Developing Multi Agent Systems on Semantic Web Environment using SEAGENT Platform

Oguz Dikenelli[1], Riza Cenk Erdur[1], Geylani Kardas[2], Özgür Gümüs[1], Inanç Seylan[1], Önder Gürcan[1], Ali Murat Tiryaki[1], and Erdem Eser Ekinci[1]

[1]Ege University, Department of Computer Engineering,
35100 Bornova, Izmir, Turkey
{oguz.dikenelli,cenk.erdur,ozgur.gumus,inanc.seylan,onder.gurcan,
ali.murat.tiryaki}@ege.edu.tr, erdemeserekinci@gmail.com
[2]Ege University, International Computer Institute,
35100 Bornova, Izmir, Turkey
geylani.kardas@ege.edu.tr

Abstract. In this paper, we discuss the development of a multi agent system working on the Semantic Web environment by using a new framework called SEAGENT. SEAGENT is a new agent development framework and platform, which includes built-in features for semantic web based multi agent system development. These features provide semantic supports such as a new specific content language for transferring semantic knowledge, specifically designed agent's internal architecture to handle semantic knowledge, a new directory facilitator architecture based on semantic service matching engine and ontology management service to provide ontology translations within the platform's ontologies. The implemented case study shows the effectiveness of these features in terms of semantically enriched multi agent system development.

1 Introduction

The standardization effort on Semantic Web [2] aims to transform the World Wide Web into a knowledge representation system in which the information provided by web pages is interpreted using ontologies. This creates an environment in which knowledge is defined in terms of these ontologies and information systems are designed and implemented in a way that these ontologies are used, transferred and regenerated. In these environments, agents place a critical role to autonomously collect, interpret and use semantic knowledge as a part of future's information systems.

In this paper, we discuss how to develop MASs in such semantically enriched environments with a new framework called SEAGENT. SEAGENT, which is introduced in [8], is a new agent development framework and platform that is specialized for semantic web based MAS development. The communication and plan execution infrastructure of SEAGENT looks like other existing agent development frameworks such as DECAF [10], JADE [1], and RETSINA [16]. However, to support and ease semantic web based MAS development, SEAGENT

includes the following built-in features that the existing agent frameworks and platforms do not have:

* SEAGENT provides a specific feature within the agent's internal architecture to handle the agent's internal knowledge using OWL (Web Ontology Language) [13].

* The directory service of SEAGENT stores agent capabilities using specially designed OWL based ontologies and it provides a semantic matching engine to find the agents with semantically related capabilities.

* Based on FIPA-RDF [9], a content language called Seagent Content Language (SCL) has been defined to transfer semantic content within the agent communication language messages.

* SEAGENT introduces a new service called Ontology Management Service (OMS). The most important feature of this service is to define mappings between platform ontologies and external ontologies. Then it provides a translation service to the platform agents based on these defined mappings.

* SEAGENT supports discovery and dynamic invocation of semantic web services by introducing a new platform service for semantic service discovery and a reusable agent behavior for dynamic invocation of the discovered services.

The paper is organized as follows: Section 2 gives a brief overview of the related work. Section 3 explains the overall architecture of the SEAGENT framework. In section 4, a case study on MAS development using SEAGENT is discussed in detail. This case study demonstrates the MAS implementation taking into account of platform initialization, agent plan development using semantic knowledge and semantic capability matching on platform's Directory Facilitator (DF). Conclusion is given in section 5.

2 Related Work

The idea of integrating the semantic web and agent research has already been realized and some systems have been developed. ITtalks [5] system offers access to information about activities such as talks and seminars related with information technology. ITtalks uses DAML+OIL for knowledge representation and lets agents to retrieve and manipulate information stored in the ITtalks knowledge base. The smart meeting room system [4] is a distributed system that consists of agents, services, devices and sensors that provide relevant services and information to the meeting participants based on their contexts. This system uses semantic web languages for representing context ontologies. Both the ITtalks and the smart meeting room system use a multi-agent development framework in their underlying infrastructure. For example, ITtalks uses Jackal [6] and smart meeting room system uses Jade [1]. In these systems, semantic web functionality is hard coded into the system together with the domain knowledge, because the agent frameworks used in the implementation of these systems do not have a built-in support for semantic web. For example, it is difficult for these systems' developers to support basic semantic web functionalities such as discovering and dynamically invoking of semantic web services inside an agent or performing

an ontology translation between different platform ontologies. Moreover, it requires knowledge for ordinary developers to handle the semantic web and agent technology details in addition to the application domain related knowledge.

