order to find similarities between operations or service parameters in the event that no similarity relationship is detected. The SAWSDL-MX2 [23] also presents a hybrid approach for I/O pairing of SAWSDL services. iSeM propose a hybrid approach. It inherits from its -MX predecessors, the technical characteristics including logical and non-logical matching techniques, and the SVM approach for aggregating the results of all measurements [24]. iSeM 1.0 has two additional features, namely that it is suitable for OWL-S and SAWSDL. SeMa<sub>2</sub> present a hybrid approach combining logical and non-logical matching techniques for OWL-S services [25]. In addition to logical comparison, SeMa<sub>2</sub> handles syntactic comparison of service names

and service descriptions. Currently, the best performing Matchmakers in terms of average accuracy for Web service discovery are iSeM (0.922) for services described with SAWSDL, and SeMa2 (0.877) for services described with OWLS.

### 3. METHOD

#### 3.1. Architecture for discovery of web services

In this section we describe our proposed web services discovery approach based on multi-agent system and ontologies. The suggested architecture is an extension of service-oriented architecture (SOA). A reading of the Figure 1 allows us to have a general idea about the structure of the proposed conceptual architecture. Indeed, as illustrated in the Figure 1, the architecture is composed of seven main elements: The user, the user interface agent, the discovery agent, the registration agent, the WS providers, the ontologies and the UDDI service registry.

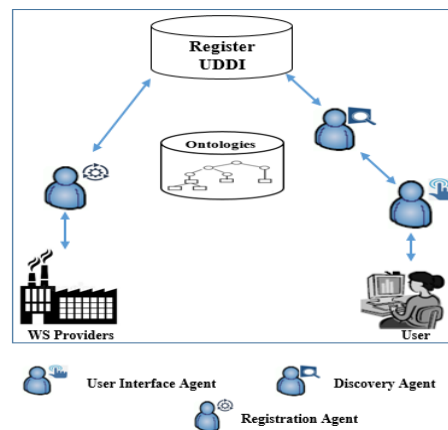


Figure 1. Proposed architecture for of web services

#### 3.2. Architecture description

##### 3.2.1. User interface agent

The user interface agent pilots the analysis phase, which constitutes a preparatory phase in the discovery cycle and presents the entry point for external requests to the system. The main role of this phase is to transform the specifications of the requested service into a semantic specification understood by the system. To do this, it follows a three-step process:

- Collect the user's needs, providing him with the right form to enter his request. Once the request is submitted by the user, an XML file containing the details of the information entered through the user interface is generated.
- Process the information contained in the XML file using a set of resonance rules as well as the domain ontologies to build the semantic query consisting of inputs, outputs, a reference on the domain ontology to be used (for example the tourist travel ontology).
- Encapsulate the request in a message and send it to the discovered agent.

Indeed, the architecture of the user interface agent is composed of 3 main modules (Figure 2). The three modules are as follows: the user communication module, the processing module, the inter-agents communication module.

##### a. User communication module

Its role is to collect the necessary information from the user and transfer it to the processing module. The inverse operation is also available, i.e. the module can receive results from the processing module to present them to the user. This module is controlled by a user interface (communication interface). The user chooses the terms to be used from a list proposed by a user interface, these terms are fed from the domain ontology. The user interface must accommodate the description of the functional characteristics of the request (the title of the service, its inputs and its outputs). Once the data is entered and the request is submitted by the user through the user interface, an XML file containing the elements entered by the user is generated. Figure 3 defines the structure of the XML file.

