Expert Systems with Applications 42 (2015) 3833–3842

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

# Semantic web service discovery system for road traffic information services

J. Javier Samper Zapater [a,*], Dolores M. Llidó Escrivá [b], Francisco R. Soriano García [a], Juan José Martínez Durá [a]

[a] Computer Sciences Department, University of Valencia, Avda La Universitat S/N, Burjassot, Valencia, Spain
[b] Languages and Systems Department, University Jaume I, Campus Riu Sec, Castellón, Spain

ARTICLE INFO

*Article history:*
Available online 16 January 2015

*Keywords:*
Semantic web services
Matchmaking
Information retrieval
Road traffic information systems
Knowledge discovery

ABSTRACT

We describe a multi-agent platform for a traveller information system, allowing travellers to find the road traffic information web service (WSs) that best fits their requirements. After studying existing proposals for discovery of semantic WS, we implemented a hybrid matching algorithm, which is described in detail here. Semantic WS profiles are annotated semantically as an OWL-S and also the traveller request is represented as a OWL-S profile. The algorithm assigns different weights and measures to each advertised WS profile parameter, depending on their relevance, type and nature. To do this we have extended Paolucci's Algorithm and adapted it to our scenario. We have added new similarity measures, in particular, the use of the 'sibling' relationship, to improve the recall, allowing relevant services to be discovered by the users yet not retrieved by other algorithms. Although we have increased the similarity concept relations, we have improved the run-time using a pre-process filter step that reduces the set of potentially useful WS. This improves the scalability of the semantic matching algorithm.



## 1. Introduction

Organizations operate in a global environment in which national business compete within a global economy and society. Information and data can be used as a strategic advantage by providing customers with specialized services tailored to suit their individual needs. Web Services (WS) enables the interoperability between information systems in real time. WS semantically annotated are handy plug and play services accessible on the Internet. As the number of WS over Internet is rising, the number of clients demanding to reuse services is also increasing. These clients ask for services using queries specified by functional (FPs) and non-functional parameters (NFPs).

In Samper-Zapater, Llidó, Durá, and Cirilo (2013) we proposed a Traveller information architecture as a multi-agent platform system (Fig. 1) to allow advertisement, request, discovery, invocation and execution of WS within it.

In this paper, we are going to describe a hybrid matching algorithm to automatic discovery road traffic information WSs. User requirements and WS, were both represented as a WS profile semantically annotated with the same ontologies.

An Automatic WS Discovery is an automated process to locate WS that can provide a particular class of service capabilities, considering the constraints specified by the client. To achieve this goal, it is quite important to match appropriately the different properties or capabilities of the desired WS, specially the outputs. The use of matching semantics is very important in this context, so we are going to use OWL-S descriptions[1] (Martin et al., 2005; Pedrinaci, Maleshkova, Zaremba, & Panahiazar, 2012).

To help providers annotate semantically WS profiles with OWL-S, in Samper, Tomás, Carrillo, and do PC Nascimento (2008) we proposed the OntoService web tool. This tool is also useful to help clients to specify their requirements semantically as an OWL-S profile and it facilitates the user interaction establishing a semi-automatic guidance. Due to the lack of specific ontologies on this domain, road traffic information, we have developed two ontologies (Samper-Zapater, Zambrano, & García, 2006): The first one, a Road Traffic Ontology to specify the road information concepts and relations. The second one, a Road Traffic Services Categorization Ontology (Fig. 2) to categorize the services on our scenario.

* Corresponding author. Tel.: +34 963543567; fax: +34 963543550.
*E-mail addresses:* jsamper@irtic.uv.es (J.J. Samper Zapater), dllido@uji.es (D.M. Llidó Escrivá), francisco.r.soriano@uv.es (F.R. Soriano García), juan.martinez- dura@uv.es (J.J. Martínez Durá).
*URL:* http://irtic.uv.es (J.J. Samper Zapater).

