

Improving semantic web service discovery

J. Javier Samper, F. Javier Adell, Leo van den Berg and J. José Martínez
 Robotics Institute, University of Valencia, Valencia, Spain
 Email: { jsamper, jadell, l.vdberg, juanjo }@robotica.uv.es

Abstract— Semantic Web Services (SWS) propose to extend the traditional Web Services technologies on the way to consolidate ontologies and semantics. This makes it possible to select, integrate and invoke services dynamically. In this way, services are able to dynamically adapt themselves to changes without human intervention.

The main purpose of this paper is to present an algorithm for matching SWS. The algorithm uses the description of the service capabilities to match the semantic values. The traditional matching has been improved using ontologies which constitute a step further in the matching algorithms. To implement the algorithm, an agent FIPA compliant architecture has been designed and developed.

The results obtained are positive. The semantic web services framework developed in combination with the use of the matchmaking algorithm, which allows finding services based on their similarities.

Index Terms—Web services, Semantic web services, Ontologies, Multiagent systems, Traffic information, Semantic descriptions

I. INTRODUCTION

As it is reflected in [1], the Semantic Web (SW) is a medium where the data can be shared and processed both by automated tools and humans. The key point is in the automation and integration of the processes through machine readable languages. In order to use and connect all the information available on the Web, the software agents should be able to understand the information, that is to say, the data should be written with semantics. Therefore, in XML or HTML documents, additional semantics or metadata should be added so that the software programs can fix the meaning of the documents labels.

In order that computers are much more useful, the SW expands the present Web with formalized knowledge and the data are processed by computers. To be able to look for and to process information relative to some matter of interest, the programs need information that must be modelled in a coherent way. An ontology models all the entities and relations inside a domain. The ontology is necessary for knowledge representation. The key of ontologies is that they can be shared and, therefore, they increase in efficiency and interoperability.

The SWS are a new computation paradigm, defined generally as the development of Web Services (WS) descriptions with Semantic Web annotations to facilitate the discovery, composition, invocation and monitoring of

the services in a chaotic and non-regulated atmosphere as the Web is [2].

The objective of this paper is to show how some improvements in the matching algorithms process can allow obtaining better results. On the other hand, a second objective is showing how they can be applied to specific domains. Section 2 explains a brief overview of web services and describes the definition of a web service using an ontological infrastructure. In section 3, previous works about semantic matching of services are presented. Next, in section 4, the proposed matching algorithm is described. In section 5, the new multiagent architecture to publish and discover information services is described. In the last section, the conclusions and the future work is presented.

II. WEB SERVICES: OVERVIEW AND EXAMPLES

A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically Web Services Description Language, WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

Web Services are based on the interchange between applications through the Web and have been conceived for the machine-machine interaction rather than for machine-man interaction, which is characteristic of web sites. It provides a standard means of interoperating between different software applications, running on a variety of platforms and/or frameworks.

The use of the services permits access to the Web with a multitude of devices, the only restriction is that clients need to communicate with the service through a set of XML-based messages.

A. Ontological Development

The use of an ontology as a knowledge representation scheme has a number of important advantages: The acknowledged theoretical fundament (the first order logic contained by ontologies are studied over 100 years), robustness and code separation makes it possible to represent specific expert knowledge which can be interpreted by machines. For this reason, ontologies are already one of the most usable forms of knowledge

representation on the Web thanks to its inclusion of semantic description of resources.

An example developed is the formal ontology definition of the hazardous goods transport domain taking in account the involved laws. To formalize the concepts and the existing relations in the different norms related to goods transport over land a standard language was used to define ontologies: OWL-DL [3]. The consistency check of these ontologies evaluation are realized using the description language reasoner Pellet [4], because of its easy integration with Web services using the administration API of Jena [5]. With all the Web infrastructure and knowing that the used ontologies are consistent (there are no conflicts in the defined concepts), new information of specific transport can be added and its correctness can be tested using the reasoner.

B. Development and test of the web services

The last step taken was the integration of the infrastructure using web services. In this way it was created access from whatever client system to this knowledge base and particularly from road-side devices.

The use of the administration API of Jena facilitates the agile development of web services for the Semantic web, particularly description logic ontologies and reasoner. For the development it was used the Eclipse Web Tools Platform (WTP) [6] together with Apache's Axis [7], a Java platform for creating and deploying web services applications, and the servlet container apache Tomcat [8].