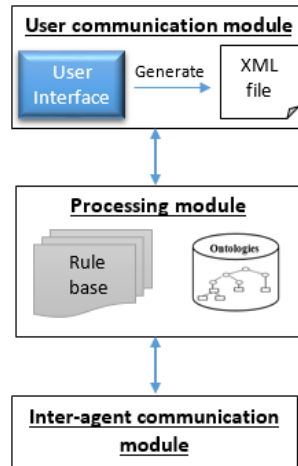


Figure 2. Architecture of the web service interface agent

```

<!-- Definition of the request-->
<request>
  <!-- Generation of the request identifier -->
  <id> texte </id>
  <description> texte </ description >
  <!-- Definition of the request parameters-->
  <params>
    <param>
      <type> in </ type>
      <value> val </ value>
    </param>
    <param>
      <type> out </ type>
    </param>
  </params>
</request>

```

Figure 3. User request xml file

### b. Processing module

This module constitutes the brain of the user interface agent, it makes it possible to reason using the different rules as well as the domain ontologies, in order to build the semantic request. It receives data from the user communication module as an XML file and translates it into a semantic query using the various translation rules stored in the rule base as well as domain ontologies. We present in Figure 4 the main translation rule used.

**Rule:** <param> tags with <value> sub-tags are query inputs, and <param> tags without <value> sub-tags are query outputs.

Figure 4. Translation rules

This rule is used to define the metric part (inputs, outputs) of the semantic query. Once the semantic query data is ready, the agent must save it in a registry and then determines if all the information needed to start the discovery phase is available. If so, it constructs a message and asks the inter-agent communication module to transmit it. If not, it requests additional information from the user which is transmitted to the user communication module; the latter will then take care of sending it to the user.

**c. Inter-agent communication module**

It receives from the processing module, requests for the transmission of messages to other agents. These messages are the semantic query. The request is encapsulated in a message and sent to the discovered agent.

**3.2.2. Discovery agent**

The role of the discovery agent is to determine the description of the web services satisfying the request sent by the user interface agent on the semantic level. This operation is guided by a matchmaking algorithm which is based on semantic techniques which make discovery a dynamic filtering mechanism. The internal architecture of the agent discovered as shown in Figure 5 is composed of two modules: inter-agent communication module and processing module requiring as input a request and producing as output a list of candidate services.

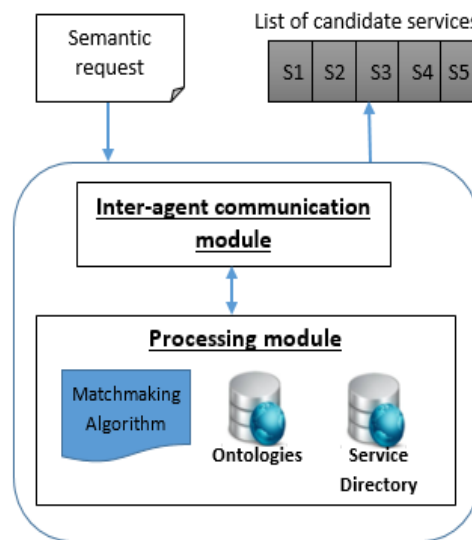


Figure 5. Architecture of discovery agent

**a. Processing module**

We choose to carry out the discovery phase based on the logical semantic matching method. This method uses the concepts of ontologies and logical rules to check the compatibility between the request and the services by comparing the I/O. The domain ontology corresponding to the request is already selected in the analysis phase, the tree structure of this ontology makes it possible to deduce generalization relations between the concepts. These relationships allow for flexible comparisons between offers and demands, based on semantic matching filters. Filters are determined based on subsumption relations between concepts within an ontology. The filters were defined on I/O in the work of [26] and are detailed in Table 1.

Table 1. Logical filter categories

Name	definition	Formal def
<i>Exact</i>	<i>Les sorties du service et de la requête sont équivalents.</i>	$Output(R) \equiv Output(S)$
<i>Plugin</i>	<i>Les sorties du service sont des généralisations des sorties de la requête.</i>	$Output(R) \sqsupseteq Output(S)$
<i>Subsume</i>	<i>Les sorties du service sont des spécifications des sorties de la requête.</i>	$Output(R) \sqsubseteq Output(S)$
<i>Fail</i>	<i>Aucun filtre ne peut être déterminé parmi ceux mentionnés ci-dessus.</i>	$Output(R) \not\equiv Output(S)$