There have been a few implementations to integrate web services and FIPA compliant agent platforms. WSDL2JADE [22] can generate agent ontologies and agent codes from a WSDL input file. WSIG (Web Services Integration Gateway) [19] supports bi-directional integration of web services and Jade agents. WS2JADE [21] allows deployment of web services as Jade agents' services at run time. But these tools only deal with agent and web service integration and do not provide any mechanism to use semantic web knowledge during MAS development.

There is an attempt called as "JADE Semantic Agent" [20], to integrate FIPA ACL semantics into a multi agent development framework. It is implemented on top of the JADE framework and it has a built-in mechanism to interpret the semantics of FIPA messages. This is a very noble attempt but it is highly dependent on the FIPA-SL language which seems to be a problem when sending OWL content between agents. We believe that it still remains a problem to define ACL semantics in a way compatible with OWL.

We can conclude from this discussion that there must be environments, which will simplify semantic web based multi agent system (MAS) development for ordinary developers and which will support the basic semantic web functionalities.

3 Platform Architecture

In this section, we explain SEAGENT's layered software architecture briefly. Each layer and packages of the layers have been specially designed to provide build-in support for MAS development on Semantic Web environment. The overall architecture is shown in Fig. 1. Although the given architecture is the implemented architecture of the SEAGENT platform, we believe that it is generic enough to be considered as a conceptual architecture of MASs those are developed and deployed for semantic web environment. In the following subsections, we discuss each layer with an emphasis on the semantic support given by that layer.

3.1 Communication Infrastructure Layer

This bottom layer is responsible of abstracting platform's communication infrastructure implementation. SEAGENT implements FIPA's Agent Communication and Agent Message Transport specifications [9] to handle agent messaging. Although Communication Infrastructure Layer can transfer any content using FIPA ACL and transport infrastructure, SEAGENT platform only supports Seagent Content Language (SCL) by default. SCL itself is a specific OWL ontology to define the ACL content. It is based on the FIPA-RDF but extends the FIPA-RDF by defining new concepts and relations. So, the language itself is not OWL like Zou et. al's work [18], but it is serialized into OWL. This allows

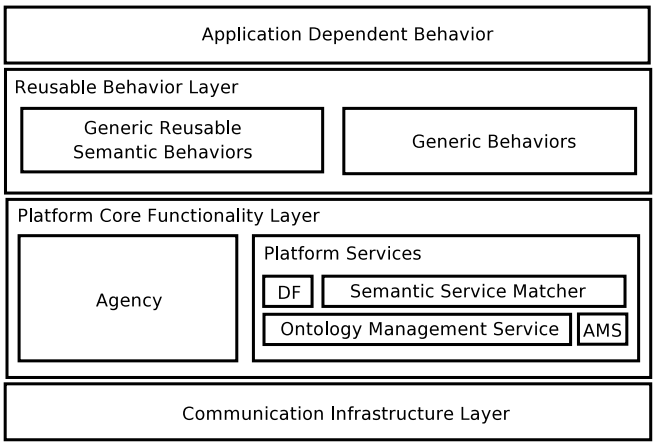

Fig. 1. SEAGENT Platform Overall Architecture

content to be easily parsed and takes advantage of directly inserting concepts / individuals from OWL ontologies which form the knowledge bases of services and agents.

In order to be used with FIPA-ACL, a content language must satisfy three requirements [3]. The first two states that the language must be capable of representing propositions and actions. This is done in SEAGENT by defining those two concepts in the SCL ontology. The third requirement is that it must be capable of representing objects, including identifying referential expressions to describe objects. To achieve this, we have defined a query and match ontology in OWL which is called "Seagent Match Ontology". The concepts of this ontology are used to define required content and are directly inserted into SCL based content to represent objects.