The phases of the proposed control process are:

1. In the first place from the web service, which access the reasoner, load the Web ontologies, which contain the generic characteristics of the goods and the vehicles used for road transport.
2. The control process starts when a specific vehicle with an OBU transmitter ("On Board Unit") providing its transport information, transmits its information to a motorway check point, which sends it to the web service.
3. When the service receives the information of the transport, it loads the specific restrictions in memory of the specific transported good. At that moment the web service contains what is permitted, based on the legal restrictions and on the data which describes the transport obtained from the OBU.
4. The inference system is able to interpret automatically all that information in such a way that whenever an inconsistency is found, a check point is contacted to inform about the possible infraction.

III. RELATED WORKS AND RESEARCH

The main foundation of the investigation related to semantic matching of services is in the studies of Software Engineering that Zaremski and Wing carried out in [16] and [17]. In [18], Sycara et al. presented LARKS, in which the services are seen as frames, and the input, output, inConstraints and outConstraints can be used to describe the essential attributes of a service. The matching process in LARKS makes both syntactic and semantic matching and is formed of a group of filters

(which are independent from each other) that restrict progressively the number of candidate advertisements to be matched.

The approximation used by [19] for service matching is an algorithm based on the subsumption tree given by a description logic (DL) reasoner. The service descriptions have been specified in DAM+OIL and, since DAM+OIL is based on DL, the DL reasoner is the heart of the matching algorithm. The different matching types for a service *S* have been defined as: equivalent concepts to *S*, subconcepts of *S*, superconcepts of *S* that are included by the concept in the service description ontology, and the subconcepts of any direct superconcept of *S* whose intersection with *S* is satisfactory.

The Intelligent Software Agents Group team at the Carnegie Mellon University (CMU) designed an architecture based on ATLAS (Agent Transaction Language for Advertising Services), a language based on DAML, for matching of services [20]. It uses two filters: matching of functional attributes of services for determining the advertisement applicability and matching of service functionalities, in order to determine if the advertised service matches the required service.

In [9] a matching system was used by Paolucci et al. that is based on the semantics described in the profile service and on the use of UDDI registers for maintaining the descriptions of such services. A matching between an advertisement and a service request consists in pairing all the request outputs with those of the advertisements; and all the advertisements inputs with those of the request. The reason why the priority of the output parameters is higher is because for the process matching the most important aspect is what the client hopes to obtain as a result, that is, the service output. One of the most significant ideas that Paolucci et al. provide in their algorithm for a better behaviour is that both client and providers use the same ontology so as to give semantic value to each one of the profiles of the services. In this way, the task is facilitated to the inference engine or reasoner since it only needs that the concept ontologies that they used for building profiles are loaded so as to measure the similitude degree that the defined inputs or outputs parameters had.

Different researchers have based their matching system on such algorithm, although they added some functionalities: Charlie Abela in [21] developed an algorithm that uses two filter operations to carry out the matching. The first filter limits the search space to the restrictions that the user defined in the request (search by ServiceProvider, ServiceCategory or ServiceName). The second filter is an output matching that the user defines in the request, which allows using only one filter or both. In one matching result there can be different services and, thus, a range needs to be created so as to specify which one fits the client's needs. Sometimes, the client's intervention will be needed so as to choose the one that he considers as most appropriate.

The Information Management Group at the University of Manchester [15] developed another approach. The prototype description of the matching algorithm is also

based on the descriptions of DAML-S. It uses the reasoner Racer Descriptions Logics for making semantic matching between services and uses JADE as the agent platform. As in the DAML-S Matchmaker design the service requests in DAML-S are matched with those of the service advertisement. In such approach, the profiles are treated as complete entities and their components are not treated separately. Once again, to the matching result a certain degree of similitude is associated.

IV. MATCHMAKING ALGORITHM PROPOSAL

The algorithm that we have developed consists also of an extension of the algorithm of Paolucci et al. for matching of semantic values that were used to describe each one of the service capabilities, as much by the provider as by the client in his request. For this reason, the client requests will be profiles such as those the providers use to advertise.

The phases of the proposed algorithm are:

- Phase 1: Filter from all the advertised services those that belong to the same category as those requested by the client. The fact that new filters are used allows us to take advantage of other characteristics that the DAML-S Profile has, that is, to advertise Web services and that were not exploited by previous algorithms.
- Phase 2: From the resulting list, it combines all the possible advertisements between the client’s request and each one of the provider advertisements. 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.
- Phase 3: Each time a couple is obtained in the previous phase, different degrees of similarity for functional parameters must be applied to them, and the relative weights must be calculated for functional parameters (outputs, inputs,...) and non-functional parameters (quality of service, provider name,...).
- Phase 4: Taking into account the weights calculated (weight output, weight input, weight of non-functional parameters) the next step is executing the ordering algorithm to then insert it in the correct place of the service list that resulted from the query. The service that heads the list will be one that is considered optimal. The rest of the list will be maintained for fault tolerance.