[1] <http://www.w3.org/Submission/OWL-S/>

http://dx.doi.org/10.1016/j.eswa.2015.01.005
0957-4174/

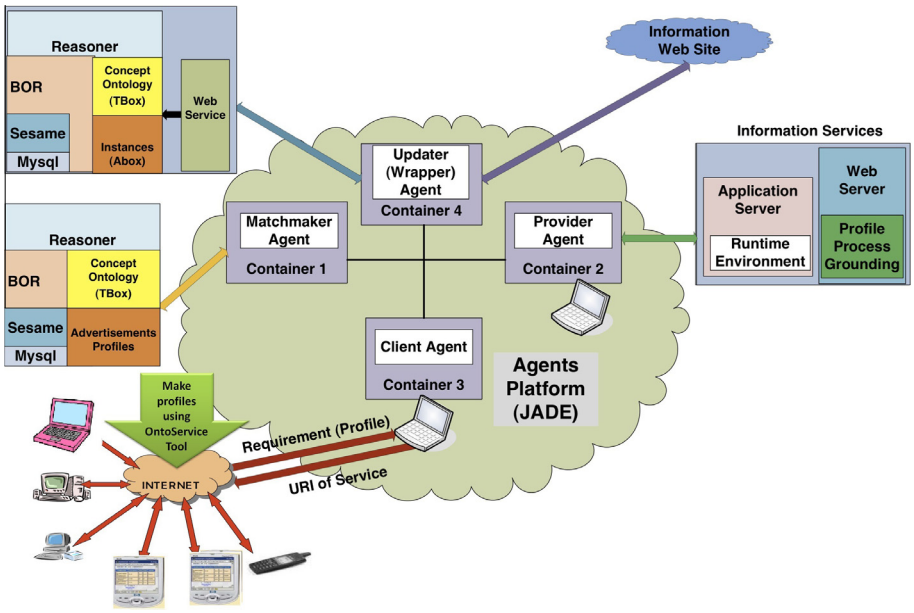



**Fig. 1.** Traveller information architecture.

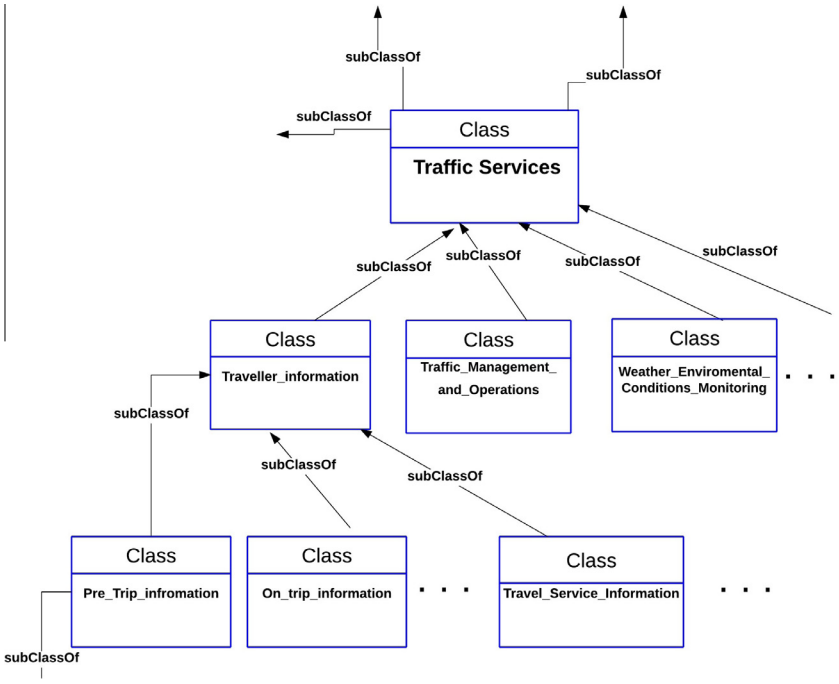



**Fig. 2.** Part of road traffic services categorization ontology.

Finally, the hybrid algorithm has been validated through the development of a software prototype of our multi-agent platform traveller architecture. It has allowed the assembly of all elements (road traffic ontologies, matching algorithm, information web services). Using the prototype, the algorithm has been evaluated and the functionality verified as a whole by studying the relationships between each one of the elements and results.

The paper has been organized as follows: Section 2 provides an overview of the background. Section 3 introduces our traveller information architecture. Section 4 deals with the matchmaking algorithm proposed in our system. Section 5 presents the results and discussions with some test cases. Section 6 analyzes the run-time improvements. And finally, in Section 7, some conclusions and future work are exposed.

## 2. Background

As indicated by Trastour, Bartolini, and Gonzalez-Castillo (2001), service profiles describe the capabilities of a service and, thus, they can describe both the capabilities of the services offered by providers (advertisements) and those expected by clients (requests). The formal OWL-S ontology enables users and software agents to automatically discover, invoke, compose, and monitor

Web resources offering services, under specified constraints (Kamaruddin, Shen, & Beydoun, 2012).

The group of researchers in CMU presented LARKS (Sycara, Klusch, Widoff, & Lu, 1999) as a dynamic matchmaking system among heterogeneous software agents, where WS were seen as frames and their slots could be used to describe the essential attributes of a service: input, output, inConstraints and outConstraints. The concepts used in textcolorcyanWS descriptions were defined through a language of concept description called ITL (Information Terminological Language). This language was used by mediator agents to match service application agents with service agent providers to solve the requested agent requirements. The integration architecture described in CMU MatchMaker,[2] used Information Retrieval, Artificial Intelligence and Software Engineering techniques to process the syntactic and semantic similarity between the capability descriptions of WS described in DAML-S (DARPA Agent Markup Language Services).

Paolucci, Kawamura, Payne, and Sycara (2002) proposed a semantic matchmaking algorithm of WS capabilities based on the use of DAML-S ontology. In this algorithm a match between an advertisement and a service request, consists of a similarity degree obtained by matching all the parameters of the request with those of the advertisement, for outputs and inputs. The implemented match is IO matching. A parameter from the provider matches with one from the request profile if there is a conceptual relation in the ontology. Four different measures are defined with a matching degree organized along a discrete scale from Exact (3), Plugging (2), Container (1) to Fail (0). This algorithm (Paolucci et al., 2002; Payne, cci, & Sycara, 2001), henceforth Paolucci's Algorithm, proposed from the CMU is a flexible matchmaking algorithm commonly used by the majority of researchers as a point of departure for their systems.

Tomaz, Labidi, and Wanghon (2003) presented an extension of Paolucci's Algorithm using preconditions. So that, they match as much as possible services providers that meet the client pre-conditions match as far as possible. Li and Horrocks (2004) extended Paolucci's Algorithm differentiating exact and subclass relations, assigning them different scores, and adding an intersection degree to find services when an advertisement and a request are compatible, that is, they have something in common.

Vu (2005) proposed to improve the matching algorithm by using the quality of service (QoS) NFP. Jaeger and Rojec-Goldmann (2005) proposed the use of the service category NFP. Other researchers as Tsai, Hwang, and Tang (2011), use a hybrid model, which considers the description of services by the users and by the providers, the labels and categories, as well as the QoS parameters. The hybrid approach uses deductive methods that integrate distance calculation. Recently, to solve the Paolucci's Algorithm problems of greedy Khater and Malki (2014) propose a shortest path based approach.

In García, Ruiz, and Ruiz-Cortés (2012) the authors proposed the inclusion of a preprocessing stage to improve the run-time. This preprocessing stage discards services which are not related to requirements and preferences stated by the user, reducing the search space before actual discovery. Using SPARQL queries, they discard all the services that do not contain the concepts related to the inputs and outputs requested by the client. This preprocessing step improves the run-time, but it has a penalty on precision.

One of the bottleneck on Semantic Web service discovery are the end users. As stated by Sangers, Frasincar, Hogenboom, and Chepegin (2013) the users are not technicians and it is necessary a bridge between keywords and the user natural language, so they need to make a word sense disambiguation to annotate automatically the user request. In our system thanks to OntoService, our end users and providers can specify the queries and services, respectively, as a WS description using the same ontology. Furthermore, using user profiles and repositories, the previous found services are retrieved.

