Journal of Computer Science 10 (9): 1628-1641, 2014
ISSN: 1549-3636

doi:10.3844/jcssp.2014.1628.1641 Published Online 10 (9) 2014 (http://www.thescipub.com/jcs.toc)

A MOBILE AGENT BASED APPROACH FOR AUTOMATING "DISCOVER-COMPOSE" PROCESS OF SEMANTIC WEB SERVICES

Latrache, A., E. Nfaoui and J. Boumhidi

LIIAN Laboratory, Sidi Mohamed BenAbdellah University, Fez, Morocco

Received 2013-11-15; Received 2014-03-08; Accepted 2014-04-14

ABSTRACT

Nowadays, the focus is not only on how to exchange data between companies but also on how to exchange services between them or between companies-customers in order to minimize IT charges and increase their profit. Thus, web services are an adequate solution for the e-business thanks to their interoperability and reusability. The combination of web services and semantic technology, which are called Semantic Web Services (SWS), allows their discovery-composition-invocation process to be automatically performed by programs or intelligent agents. However, this process is still a challenging task that includes several issues such as the complexity of finding and composing distributed SWS. In this study we present a mobile agent based approach to discover and compose SWS in a distributed environment and extend some algorithms related to this field. The article reports examples and experimental results in order to illustrate as well as to assess the benefits of the proposed approach.

Keywords: Semantic Web Services, Semantic Web Services Discovery, Semantic Web Services Composition, Ontology, Mobile Agent, JADE

1. INTRODUCTION

A web service is a computer program for communication and data exchange between heterogeneous applications and systems in distributed environments i.e., a set of functionalities exposed on the internet or an intranet (either by or for applications). The main benefit of this vision is the facility of the application maintenance and interoperability to change a component (service) to replace it with another easily. Moreover, web services reduce the complexity of an application because the developer can focus on the service regardless to the rest of the application. There are two major classes of web services, the first type is named REST-compliant web services, in which a set of web services functionality is presented as a set of URI accessible via http protocol while in the second type, on which we will focus in our study, web services are presented as a set of remotely executable services on the basis of SOAP and WSDL standards. Thus, a complete description of the web service-by the means of the WSDL document as discussed by Chinnici *et al*. (2007)-leads to the effectiveness of the web service discovery and invocation process while the WSDL does not have the capacity to specify the meaning and semantic constraints of data involved in web services because it is based only on a syntactic description. This brings up the vision of semantic web services and semantic web service discovery which make use of the semantic web technologies to enrich the semantics of service descriptions.

Adding semantics to the web service descriptions remains as one of the promising axis in the semantic web area. Several alternatives have been proposed in the literature such as DAML-S by Ankolekar *et al*. (2002), WSMO by Bruijn *et al*. (2004), WSDL-S by Akkiraju *et al*. (2005) Martin *et al*. (2007) have proposed OWL-S which provide a set of sub ontology

Corresponding Author: Latrache, A., LIIAN Laboratory, Sidi Mohamed BenAbdellah University, Fez, Morocco

that one can use to describe the service, for example to describe what does the service do we can use a subontology called (profile.owl) and to describe how does the service work we can use a subontology called (process.owl) while the third subontology called (grounding.owl) can be used to describe how is the service invoked. Also, the OWL-S introduces one final subontology called (service.owl) to describe how these three pieces work together to completely describe a web service. Adding semantics to web services have several advantages, on the one hand, an explicit semantic description of their functionality understood by software agents as well as by human user's andon the other hand, a correct interpretation of information sent and received as discussed by Mannan *et al*. (2014).

Nowadays, web services aren't published in centralized registry such as Universal Description Discovery and Integration (UDDI) but in different hosting sites, the web service providers publish their services in their own web sites or in a private portals as discussed by Yu (2007). For that, web services discovery process becomes a challenging activity which aims at providing suitable tools to discover and invoke these services. Under these conditions, finding an appropriate web service may lead to several problems such as inaccuracy in the discovery and the incompleteness of the description may appear in the evaluation of the similarity mechanism for the discovery, which is called matching. Enrich the web service by semantic information has been studied in web service discovery to overcome these issues and improve the accuracy of the service discovery task. Another key in the semantic web services field is that they can be composed into more complex processes in order to achieve a given business goal. This is an important feature, since it allows atomic services to be combined in a flexible way to complete complex tasks. Thus, it can minimize the integration costs within the enterprise or allow developers to reuse existing services rather than developing new ones.

In this study, we propose a mobile agent based approach to discover and compose semantic web services in a distributed environment. Mobile agents-software programs with the feature of autonomy, social ability, learning and most importantly, mobility-represent a suitable technology to exploit autonomously the semantics of web services in order to supply consumers' applications. This study is organized as follows. Section 2 discusses some related works to the discovery and composition process of semantic web services. Section 3 describes our proposed approach and the related algorithms. Section 4 illustrates a concrete example to demonstrate the viability of our approach while the last section outlines the conclusions and future works.

2. BACKGROUND REVIEW

2.1. Background Review of Web Services Discovery Process

Due to the increasing number of web services over the web, the service requesters need a convenient tool to search the appropriate web service that meets their needs and expectations. This issue has motivated researchers to propose several mechanisms to select the appropriate and relevant web services for a service requester on the basis of the service descriptions and service requester's needs, i.e., improving the quality of the web service discovery process to improve the satisfaction of the web service requesters.

Several approaches have been proposed in literature to enable web service discovery. We classified these approaches from three perspectives as follows:

Semantic or syntactic based research perspective: The web service discovery mechanism depends on the web service description i.e., if this description is based just on syntactic description (WSDL document) or on semantic one (WSDL-S, OWL-S…).

Centralized or distributed architecture perspective: Web service centralized architecture means that the web Service descriptions are stored in a central repository such as UDDI or web portals while the distributed architecture means that these descriptions are stored in different hosting sites.

Functional or no-functional perspective: No-functional properties or also named Quality of Service (QoS) properties aim to satisfy expectations of service requesters such as how the selected web service is similar to the request or what is the time needed to find the appropriate web service.

A summary of some relevant approaches according to the previous perspectives is illustrated in **Table 1**.