3.2 Platform Core Functionality Layer

Agency Package The second layer includes packages, which provide the core functionality of the platform. The first package, called as Agency, handles the internal functionality of an agent. Agency package supports the creation of general purpose and goal directed agents. In this sense, Agency package provides a built-in "agent operating system" that matches the goal(s) to defined plan(s), which are defined using HTN planning formalism [14]. It then schedules, executes and monitors the plan(s). From semantic web based development perspective, an agent's internal architecture must support semantic web ontology standards for messaging and internal knowledge handling to simplify semantic based development. For this purpose, Agency package provides a build-in support to parse and interpret SCL content language to handle semantic web based messaging. On the other hand, Agency provides two interfaces for semantic knowledge handling,

one for local ontology management and the other one for querying. Although the current version includes the JENA [11] based implementation of these interfaces, other semantic knowledge management environments and query engines can be integrated to the platform by implementing these interfaces.

Platform Services The second package of the Core Functionality Layer includes service sub-packages, one for each service of the platform. SEAGENT provides all standard MAS services such as DF Service and Agent Management Service (AMS) following the previous platform implementations and FIPA standards. But these standard services are implemented in a different way by using the capabilities of a semantic web infrastructure.

In SEAGENT implementation, DF uses an OWL ontology to hold agent capabilities and includes a semantic matching engine to be able to return agent(s) with semantically similar capabilities to the requested ones. Matchmaking process in case is realized within the built-in capability matching engine of the DF which is called *Seagent Matching Engine.* This engine matches advertised agent services with the received service request. It stores agent service definitions in a database. Actually this database is an ontology model of the agent services in which agent service ontology individuals are included. Therefore each agent service that is registered to the DF is also represented in this ontology with an individual as it is discussed above. The matching engine uses those individuals and compares them with given service requests semantically. Seagent Matching Engine uses a basic reasoner called *Ontolog* to determine semantic relation between agent services. We have adapted the service matching algorithm originally proposed in [15] for semantic web services into the matchmaking process of agent services. The Ontolog works on ontology hierarchy tree of service concepts and finds distance between any given two classes (i.e. service types of requested and advertised agent services). Based on the results returned from the Ontolog, Seagent Matching Engine defines and uses a degree of match function named DoM(C_1, C_2) which determines semantic match degree between concepts, C_1 and C_2:

DoM(C_1, C_2) = EXACT if C_1 is a direct subclass of C_2 or $C_1 = C_2$
DoM(C_1, C_2) = PLUG-IN if C_1 is a distant subclass of C_2
DoM(C_1, C_2) = SUBSUMES if C_2 is a direct or distant subclass of C_1
DoM(C_1, C_2) = FAIL otherwise

Matching engine of the agent platform takes the above defined relations into account and determines the suitability of the advertised agent services with the requested one. The internal architecture and theoretical base of the engine is introduced in [12].

Similarly, AMS stores descriptions of agents in OWL using FIPA Agent Management Ontology [9] and can be queried semantically to learn descriptions of any agent that is currently resident on the platform.

Besides implementing standard services in a semantic way, SEAGENT platform provides two new services to simplify semantic web based MAS development. The first one is called as Semantic Service Matcher (SSM). SSM is re-

sponsible for connecting the platform to the semantic web services hosted in the outside of the platform. SSM uses "service profile" construct of the Web Ontology Language for Semantic Web Services (OWL-S) [17] standard for service advertisement and this knowledge is also used by the internal semantic matching engine for discovery of the service(s) upon a request. SSM and DF services are implemented by extending a generic semantic matching engine architecture, which are introduced in [7] and [12] in detail.

The second unique service is the Ontology Management Service (OMS). It behaves as a central repository for the domain ontologies used within the platform and provides basic ontology management functionality such as ontology deployment, ontology updating and querying etc. The most critical support of the OMS is its translation support between the ontologies. OMS handles the translation request(s) using the pre-defined mapping knowledge which is introduced through a specific user interface. Through the usage of the ontology translation support, any agent of the platform may communicate with MAS and/or services outside the platform even if they use different ontologies.