## 3. Traveller information architecture

In Samper-Zapater et al. (2013) a multi-agent platform of a Traveller information system is presented (Fig. 1). The platform has been implemented using JADE Yu (2012) which follows FIPA standards as a matchmaking model. It allows the complete livecycle: advertisement, request, discovery, invocation and execution. In our prototype, both Client and Provider Agents use the 'OntoService' tool, proposed by Samper et al. (2008) to annotate semantically WS. The following agents components can be identified in our Traveller information architecture:

| | |
|---|---|
| Provider Agent: | It represents the WS within the platform. The provider using this agent annotates semantically the WS and registers the corresponding OWL-S profile on the repository. |
| Client Agent: | It is the representation of the client within the platform. It helps the user in the information search process allowing to specify the user requirements as an OWL-S profile using OntoService. When the user selects a WS from the result set, this agent also helps the user to retrieve the information from the WS. |
| Matchmaker Agent: | This agent requires a matchmaking algorithm to retrieve the best profiles that fit the user requirements. It is explained in Section 4. |
| Updater Agent: | This wrapper agent replicates web applications or static web pages as a semantic WS, and wraps the information from the web periodically. |
| Directory Facilitator: | This agent is necessary in multi-agents platforms. It can be considered the 'yellow pages' of the system. Other agents register their services on this agent or search for other agents. |

The basis of this architecture is a knowledge-base model stored on the Sesame Repository. Sesame[3] supports as query languages SPARQL and SeRQL. In our system we use SeRQL (Sesame RDF Query Language) to access to the knowledge-base and to allow interoperability with a logic description reasoner, BOR. The reasoner is based on logical descriptions and has support for inferences about instances and concepts. In a knowledge-base we can identify two parts: TBox (terminological part), formed by terms and their relations and ABox (instantial part) which will be formed by instances of the concepts defined in the TBox. An ABox is an assertion component, a fact associated with a terminological vocabulary within a knowledge-base. ABox are TBox compliant statements of that vocabulary. The terms ABox and TBox are used to describe two different types of statements in ontologies. Together, ABox and TBox statements make up a knowledge-base. For the characterization of web services on the knowledge-base, the starting point has been the OWL-S ontology and the road traffic ontology concept, both stored in our knowledge-base as concept on the Tbox. And each web service

[2] <http://www-2.cs.cmu.edu/~softagents/daml_Mmaker/daml-s_matchmaker.htm>

[3] <http://www.openrdf.org/doc/sesame/users/ch06.html>

profile semantically annotated, could be stored on our knowledge-base as instance of a concept on the Abox.

## 4. MatchMaking algorithm

As other research on the WS discovery systems, we determined to use a hybrid algorithm taking as a starting point Paolucci's Algorithm, and trying to improve it in our scenario. After implement Paolucci's Algorithm and testing it, some questions have arisen: what kind of filtering could be applied before the matching process to improve the run-time? Which parameters of the semantic WS profile could be used? Which similarity measures should be used?

After a literature background analysis we decided to improve the algorithm focusing on four aspects:

1. Reduce the run-time: Using a pre-process step to filter the existing provider profiles.
2. Improve precision using non-functional parameters which add more information about user requirements.
3. Improve the recall selecting different relationship concepts.
4. Improvements in the management of results, adapting the weight of the different measures.

A hybrid approach uses deductive methods that integrate distance calculation, so it combines the advantages of both methods. Different matching measures depending on the nature of the parameters could be used. Our hybrid matchmaking algorithm (Algorithm 1, see Section 4.3), assigns a weight to each similar WS and returns an ordered list, where the first element is the most similar WS profile to the requested profile. The Algorithm 7 applies different measures according to their parameters nature, due to the relevance of the parameter. For example, outputs are more relevant than inputs and between the NFP the geographic radius and quality of service parameters are considered the best (semantic exact matching). After some tests we have tuned the similarity degree of each type of measure and type of parameter (see Section 4.1 FPs and Section 4.2 NFPs).

**Algorithm 1.** Hybrid Matching Algoritm

```
HYBRID_ALGORITHM (REQ,ADV)
{
 Candidates = FilterByCategory (REQ,ADV)
 ListCouples = Combiner (REQ,Candidates)
 WeightedCandidatesOrdered = Null
 forall Couples in ListCouples
 {
  Weight = MatchingWeight (Couples)
  if Weight!=Null
    (WeightedCandidatesOrdered =
  Sort(Weight,WeightedCandidatesOrdered)
  }
 return WeightedCandidatesOrdered
 }
```

### 4.1. Similarity matching for FPs

FPs represent WS functional behavior. The OWL-S Profile represents two aspects of the functionality of the service: the information transformation (represented by inputs and outputs) and the state change produced by the execution of the service (represented by preconditions and effects).

As pointed by Paolucci et al. (2002) an advertisement matches a request, when the advertisement describes a service that is "sufficiently similar" to the requested service. A flexible matchmaking algorithm uses different similarity measure according to the relation between the concepts and assigns a similarity degree taking into account the relevance of the matching. Exact relationship is not enough, since rejecting similar concepts, you can discard some valid services. Exact matches are, of course, preferable to any another, and Fail is the lower level and it represents an unacceptable result.

Algorithm 2. Flexible Matching Algorithm

```
MeasureFP(Pairs)
{
Weight = 0
Fails = 0
for (REQ,ADV) in Pairs
  Degree = SimilarityDegree
  if Degree = 0 fails+=1
 else Weight+=degree
return(Weight,Fails)
}
```

**Algorithm 3.** Similarity degree

```
SimilarityDegree(C,P)
{
  degree = 0
  if Exact(C,P) then weight+=6
  else if SubClasssOf(C,P) then weight+=5
  else if SubClasssOf(C,P) then weight+=4
  else if Subsume(C,P) then weight+=3
  else if Subsume(C,P) then weight+=2
  else if Sibling(C,P) then weight+=1
return(weight)
}
```

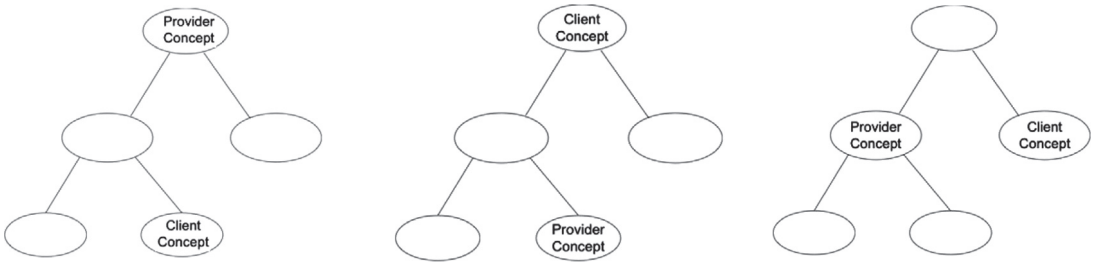
In our Algorithm 2 as in Paolucci's, we use a flexible matching algorithm, but we use 7 grades of similarity using the concept semantic relation defined in Section 4.1.1. The Algorithm 2 shows the procedure to assign the similarity measure for FPs: inputs or outputs. The input of this algorithm is a list of couples of concepts, obtained by combining the inputs or outputs of the WS profiles from the provider (ADV) and the client (REQ). The algorithm returns a weight and the number of matching failures. Algorithm 3, for each couple concept C (from the client) and P (from the provider) using semantic relationships of concepts, obtains a similarity degree that is organized along a discrete scale from Exact (with 6) to Fail (with 0).