A. Obtaining Matching Degree for Functional Parameters

The degree of similarity depends on the relation between concepts (taken from the ontologies) that are being compared, and generally is reduced to the minimum distance between them in the taxonomic tree. The denomination of the degrees varies according to the literature.

In our algorithm, modifications are proposed in the degrees of matching since in these projects they were very general; therefore, a composed version of seven degrees of matching is proposed:

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

- CsubclassP: Within the taxonomic concept tree the distance between the demanded concept and that offered by the provider is equal to 1, this way being the one described by the client, direct subclass of the 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 that 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. It would be equivalent to the Plug-in defined in [9] although it is differenced by not allowing the client to specify the level of maximum depth of the search. It has been chosen not to allow the client to specify this because we consider that concepts at distances greater or equal to 3 levels do not have hardly any semantic relation, due to the way in which the 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, in other words, in the case opposite to the previous one, and is equivalent to the Subsume degree defined in [9].

- CsiblingP: The concept provided by the client coincides in some restriction defined for its properties in common with those of the provider’s concept, 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.

The pseudocode of the method for the assignment of matching degrees appears in Fig. 1.

1. Queries for Degrees of Similitude in Functional Parameters

The queries were carried out by means of SeRQL (Sesame RDF Query Language) [10]. In Fig. 2 one of them is summarized. This query is based on taking the restrictions of the client concept, to then browse through the restrictions of the siblings of this class, to find

```

DegreeOfMatch(outR, outA)
{
    if Exact(outA, outR)
        return EXACT
    else if SubClassOf(outR, outA)
        return CSUBCLASEP
    else if SubClassOf(outA, outR)
        return PSUBCLASEC
    else if Subsumes(outA, outR)
        return PSUBSUMEC
    else if Subsumes(outR, outA)
        return CSUBSUMEP
    else if Siblings(outR, outA)
        return CSIBLINGP
    else
        return FAIL
}
    
```

Figure 1. Matching Pseudocode

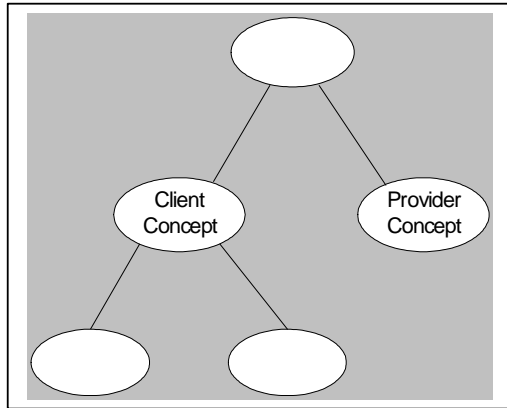


Figure 2. Sibling Concepts Query

restrictions that match. The corresponding SeRQL query implemented is showed in Fig. 3. In case it is obtained as result the given URIs, it can be concluded that the classes are siblings and they coincide in some of the restrictions.

B. Exact Coincidence in Non-functional Parameters.

The Paolucci et al. algorithm does not use other Profile properties, such as the non-functional parameters, which also contribute semantic information from the Web service, one of whose objectives would be to cover this area. The matching algorithm exploits the Profile possibilities to the maximum, with the notion of making use of all the service capabilities, described semantically.

- Region filter: With this, the geographical radio given by the client will be checked, whether it is the same given by the provider.
- Quality service filter: It consists in checking in the repository whether the quality provided by Web services is the same as that requested by the client. A ranking for quality of traffic services was made.
- Service name filter: In this type of query, it is checked whether the client finds any service in particular with the name provided. This matching is syntactic, since the way in which the service-name profile is defined in the Profile subontology has as a value range, the String datatype from XML Schema, with which only syntactic comparisons can be made.
- Provider name filter: It allows checking whether the Web service is provided by the company in which the client is interested. As the previous filter only a syntactic match is made since this Profile property is defined as an

```

Select distinct X, Y
From {X} <daml:subClassOf> {Father},
{Y} <daml:subClassOf> {Father},
{X} <rdfs:subClassOf> {Rest},
{Y} <rdfs:subClassOf> {Rest2},
{Rest} <rdf:type> {<daml:Restriction>},
{Rest2} <rdf:type> {<daml:Restriction>},
{Rest} <daml:onProperty> {Prop1},
{Rest2} <daml:onProperty> {Prop2},
{Rest} Z {A},
{Rest2} Z {B}
Where X = <URI of Provider's semantic concept>
and Y = <URI of Client's semantic concept>
and Prop1=Prop2
and A = B
    
```

Figure 3. SeRQL Query

XML Schema String.