3.3 Reusable Behaviour Layer

Third layer of the overall architecture includes pre-prepared generic agent plans. We have divided these generic plans into two packages. Generic Behavior package collects domain independent reusable behaviors that may be used by any MAS such as well known auction protocols (English, Dutch etc.). On the other hand, Generic Semantic Behaviors package includes only the semantic web related behaviors. In the current version, the most important generic semantic behavior is the one that executes dynamic discovery and invocation of the external services. This behaviour is defined as a pre-prepared HTN structure and during its execution, it uses SSM service to discover the desired service and then using OWL-S "service grounding" construct, it dynamically invokes the found atomic web service(s). Detail of this behaviour is explained in [7]. Hence, developers may include dynamic external service discovery and invocation capability to their plan(s) by simply inserting this reusable behavior as an ordinary complex task into their HTN based plan definition(s).

4 Developing a MAS with using SEAGENT through a Case Study

In this section, development of a simple MAS using SEAGENT framework is discussed to demonstrate semantic knowledge handling capability of the framework. We first describe scenario of the implemented case study. Then initialization of the MAS on semantic web enviroment, plan and behaviour structure of the working agents and internal workflow of the system's semantic DF service are explained respectively.

4.1 Scenario

The agent environment in case is about Tourism domain in which traveler agents try to reserve hotel rooms on behalf of their users while some other agents are offering hotel services for those ones. In our prototype MAS, we have a traveler agent and four hotel agents. Initially each one is unaware of the others. Those four hotel agents are registered to the DF of the MAS with their service advertisements. Those agents use an agent description ontology called "fipa-agent-management.owl" to advertise themselves (including their services and related information) in DF. "df-agent-description" and "service-description" concepts defined in this ontology are given in Fig. 2. This ontology involves the concepts given in FIPA Agent Management Specification [9], thus making the platform compatible with FIPA Specifications.

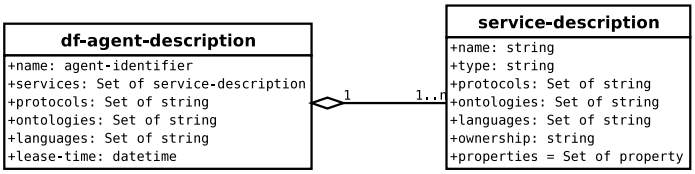

Fig. 2. DF Agent Description and Agent Service Description concepts

On the other hand, we use two properties of service description *(type* and *properties)* to define agent services semantically. Values of these properties may come from various domain ontologies. Therefore they involve URI of related ontology concepts. For example in our case, hotel agents use "HotelInfoService" concept which is defined in the OWL ontology called "TourismServices.owl" to set *type* property of service description instance and to advertise themselves in DF. This means that hotel agents provide a service called "HotelInfoService" to other agents.

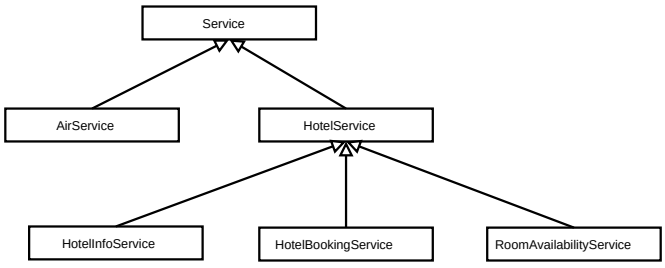

Fig. 3. TourismServices ontology

According to our scenario, hotel agents in here provide activities to their customers. Hence, service descriptions include a service property called "activity" within the set of service description *properties*. The range of the *activity* property is an individual of the concept that comes from another domain ontology called "Hotel.owl". TourismServices ontology and a fragment of Hotel Ontology are given in Fig. 3 and Fig. 4 respectively. Service types and activities available in each hotel agent's service description are given in Table 1.