Yu (2007) highlighted another key in web services discovery, especially in semantic matching algorithms, it is the degree of matching between two concepts on the basis of their semantic description. These degrees are classified into four classes, namely (Exact matching, Plug-in matching, Subsume matching and Fail matching).

The proposed techniques used for semantic web services discovery-as shown in **Table 1** don't assure all web services' requests especially when the requested service is a composite service, this issue remains as one of the main motivations of the semantic web services composition. A summary of the proposed techniques to overcome this issue is illustrated in the following section.

Table 1. Summary of web services 'discovery approaches

The approach	SD?	NFC?	CA?	Limits and advantages
User-centric design for Semantic discovery of mathematical web Sheeba *et al.* (2014)	Yes	Yes	Yes	It's a semantic discovery engine to allow efficient mathematical web service registration and discovery. The weak point in this approach services is that it requires the use of specific registration.
Using semantic information for distributed web service discovery. Canturk and Senkul (2011)	Yes	Yes	No	It's a web service discovery process in both business registries and private sites. The advantage of this approach is the use of distributed and domain-specific crawling that improves the QoS.
CoWS: An internet-enriched and quality- aware web services Search engine Li *et al*. (2011).	Yes	Yes	No	it's a novel web services search engine that it's not limited to the information in UDDI repositories or WSDL but it enriches web services information using the information captured from the internet.
Discovering semantic web services via advanced matching. Cuzzocrea and Fisichella (2011).	Yes	No	Yes	It's a graph-based method for matching not just atomic services but also composite OWL-S services.
Discovering Semantic Web services using SPARQL and intelligent agents. Sbodio *et al.* (2010)	Yes	No	Yes	Intelligent agents represent the main advantage of this approach in the other side, the main limitation is: It requires a specific description of web services.
WSExpress: A QoS aware search engine for web services Zhang *et al.* (2010)	No	Yes	Yes	A powerful search engine that provides three searching styles, which can adapt to the scenario of finding an appropriate Web service.
A QoS-aware model for discovery. Ye *et al*. (2009)	No	Yes	Yes (UDDI)	The main advantage of this approach is the use of a middleware web service called a broker for the service discovery.
QOS-aware web service discovery in P2P network. Xilu *et al.* (2009)	Yes	Yes	No	The main advantage of this approach is the use of functional and no-functional requirements in the discovery process
A recommender system for web services discovery registry environment. Sellami *et al.* (2009)	Yes	No	No	It's based on Peer-to-Peer (P2P) networks to structuring web services into groups, as result the web service discovery will increase fault tolerance and search efficiency but there is a lack of information security over the peers.
Discovering web services in search engines Masri and Mahmoud (2008)	No	No	Yes (UBR)	The proposed Web Service Crawler Engine(WSCE) for The UDDI Business Registry(UBR) gives a significant result in the verification and validation test.
Reputation-enhanced QoS-based web services Xu *et al.* (2007)	No	Yes		Yesit's a hybrid approach that uses several(UDDI) components for the service discovery.

SD: Semantic Description, NFC: No-Functional Criteria, CA: Centralized Architecture

2.2. Background Review of Web Services Composition Process

The main expectations of combining web services and semantic web technologies are to automate the following processes:

- Automatic discovery and invocation of web services: This phase aims at finding and invoking the desired web services automatically on the basis of the service requester's needs and the semantic description of the web services
- Automatic composition of web services: Web service requesters often require the use of several web services to achieve their requests; this phase aims at composing the necessary services to achieve these requests
- Automatic monitoring of the web service execution: This phase aims at verifying if the requested services are executed successfully and correctly

In the previous sections, we discussed the process of creating, publishing and discovering a web service. Thus, in the current section we will focus on the proposed mechanisms which are related to the web services composition field. These mechanisms can be classified into two categories namely static composition techniques and dynamic composition techniques.

The static composition techniques are based on business process. They can be divided into two categories; the first one is the orchestration in which a sequence of web services is executed according to a predefined template through orchestrations scripts as stated by (Albrechne *et al*., 2009); the second category is the choreography which is more collaborative. According to (Yeung, 2011; Hwang *et al*., 2011; Kuo, 2012), Choreography does not depend on a central orchestrator. Each web service involved in the choreography has to know exactly when to become active and with whom to interact in contrast to the orchestration techniques.

The dynamic composition techniques for web services are an aggregation of web services in order to solve a specific goal submitted by a user taking into account their preferences. Several approaches have been proposed to achieve this goal; the first solution is to generate a dynamic workflow on the basis of choreography or orchestration techniques as shown by Martinez *et al*. (2005) but this solution still has several limitations: It's not totally automatic and it requires the user intervention. This issue remains a need for intelligent approaches to automate web service composition, using intelligent mechanisms can solve this issue such as the artificial intelligence, (Bertoli *et al*., 2010) proposed an approach based on IA planning to automatically composing services. Also, using intelligent agents can solve this issue such as presented by Liu *et al*. (2006). Ouassila and Zizette (2011) proposed to combine these two mechanisms, i.e., intelligent agent and IA planning to ensure an efficient composition of services.

Several other technologies are proposed to enable the web services composition such as neural network by (Maamar *et al*., 2005), petri net by (Zhu and Du, 2010) and genetic algorithm by (Liu *et al*., 2010). However, all these approaches take into account just the user's requirements and neglect the service providers needs i.e., in increasing the profitability for the business providers, this point is highlighted by (Sandhya and Lakshmi, 2013). When it comes to a specific domain of application, there is no better or worse approach, as each one has its limitation. That's why until now there is no standard to automate web services composition.

The web services field has a great success in the web that's why researchers have proposed several approaches to increase the scope of use of this technology, the semantic web and the distributed environments are the better candidate to achieve this goal. Crasso *et al*. (2011) have proposed a platform for building prolog-based agents on the semantic web but this approach is limited just to the SWAM enabled sites, so it still the need of mechanism to discover the web services in other hosting sites. Yu (2007) proposed architecture to discover semantic web services in different hosting sites due to the use of a web crawler but he neglected the composite services. Thus, we identify a need of an approach (see section 3) that supports:

- Automatic discovery of semantic web services indifferent hosting sites
- Automatic composition of semantic web services

There is also a need for an approach that does not enforce the providers or the consumers of the semantic web services to use any specific techniques or annotations to publish or discover these services which is the case in the private central registry or in the approach proposed by Crasso *et al*. (2011). The web service providers can publish their services in their web sites and the consumers can easily invoke a search engine to discover these services, these are the main motivations of the proposed architecture in the following section.