The weight of these parameters is globally calculated, although they affect in different ways depending on their nature (syntax or semantics)

C. Evaluation Tests.

A set of different test has been developed to evaluate the efficiency of our algorithm versus the Paolucci et al. The results show how by using the parameter of the profile class in the OWL-S / DAML-S ontology the outcomes obtained are better. The outcomes are better related to service search precision and the response times obtained are similar to Paolucci et al. except in extreme situations (best and worst cases)

In order to develop the tests, a set of different services profiles were created. These profiles were similar between them. To analyse the results the system was queried with several client requests. In Fig. 4 the location in the taxonomy is showed. It includes the main concepts to specify the different service parameters.

1. Test case definition

The algorithm of Paolucci et al. does not take into consideration the fraternal degree. Thus, the test cases are focused to evaluate the results taking into account that the service is closest to the one searched and it has parameters that belong to the fraternal relation but the Paolucci et al. algorithm is unable to obtain the related service profile. Opposite, the proposed algorithm obtains a supplier of the sibling concept required by the client.

Table 1 contains the specifications of the profiles used in the tests. The profiles describe services including the parameters related to ontology concepts. These concepts belong to the established hierarchy (See Fig. 4). In the hierarchy is possible to distinguish traffic events. These events are classified in: Accidents, incidents and traffic measures. The type of the events appears in the hierarchy. The possible types are: forecasted situations, prognosis and occurrences. Moreover, the main alert services are included: e-mail, SMS, and phone voice alerts.

Table 2 shows the proposed scenario, "ClientForecast", and the results for this test case (required profile by client and service profiles offered by providers).

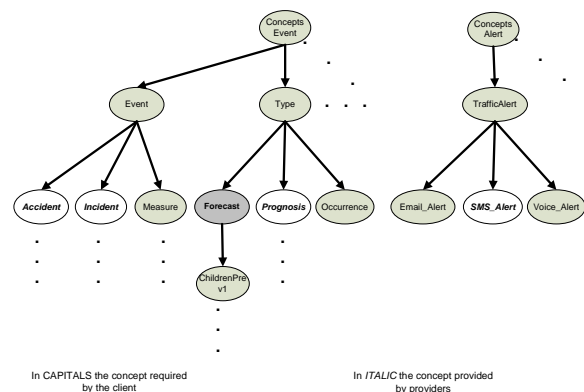


Figure 4. Concepts hierarchy

TABLE I.
TEST SPECIFICATIONS PROFILES

Profile Name	Inputs	Outputs / GR / QoS
InvoiceProfile	User: XMLSchema.xsd#string Password: XMLSchema.xsd#string Road: Roads.daml#Road Init_Km: XMLSchema.xsd#decimal End_Km: XMLSchema.xsd#decimal address: Relations_Events.daml#Way Days: Time.daml#DayofWeek Init_time: Time.daml#Time End_time: Time.daml#Time	Outputs: Confirmation: XMLSchema.xsd#String Alert: Croute.daml#Alert_SMS Geographical radius Geography: geography.daml#Spain QoS : Concepts.daml#Very_good
TrafficCatProfile	Incident: Events.daml#Type_Incident Order: TrafficCatProcess.daml#Order Province: Geography .daml#Province Region: Geography .daml#Region	Inputs: Outputs: Incident_out: Events.daml#Incident Geographical radius : Geography .daml#Catalonian QoS : Concepts.daml#Very_good
TrafficProfile	Incident: Events.daml#Type_Incident Province: Geography .daml#Province State: Geography .daml#State ShowInc: Concepts.daml#ShowType	Outputs: Incident: Events.daml#Incident Geographical radius : Geography .daml#Spain QoS : Concepts.daml#Very_good
PrognosisProfile	Day: Time.daml#Date	Outputs: Prognosis: Events.daml#Prognosis Geographical radius : Geography .daml#Spain QoS : Concepts.daml#Good
AccidentDateProfile	Date: Time.daml#Date	Outputs: Output: Events.daml#Accident Geographical radius : Geography .daml#Spain QoS : Concepts.daml#Good

In the test, as Paolucci et al. algorithm does not take in consideration the fraternal degree, a client request could not have an available response using concept backgrounds and predecessors information. In these cases, it is important to take into account the degree of comparison of sibling nodes. We can appreciate, that the profile resulting of our algorithm corresponds to PrognosisProfile (maximal score: 1+6) but in the Paolucci et al. the result obtained corresponds to AccidentDateProfile (0+6).

