Communication and interaction in a multi-agent system devised for transport brokering

Lucian Luncean
Romanian-German University of Sibiu
Sibiu, Romania
lucian.luncean@gmail.com

Alex Becheru
University of Craiova
Craiova, Romania
becheru@gmail.com

Abstract

In multi-agent systems communication and interaction between individual agents are fundamental characteristics. In order to fulfil those characteristics, agents have to exchange messages that have a predefined and agreed format and semantics. Our chosen format for messages is FIPA-ACL, a language widely accepted for agent platforms. In our previous work we proposed a multi-agent system for transport brokering. This paper complements our previous work by defining the process of communication and interaction between agents. An ontology is introduced and evaluated through usage scenarios, with the purpose of defining the messages' meanings.

1 Introduction

This paper describes an ongoing effort in developing a multi-agent system for transport brokering. Our proposed logistics service with brokering functionality is implemented through agent-based negotiation. When a request arrives, a virtual supply chain emerges from the system through automated or semi-automated negotiation processes between software agents. A software agent is the fundamental actor in our domain. Across this paper we shall be using the term *agent* instead of *software agent*.

The main purpose of this paper is to define and present the means of communication between agents in our first proposed system [14] and improved in [13]. The messages exchanged in our multi-agent system (MAS) shall be represented as semantic content. Agents exchange information in order to achieve their goals. Therefore agents must speak the same language to understand each other, but they also need a common representation of concepts. Thus an ontology should be part of every agent's knowledge base to describe concepts and relations among them.

Another aim of the paper is to rethink and improve the information flow within the system, introduced in [15]. We will focus our discussion on two scenarios: the information flow starting with the arrival of a transport request for a cargo and the information flow upon picking up a request for adding a transport vehicle to the system.

In order for agents to be able to converse, we needed to use a certain Agent Communication Language (ACL). The most popular ACLs are: FIPA-ACL that was proposed by the Foundation for Intelligent Physical Agents (FIPA) [17] and Knowledge Query and Manipulation Language (KQML) [19].

In this paper we chose to use FIPA-ACL, as it has a high degree of acceptance in the agent programming community and moreover in open systems. Also FIPA[1] maximises the interoperability across agent-based applications, services and equipment as it has been implemented in many projects.

An Agent Communication Language is a language with a precisely defined syntax, semantics and pragmatics that is the basis of communication between independently designed and developed agents. ACL standard allows encoding/decoding of information exchange by the agents.

A basic FIPA-ACL message is composed of several parameters including a *performative* (i.e. type of the communication act of the message), the sender, the receiver and the reply-to (i.e. participants in communication), the content (i.e. the content of the message), the used ontology (i.e. description of content) and others. Among the parameters listed above only *performative* is mandatory. A conversation between agents is defined as a sequence of messages exchanged by them. Control of conversation is done through a protocol parameter that defines the interaction protocol in which the ACL message is generated. Certainly

In: A. Bădică, M. Colhon (eds.): Proceedings of the 2015 Balkan Conference on Informatics: Advances in ICT

[1] http://www.fipa.org

one of the most complex tasks in defining the conversation between agents, is to study all the possible sequences of messages exchanges that can be performed during the conversation. A FIPA-ACL message example is illustrated in Listing. 1.

```
1 [<performative> sender: <sender>
2 receiver: <receiver> content: <content>..]
```

Listing 1: FIPA-ACL message structure

More information about the structure and the language content of FIPA-ACL messages can be found in [5] and [8].

ACLs rely on speech act theory which defines a set of performatives called Communicative Acts. In [6] you can find more details about FIPA Communicative Act Library (CAL).

To define the semantics of the FIPA ACL we need an formal language called Semantic Language (SL). On-line information about FIPA SL is available in [7].

The paper is organised as follows. The previous work is presented in Section 2. Section 2.1 is dedicated to the description of the system architecture. Two real use case scenarios are given in Section 2.2 together with the agent design and agent collaboration. Information flow in the system can be found in Section 3. Also in the previous section we define the message structure and message content ontology. Section 4 presents some relevant related work and Section 5 is dedicated to conclusions and future work.