3. PROPOSED ARCHITECTURE AND ALGORITHMS

3.1. Mobile Agent Architecture for Discovering and Composing SWS

This architecture is based on mobile agents to discover the requested SWS in different hosting sites and the graphs to achieve the composition process. The main components of this architecture are illustrated in **Fig. 1**.

User interface provides a set of features that can help the user to better express his request. This interface-as shown in **Fig. 3**-has several fields: The web service's name in which the user enters the desired value. Then, he selects the requested inputs and outputs on the basis of predefined domain ontology. Also, the user can give a little description of the desired service. After gathering user requirements, an OWL-S request is automatically generated.

Manager agent analyzes the owl-s request that contains the semantic description of the user request. This agent has the following roles:

- Decide if an index of the requested service is in the local repository or if he can compose the requested service on the basis of the local repository services
- If the requested service is not found in the local repository, the manager agent should send the request to the pool of mobile agents to search for the requested service in different sites, the number of involved agents depends on the complexity of the user request
- Update the data included in the repository

Local repository contains a set of useful information related to the collected web services such as the semantic description, the provider site link, These information aren't stored arbitrarily but according to a graph representation. Each graph node represents a web service and it's linked to another node (i.e., Web service) if the

outputs of the first one matches the inputs of the second one. The manager agent uses this representation to facilitate the composition process of the web services.

Several web services providers prefer to publish their services directly on their hosting sites, for that there is a need for a tool to discover and invoke these services. Our solution to overcome this issue is the use of mobile agents to collect the desired semantic web services descriptions.

Mobile agents collect the semantic web services that fulfill the manger agent request; this is done by visiting the web and finding the hosting web sites that contain SWS. The use of mobile agent instead of using a web crawler is recommended by Kumari and Rajput (2012), it has several advantages as follows:

- The use of several mobile agents increases the crawling speed
- It minimizes the network overhead
- It can adapt dynamically while solving a critical problem. This feature provides robust and fault tolerance capability. Other features such as mobility, social ability and learning allow the use of mobile agent in different domains like intelligent product as presented by (Boulaalam *et al*., 2013)

3.2. Proposed Algorithm to Discover and Compose SWS in a Distributed Environment

Figure 2 shows an UML activity-diagram that describes our proposed algorithm. Firstly, the user selects the inputs, outputs and the category of the desired web service; according to these criteria an OWL_S file is generated. Then the manager agent looking in the local repository, on the basis of this file, if there is any index of a web service that satisfies the user request, if not the manager agent sends the request to the mobile agents in order to visit different sites and search for the requested services. Finally, these agents send the discovery process result list (i.e., a set of web services) to the manager agent which decides if there is an atomic service that satisfies the user request; if not, he looks if there is any possible composition on the basis of these selected services. At this step, if the manager agent notices that there is a need of a novel web service to accomplish the composition process; a new discovery process is started, to never have an indeterminate loop at this step; the number of attempts is fixed by the user in the early stage.

3.3. Semantic Web Services Discovery in the Proposed Algorithm

The first main phase in this algorithm is invoking mobile agents to search the requested web service in different web sites. This phase can be divided into two sub-phases, the first one is crawling the web to find semantic web services while the second aims to select-based on the founded semantic web services-services that satisfy the user request.

3.4. Intelligent Crawler for the Semantic Web Services

If an index of the required SWS does not exist in the local repository, the next step in the proposed algorithm aims to discover the location of the required SWS by browsing different web sites using mobile agents. In this study we propose a focused web crawler-by adapting the crawler proposed by (Batzios *et al*., 2008)-to collect the SWS.The proposed crawler selects a random link from the URLs queue; then for each valid link i.e., a link which doesn't exist in robots.txt and it is not an image or PDF document, the crawler verifies if it contains SWS that may satisfy the user's request. The main steps of the proposed algorithm are summarized as follows:

```
Inputs: (starting URL in queue, user' request)
Begin:
  While (URLs queue is not empty){
   Get first URL from queue
     If (the actual link is valid)
       Visiting the actual link
         If (the OWL_S descriptions exist)
           For each SWS
             {Invoke the matchmaking algorithm
               If the SWS is a candidate to satisfy the
               user request
                 Store an index of the selected SWS
             }
  Else
    Extract links from the actual page
    Add linked URLs in queue
  }
End;
```

There are not many semantic web service descriptions on the internet for collection and most pages the crawler visits would have nothing to do with semantic web services. Therefore, precious time and system resources could be wasted. A possible solution to this problem is to come up with a better set of seed URLs. For instance, we can use Swoogle, Sindice or other semantic search engine to discover the documents that use the OWL-S upper ontology; the results returned by this engine are then used as an initial set of seed URLs.