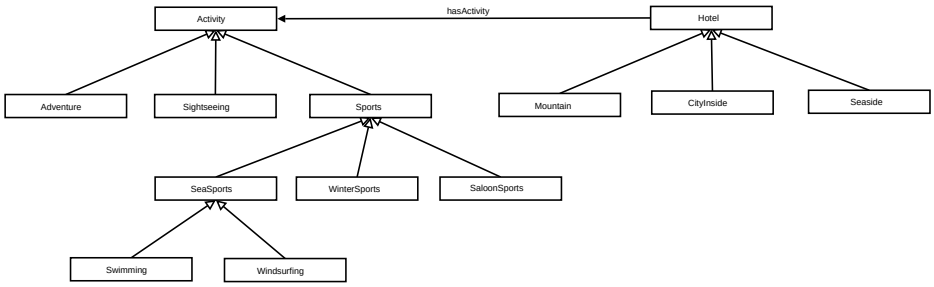

Fig. 4. A Fragment of Hotel Ontology

In the scenario, our traveler agent looks for suitable hotels in which "SeaSports" activity is available. Hence, it first communicates with DF of the MAS, receives DF descriptions of the suitable agents and then calls "HotelInfoService" service of those agents to get further information about the hotel.

Table 1. Four hotel agent services registered into the directory facilitator

Agent's name	Service type	Activity type
hotel1@seagent.com	TourismServices.owl#HotelInfoService	Hotel.owl#Windsurf
hotel2@seagent.com	TourismServices.owl#HotelInfoService	Hotel.owl#WinterSports
hotel3@seagent.com	TourismServices.owl#HotelInfoService	Hotel.owl#Swimming
hotel4@seagent.com	TourismServices.owl#HotelInfoService	Hotel.owl#SeaSports

4.2 Initiating the Platform

To instantiate the platform, the standard platform services are started first. These are AMS, ACC (Agent Communication Channel) and DF in order. Agents and services which aren't registered to an AMS are not considered to be a part of a platform, therefore registration to AMS is the initial behavior of all entities (agents and services). The AMS maintains an OWL instance of the FIPA Agent Management Ontology. When agents register themselves, their agent descriptions

are kept in this instance. The interaction between AMS and the agents are as stated by FIPA specification [9]. If there is no problem with the content delivered to AMS, it sends an *agree* message and if the agent is successfully registered, an *inform* message is sent back by the AMS.

After the initialization of AMS, ACC is started. All communication is done through ACC, thus it is needed when agents send their "register to AMS" message. Finally, DF is created. It is not mandatory that agents register themselves with DF on their creation as in AMS. This is why all entities take AMS and ACC address in the construction but not the DF address. After the DF starts its operation, it broadcasts that it is working. Then platform's AMS and ACC advertises their service descriptions to the DF. The types of these services are reserved – *fipa-acc* and *fipa-ams* – so no other agent can advertise themselves with those parameters.

When the instance of the standard Seagent platform is ready, it is then populated with five agents mentioned above. These agents are supplied with their AMS agent descriptions when they are created. They then use these agent descriptions to register to AMS, which is already stated as a mandatory operation. The agents are also planned to register their services with the DF. As in AMS, the knowledge base of DF is an OWL ontology. This ontology has instances of DF agent descriptions. An instance of DF agent description for one of the hotel agents in N3 format is given in Fig. 5.

```
@prefix ts:      <http://aegeants.ege.edu.tr/ont/TourismServices.owl#> .
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .
@prefix hotel:   <http://aegeants.ege.edu.tr/ont/Hotel.owl#> .
@prefix am: <http://aegeants.ege.edu.tr/ont/fipa-agent-management.owl#>.
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix :        <#> .
@prefix owl:     <http://www.w3.org/2002/07/owl#> .

[]     a       am:DFAgentDescription ;
       am:language "http://aegeants.ege.edu.tr/ont/scl.owl" ;
       am:name [ a       am:AgentIdentifier ;
                 am:name "hotel4@seagent.com" ] ;
       am:ontology _:b1 ;
       am:service
                [ a       rdf:Bag ;
                  rdf:_1  [ a       am:ServiceDescription ;
                            hotel:activity hotel:SeaSports ;
                            am:name "sea sports service" ;
                            am:ontology _:b1 ;
                            am:ownership "Foo Co." ;
                            am:type ts:HotelInfo ] ] .
_:b1   a       rdf:Bag ;
       rdf:_1  "http://aegeants.ege.edu.tr/ont/TourismServices.owl" ;
       rdf:_2  "http://aegeants.ege.edu.tr/ont/Hotel.owl" .
```