2 Background

In recent years the development of Internet applications has known a considerable growth both in quality and quantity. Many entrepreneurs have identified an opportunity to increase their business. One of these opportunities is the constant struggle to automate operations. In [15] we proposed an agent-based system for brokering of logistics services. We consider a broker position in a transport company holding a Website where requests and offers are posted. Requests are posted by the cargo owners and offers are posted by the providers of freight. Both the transport provider and the cargo owner are continuously seeking transport opportunities. The role of the freight transportation broker is to match requests with offers, such that goods are transported in optimal conditions. The broker has to take in consideration the time constraints and vehicle capabilities. An important issue is the minimisation of the movement without cargo along the routes chosen between the points of charge and discharge.

Thus the broker has to determine the optimal routes for the transport vehicles in order to eliminate as much as possible the movement of vehicle without cargo, thus reducing transport costs (i.e. reducing logistics costs).

In [14] we proposed four ontologies and we introduced only three ontologies such as: *Transport Request Ontology*, *Transport Resource Ontology* and *Freight Ontology*. After a detailed analysis of the three ontologies we decide to improve them in [13]. In the same paper we evaluated all three ontologies through a simple scenario and another complex scenario. In this paper we shall introduce the forth ontology *Messages Ontology*.

2.1 System Architecture

Through the proposed system we plan to build a business model for logistics brokering in the transport domain. In the context of our work we identified two types of customers, on one hand cargo owners (named *Cargo owners* in our system) and on the other hand providers of transport (named *Transport providers*). These customers are represented in our system by agents such as *aCAgent*, respectively *aFTPAgent*. Other agents that contributing to the functioning of the system are *aFBAgent* (Freight Broker Agent) that represents the transport brokering service and *aFBRAgent* (Freight Broker Registry Agent) that manages the requests of customers and transport providers. Thus we identified four types of agents that represent both external users of the system, as well as internal system components with specific capabilities.

We distinguished four major parts of the system: (1) the *Request Side* for agents and activities representing customers, (2) the *Freight Broker Side* where the freight broker's agent act, (3) the *Freight Transport Provider Side* where activities related to the carrier take place and (4) the *Freight Broker Registry side* where *Customer Data* and *Freight Transport Provider Data* are stored.

The system diagram illustrating the types of agents and their acquaintance relations are presented in Figure. 1. In [15] you can find more details about description and comportment of all agents mentioned above. The diagram is of UML (Unified Modelling Language) type, while agent types are represented using *agent* stereotype. Agent acquaintance relations are represented as UML associations [9].

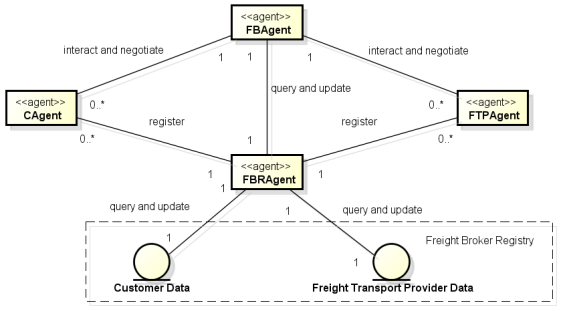

Figure 1: System Diagram

2.2 Usage Scenario

Following the analysis made in [15] we identified two main scenarios. The first scenario refers the possibility

of allowing customers (i.e. cargo owners) to make transport requests. The second scenario should allow transport providers (i.e. transport providers) to add their resource (i.e. vehicle) to a freight broker.

The two proposed usage scenarios are very similar with those in [4]. In the first case a set of requirements (i.e constrains) is matched against a description of a resource, while in the second case a description of the resource is matched against a set of requirements. Let us assume that customers, both the cargo owners and the transport providers are already registered and logged in our system (i.e. Freight Broker's Website).

First, let us to assume that one of the customers formulates a request to the transport company to solicit transport of *flat doors* between addresses *A* and *B*. Also the customer mentions in the request form that the weight of the flat doors is *3000 kg*.

Secondly, let suppose that a transport provider (also named customer from the point of view of the broker) wants to make available its transport vehicle on the Freight Broker's Website. The transport resource must be a vehicle with some of the following capabilities: type of vehicle (regarding the number of axes), vehicle dimensions, dedicated vehicle, etc.