The discovery agent applies a subsumption function (Figure 5) to compare the outputs of the request and the outputs of the service offered (Figure 6). The level of semantic correspondence between input parameters is assigned in the same way as for output parameters (Figure 7).

```

Function Subsume (E1: string, E2: string): Boolean
% This function returns true if E1 subsume E2 false otherwise
% E1 is an element of the Input or Output clause of the Offer
% E2 is an element of the Input or Output clause of Request
% A represents the ontology (in tree form)
% The following high-level functions are used:
% Father (E): returns the father of E in A
% Root (A): returns the root of A
Variables
CurrentVertex: AVertex % Vertex of A being examined
Ancestors: SetofVertices % The ancestors of E2
Beginning
Ancestors ← ∅
If E2 = root (A) Then
% E2 has no ancestor and cannot be subsumed
Ancestors ← ∅
Otherwise
CurrentVertex ← Father(E2)
Ancestors ← Father(E2)
While (CurrentVertex < >Root(A)) Do
VertexCurrent ← Father(VertexCurrent)
% "+" denotes adding a new item
% in the set Ancestors
Ancestors ← Ancestors + CurrentVertex
End While
End if
Subsume (E1 ∈ Ancestors)
End

```

Figure 6. Function subsume

```

Procedure degreeOfMatch(OutR, OutS : string )
% This Procedure returns comparison result
% OutR, OutS are the output of demand and Offer respectively
Variables Ch: string
Beginning
If OutR = OutS Then ch ← "Exact"
If Subsume(OutS, OutR) Then ch ← "PlugIn"
If Subsume(OutR, OutS) Then ch ← "Subsume"
Otherwise ch ← "Fail"
End if
Return ch
End

```

Figure 7. Output matching procedure

Relation (1) generalizes the comparison between the concept of the request R and the corresponding concept of the service offer S. This relation is assigns a score for each matching mode.

$$\text{match}(\text{OutR}, \text{OutS}) = \begin{cases} 3 & \text{if degreeOfMatch}(\text{OutR}, \text{OutS}) = \text{Exact} \\ 2 & \text{if degreeOfMatch}(\text{OutR}, \text{OutS}) = \text{PlugIn} \\ 1 & \text{if degreeOfMatch}(\text{OutR}, \text{OutS}) = \text{Subsume} \\ 0 & \text{else} \end{cases} \quad (1)$$

Suppose that we have m concepts of the service offer and n corresponding concepts of the request, the global match between the request R and the service offer S can be deduced by the relations (2) and (3).

$$\text{globMatch}(\text{OutR}, \text{OutS}) = \frac{1}{n} * \sum_{i=1}^{n*m} \text{match}(\text{OutR}, \text{OutS}) \quad (2)$$

$$\text{globMatch}(R, S) = (\text{globMatch}(\text{OutR}, \text{OutS}) + \text{globMatch}(\text{InR}, \text{InS})) / 2 \quad (3)$$

We retain the service S, if its global match (similarity) is greater than or equal to the threshold  $\theta$  (the threshold  $\theta$  is chosen by the user). Then we sort the results in descending order of the scores.

**b. Inter-agent communication module**

It receives the request from the user interface agent in the form of a message and following that, it calls the processing module. It also receives processing module message transmission requests. These transmission requests constitute responses to the requests received.

**3.3. Synthesis**

The proposed architecture for discovering web services (see Figure 1) is an extension of service-oriented architecture (SOA). This agent-based architecture leverages domain ontologies that integrate software components and facilitate automatic discovery of services. This allows you to implement filtering mechanisms (comparisons) between requests and offers, so you can implement more than simple equality.

In this work, we adopted a logical semantic discovery approach. Logical methods have been used in various papers in several research studies to verify that the I/O parameters of the service are compatible with the I/O parameters of the request. A common approach to logic-based matching is to define a set of rules (filters) that determine the types of logical relationships allowed between service I/O parameters and request I/O parameters. This type of matching considers the entire I/O signature, so it is not possible to calculate the degree of matching between services and requests. Evaluating the match between services and requirements requires a more flexible approach than rigid match filters. For these reasons, we propose a logical mapping method based on individual associations between services and query parameters.