#### 4.1.1. Concept semantic relation

FPs in a web service are represented by concepts, with a lot of synonyms, categories and granularities. Considering C the concept that stands for the client parameter and P for the provider, after some test experiments we have decided to use the next 7 similarity measures to improve the flexibility of the algorithm:

Exact: The concept defined by the client and the provider is the same.

CsubclassP: Within the taxonomic concepts tree, the distance between the demanded concept and that offered by the provider is equal to 1, i.e the concept described by the client is a direct subclass of the




(a) PsubsumeC (b) CsubsumeP (c) CsiblingP

**Fig. 3.** Concept semantic relations.

concept defined by the provider. In this case the provider offers a more general concept than the client requested. The client concept is more restrictive but is included in the one supplied by the provider.

PsubclassC: The concept described by the provider is direct subclass of that requested by the client, in other words, the provider offers a more restrictive concept than the requested by the client.

PsubsumeC: The concept described by the client is found within the subtree of concepts that hangs from the one defined by the provider (Fig. 3a). It would be equivalent to the Plug-in defined in Paolucci's Algorithm but with a little difference because we applied maximum depth to the search. We consider that concepts at distances greater or equal to 3 levels have hardly no semantic relation, due to the way in which our concept hierarchies are built.

CsubsumeP: The concept described by the provider is found within the concept subtree that hangs from that defined by the client (Fig. 3b), in other words, it is the opposite to the previous one, and is equivalent to the subsume degree defined in Paolucci's Algorithm.

CsiblingP: The concept provided by the client coincides in some restriction with those of the provider's concept (Fig. 3c), and besides, the offered concept as well as the one demanded by the client hang from the same father, that is, they are 'sibling' concepts.

Fail: When none of the above cases takes place, in other words the provider's concept and the client's concept have no relation whatsoever.

Fig. 4 shows the query statement in SeRQL to implement the CsiblingP measure similarity. This query takes the restrictions of the client concept, and then browse through the restrictions of the siblings of this class, to find common restrictions. If the concepts are siblings and they coincide in some of the restrictions it returns a record with the URI's from the provider and the consumer services.

### 4.2. Similarity measure for NFPs

As in the approach of a hybrid model proposed by Tsai et al. (2011), we decided to use some NFPs to improve the results of our system. Fundamental research related to selection and discovery of suitable Web services focuses on the analysis of quality of service properties in order to make the best service selection (Parejo, Segura, Fernandez, & Ruiz-Cortés, 2014).

In our road traffic scenario, after asking some experts and potential clients of our system, we have decided to use these NFPs:

1. Category, refers to an entry in some ontology or taxonomy of services.
2. Geographical Radius, which helps to check whether or not the geographical radius given by the client is similar to the offered one.
3. Quality of Service, which consists in checking if the quality in the repository if the quality is the same as that requested by the client.
4. Service's Name, syntactic comparison using the name value.
5. Provider's Name, which allows the client to specify the desired WS organization provider. It requires a syntactic comparison too.

As cited by Cardoso and Sheth (2003), Cardoso, Sheth, Miller, Arnold, and Kochut (2004) the management of QoS metrics directly impacts the success of organizations participating in e-commerce. The QoS metrics is presented in many current research (Rabea & Fraihat, 2012). In our system, QoS is defined in the ontology as a set of values from Excellent to Poor. To compute the quality of service, the system takes into account the following factors: Cost (is it free?), Frequency Update, Offered by Public Administrations (if the web service is provided by them, they are considered better), Usability, Reliability and Integrity and finally Security aspects.

**Algorithm 4.** Measures NFP

```
MeasureNFP(couples)
{
  TotalNF = 0
  for (REQ,ADV) in couples[geoRadius]
    if exact(REQ,ADV) then TotalNF +=5
  for (REQ,ADV) in couples[Quality]
    if exact(REQ,ADV) then TotalNF +=5
  for (REQ,ADV) in couples[ServiceName]
    if REQ == ADV then TotalNF +=3
  for (REQ,ADV) in couples[ProvidersName]
    if REQ == ADV then TotalNF +=2
  return TotalNF
}
```

As the number of selected parameters increase, the run-time of the algorithm also increases. Our hypothesis is that using a pre-process step to filter the collection of WS to be candidates to match the user requirements will be a good solution to improve the run-time. Despite García et al. (2012) we only filter by category, a mandatory NFP in our WS profiles which helps not to decrease the precision.

The category, service profile attribute, allows to specify the service category within the UNSPSC[4] classification system or other

[4] <http://www.unspsc.org>

taxonomies. Road traffic information is generally available via road operators' web sites, for this scenario a standard taxonomy has been established by ISO TC 204 (ISODIS14813-1.2, 2004) but as UNPSC is too broad. The use of service categories may have little meaning if the categories allowed in the ontology are too wide because the number of candidates won't decrease significantly with the pre-process step. The use of less precise ontologies will be needed, although service descriptions should be more exact and the matchmaking process should be more complex. For that purpose, we defined the categorization ontology (Fig. 2) to be used as the taxonomy to classify our WS. This ontology has been defined taking into account the ISO TC204 as basis.

Algorithm 4 presents the similarity measure for the NFPs Geographical Radius, Quality of Service, Service's Name, and Provider's Name. For NFPs, the value of the similarity degree depends on whether it has a semantic (exact match) or syntactic (string match) type. Service's Name and Provider's name do not allow semantic comparisons and the match is only syntactic. Both matchings have the same weight. The algorithm counts the number of matches, and accumulates the fails.

### 4.3. General issues about the algorithm proposal

The Hybrid Matching algorithm (Algorithm 1) allows us to obtain an ordered list of the best WS of our advertisement for the desired or requested WS profile. In our process each WS profile will be represented in our algorithm as a collection of concepts grouped by Inputs, Outputs and NFPs (category, geographical radius, quality of service, service's name, provider's name). ListCouples, is a multidimensional array, where each dimension has a list of pairs organized by type of parameters: Inputs, Outputs and NFPs. A pair is a couple with two concepts of the profiles representing this dimension, one from the request and another one from the advertisement. This structure facilitates the process to obtain a detached similarity degree for each type of parameter within the combiner algorithm (Algorithm 7).

The steps of the proposed Hybrid Matchmaking Algorithm are:

1. Filter. The pre-process step is to filter among all advertised services, those which belong to the same category as requested by the client (Algorithm 5). For this, we use the Road Traffic Services Categorization Ontology (Fig. 2). The filter algorithm is a pre-step process to obtain a subset of OWL-S profiles stored on the knowledge-base similar to the user request (Candidates). This process has been created to decrease the algorithm runtime.