2. Response Time Evaluation

The test to evaluate the response time has been developed varying the number of services profiles (1, 3, 5, 8, 10, 15, 20, 25, 30 and 33) and 3 cases are presented:

- Case 1: All available advertisements belong to the same category. The results prove that this is the worst case for the proposed algorithm. This is due to the fact that all service providers inserted belong to the same service category, including the service requested by the client. In this case the service category filter does not reject profiles, so the matching is done with the same number of profiles. Moreover, our algorithm cost is greater, due to the greater number of comparisons. Thus, the Paolucci algorithm obtains the result faster than our algorithm. See Fig. 5.

- Case 2: In this situation, more usual, not all profiles belong to the same category as the profile requested by the client (only a 20%). In this case, the results between two algorithms are closer. See Fig. 6.

- Case 3: Only 1 service belongs to the same category searched by the client. The results probe that this case is the best for our proposal. This is due to the fact that our algorithm only has to match for one service. On the

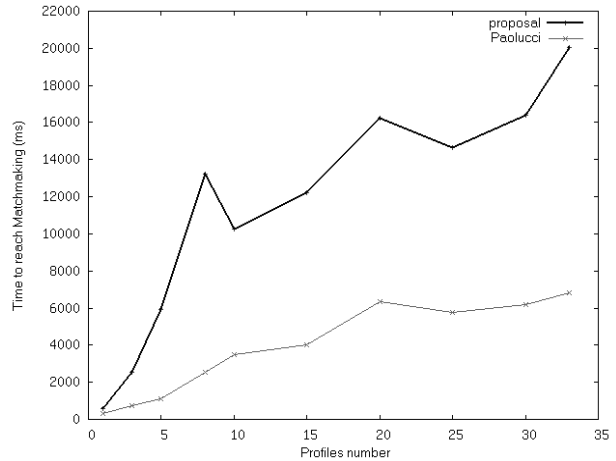


Figure 5. Results of case 1: All the available advertisements belong to the same category

contrary, the Paolucci et al. algorithm has to search for the all the available services. See Fig. 7.

Starting from the tests realized, it was observed how using our algorithm it was possible to find much more compatible services that the ones found by previous research.

V. INTEGRATED SYSTEM OF PUBLICATION AND DISCOVERY OF INFORMATION SERVICES: PROPOSAL FOR A NEW ARCHITECTURE.

At this point we think that the best way to build the SWS environment for mobile devices is to integrate the contributions described previously within a multiagent discovery system. In our case we integrated our contribution within a multiagent discovery system of traffic information Web services starting from existing Web portals. These web portals were adapted to be accessed as web services using auxiliary software agents. The matching proposals that were found [11], [12] do not use the ontologies for semantic service description. The architecture of integration obtained constitutes an extension to that defined in [2] starting from its capability to process the syntactic and semantic similarity between

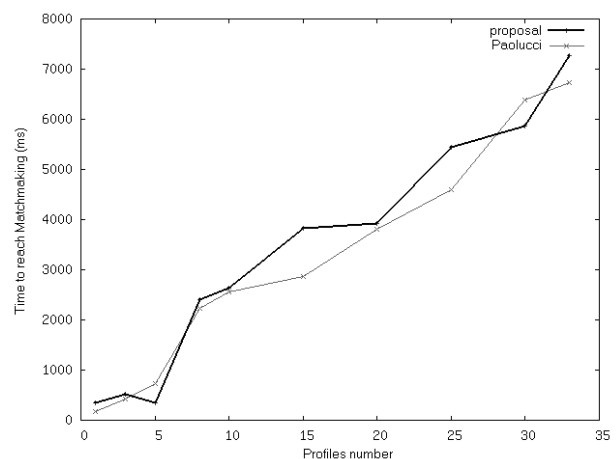


Figure 6. Results of case 2: Only a 20% of the available advertisements belong to the same category.

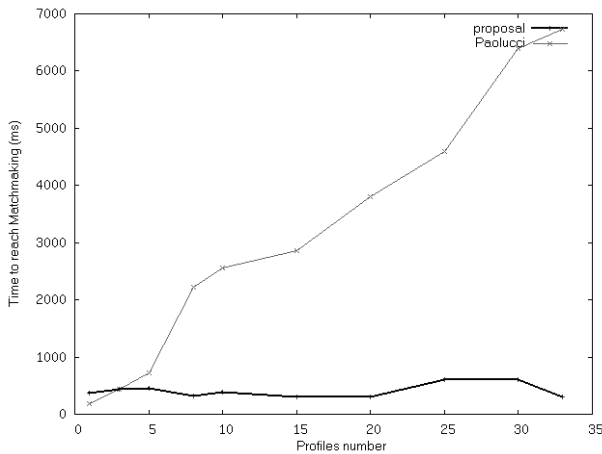


Figure 7. Results of case 3: Only 1 service belongs to the same category required by the client.

capability descriptions of services described in DAML-S.