Filters on individual links between the different description elements allow the similarity between a service and a request to be calculated in a more flexible and precise way compared to logical methods which define filters on the entire signature of the service. However, these filters can tolerate a large number of false positives at the level of the subsumption score, especially if the subsumption chain, linking the two concepts to be compared, is long. The use of these filters combined with the similarity calculation rules (relation (1), (2) and (3)) makes it possible to assign a numerical score which represents the semantic similarity between two concepts by respecting the logic defined by these filters.

**4. RESULTS AND DISCUSSION**

In this paper, we will consider, as an illustrative example, the problem of e-healing system. Suppose there are three web services: appointment service, calendar service, and consultation service published on the web. The input and output parameters are presented in Figure 8. Consider a user request R contains tree inputs and two outputs (Figure 9). Given the ontology fragment displayed in Figure 10. The results of applying the matching algorithm described above in the e-healing example are presented in Figure 11.

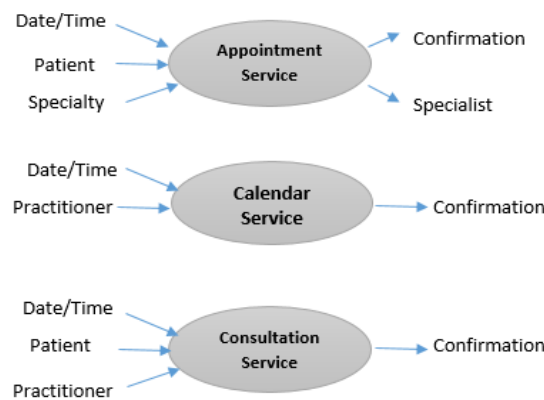


Figure 8. Appointment, diary and consultation service



Figure 9. Parameters of the request



Figure 10. A part of the practitioner ontology

- **Appointment Service**
  - Calculation of  $globalMatch(InR, InS)$ 
    - $degreeOfMatch(Date/Time, Date/Time) = Exact \rightarrow match(Date/Time, Date/Time)=3.$
    - $degreeOfMatch(Date/Time, Patient) = Fail \rightarrow match(Date/Time, Patient)=0.$
    - $degreeOfMatch(Date/Time, Spacialty) = Fail \rightarrow match(Date/Time, Spacialty)=0.$
    - $degreeOfMatch(Patient, Patient) = Exact \rightarrow match(Patient, Patient)=3.$
    - $degreeOfMatch(Patient, Date/Time) = Fail \rightarrow match(Patient, Date/Time)=0.$
    - $degreeOfMatch(Patient, Spacialty) = Fail \rightarrow match(Patient, Spacialty)=0.$
    - $degreeOfMatch(Spacialty, Spacialty) = Exact \rightarrow match(Spacialty, Spacialty)=3.$
    - $degreeOfMatch(Spacialty, Patient) = Fail \rightarrow match(Spacialty, Patient)=0.$
    - $degreeOfMatch(Spacialty, Date/Time) = Fail \rightarrow match(Spacialty, Date/Time)=0.$
    - $globalMatch(InR, InS)=(3+3+3)/3 = 3$
  - Calculation of  $globalMatch(OutR, OutS)$ 
    - $degreeOfMatch(Confirmatin, Confirmatin) = Exact \rightarrow match(Confirmation, Confirmation)=3.$
    - $degreeOfMatch(Confirmation, Specialist) = Fail \rightarrow match(Confirmation, Specialist)=0$
    - $degreeOfMatch(Practitioner, Specialist) = Subsum \rightarrow match(Practitioner, Specialist) = 1.$
    - $degreeOfMatch(Practitioner, Confirmation) = Fail \rightarrow match(Practitioner, Confirmation) = 0.$
    - $globalMatch(OutR, OutS)=(3+1)/2 = 2$