**Algorithm 5.** Filter Algorithm

```
FilterByCategory (REQ,Advertisements)
{
 Candidates = Null
 forall ADV in Advertisements
  if (REQ[Category]= ADV[Category]) then
  Candidate.append(ADV)
 return Candidates
}
```

2. Combiner. This simple phase consists in establishing the different possible combinations between the request given by the client and the advertisement published by the providers, that due to the previous phase belong to the same service category. For each WS from the provider, the result of this step will be a multidimensional array of pair of concepts. Each pair list contains all the combinations between the different parameters from user requirements and the provider service, related with a type of parameter. Algorithm 6 shows in detail this combination process.

```
Select distinct URIPS, URIRS
From
    {URIPS} <owl:subClassOf> {Father},
    {URIRS} <owl:subClassOf> { Father },
    {URIPS} <rdfs:subClassOf> {Rest},
    {URIRS} <rdfs:subClassOf> {Rest2},
    {Rest} <rdf:type> {<owl:Restriction>},
    {Rest2} <rdf:type> {<owl:Restriction>},
    {Rest} <owl:onProperty> {Prop1},
    {Rest2} <owl:onProperty> {Prop1},
    {Rest} Z {A},{Rest2} Z {B}
Where URIPS = <URI of Provider's Concept>
     and URIRS = < URI of Client's Concept >
     and A = B and Prop1=Prop2
```

**Fig. 4.** Sibling SeRQL Query (CsiblingP).

**Algorithm 6.** Combiner Algorithm

```
Combiner ALGORTIHM(REQ,ADV)
{
  PairList = Null
  PairList[Inputs]=combine(REQ[inputs],
  ADV[inputs])
  PairList[Outputs]=combine(REQ[inputs],
  ADV[inputs])
  PairList[geoRadius]=combine(REQ[geoRadius],
  ADV[geoRadius])
  PairList[Quality]=combine(REQ[Quality],
  ADV[[Quality])
  PairList[ServiceName]=combine
  (REQ[ServiceName],ADV[ServiceName])
  PairList[ProviderName]=combine
  (REQ[ProviderName],ADV[ProviderName])
  return PairList
}
```

3. Similarity measure algorithm. This step will obtain a detached degree/weight for inputs, outputs and NFPs and failures (Algorithm 7). Each type of parameter due to its nature need to use different measures, the similarity measures for each type of parameters are explained in Sections 4.1 and 4.2. The algorithm detached all the measures accumulated for inputs, outputs, NFPs and failures. Algorithm 7 analyzes each weight using the order relevance of the parameters.

**Algorithm 7.** Matching Weight

```
MatchingWeight(Couples)
{
  Weight = Null
  (Weight[Outputs], Weight[Fails]) =
  measureFP(Couples[Outputs])
  If Weight[Outputs]>0
  {
      (Weight[Inputs], Fails) =
  measureFP(Couples[Inputs])
      Weight[Fails]+=Fails
      Weight[NFP]= measureNFP(Couples)
  }
  return(Weight)
}
```

4. Sort algorithm. Taking into account the measures calculated previously detached by Outputs, Inputs, NFP and Fails, the next step is to sort the matching services, so the service that heads the list will be one that is considered the best. Algorithm 7 will create an ordered list using the insert in an ordered list algorithm. The order is obtained with the New Sort Rule function (Algorithm 9). This function will return true if the first WS ADV1 is a better candidate than the second ADV2. The proposed New Sort Rule gives more importance to the weight of the outputs as in sortRule function in Paolucci's Algorithm, because the most important is that the one customer gets. What customer wants, is defined by the output parameters. It is considered that inputs are the least important parameters to find the SW, so the inputs parameters are the latest to be compared. As the required service needs to have some common output to the user requirements notice that if there is no similarity between the outputs (weight==0), the procedure will not continue evaluating the others measures. This solution will allow to improve the run-time and save on computation of unnecessary similarity measures.

**Algorithm 8.** Sort Algorithm

```
Sort (ADV1, ADVList)
{ List = Null
 for i in range(len(ADVList)):
  { if bestWS(ADV1,ADVList[i])
  List = ADVList[:i] + ADV1 + ADVList[i:]
  return List
  }
 List = ADVList + ADV1
 Return List
}
```

**Algorithm 9.** NewSortRule

```
BestWS(ADV1,ADV2)
{
if ADV1.Outputs.Weight > ADV2.Outputs.Weight
  return true
else if ADV1.Outputs.weight < ADV2.Outputs.weight
  return false
else if ADV1.SumNF.weight > ADV2.SumNF.weight
  return true
else if ADV1.Inputs.weight < ADV2.Inputs.weight
  return false
else if ADV1.Fails < ADV2.Fails
  return true
else return false
}
```

## 5. Results and discussion

In this section we are going to evaluate the efficiency of our matchmaking system and it will be compared with Paolucci's Algorithm. A set of different web services profiles from the provider repository has been selected to test our tool (Table 1).

### 5.1. Experimental scenario

To test our assumptions we have defined two queries representing the user's requirements, and in Tables 2 and 3 are represented the query profiles and the scores of the services that best meet our user requirements. Using the Road Traffic Ontology (Fig. 5), we can find the semantic relations between the parameters of these services.

Query 1. The first user query is represented by the 'ClientForecast' profile. Looking at Table 2, there is an exact match between all the parameters of the candidates WS profiles, except for the outputs (parameter 'Forecast'). The concept 'forecast' is a direct subclass of the concept 'Type', while the output 'MeteorologicalForecast' is a direct subclass of 'Forecast' (Fig. 5).

Query 2. In this second query (Table 3), represented by 'MeteorologicalForecast' profile, the output selected is 'MeteorologicalForecast', a direct subclass of 'Forecast'. There is a unique service which has a different NFP ('Geographical Radius'). 'ForecastCat' profile represents a service similar to 'ForeCast', but restricted to the Catalonian region.