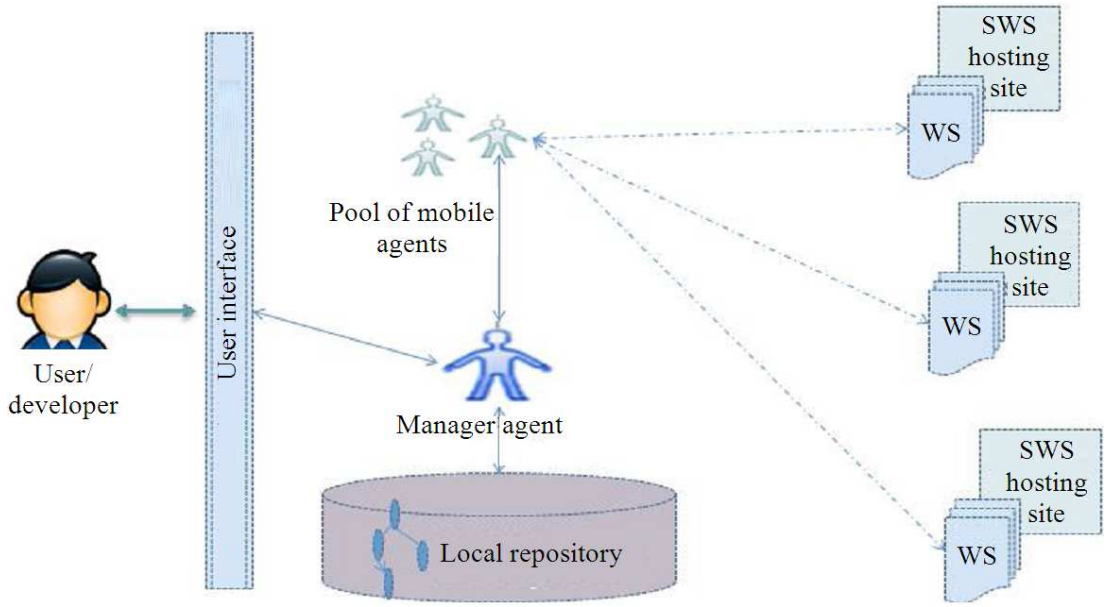

Fig. 1. Mobile agent architecture for discovering and composing SWS

3.5. Matchmaking Algorithm

The second role of mobile agents is to verify if the founded SWS is a candidate to satisfy the user request, by invoking a matchmaking algorithm. The main key concept in any matching algorithm is the concept of degree of match; it describes the degree of matching between two concepts. More precisely, the matching between two concepts is not syntactic, but it is based on the relation between these concepts in their OWL ontologies. A matching algorithm normally recognizes the following four degrees of matching between two concepts:

- Exact matching: Two concepts exactly match each other if they are the same concepts; i.e., if concept A subsumes concept B and concept B subsumes concept A
- Plug-in matching: If concept A subsumes concept B, then concept A is a set that includes concept B; in other words, concept A could be plugged in place of concept B
- Subsume matching: This is similar to the afore mentioned situation, except that the request's concept subsumes the advertised concept. In this case, the service may not satisfy the needs of the request, but it is still a possible candidate
- Fail matching: In this case, there is no exact matching andthere is no subsumption relationship, either. In other words, the two concepts are unrelated and a failed match is returned

The matchmaking algorithm used in our approach is based on Berdjouh and Okba (2009); the main function in this algorithm is to verify if a concept E1 subsume a concept E2. This function is used to verify the matchmaking degree between inputs and outputs of the selected services. A prototype of this function is summarized as follows:

```
FUNCTION Englobe (E1: Concept, E2: Concept): Booléen
VARIABLES ActualNode: The actual node
                  Parents: A set of previous nodes to E2
                  A: Represents the ontology (tree form)
BEGIN
 Parents←∅
  If E2 = racine(A) then
     Parents←∅
   else
    ActualNode←Previous(E2)
     Parents←Previous(E2)
     While(ActualNode<>Racine(A)) do
          ActualNode←Previous(ActualNode)
           Parents←Parents+ActualNode
     End while
    end if
  Englobe← (E1∈ Parents)
END;
```

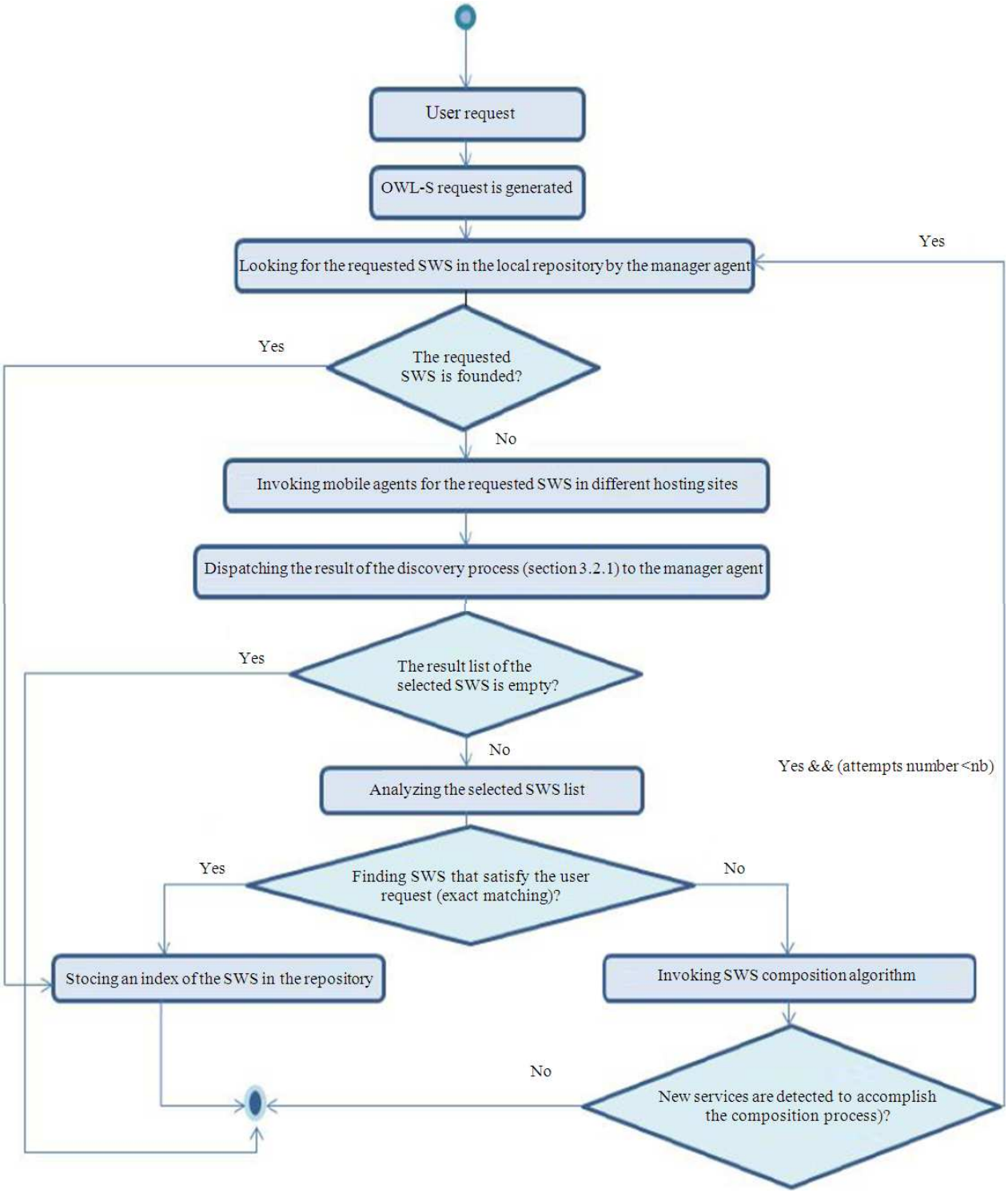

Fig. 2. Proposed algorithm to discover and compose SWS in a distributed environment.