  - Calculation of  $globalMatch(R, S)$ 
    - $globalMatch(R, S)=(3+2)/2 = 2,5$
- **Calendar Service**
  - Calculation of  $globalMatch(InR, InS)$ 
    - $degreeOfMatch(Date/Time, Date/Time) = Exact \rightarrow match(Date/Time, Date/Time)=3.$
    - $degreeOfMatch(Date/Time, Practitioner) = Fail \rightarrow match(Date/Time, Practitioner)=0.$
    - $degreeOfMatch(Patient, Date/Time) = Fail \rightarrow match(Patient, Date/Time)=0.$
    - $degreeOfMatch(Patient, Practitioner) = Fail \rightarrow match(Patient, Practitioner)=0.$
    - $degreeOfMatch(Spacialty, Date/Time) = Fail \rightarrow match(Spacialty, Date/Time)=0.$
    - $degreeOfMatch(Spacialty, Practitioner) = Fail \rightarrow match(Spacialty, Practitioner)=0.$
    - $globalMatch(InR, InS)=3/3 = 1$
  - Calculation of  $globalMatch(OutR, OutS)$ 
    - $degreeOfMatch(Confirmatin, Confirmatin) = Exact \rightarrow match(Confirmation, Confirmation)=3.$
    - $degreeOfMatch(Practitioner, Confirmatin) = Fail \rightarrow match(Confirmation, Confirmatin)=0.$
    - $globalMatch(OutR, OutS)=3/2 = 1,5$
  - Calculation of  $globalMatch(R, S)$ 
    - $globalMatch(R, S)=(1+1,5)/2 = 1,25$
- **Consultation Service**
  - Calculation of  $globalMatch(InR, InS)$ 
    - $degreeOfMatch(Date/Time, Date/Time) = Exact \rightarrow match(Date/Time, Date/Time)=3.$
    - $degreeOfMatch(Date/Time, Patient) = Fail \rightarrow match(Date/Time, Patient)=0.$
    - $degreeOfMatch(Date/Time, Practitioner) = Fail$
    - $match(Date/Time, Practitioner)=0.$
    - $degreeOfMatch(Patient, Patient) = Exact \rightarrow match(Patient, Patient)=3.$
    - $degreeOfMatch(Patient, Practitioner) = Fail \rightarrow match(Patient, Practitioner)=0.$
    - $degreeOfMatch(Patient, Date/Time) = Fail \rightarrow match(Patient, Date/Time) = 0.$
    - $degreeOfMatch(Patient, Spacialty) = Fail \rightarrow match(Patient, Spacialty)=0.$
    - $degreeOfMatch(Spacialty, Practitioner) = Fail \rightarrow match(Spacialty, Spacialty)=0.$
    - $degreeOfMatch(Spacialty, Patient) = Fail \rightarrow match(Spacialty, Patient)=0.$
    - $degreeOfMatch(Spacialty, Date/Time) = Fail \rightarrow match(Spacialty, Date/Time)=0.$
    - $globalMatch(InR, InS)=(3+3)/3 = 2$
  - Calculation of  $globalMatch(OutR, OutS)$ 
    - $degreeOfMatch(Confirmatin, Confirmatin) = Exact \rightarrow match(Confirmation, Confirmation)=3.$
    - $degreeOfMatch(Practitioner, Confirmatin) = Fail \rightarrow match(Confirmation, Confirmatin)=0.$
    - $globalMatch(OutR, OutS)=3/2 = 1,5$
  - Calculation of  $globalMatch(R, S)$ 
    - $globalMatch(R, S)=(2+1,5)/2 = 1,75$

Figure 11. Application of matchmaking algorithm in e-healing services



Suppose the threshold is set to  $\theta = 1.5$ , so the Calendar service is excluded. We rank the appointment and consultation services in descending order so the best corresponding service to the request the appointment service.