**Table 1**
WS Repository.

| Profile name | Inputs | Outputs/GR/QoS |
|---|---|---|
| ForecastCat | Time.owl#Date | Outputs:<br>Events.owl#Forecast<br>Geographical radius:<br>Geography.owl#Catalonian<br>Quality of service:<br>Concepts.owl#Good |
| TypeProfile | Time.owl#Date | Outputs:<br>Events.owl#Type<br>Geographical radius:<br>Geography.owl#Spain<br>Quality of service:<br>Concepts.owl#Good |
| Forecast | Time.owl#Date | Outputs:<br>Events.owl#Forecast<br>Geographical radius:<br>Geography.owl#Spain<br>Quality of service:<br>Concepts.owl#Good |
| Prognosis | Time.owl#Date | Outputs:<br>Events.owl#Prognosis<br>Geographical radius:<br>Geography.owl#Spain<br>Quality of service:<br>Concepts.owl#Good |
| Meteorological forecast | Time.owl#Date | Outputs:<br>Events.owl#meteorologicalforecast<br>Geographical radius:<br>Geography.owl#Spain<br>Quality of service:<br>Concepts.owl#Good |
| GrandChild forecast | Time.owl#Date | Outputs:<br>Events.owl#GrandChildForecast<br>Geographical radius:<br>Geography.owl#Spain<br>Quality of service:<br>Concepts.owl#Good |
| GreatGrandChild forecast | Time.owl#Date | Outputs:<br>Events.owl#GreatGrandChildForecast<br>Geographical radius:<br>Geography.owl#Spain<br>Quality of service:<br>Concepts.owl#Good |
| AccidentDate | Time.owl#Date | Outputs:<br>Events.owl#Accident<br>Geographical radius:<br>Geography.owl#Spain<br>Quality of service:<br>Concepts.owl#Good |

**Table 2**
ClientForeCast query.

| Request | Available services | Score (O/I/NF) | |
|---|---|---|---|
| ClientForecast | | Proposal | Paolucci |
| Inputs: | TypeProfile | 5/6/8 | 3/3 |
| Time.owl#Date | Forecast | 6/6/8 | 3/3 |
| Outputs: | MeteorologicalForecast | 4/6/8 | 2/3 |
| Events.owl#Forecast | GreatGrandChildForecast | 0/6/8 | 2/3 |
| Geographical radius | GrandChildForecast | 2/6/8 | 2/3 |
| Geography.owl#Spain | Prognosis | 1/6/8 | 0/3 |
| Quality of service: Concepts.owl#Good | AccidentDate | 0/6/8 | 0/3 |

**Table 3**
Client MeterologicalForeCast Query.

| Request | Available services | Score (O/I/NF) | |
|---|---|---|---|
| MeteorologicalForecast | | Proposal | Paolucci |
| Inputs: | Forecast | 5/6/8 | 3/3 |
| Time.owl#Date | ForecastCat | 5/6/5 | 3/3 |
| Outputs: | GrandChildForecast | 4/6/8 | 2/3 |
| Events.owl# MeteorologicalForecast | GreatGrandChildForecast | 2/6/8 | 2/3 |
| Geographical radius | TypeProfile | 3/6/8 | 2/3 |
| Geography.owl#Spain | Prognosis | 1/6/8 | 0/3 |
| Quality of service: Concepts.owl#Good | AccidentDate | 0/6/8 | 0/3 |

### 5.2. Results analysis for adopted similarity measures

Here we are going to evaluate how our measures improve the results of our algorithm comparing it with Paolucci's algorithm, and analyzing the scores of our test case.

1. **Exact and direct subclass.** It should be noted that in Paolucci's Algorithm the same weight has been assigned to these measures. Due to this, on Table 2 the results using Paolucci's Algorithm of the first two services have the same weight (3/3), 'exact match' concept relation for the output parameters. Using concept taxonomy in Fig. 5, the request concept 'MeteorologicalForecast' is a direct subclass of 'Forecast'. So Forecast is more relevant for the user request, as it is pointed out in our algorithm results scores. We can conclude that our proposal, thanks to the distinction between exact and direct subclass, resulted in a more adequate service to the requirements.
2. **Fraternal relationship (sibling).** The last two profiles resulting of our algorithm: Prognosis and AccidentDate profiles (Table 2), obtain in Paolucci's Algorithm the same score (0/3) because they do not recognize the fraternal degree. Observe in the taxonomic tree (Fig. 5), that the output parameter 'Forecast' and 'Prognosis' are sibling concepts, while there is not a close relationship between 'Forecast' and 'Accident'. So if there are no other WS with higher weight than 'Prognosis', our algorithm at least presents a possible matching service, that is better than nothing. Our algorithm scores 'Prognosis' (1/6/8) and for 'AccidentDate' (0/6/8); This case is focused on evaluating the results taking into account that the most appropriate service is one that has parameters, whose concepts are on the same level in the tree and have a fraternal relationship with the concepts related to the parameters of the provider's services.
3. **PsubsumeC/CsubclassP.** We request for 'MetheorologicalForecast' in Table 3. Paolucci's Algorithm, assigns the same relevance to 'TypeProfile'(2/3) than to 'GrandChildForecast'(2/3). Note in the taxonomic tree (Fig. 5), that 'GrandChildForecast' is a direct subclass of 'MeteorologicalForecast', while it subsumes 'Type' with a distance of two, so our algorithm scores 'GrandChildForecast' as (4/6/8) and 'TypeProfile' as (3/6/8). In our algorithm, the provider's advertisement that has more priority is one whose concepts are direct subclass, rather than the more general provider's advertisements (grandfather, great-grandfather, and so on), since they are far away within the hierarchy.
4. **PsubsumeC with maximun distance 2.** In our algorithm there is an exclusion of concepts in distances bigger than 2, while Paolucci's Algorithm gives the same similarity value to concepts descending or ascending in the hierarchy, because for them, the distance on the ontology taxonomy is not an important factor. In this scenario, we found out that returning some results without imposing a maximum limit to the tree distance, can result in concepts (profiles) that hardly have characteristics in common with the concept required. In Table 2, we can see how in Paolucci's Algorithm, two concepts with different distances from the concept 'ForeCast' obtain the same result ('GrandChildForecast' and 'GreatGrandChildForecast'). However, in our algorithm 'GrandChildForecast' is better and 'GreatGrandChildForecast' is not considered because its distance is too big.
5. **Non-Functional Parameters.**In the second user query, the client determines a NFP as a Geographical Radius requirement. The user's request for a Meteorological Forecast information service on the Spanish region. First two profiles on Table 3: 'Forecast' and 'ForecastCat' are characterized by the output parameter 'Forecast' and only differs on geographical radius parameter. In our repository (Table 1) 'Forecast' is a service whose provider, the DGT (Spanish traffic administration) has specified Spain in the Geographical Radius, and on the other hand, the SCT (Catalonian traffic administration) has specified for the 'ForecastCat' the Catalonian region. Paolucci's Algorithm does not distinguish between them so our suggested algorithm prefers the 'Forecast' (5/6/8), vs. 'ForecastCat' (5/6/5) as our end users.