A. Description of the Multiagent Platform

The platform allows the development of the complete cycle of advertisement, request, discovery, invocation and execution of web services within it.

The multiagent system is the environment in which the clients and service providers interact. It has been implemented using JADE Platform [13]. JADE presents several advantages:

- It follows FIPA standards. It allows agents to communicate with other agents independently of the agent platform.
- It is independent of the development platform, thus the Operative System is not important.
- It implements mechanisms of mobility, security and fault tolerance that makes it easier to develop a more secure environment.
- It is prepared to be used with mobile devices. So, it is possible to design and execute the platform on lightweight devices.

The prototype proposed consists in the implementation of a matchmaker agent that uses a matching algorithm, a provider agent and another client, which are used as representatives, within the platform, of clients and providers respectively, that specify the services required or offered through DAML-S or OWL-S profiles.

In Fig. 8 the collaboration diagram of the matchmaker system can be observed.

Where:

- DF: JADES’s directory facilitator.
- Service: Web service.
- Provider Agent: Its objective is to represent the Web service within the platform, indicating to the matchmaker the DAML-S profile that describes it (URI). The provider

TABLE II. RESULTS OF CLIENT/FORECAST SCENARIO

Request ClientForecastProfile	Available_Services	Score (O/I)	
		Proposal	Paolucci
INPUTS: Day: Time.daml#Date	InvoiceProfile	0/0	0/0
	TrafficCatProfile	0/0	0/0
OUTPUTS: Forecast: Events.daml#Forecast	TrafficProfile	0/0	0/0
	PrognosisProfile	1/6	0/0
Geographical radius : Geography.daml#Spain	AccidentDateProfile	0/6	0/6
QoS: Concepts.daml#Good			

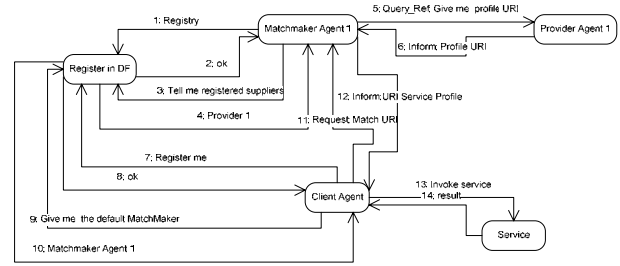


Figure 8. Collaboration Diagram of the Matchmaker system.

agents appear and disappear from the platform as many times as necessary. The services are only available while the agent is active. We decided that the user who wanted to advertise a service within the platform should launch it giving as a parameter the URI of that service description. Besides registering in the DF as provider type only a behaviour that would handle sending the profile service URI should be planned, which represents the matching agent each time it would require it, always following the protocol performances.

- Client Agent: It has as its main objective the representation of the client within the platform. It informs the matchmaker of the service description that is being searched, or how to access it (URI). Specific clients for each service do not exist, since the available services appear and disappear from the platform, and having an available client for each type of service was not a flexible enough solution; therefore it has been established that the client agent be as generic as possible, and as a consequence of this, a graphic interface must be dynamically generated, in function of the entries required by the service with which it has been matched.

- Matchmaker Agent: It implements the matchmaking algorithm. It employs web service matchmaker systems, being the matching the process that requires a repository host to take a query or advertisement as input, and to return all advertisements which may potentially satisfy the requirements specified in the input query or advertisement [15]. This matching serves an alternative or complement to UDDI and has as its main characteristic the fact that it is based on semantics. The matchmaking is in charge of receiving descriptions of service requests and then contrasts them with the descriptions of advertised services to identify which one of them more closely resembles the needs of the client making the request. Each couple obtained is assigned a weight that indicates the semantic proximity between the two concepts that are related. This weight is obtained by querying a reasoner about the hierarchy defined by an ontology of concepts that define the common vocabulary to be used.

After the matching process, once the service profile that most closely resembles what is being searched, the client who has made the request is informed of the existence of that profile (its URI). Then, the client uses this profile to access the service and the grounding files associated to it, to obtain the necessary data for the correct invoking of the service. Besides, the invocation of the service can be made without having to recompile the

client by means of WSIF [14]. The pseudocode of the client and the matchmaker agent can be seen in Fig. 9 and Fig. 10 respectively.