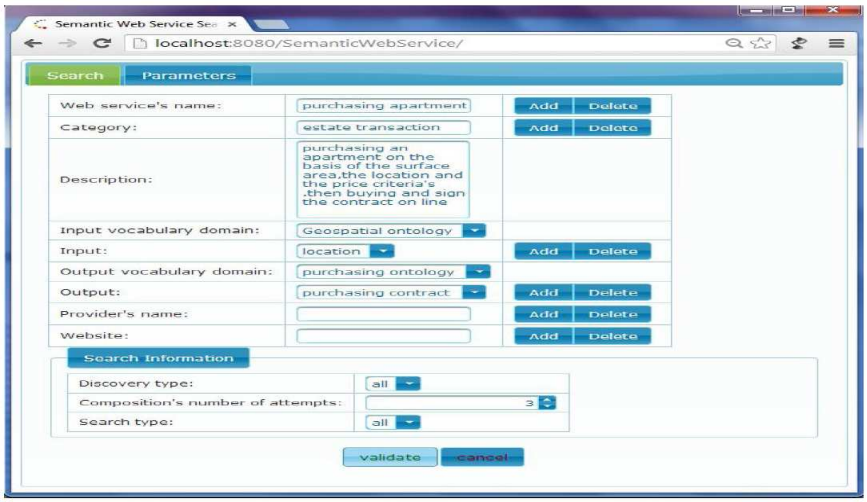

Fig. 3. The user interface of our application to discover and compose SWS

```
<owl:Ontology rdf:about="">
  <owl:imports rdf:resource="&profile;"/>
  <owl:imports rdf:resource="&the_profile;"/>
  <owl:imports rdf:resource="&profileHierarchy;"/>
</owl:Ontology>
<profile:Profile rdf:ID="purchasing_apartment">
<profile:serviceName> purchasing apartment</profile:serviceName>
<profile:textDescription>
Purchasing an apartment on the basis of the surface area,the location  and the price criteria's, then buying and sign the contract on line
</profileHierarchy:purchassing_apartment_Information  rdf:ID="purchassing_apartment_request">
</profile:textDescription>
<profile:hasInput>
<profile:hasInput rdf:resource="&geospatial;#location"/>
<profile:hasInput rdf:resource="&areas;#surface"/>
<profile:hasIOutput rdf:resource="&purchassing;#contract"/>
</profileHierarchy:purchassing_apartment_Information>
</rdf:RDF>
```

Fig. 4. A part of the OWL-S request

Then a score is assigned for each mode of matching: Exact (score = 3), Plugin (score = 2), Subsume (score = 1), Fail (score = 0), the following Equation generalizes the comparison between the concept of the service offer C^O (Offer) and C^D (Demand) corresponding to the query concept:

$$\text{Match}\left(C_i^D, C_i^O\right)=\begin{cases}3 & \text{if} \quad C_i^D = C_i^O \\ 2 & \text{if} \quad C_i^D \subset C_i^O \\ 1 & \text{if} \quad C_i^D \supset C_i^O \\ 0 & \text{else}\end{cases}$$

Suppose we have m concepts in the description of the service offering and m corresponding concepts in the description of the application, the similarity or the overall match between the demand (request) D and O supply can be derived by taking the sum score of the pair of concepts, so the requested service is the one that have the highest score of match:

$$\text{Similarity}\left(D, O\right) = \sum_{i=1}^{m} \text{Match}\left(c_i^D, c_i^O\right)$$

For example, a user looks for a web service which has as inputs “apartment “and “region” and a single output that is “price”. Let’s suppose that there are three web services S1, S2 and S3 published on the web.
Functional parameters (inputs, outputs) are:

- S1 has two inputs “building” and “region” and a single output “price
- S2 has two inputs “region” and “apartment” and a single output “price
- S3 has two inputs “product” and “provider” and a single output “price

To select which SWS-between S1, S2 and S3-that better satisfy the user request, we calculate the matching degree between their inputs and outputs on the basis of their ontologies, a fragment of the used ontology in our example is illustratted in **Fig. 6.**

In our example, the first step aims to calculate the inputs matching degree between desired SWS and the offered SWSs as follows:

S1:

- Apartment→building, mode = Plug-in, score = 2, Total = 2
- Apartment→region, mode = Fail, score = 0, Total = 2
- Region→building, mode = Fail, score = 0, Total = 2
- Region→Region, mode = Exact, score = 3, Total = 5

Inputs’ total score for S1= 5

S2:

- Apartment→Region, mode = Fail, score = 0, Total = 0
- Apartment→apartment, mode = Exact, score = 3, Total = 3
- Region→region, mode = Exact, score = 3, Total =6
- Region→apartment, mode = Fail, score = 0, Total = 6

Inputs’ total score for S1 = 6

S3:

- Apartment→product, mode = Fail, score = 0, Total = 0
- Apartment→provider, mode = Fail, score = 0, Total = 0
- Region→product, mode = Fail, score = 0, Total = 0
- Region→provider, mode = Fail, score = 0, Total = 0

The second step aims to calculate the outputs matching degree between desired SWS and the offered SWSs as follows:

S1: price→price, mode = Exact, score = 3, Total =3
Outputs’ total score = 3
S2: price→price, mode = Exact, score = 3, Total =3
Outputs’ total score = 3
S3: price→price, mode = Exact, score = 3, Total =3
Outputs’ total score = 3

The last step aims to calculate the global matching degree between desired SWS and the offered SWSs as follows:

- S1: Total score (total score of inputs + outputs total score) = 5 + 3 = 8→ good
- S2: Total score = 6 + 3 = 9, →the best
- S3: Total score = 0 + 3 = 3, →Not good
- So, the S2 is considered the best one that correspond to the user request

3.6. Semantic Web Services Composition in the Proposed Algorithm

This phase depends on the result of the previous one, so two alternatives are possible:

If we find an exact matching of the requested semantic web service; it means that the user request is fulfilled.