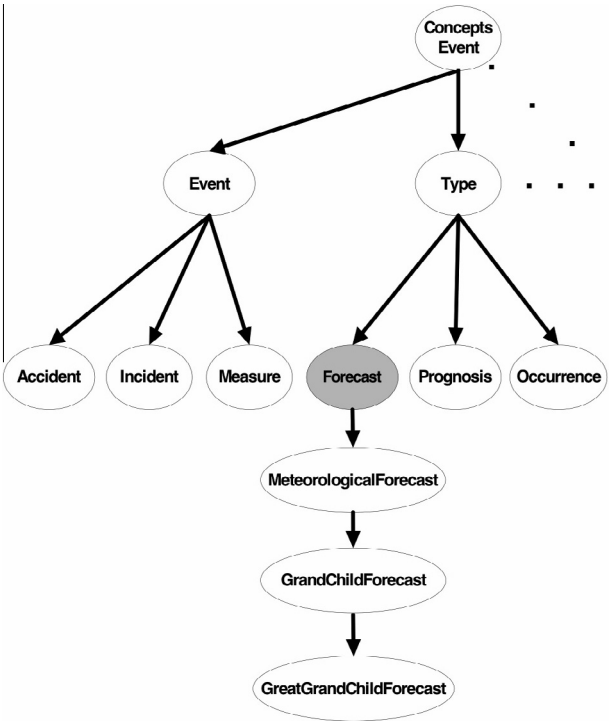
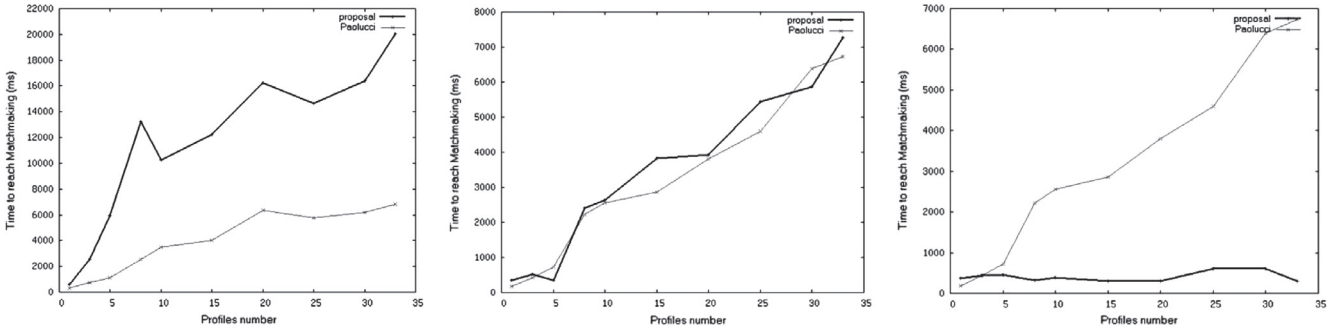



**Fig. 5.** Part of road traffic ontology with test concepts.




(a) Same category for all. (b) Request and 20% advs same category. (c) Only one advertisement by category.

**Fig. 6.** Run-time/number of services.

## 6. Results analysis for run-time

To demonstrate the importance of using filters before the matching process (Filter step in Section 4.3) in order to the run-time, it was compared the run-time with Paolucci's Algorithm, attending to a non-functional criteria: 'belong to the same category'. The test to evaluate the run-time has been developed varying the number of service profiles (1, 3, 5, 8, 10, 15, 20, 25, 30 and 33) and 3 cases are presented:

**Case 1** All advertisements available belong to the same category. This is the worst case for the proposed algorithm. This is due to the fact that all advertisements inserted belong to the same service category, including the service requested by the client. In this case the service category filter does not reject any candidate, so the match is made with the same number of profiles in both algorithms. Moreover, our algorithm cost is greater, due to the number of comparisons. Thus, Paolucci's Algorithm results are reached faster than our algorithm (Fig. 6a).

**Case 2** Only 20% of the available advertisements belong to the same category. In this situation, which is more realistic, not all the profiles belong to the same category as the profile requested by the client (only 20%). In this case, results between the two algorithms are closer (Fig. 6b).

**Case 3** Only one service belongs to the same category as that required by the client. This is the best case for our proposal, because our algorithm only has to match one service. However, Paolucci's Algorithm has to search all the services available (Fig. 6c).

## 7. Conclusions

Using our repository we have run different experiments to assign a degree of similarity to each parameter to obtain the WS profile that best fits user requirements. During the testing process we defined several queries to test the functionality of our system and to compare it with Paolucci's Algorithm. Afterward, we included a result analysis section to present the improvements obtained due to the use of these measures and also to the use of NFPs, some of which had not been considered in previous investigations. After defining our research problem, we found two important limitations: the lack of existing WS profiles annotated semantically, and the lack of ontologies for this scenario. So we were forced to implement a couple of ontologies: Road Traffic Ontology for specifying the road information concepts and a Road Traffic Services Categorization Ontology. The latter allows us to annotate the WSs semantically using OntoService Samper et al. (2008).

The problems of greedy algorithms for WS semantical matching is a recent research area, as shown in the proposals in the background section. In a discovery WS system, it is not enough to return the WSs that correspond to the inputs and outputs requested. Most likely the client requires some non-functional constraints, i.e. only wants information from a specific city. We have noticed that the use of non-functional parameters can improve the precision of the algorithm, and the use of several semantic matching degrees for the functional parameters increase the recall. Each parameter or measure that we introduce increases the algorithm cost.

The major contributions of this research are outlined as follows. First, the use of the 'sibling concept relation measures', which had not been studied previously, improves the recall. Second, use of the concept relation measure PsubsumesC, that would be equivalent to the Plug-in defined in Paolucci's Algorithm yet with a little difference because we applied maximum depth (3 levels) to the search. This restriction will not decrease the recall, but does improve the run-time. Third, the use of a limited number of non-functional parameters relevant to our scenario. For example the use of geographic radius is very specific to our context and its weight is the same as the more generally relevant and used QoS parameter. Finally, but not less important, the use of a pre-process step, which decreases the run-time due to reduced number of WSs candidates, filtering by service category.

Furthermore we can state that the algorithm suggested has a higher cost, because it has more matches to make, due to the inclusion of new measures and the use of more parameters. However, thanks to the preprocessing filter step, the run-time is not overly inflated except in extreme cases where there is a high number of providers' advertisements that belong to the same service category as that of the client's request. In realistic situations, where roughly 20% of the advertisement belongs to the same service category, run-time is similar to Paolucci's Algorithm, and is even better in situations where the number of advertisements belonging to the request category is small. Therefore, we conclude that precision in searches has risen without harming the running-time in normal situations.