Fig. 5. DF Agent Description of a Hotel Agent in N3 format

4.3 Internal Plan of the Traveler Agent

In the Seagent Platform, there is a generic plan in which agents query on DF and evaluate the match results to select agent(s) with appropriate service(s). This plan simply contains one behaviour called "Find Agent from DF" and this behaviour is composed of two actions: "Create Query and Send It to DF" and "Select Agent". The HTN structure of this plan is given in Fig. 6.

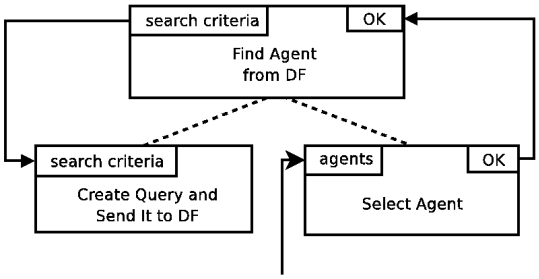

Fig. 6. HTN strcuture of the generic "Find Agent" plan

In the first action, the agent retrieves search criteria provision which is created according to the previously mentioned Seagent Match Ontology. The search criteria in here can define anything that can be retrieved from DF such as a service description or an agent DF description. To model the criteria, a concept which is called SeagentMatchRequest as a part of the Seagent Match Ontology is used [12]. A SeagentMatchRequest has properties such as "hasPremise", "hasQuery" and "hasSemanticMatch" to define RDF (Resource Description Framework) triples which will be used in semantic match on DF. In "hasPremise" and "hasQuery" lists, the requester defines RDF sentences of the RDQL (RDF Data Query Language) [11] query which will be executed before semantic match to filter result set according to non-semantic parameters. On the other hand, the requester specifies each semantic parameter and its ontological value in "hasSemanticMatch" list to be used during semantic match on filtered query results. Result type and desired semantic match degree of the match process is given in "mustBindVariable" and "matchDegree" properties of the SeagentMatchRequest.

In our scenario, the traveler agent prepares the proper SeagentMatchRequest instance in which a DF Agent Description is requested with *TourismServices.owl#HotelInfoService* service and *Hotel.owl#SeaSports* activity. In this request, match degree is emphasized as *SUBSUMES* and semantic matching is requested on these two fields. That means the traveler agent accepts all agent services which are semantically related with *TourismServices.owl#HotelInfoService* service. Likewise the traveler agent also requests hotel activities which are semantically related with *Hotel.owl#SeaSports* activity.

In order to be sent to DF, the instance of SeagentMatchRequest is serialized in the outgoing ACL message based on Seagent Match Ontology. The content of the message is shown in Fig. 7. Due to space limitations, the namespaces in Fig. 5 are not given here again. As it is seen, the content language (SCL) itself is an OWL ontology. Therefore the SeagentMatchRequest instance corresponds to an individual in this ontology. It is the argument of the *search* action that is requested from DF.

```
@prefix match:   <http://aegeants.ege.edu.tr/ont/
                  seagent-match-ontology.owl#> .
@prefix scl:     <http://aegeants.ege.edu.tr/ont/scl.owl#> .

[]     a   scl:Action ;
       scl:act "search" ;
       scl:actor
          [ a       am:AgentIdentifier ;
            am:name "df@seagent.com" ] ;
       scl:argument _:m1 .
_:m1   a   match:SeagentMatchRequest ;
       match:hasPremise
          [ match:object "fipa-agent-management.owl#ServiceDescription" ;
            match:predicate "http://www.w3.org/02/22-rdf-syntax-ns#type";
            match:subject "?s" ] ;
       match:hasPremise
          [ match:object "fipa-agent-management.owl#DFAgentDescription" ;
            match:predicate "http://www.w3.org/02/22-rdf-syntax-ns#type";
            match:subject "?x" ] ;
       match:hasPremise
          [ match:object "http://.../ont/Hotel.owl#Activity" ;
            match:predicate "http://www.w3.org/02/22-rdf-syntax-ns#type";
            match:subject "?a" ] ;
       match:hasQuery
          [ match:object "?a" ;
            match:predicate "http://.../ont/Hotel.owl#activity" ;
            match:subject "?s" ] ;
       match:hasQuery
          [ match:object "?s" ;
            match:predicate "http://../fipa-agent-management.owl#service
                ";
            match:subject "?x" ] ;
       match:hasSemanticMatch
          [ match:object "http://.../ont/TourismServices.owl#HotelInfo" ;
            match:predicate "http://.../fipa-agent-management.owl#type" ;
            match:subject "?s" ] ;
       match:hasSemanticMatch
          [ match:object "http://.../ont/Hotel.owl#SeaSports" ;
            match:predicate "http://www.w3.org/02/22-rdf-syntax-ns#type";
            match:subject "?a" ] ;
       match:matchDegree "SUBSUMES" ;
       match:mustBindVariable "?x" .
```