If we find a plug-in or subsume matching; we invoke an algorithm of SWS composition, the mains steps of this algorithm are:

- This algorithm has as input the set of the plug-in and subsume matching of SWS
- The graph construction phase matching inputs and outputs of services are logically connected in a way that they form a potential workflow
- Each SWS has a table of service information that contains four columns namely input, output, canal and jumps. The canal represents the service names of the SWS that can be used to get the required output while the jumps represent the distance (the number of jumps) between the initial node and the desired node (that have the required output)
- The global table that contains all information included in these tables is created by the manager agent
- On the basis of this table, the manager agent seeks if there is a canal (path in the graph) that has as input the requested input and as output the requested output
- If the canal exist: The manager agent send the set of the SWS involved in this canal to the user
- If the canal does not exist: It means that the user request can’t be fulfilled either by atomic WS or composite WS

3.7. Advantages of the Proposed Approach

When it comes to compare this approach with other approaches proposed in the literature to discover and compose distributed semantic web services, the most successful approach is the one that present the maximum advantages. Using mobile agents and graphs in our approach leads to overcome several lacks and providing several benefits. Thus, the main advantages of our approach are:

Several functionalities are provided in a convivial user interface which helps the user to better express his request. On the basis of this request an OWL-S document is generated that contains the semantic description of the requested web service. So, the obtained document is semantically richer than a document generated by other search engines which is based just on a keyword research.

The use of the graph representation to store the web services information contributed to improve the response time to a user request and to find an optimum path for the composite services.

It's a distributed architecture that enables the discovering of web services in several hosting sites. Thus, it's not limited to the UDDI which has several limits as discussed by Yu (2007). It's based on mobile agents to achieve this goal; the advantages of using mobile agents are discussed in the previous parts.

It does not enforce the providers or the consumers of SWS to use a specific annotations or technologies either than OWL-S, to perform their tasks.

4. EXAMPLE OF SWS APPLICATION IN E-BUSINESS

Web services are emerging as a promising technology for e-business; it allows a company to link its applications with those of its partners, customers and suppliers via the Internet. Therefore, businesses can view and use partners' information as if it were their own. Not only that but companies also can link their own applications within the enterprise, even those coded in different programming languages, to reduce redundancy and increase efficiency. For example, a corporation might link inventory programs with accounting applications so that changes made in one area automatically affect data in other departments.

The use of semantic web services instead of simple web services can bring several benefits as explained in the previous sections. Therefore, we proposed an approach for using semantic web services in a distributed environment that can be applied successfully in e-business domain, an example illustrated this case is demonstrated in the following section.

4.1. A Case-Study: Real Estate Company

A conventional scenario of purchasing an apartment in a real estate company includes several steps:.

- The costumer arrives at the real estate company and expresses his request
- A receiving employee serves the request and proposes several apartments to the costumer
- The costumer chooses the appropriate apartment that satisfies its request
- The costumer signs the contract offered by the notary agency

This is just a simplified scenario of purchasing an apartment but it shows clearly how this operation is a time consuming task that involves the coordination of several component within and outside the company such as the notary agency. The same scenario (task) can be carried out using SWS and our approach, in which companies services must be published in their web sites as SWS, for example a service that offer a contract for the costumer and the company can be a SWS published in the notary agency web site while a service that offers a list of available apartments to buy in a specific region can be a SWS published in the estate company web site. The task of developing or consuming those SWS is beneficial in term of time consuming and company profit than in the conventional scenario. For that, we will develop SWS of purchasing apartment to demonstrate this result on the basis of our approach.

Before developing the required service we must verify if there is any similar service or if it can be composed using several atomic services by fulfilling the different fields in our application as shown in **Fig. 3**.

The interface contains several fields such as:

- Web service' name: The name of the required service in our case is "purchasing apartment", it corresponds to the <profile:serviceName> in the owl-s profile
- Category service: Defines the category of the service (using ontology or any international categorization scheme), it corresponds to the <categoryBag> in the OWL-S profile
- Description: The description of the required web service, it can be used to get more information about

the requested web service and consequently find the suitable one, it corresponds to the <profile: Textdescription>in the owl-s profile

- Input Output: Defines successively the input and output of the required web service on the basis of predefined ontologies. It corresponds to <profiles: Hasinput> and <profile: Hasoutput> in the owl-s process
- Provider name: The name of the provider of the required web service if it is known <profile: Contactinformation>
- Site: A web site of the possible provider(s), it serves as addresses to be visited by the mobile agents to look for the desired web services
- Discovery type: Contains four options (all, subsume matching, exact matching, Plug-in matching, subsume matching), it defines the desired types of WS discovery techniques
- Composition attempts: During the composition process, sometimes a need of a new web service(s) is detected, so the number of attempts represents the number of these services that must be fixed to never come with an infinite loop
- Search type: Contains four options namely local repository, different-web sites, offered-web sites or all, it defines where the developer wants to search for the adequate WS

After fulfilling the interface' fields, an OWL_S file shown in (**Fig. 4**) is sent to the manager agent as a request. Then, the manager agent looks in the local repository if there is any SWS that corresponds to this request (by matching semantically the SWS name, the inputs, the outputs…). In our case, there is no SWS in the local repository that satisfy directly this request, there is just a "list appartment" service that has as outputs the list of candidate apartments addresses with their prices and as inputs a specific location and having a specific surface while we search for a purchasing apartment service that has as inputs (location and surface) and as output (the purchasing contract).

According to the proposed algorithm (**Fig. 2**). the next step is invoking the mobile agents to search for the requested SWS in different web sites. Then, the list of this operation is sent to the manager agent which analyses this list according to the matchmaking algorithm, the contract_purchasing is selected.

A new web service is detected to accomplish the composition process, this service has as inputs price or/and address and has as output an e-payment receipt. Thus, a new request is sent to the mobile agents to look for this service. An appropriate service is found in an e-payment site.