B. General view of the system and interactions between the different agents

The phases that summarize the complete cycle would be as follows:

- a) When a new service A is advertised, its Provider Agent, representative within the platform, registers in the DF, and then DF informs the Matchmaker Agent, which in turn asks the provider agent regarding the semantic description of the service it offers (URI).
- b) A user launches a Client Agent on the platform with the semantic description of the service he is searching. The Client Agent contacts the Matchmaker Agent and informs of the service description being searched. The Matchmaker Agent asks to the reasoner and contrasts the description of R service with those of all the available providers, deciding that service A is what most closely resembles that described by R and it passes the URI to the client agent.
- c) Once the Client Agent dynamically knows the description that closely resembles R semantically, it generates a graphic interface to obtain the entry data necessary to make a correct invocation of service A.
- d) When the user fills out the form that asks for service entry data, the client agent invokes it directly with that data and obtains the result that is returned.
- e) The Client Agent contacts the user again and shows

- 1) Register in DF
- 2) Ask to DF for Matchmaker identifier
- 3) Send Matchmaking request to the matchmaker agent
- 4) Receive service profile
- 5) Process service profile to obtain input parameters
- 6) Create graphic interface depending on service profile
- 7) gather introduced data and invoke the service
- 8) gather result of the invocation
- 9) show the result

Figure 9. Pseudocode of the client agent

- 1) Create list of request profiles
- 2) Create list of result couples
- 3) Create list of URI's profiles from providers to insert
- 4) Process input arguments
- 5) Registry in DF like Matchmaker type
- 6) listprov= look_for in DF all provider agents
 - For every element from listprov
 - {
 - ask the profile
 - }
- 7) Run matchmaker thread
- 8) Subscribe in DF to be notified when a provider is registered
- 9) Create behaviour 0 to solve notifications from DF
- 10) Create behaviour 1 to solve request from client and provider agents
- 11) Create behaviour 2 to check every N seconds if there are results for notifying
- 12) Insert in the scheduler the behaviours

Figure 10. Pseudocode of the matchmaker agent

the result of the service invocation.

In Fig. 10 the components of the complete system can be seen.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, a new architecture to retrieve information is presented. The architecture has been designed to be used in multiple devices, especially in mobile devices, from Web using Semantic Web Services.

A semantic web services framework to support mobile web services has been designed and implemented. The framework in combination with the use of the proposed matchmaking algorithm present positive results: they allow finding services based on their similarities. The algorithm uses ontologies, taking account concepts, to find the Web Services. These Web Services appear at the same level, i.e. sibling concepts, but they have different restrictions.

The tests carried out to the matchmaking algorithm confirm that it is possible to find services based on similarities of the "sibling" type. This type of similarity could not have been held in consideration in the case of finding similarities between sibling nodes, since a design point of view is better to create an intermediary father node that contains the common characteristics of the children.

The proposed algorithm holds them starting from the premise that it is possible that this type of similarities have not been considered by the provider that semantically defined these relations (due to considerations of performance or simply because this type of relations were not detected in the design phase), by which the discovery of services based on conventional algorithms would not produce the best results.

Other decisions such as the establishment of filter previous to the algorithm as well as the use of other non-functional parameters considerably improved the cost and the election of the ideal service, respectively.

Nowadays, the authors go on developing complex ontologies of traffic information for applying them to the real systems belonging to traffic administrations.

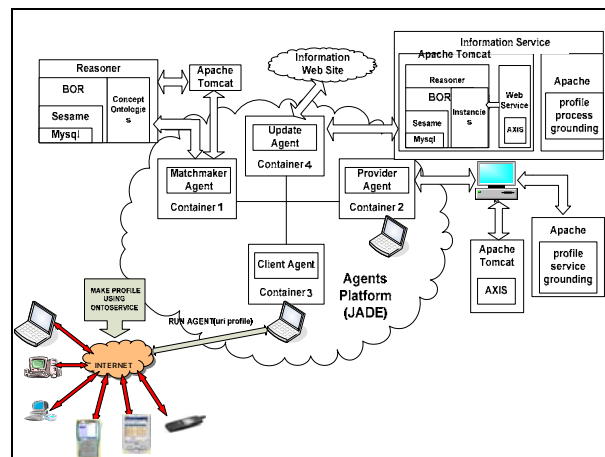


Figure 11. Detail of the multiagent architecture.