## 5. CONCLUSION

This document brings together different contributions, in particular we presented the architecture of the proposed approach to web service discovery based on a multi-agent system composed of three agents: user interface agent, discovery agent, registration agent. Since there are three agents in the architecture, there are three phases in the life cycle of the proposed approach, in particular: the analysis phase, the discovery phase, the registration phase. Starting with the request sent by the user, it is a request reviewed by the user interface agent, the latter is endowed with a semantic analysis capacity allowing to translate the informal specifications of the users into a semantic request. To optimize the processing of this request, a discovery mechanism has been designed. This mechanism is based on a pre-calculation of the semantic correspondences relating to the abstract services compatible with the overall objective to be achieved. Finally, the last step is to publish in the services directory.




## REFERENCES

- [1] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific American*, vol. 284, no. 5, pp. 34–43, 2001, doi: 10.1038/scientificamerican0501-34.
- [2] L. Z. Varga and A. Hajnal, "Semantic web services description based on web services description," in *Position paper for the W3C Workshop on Frameworks for Semantics in Web Services*, 2005, pp. 3–7.
- [3] M. Klusch, P. Kapahnke, S. Schulte, F. Lecue, and A. Bernstein, "Semantic web service search: a brief survey," *KI - Künstliche Intelligenz*, vol. 30, no. 2, pp. 139–147, Jun. 2016, doi: 10.1007/s13218-015-0415-7.
- [4] M. C. Jaeger, G. Rojec-Goldmann, C. Liebetruh, G. Mühl, and K. Geihs, "Ranked matching for service descriptions using OWL-S," in *Kommunikation in Verteilten Systemen (KiVS), Berlin/Heidelberg: Springer-Verlag*, 2005, pp. 91–102.
- [5] N. Srinivasan, M. Paolucci, and K. Sycara, "Semantic web service discovery in the OWL-S IDE," in *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06)*, 2006, vol. 6, pp. 109b-109b, doi: 10.1109/HICSS.2006.431.
- [6] P. Bartalos and M. Bielikova, "Automatic dynamic web service composition: A survey and problem formalization," *Computing and Informatics*, vol. 30, no. 4, pp. 793–827, 2011.
- [7] Q. Z. Sheng, X. Qiao, A. V. Vasilakos, C. Szabo, S. Bourne, and X. Xu, "Web services composition: A decade's overview," *Information Sciences*, vol. 280, pp. 218–238, Oct. 2014, doi: 10.1016/j.ins.2014.04.054.
- [8] T. Bellwood, et al., "UDDI Version 3.0, Published Specification," *Oasis*, 2003.
- [9] K. Mohebbi, S. Ibrahim, M. Khezrian, K. Munusamy, and S. G. H. Tabatabaei, "A comparative evaluation of semantic web service discovery approaches," in *Proceedings of the 12th International Conference on Information Integration and Web-based Applications & Services - iiWAS '10*, 2010, p. 33, doi: 10.1145/1967486.1967496.
- [10] A. Huang, "Similarity measures for text document clustering," in *New Zealand Computer Science Research Student Conference, NZCSRSC 2008 - Proceedings*, 2008, pp. 49–56.
- [11] D. A. Jackson, K. M. Somers, and H. H. Harvey, "Similarity Coefficients: measures of co-occurrence and association or simply measures of occurrence?," *The American Naturalist*, vol. 133, no. 3, pp. 436–453, Mar. 1989, doi: 10.1086/284927.
- [12] S. K. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation*, vol. 60, no. 5, pp. 493–502, 2004.
- [13] Z. Wu and M. Palmer, "Verbs semantics and lexical selection," in *Proceedings of the 32nd annual meeting on Association for Computational Linguistics -*, 1994, pp. 133–138, doi: 10.3115/981732.981751.
- [14] U. Lampe and S. Schulte, "Self-adaptive semantic matchmaking using COV4SWS.KOM and LOG4SWS.KOM," in *Semantic Web Services: Advancement Through Evaluation, Berlin, Heidelberg: Springer Berlin Heidelberg*, 2012, pp. 141–157.
- [15] D. Lin, "An information-theoretic definition of similarity," *Icml*, pp. 296–304, 1998.
- [16] P. Resnik, "Using information content to evaluate semantic similarity in a taxonomy," *arxiv preprints*, Nov. 1995, [Online]. Available: <http://arxiv.org/abs/cmp-lg/9511007>.
- [17] P. Plebani and B. Pernici, "URBE: web service retrieval based on similarity evaluation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 11, pp. 1629–1642, Nov. 2009, doi: 10.1109/TKDE.2009.35.
- [18] M. Klusch, P. Kapahnke, and I. Zinnikus, "Hybrid adaptive web service selection with SAWSDL-MX and WSDL-analyzer," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5554 LNCS, 2009, pp. 550–564, doi: 10.1007/978-3-642-02121-3\_41.
- [19] M. C. Jaeger and S. Tang, "Ranked matching for service descriptions using daml-s," in *Proceedings of CAiSE'04 Workshops*, Riga Technical University, Riga, Latvia, 2004, pp. 217–228.
- [20] M. Klusch and P. Kapahnke, "OWLS-MX3: An adaptive hybrid semantic service matchmaker for OWL-S," *CEUR Workshop Proceedings*, vol. 525, 2009.
- [21] M. Klusch, and P. Kapahnke, "OWLS-MX3: An Adaptive Hybrid Semantic Service Matchmaker for OWL-S," in *Proceedings of 3rd International Workshop on Semantic Matchmaking and Resource Retrieval (SMR2) at ISWC*, 2009.
- [22] M. Klusch, and P. Kapahnke, "Semantic Web Service Selection with SAWSDL-MX," in *Proceedings of 2nd International Workshop on Service Matchmaking and Resource Retrieval in the Semantic Web (SMR2)*, 2008.
- [23] M. Klusch and P. Kapahnke, "iSeM: Approximated reasoning for adaptive hybrid selection of semantic services," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6089 LNCS, no. PART 2, 2010, pp. 30–44, doi: 10.1007/978-3-642-13489-0\_3.
- [24] N. Masuch, B. Hirsch, M. Burkhardt, A. Heßler, and S. Albayrak, "SeMa2: A hybrid semantic service matching approach," in *Semantic Web Services, Berlin, Heidelberg: Springer Berlin Heidelberg*, 2012, pp. 35–47.