The composition process is done successfully and we found a set of SWS that satisfy the user request by combining three different SWS namely list-appartment (saved in the local repository), e-payment (offered by epayment site) and contact-purchasing-flat (offered by a notary agency web site). The result of these steps- which are carried out automatically- is illustrated in **Fig. 5**.

The final step is publishing the qualified SWS in the hosting estate' web site for the probable use by external developers and link this service with a suitable graphic web interface which can be used by the customers to accomplish the apartment purchasing' transaction.

Building an e business application based on SWS and our approach can bring several benefits to the companies such as:

- Thanks to their interoperability, WSs can lead to decrease the company charges especially the services integration' costs
- Reusability a characteristic of WS that makes them suitable for building composite web services. It can reduce the development tasks 'charges

4.2. Implementation

The prototype has been implemented using Java Agent Development Framework (JADE) from CSELT, Turin, Italy.

JADE is a middle-ware that could be used to develop agent-based applications in compliance with the FIPA specifications for inter-operable intelligent multiagent systems. The SWS dataset was implemented using OWL_S editor which is an easy-to-use editor for creating OWL-S services supported by the university of Malta Department of Computer Science and A.I. This tool is divided into three main parts Creator, Validator and Visualizer. **Table 2** shows our implementation environments for the prototype.

Table 2. Implementation environments for the prototype

Implementation environment	JADE 4.3.0
Implementing language	Java
DBMS	Oracle
SWS editor	OWL-S editor
Ontology editor	Protégé
Operating system	Windows 7

Fig. 5. The final result of the discovery and composition process

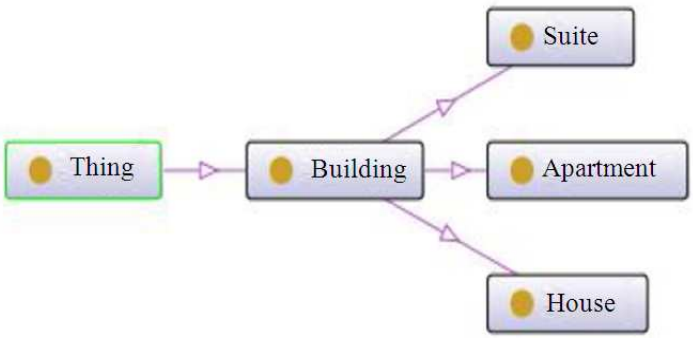

Fig. 6. A fragment of building ontology

5. CONCLUSION

In this study we have presented an architecture to discover and compose SWS in a distributed environment, unlike the most of the proposed architectures in the literature which are based on a central repository to store the web services. In the other side, the few approaches that handle this issue and taking into account the distribution of the web services require the use of a specific annotations to describe the web service or to find it (such as UDDI) while we can't enforce the web service' publisher to use these specific annotations, the only annotation to consider for him is the standard semantic one.

This architecture is based on mobile agents which are able to migrate within a network to interact with locally accessible resources and exploit the semantics of web services to supply consumers' applications. In the other hand, the graphs are used to compose several SWS to perform a user request if it is needed. We have also enhanced this architecture by adopting some related algorithms.

This study has highlighted also the interest of using SWS in several domains especially in e-business domain. Using SWS allow a company to link its applications with those of its partners, customers and suppliers via the internet. Not only that but companies also can link their own applications within the enterprise to reduce redundancy and increase efficiency. Therefore, it can minimize IT charges and increase the company profit.

There are some issues that are subject of future work. On one hand, the result of the discovery process is to return all services that are "sufficiently similar" to the request, this means that there is a lack of an accurate measure to determine whether a "sufficiently similar" service is the desired one for the user. To tackle this issue, we will extend our approach by adding the nonfunctional parameters in the selection process. Also, using recommender system' techniques can be beneficial at this stage. On the other hand, when the number of services or ontologies complexity increase, scalability and interoperability issues are highlighted; this issue will be taken in consideration in our future work.

6. REFERENCES

Akkiraju, R., J. Farrell and J. Miller, 2005. Web service semantics-wsdl-s. University of Georgia and ² IBM,

Albrechne, A., P. Fuhrer and J. Pasquier, 2009. Web services orchestration and composition.

Ankolekar, A., M. Burstein, J.R. Hobbs, O. Lassila and D. Martin *et al*., 2002. DAML-S-web service description for the semantic web. Proceedings of the 1st International Semantic Web Conference on Semantic Web, Jun. 9-12, Springer Berlin Heidelberg, Italy, pp: 348-363. DOI: 10.1007/3-540-48005-6-27

Batzios, A., C. Dimou, A.L. Symeonidis and P.A. Mitkas, 2008. Bio crawler: An intelligent crawler for the semantic web. Expert Syst. Applic. J., 35: 524-530. DOI: 10.1016/j.eswa.2007.07.054

Berdjouh, C. and K. Okba, 2009. An agent-based approach for discovering web services. Proceedings of the CIIA of CEUR Workshop, (CCW '09), pp: 547-547.

Bertoli, P., M. Pistore and P. Traverso, 2010. Automated composition of web services via planning in asynchronous domains. Artificial Int. J., 174: 316-361. DOI: 10.1016/j.artint.2009.12.002

Boulaalam, A., E.H. Nfaoui and O. Beqqali, 2013. Intelligent product based on mobile agent to accelerate the new product development process. J. Comput. Sci., 9: 856-865. DOI: 10.3844/jcssp.2013.856.865

Bruijn, J.D., C. Bussler and J. Domingue, 2004. Web service modeling ontology. National University of Ireland, Galway.

Canturk, D. and P. Senkul, 2011. Using semantic information for distributed web service discovery. Inter J. Web Sci., 1: 21-35. DOI: 10.1504/IJWS.2011.044080

Chinnici, R., J.J. Moreau, A. Ryman and S.Weerawarana, 2007. Web Services Description Language (WSDL) version 2.0 Part 1: Core language. W3C.