ACKNOWLEDGMENT

Our thanks to ESAM (Estudios de Software Avanzado y Mantenimiento de Tecnología, S.L.). This research forms part of the Automatic Incident Management Project of the Dutch Ministry of Transport.

REFERENCES

- [1] A. Malik. "XML, Ontologies, and the semantic Web", The second generation of the web, M2002 Published in XML Journal, January 2003
- [2] T. Payne and O. Lassila, "Semantic Web Services". IEEE Intelligent Systems. Special Issue on Semantic Web Services. vol. 19, pp. 14-15, 2004
- [3] OWL Web Ontology Language, W3C Recommendation, available on: <http://www.w3.org/TR/owl-features/>
- [4] Pellet home page: <http://pellet.owldl.com/>
- [5] Jena – A Semantic Web Framework for Java, Hewlett-Packard Development Company , available on: <http://jena.sourceforge.net/>
- [6] Web Tools Platform (WTP) Project home page: <http://www.eclipse.org/webtools/>
- [7] Apache Axis home page: <http://ws.apache.org/axis/>
- [8] Apache Tomcat home page: <http://tomcat.apache.org/>
- [9] M. Paolucci, T. Kawamura, T. Payne, & K. Sycara, "Semantic Matching of Web Services Capabilities". The First International Semantic Web Conference (ISWC), 2002
- [10] J. Broekstra. "SeRQL: Sesame RDF query language". In M. Ehrig et al., editors, SWAP Deliverable 3.2 Method Design, pages 55-68. 2003.
- [11] M. Klein and A. Bernstein, "Searching for Services on the Semantic Web using Process Ontologies". in The First Semantic Web Working Symposium (SWWS-1). 2001. Stanford, CA USA.
- [12] D. Trastour, C. Bartolini, and J. Gonzalez-Castillo, "A Semantic Web Approach to Service Description for Matchmaking of Services", Proc. 1st Semantic Web Working Symposium, CA, 2001.
- [13] F. L. Bellifemine (Editor), G. Caire (Editor), and D. Greenwood (Editor), "Developing Multi-Agent Systems with JADE", Wiley Publishers, 2006.
- [14] M. Duftler et al., Web Services Invocation Framework (WSIF), OOPSLA Workshop on Object Oriented Web Services, October 2001.
- [15] L. Li and I. Horrocks, "A Software Framework for Matchmaking Based on Semantic Web Technology". Int. J. of Electronic Commerce, 8(4):39-60, 2004.
- [16] Zaremski, Amy Moormann, and Wing Jeannette M. "Signature matching: a Tool for Using Software Libraries", 1995.
- [17] Zaremski, Amy Moormann, and Wing Jeannette M., "Specification Matching of Software Components", 1997.
- [18] K. Sycara, M. Klusch, S. Widoff, and J. Lu "Dynamic Service Matchmaking among Agents in Open Information Environments", In A. Ouksel and A. Sheth Eds. Journal ACM SIGMOD Record, Special Issue on Semantic Interoperability in Global Information Systems, 1999.
- [19] J. Gonzalez-Castillo, D. Trastour and C. Bartolini. "Description Logics for MatchMaking of Services", HP Technical Reports, October 2001.
- [20] T. R. Payne; M. Paolucci and K. Sycara, "Advertising and Matching DAML-S Service Descriptions", Semantic Web Working Symposium (SWWS), 2001.
- [21] C. Abela and M. Montebello. "DAML enabled Web Services and Agents in the Semantic Web", WS-RSD'02, Erfurt Germany, October 2002.

J. Javier Samper received his PhD degree in Computer Science from University of Valencia (Spain) in 2005.

He is a scientific assistant at the Robotics Institute and teacher at the University of Valencia. He works in the areas of software engineering and consulting in the Intelligent Transportation Systems (ITS) sectors. His research focus is on information traffic distribution and the access to it.

F. Javier Adell received his M.S degree in computer engineering from University of Valencia, in 2007.

He works at the moment at the University of Valencia and his research interests include web services, ontologies and semantic web.

Leo van den Berg works at the moment at the University of Valencia and has over 25 years of experience in the transportation domain. He was project manager for different large scale traffic information projects and he is also well known for his experience in the area of ITS-architectures and traffic enforcement.

J. José Martínez received his PhD degree in Computer Science from University of Valencia (Spain) in 1995.

He works for the University of Valencia (Computer Sciences Department). He has participated in the EU projects from the beginning of their research work. Currently, he is the head of Robotics Institute and LISITT laboratory, which is a working group dedicated to ITS and telematic applications, mainly to the traffic and transport fields.