The cloud computing paradigm is emerging and shows similar problems to these in WS. There is limited published research in this area. Rezaei, Chiew, Lee, and Shams Aliee (2014) propose a framework to allow a semantic interoperability for software as a service system. We are now applying our results on the cloud computing paradigm (Cortázar, Samper-Zapater, & Sánchez, 2012), and in our first research Samper-Zapater et al. (2013), we have defined the ontology to facilitate a semantic identification, discovery and access to the services in the cloud.

Although the domain chosen for testing was road traffic, it could be applied to other domains by adapting the ontology and

probably modifying the selection of NFPs. It is accepted that the semantic WS promote their interfaces (inputs/outputs) using the same ontology on our repository. This restriction is due to the lack of mature solutions for mapping ontologies, specially in this context. Nowadays, LinkedData[5] is helping on the creation of new domain ontologies and on the definition of semantic relations between concepts from different ontologies. As future work to be undertaken, it is important to highlight the improvements of the algorithm using different ontologies, and relating the concepts, taking advantage of information from LinkedData (Heath & Bizer, 2011).

## References


Al Rabea, A. I., & Al Fraihat, M. M. (2012). A new matchmaking algorithm based on multi-level matching mechanism combined with fuzzy set.

Cardoso, J., & Sheth, A. (2003). Semantic e-workflow composition. *Journal of Intelligent Information Systems, 21*(3), 191–225.

Cardoso, J., Sheth, A., Miller, J., Arnold, J., & Kochut, K. (2004). Quality of service for workflows and web service processes. *Web Semantics: Science, Services and Agents on the World Wide Web, 1*(3), 281–308.

Cortázar, G. O., Samper-Zapater, J. J., & Sánchez, F. G. (2012). Adding semantics to cloud computing to enhance service discovery and access. In *6th Euro American Conference on Telematics and Information Systems*, pp. 1–6.

García, J. M., Ruiz, D., & Ruiz-Cortés, A. (2012). Improving semantic web services discovery using sparql-based repository filtering. *Web Semantics: Science, Services and Agents on the World Wide Web, 17*, 12–24.

Heath, T., & Bizer, C. (2011). Linked data: Evolving the web into a global data space. *Synthesis Lectures on the Semantic Web: Theory and Technology, 1*(1), 1–136.

ISODIS14813-1.2, 2004. Iso tc 204/sc, iso/wd 14813-1 intelligent transport systems - reference model architecture(s) for the its sector - part 1.

Jaeger M. C., Rojec-Goldmann G., M. G. L. C., & K., G., 2005. Ranked matching for service descriptions using owl-s. Kommunikation in verteilten Systemen 41(13), 91–102.

Kamaruddin, L. A., Shen, J., & Beydoun, G. (2012). Evaluating usage of wsmo and owl-s in semantic web services. In *Proceedings of the Eighth Asia-Pacific Conference on Conceptual Modelling*, vol. 130. Australian Computer Society, Inc., pp. 53–58.

Khater, M., & Malki, M. (2014). Improving the performance of semantic web services discovery: Shortest path based approach.

Li, L., & Horrocks, I. (2004). A software framework for matchmaking based on semantic web technology. *International Journal of Electronic Commerce, 8*(4), 39–60.

Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., et al. (2005). Bringing semantics to web services: The owl-s approach. In *Semantic Web Services and Web Process Composition* (pp. 26–42). Springer.

Paolucci, M., Kawamura, T., Payne, T. R., & Sycara, K. (2002). Semantic matching of web services capabilities. In *The Semantic Web–ISWC 2002* (pp. 333–347). Springer.

Parejo, J. A., Segura, S., Fernandez, P., & Ruiz-Cortés, A. (2014). Qos-aware web services composition using grasp with path relinking. *Expert Systems with Applications, 41*(9), 4211–4223.

Payne, T. R., cci, M., & Sycara, K. (2001). Advertising and matching daml-s service descriptions. *Position Papers for SWWS, 1*, 76–78.

Pedrinaci, C., Maleshkova, M., Zaremba, M., & Panahiazar, M. (2012). Semantic web services approaches. In *Handbook of Service Description* (pp. 159–183). Springer.

Rezaei, R., Chiew, T. K., Lee, S. P., & Shams Aliee, Z. (2014). A semantic interoperability framework for software as a service systems in cloud computing environments. *Expert Systems with Applications, 41*(13), 5751–5770.

Samper, J. J., Tomás, V. R., Carrillo, E., & do PC Nascimento, R. (2008). Visualization of ontologies to specify semantic descriptions of services. *IEEE Transactions on Knowledge and Data Engineering, 20*(1), 130–134.

Samper-Zapater, J. J., Llidó, D. M., Durá, J. J. M., & Cirilo, R. V. (2013). Semantic multi-agent architecture to road traffic information retrieval on the web of data. In *Distributed Computing and Artificial Intelligence* (pp. 465–472). Springer.

Samper-Zapater, J. J., Zambrano, E. C., & García, A. C. (2006). Semantic modelling of road traffic information elements. *Revista Colombiana de Computación, 7*(1).

Sangers, J., Frasincar, F., Hogenboom, F., & Chepegin, V. (2013). Semantic web service discovery using natural language processing techniques. *Expert Systems with Applications, 40*(11), 4660–4671.

Sycara, K., Klusch, M., Widoff, S., & Lu, J. (1999). Dynamic service matchmaking among agents in open information environments.

Tomaz, R.F., Labidi, S., & Wanghon, B. (2003). A semantic matching method for clustering traders in b2b systems. In Web Congress, 2003. Proceedings. First Latin American. IEEE (pp. 144–153).

Trastour, D., Bartolini, C., & Gonzalez-Castillo, J. (2001). A semantic web approach to service description for matchmaking of services. In *SWWS*. pp. 447–461.

Tsai, Y.-H., Hwang, S.-Y., & Tang, Y. (2011). A hybrid approach to automatic web services discovery. In *International Joint Conference on Service Sciences (IJCSS)*, 2011. IEEE, pp. 277–281.

Vu L-H., H.M., K., A., 2005. Owards p2p-based semantic web service discovery with qos support. In Business Process Management Workshops (pp. 18–31).

Yu, W. H. (2012). Application of java serialization into jade agent communication. *Advanced Materials Research, 433*, 7357–7361.


[5] <http://www.linkedata.org>