Crasso, M., C. Mateos, A. Zunino and M. Campo, 2011. SWAM- A logic-based mobile agent programming language for the semantic web. Int. J. Expert Syst. Applic., 38: 1723-1737. DOI: 10.1016/j.eswa.2010.07.098

Cuzzocrea, A. and M. Fisichella, 2011. Discovering semantic web services via advanced graph-based matching. Proceedings of the IEEE International Conference, on Systems, Man and Cybernetics, Oct. 9-12, IEEE Xplre Press, Anchorage, AK, pp: 608-615. DOI : 10.1109/ICSMC.2011.6083778

Hwang, S.Y., W.F. Hsieh and C.H. Lee, 2011. Verifying web services in a choreography environment. Proceedings of the IEEE International Conference, on Service-Oriented Computing and Applications, Dec. 12-14, IEEE Xplre Press, Irvine, CA, pp: 1-4, DOI: 10.1109/SOCA.2011.6166254

Kumari, V. and P. Rajput, 2012. Web crawler based on secure mobile agent. Res. J. Comput. Syst. Engineer., 3: 419-423.

Kuo, W., 2012. A π-calculus based approach for web services composition in choreography environment. MSc Thesis, Department of Information Management.

Li, M., J. Zhao and L. Wang, 2011. CoWS: An internet-enriched and quality-aware web services search engine. Proceedings of the IEEE International Conference on Web Services, Jul. 4-9, IEEE Xplre Press, Washington, DC., pp: 419-427. DOI: 10.1109/ICWS.2011.49

Liu, S., P. Küngas and M. Matskin, 2006. Agent-Based Web Service Composition with JADE and JXTA. Norwegian University of Science and Technology.

Liu, SL., YX. Liu, F. Zhang and G.F. Tang, 2010. Genetic algorithm for QoS-aware dynamic web services composition. Proceedings of the International Conference on Machine Learning and Cybernetics, Jul. 11-14, IEEE Xplre Press, Qingdao, pp: 3246-3251. DOI: 10.1109/ICMLC.2010.5580691

Maamar, Z., I. Younas and D. Benslimane, 2005. On self-coordinating web services using similarity and neural networks. Proceedings of the IEEE International Conference on e-Technology, e-Commerce and e-Service, Mar. 29-Apr. 1, IEEE Xplre Press, pp: 171-176. DOI: 10.1109/EEE.2005.95

Mannan, M.J., M. Sundarambal and S. Raghul, 2014. Selection of ontology for web service description language to ontology web language conversion. J. Comput. Sci., 10: 45-53. DOI: 10.3844/jcssp.2013.45.53

Martin, D., M. burstein, D. Mcdermott, S. Mcilraith and M. Paolucci *et al*., 2007, Bringing semantics to web services with OWL-S. World Wide Web, 10: 243-277. DOI: 10.1007/s11280-007-0033-x

Martinez, A., M. Patino-Martinez and R. Jimenez-Peris, 2005. ZenFlow: A visual web service composition tool for BPEL4WS. Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Comput, Sept. 20-24, IEEE Xplre Press, pp: 181-188. DOI: 10.1109/VLHCC.2005.74

Ouassila, H. and B. Zizette, 2011. An agent based architecture (using planning) for dynamic and semantic web services composition in an EBXML context. Int. J. Data Base Manage. Sys., 1: 22-22. DOI: 10.5121/ijdms

Masri, E. and Q.H. Mahmoud, 2008. Discovering web services in search engines. IEEE Int. Comput., 12: 74-77. DOI: 10.1109/MIC.2008.53

Sandhya, P. and M. Lakshmi, 2013. Strategic composition of semantic web services using SLAKY composer. Proceedings of the 2nd International Conference on Advances in Computing and Information Technology, Jul. 13-15, Springer Berlin Heidelberg, Chennai, India, pp: 411-420. DOI: 10.1007/978-3-642-31600-5_40

Sbodio, M.L., D. Martin and C. Moulin, 2010. Discovering semantic web services using SPARQL and intelligent agents. Web Semant. Sci. Services Agents World Wide Web, 2: 310-328. DOI: 10.1016/j.websem.2010.05.002

Sellami, M., S. Tata and Z. Maamar, 2009. A recommender system for web services discovery in a distributed registry environment. Proceedings of the 4rth International Conference on Internet and Web Applications and Services, May 24-28, IEEE Xplre Press, Venice/Mestre, pp: 418-423. DOI: 10.1109/ICIW.2009.68

Sheeba, A., A. Chandrasekar and V. Shanthi, 2014. User-centric design for semantic discovery of mathematical web services. Am. J. Applied Sci., 11: 639-647. DOI : 10.3844/ajassp.2014.639.647

Xilu, Z., W. Bai and W. Gengyu, 2009. Qos-aware web service discovery in P2P network. Proceedings of the 2nd IEEE International Conference on Broadband Network and Multimedia Technology, Oct. 18-20, IEEE Xplre Press, Beijing, pp: 650-654. DOI: 10.1109/ICBNMT.2009.5347830

Xu, Z., P. Martin, W. Powley and F. Zulkernine, 2007. Reputation-enhanced QoS-based web services discovery. Proceedings of the IEEE International Conference on Web Services, Jul. 9-13, IEEE Xplre Press, Salt Lake City, UT, pp: 249-256. DOI: 10.1109/ICWS.2007.152

Ye, G., C. Wu, J. Yue and S. Cheng, 2009. A QoS-aware model for web services discovery. Proceedings of the 1st International Workshop on Education Technology and Computer Science, Mar. 7-8, IEEE Xplre Press, Wuhan, Hubei, pp: 740-744. DOI: 10.1109/ETCS.2009.700

Yeung, W.L., 2011. A formal and visual modeling approach to choreography based web services composition and conformance verification. Expert Syst. Applic., 38: 12772-12785. DOI: 10.1016/j.eswa.2011.04.068

Yu, L., 2007. Introduction to the semantic web and semantic web services. 1st Edn., Chapman and Hall/CRC ISBN-10: 1584889330, pp: 368.

Zhang, Y., Z. Zheng and M.R. Lyu, 2010. WSExpress: A QoS-aware search engine for web services. Proceedings of the IEEE International Conference on Web Services, Jul. 5-10, IEEE Xplre Press, Miami, FL., pp: 91-98. DOI: 10.1109/ICWS.2010.20

Zhu, C.Y. and Y. Du, 2010. Application of logical petri nets in web service composition. Proceedings of the International Conference on Mechatronics and Automation, Aug. 4-7, IEEE Xplre Press, Xi'an, pp: 913-918. DOI: 10.1109/ICMA.2010.5589966