- [25] M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara, "Semantic matching of Web services capabilities," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2342 LNCS, 2002, pp. 333–347, doi: 10.1007/3-540-48005-6\_26.

## BIOGRAPHIES OF AUTHORS






**Mohamed Halim**    received the Master's degree in computer networks from the Faculty of Sciences, Sidi Mohammed Ben Abdellah University, Fez, Morocco, in 2010. He is currently an administrator at ENS and PhD student in IPI Laboratory in USMBA Fez, Morocco. He can be contacted at email: mohamed.halim@usmba.ac.ma.






**Nouha Adadi**    received the Ph.D. degree in Computer Sciences from the Faculty of Science and Technology, Sidi Mohammed Ben Abdellah University, Fez, Morocco, in 2018. She is currently a member of IPI Laboratory and a professor of Computer Sciences at ENS, Fez, Morocco. She can be contacted at email: nouha.adadi@usmba.ac.ma.



**Mohammed Berrada**    received the Ph.D. degree in Computer Sciences from the Faculty of Sciences, Sidi Mohammed Ben Abdellah University, Fez, Morocco, in 2008. He is currently a member of IASSE Laboratory and a professor of Computer Sciences at ENSA, Fez, Morocco. He can be contacted at email: mohammed.berrada@usmba.ac.ma.



**Driss Chenouni**    received the Ph.D. degree in physics from the University of Montpellier II, France, in 1989. He is currently the director of IPI laboratory and the director of Higher Normal School (ENS) Fez, Morocco. He can be contacted at email: driss.chenouni@usmba.ac.ma.