Fig. 7. N3 formatted Seagent Match Request transferred in the ACL Message content

In the second action, the traveler agent receives DF Agent Descriptions those are matched with the above request in a SeagentMatchResultSet instance. Each element in this result set is a SeagentMatchResult and they are ordered according to their match degrees. It should be noted that the DF of the MAS uses OWL

representations of those results to put them into the ongoing ACL message. So the traveler agent parses the content and de-serializes each result object to proceed on its task. During this de-serialization it uses "seagent-match-ontology.owl" to understand ontological content of the result objects. Since match results are also semantically defined in the "seagent-match-ontology" as match requests, the traveler agent retrieves query results to properly use in its plan execution. After all, the traveler agent successfully retrieves appropriate services and it communicates with hotel agents starting from the first element of the result set.

4.4 Semantic Capability Matching on DF

When DF of the MAS receives request of the traveler agent; it determines proper hotel agents - that means semantically "right" agents - and returns their descriptions back to the traveler agent. As first, the engine performs an RDQL query on the advertised hotel agent services and filters them according to the non-semantic parameters. Then, it uses its reasoner (Ontolog) to determine semantic relationship between the given request and recently filtered service advertisements. As given in the request of the traveler, semantic query is performed on service type and activity property of the descriptions. For both semantic parameters, match degree is desired as "SUBSUMES" in the request. Semantic match on service type is straightforward and all the advertised services are acceptable. However, the traveler agent have asked for the hotel info services those have at least a subsumes relationship between the given request activity type (SeaSports in case). So, the engine matches the service descriptions hotel1, hotel3 and hotel4 with the given request and sorts the match results starting from the most exact one(s) in the following order: hotel4, hotel1, hotel3 with EXACT, SUBSUMES and SUBSUMES match degrees respectively. Each match result is returned in a SeagentMatchResult object.

5 Conclusion

The main contribution of this study is to present how to develop a MAS running on the Semantic Web environment. The case study, that is discussed in here, has been implemented successfully by using the semantic features of the SEAGENT platform. SEAGENT both presents a new development framework and a platform that developers can use to create semantically enriched MASs. That means ACL content transfer, agent service discovery and agent planning would all be performed via processing the semantic knowledge of the environment.

Acknowledgments

This work is supported in part by the Scientific and Technical Research Council of Turkey (TÜBITAK), Project Number: 102E022. This support is gratefully acknowledged.

References

1. Bellifemine, F., Poggi, A., and Rimassa, G.: Developing Multi-agent Systems with a FIPA-compliant Agent Framework, Software Practice and Experience, 31 (2001) 103-128
2. Berners-Lee, T., Hendler, J. and Lassila, O.: The Semantic Web, Scientific American, 284(5) (2001) 34-43
3. Botelho, L., Willmott, S., Zhang, T., and Dale, J.: A review of Content Languages Suitable for Agent-Agent Communication, EPFL I&C Technical Report #200233.
4. Chen, H., et al.: Intelligent Agents Meet Semantic Web in a Smart Meeting Room, in the proc. of Autonomous Agents and Multi Agent Systems 2004 (AAMAS'04), NY, USA
5. Cost, R. S., et al.: Ittalks: A Case Study in the Semantic Web and DAML+OIL, IEEE Intelligent Systems, January-February (2002) 40-46
6. Cost, R. S., et al.: Jackal: A Java-Based Tool for Agent Development, in the proc. workshop tools for Developing Agents (AAAI98), AAAI Pres, Calif. (1998) 73-82
7. Dikeneli, O., Gümüs, O., Tiryaki, A. M. and Kardas, G.: Engineering a Multi Agent Platform with Dynamic Semantic Service Discovery and Invocation Capability, Multiagent System Technologies - MATES 2005, Lecture Notes in Computer Science (Subseries: Lecture Notes in Artificial Intelligence), Springer-Verlag, Vol. 3550 (2005) 141-152
8. Dikeneli, O., Erdur, R. C., Gumus, O., Ekinci, E. E., Gurcan, O., Kardas, G., Seylan, I. and Tiryaki, A. M.: SEAGENT: A Platform for Developing Semantic Web Based Multi Agent Systems, AAMAS'05, ACM AAMAS (2005) 1271-1272
9. FIPA (Foundation for Intelligent Physical Agents): FIPA Specifications, available at: http://www.fipa.org
10. Graham, J. R., Decker, K. S. and Mersic, M.: DECAF - A Flexible Multi Agent Systems Infrastructure, Journal of Autonomous Agents and Multi-Agent Systems, 7 (2003) 7-27
11. JENA - A Semantic Web Framework for Java, available at: http://jena.sourceforge.net
12. Kardas, G., Gümüs, Ö. and Dikeneli, O.: Applying Semantic Capability Matching into Directory Service Structures of Multi Agent Systems", Computer and Information Sciences - ISCIS 2005, Lecture Notes in Computer Science, Springer-Verlag, Vol. 3733 (2005) 452-461
13. McGuiness, D. L., and van Harmelen, F.: OWL Web Ontology Language Overview, (2004), available at: http://www.w3.org/TR/owl-features/
14. Paolucci, M., et al.: A Planning Component for RETSINA Agents, Intelligent Agents VI, LNAI 1757, N. R. Jennings and Y. Lesperance, eds., Springer Verlag, 2000
15. Sycara, K., Paolucci, M., Ankolekar, A., and Srinavasan, N.: Automated discovery, interaction and composition of Semantic Web Services, Journal of Web Semantics, Elsevier 1 (2003) 27-46
16. Sycara, K., Paolucci, M., Van Velsen, M. and Giampapa, J.: The RETSINA MAS Infrastructure, Journal of Autonomous Agents and Multi-Agent Systems, 7 (2003) 29-48
17. The OWL Services Coalition: OWL-S: Semantic Markup for Web Services, availabe at: http://www.daml.org/services/owl-s/1.1/
18. Zou, Y., Finin, T., Ding, L., Chen, H., and Pan, P.: Using Semantic Web Technology in Multi-Agent Systems: A Case Study in the TAGA Trading Agent Environment, ICEC 2003, Oct 2003, Pittsburgh PA.

19. Greenwood, D., and Calisti, M.: Engineering Web Service - Agent Integration, IEEE Systems, Cybernetics and Man Conference, 10-13 October, 2004, The Hague, The Netherlands.
20. Louis, V. and Martinez, T.: An Operational Model for the FIPA-ACL Semantics, Proceedings of the AAMAS'05 Workshop on Agent-Communication (AC'2005), Utrecht, The Netherlands. (2005)
21. Nguyen, T. X. and Kowalczyk, R.: WS2JADE: Integrating Web Service with Jade Agents, Proceedings of the AAMAS'05 Workshop on Service-Oriented Computing and Agent-Based Engineering (SOCABE'2005), Utrecht, The Netherlands (2005)
22. Varga, L. Zs., Hajnal, A.: Engineering Web Service Invocations from Agent Systems, Proceedings of the 3rd International Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2003, June 16-18, Prague, Czech Republic (2003) 626-635