2.3 Agent design

According to [2] agents are fundamentally a form of distributed code process and thus comply with the classic notion of a distributed computing model comprising two parts: components and connectors. Components are consumers, producers and mediators of communication messages exchanges via connectors. FIPA-ACT theory states that messages represent actions or communicative acts (i.e. speech acts or performative). The agents must be designed to allow fluent and flexible natural language communication in a manner similar to human negotiators.

2.4 Agent collaboration

Collaboration among different agents operating in the system implies communication between them, which is a fundamental characteristic of MAS. During our research we discovered challenging problems that arise when agents try to communicate and collaborate with each other in order to reach and agreement (i.e contract). In our case an example of such collaborations is the price negotiation, transport time and quality. According to [20] the following question emerge in the design phase of such systems:

- How should messages be generated, transmitted and represented?
- How can the content of messages be standardised?
- What principles (e.g. concepts, mechanisms and patterns) can be used?
- What heuristics and guidelines tell us when to apply what principle?

In [18] agent collaboration was defined as the exchange of data among information-processing agents, regardless of whether the exchange is productive or not.

Thus we designed communication diagrams for the two usage scenarios that are presented in Figures. 2 and 3. During the design we identified some similar features of ACL messages such as intentions (e.g. request, agree, forward, inform, query, search, response), attendees (*aCAgent*, *aFBAgent*, *aFTPAgent* and *aFBRAgent*), a content (i.e. the certain information that is exchanged), content description and conversation control (e.g. communication protocol).

Let us now focus our attention on the flow of information that is necessary to facilitate the two proposed scenarios mentioned before.

3 Information flow in the system

The aim of this section is to discuss the way in which the information flow and data transformations involved in it are implemented. In our system information exchange takes place separately between agent that represent the *Freight Broker* (FBAgent) and each agent that represent the *Customer* (CAgent), the *Freight Broker* (FBAgent), the *Freight Transportation Provider* (FTPAgent) and the *Freight Broker Registry* (FBRAgent).

The two main interaction scenarios are: (i) the cargo owners, represented by their *CAgent*s make a transport request and pay for provided service through the broker and (ii) the transport providers, represented by the *FTPAgent*s want to earn capital by lending their resources to the broker.

We assume that in both scenarios the customers, the cargo owners and the transport providers are logged into the system as in [12]. The interaction between an agent in the system and an external agent is not the aim of our current paper, this subject is presented in [11].

Figure. 2 represents the process taking place upon the arrival of a request from a cargo owner on the Freight Broker's Website.

Listing. 2 presents in an algorithmic detailed fashion the above mentioned figure. The first line of the algorithm represents communication (i.e. interaction) between a human user and its respective agent in our system, *aCAgent*. Next the *aCAgent* forwards the request to *aFBAgent* using the *messageForwardRequest* class from our ontology. On the third line, the *aFBAgent* sends a query message to the *aFBRAgent*. The query is an interrogation of the database for retrieving all matching vehicles, based on the constraints defined by the cargo owner. If matching vehicles could be found, the list of vehicles is sent by the *aFBRAgent* to *aFBAgent* (lines 4-8). The result of the negotiation process, either success of failure, is sent to the collaborating agents (lines 9-19). Else if not matching vehicle could be found

the *aFBAgent*, *aCAgent* and *CargoOwner* are notified with mismatch messages.

Figure. 3 presents a sequence of actions started by the transport provider, represented by the *aFTPAgent*, which desires to add a transport vehicle on the Freight Broker's Website. The detailed version of the above mentioned figure is presented in Listing. 3 in an algorithmic fashion.

3.1 Message Structure

Here we present our proposed message structure based on FIPA-ACL. As an example of a request communicative act, consider agent *aCAgent* (that represents the cargo owner) requesting agent *aFBAgent* (that represents the Freight Broker) to process an order from "TRANS LTD" consisting of *10 pallets* with *flat doors* to be transported from *Sibiu* to *Bucureşti*. This offer is only available for the next *24 hours*. The request was made in *2015.05.01*, date of charge was on *2015.05.03* and date of discharge was on *2015.05.04*. In FIPA notation, we express the request as in Listing. 4.

```
(request
  :sender aCAgent
  :receiver aFBAgent
  :reply-by: hasTTL=24:00:00
  :content (deliver hasOwnerName = "TRANS LTD"
    hasDateOfRequest="2015.05.01"
    hasNumberOfBoxOfFreight="10"
    hasNameOfFreight="flat doors"
    hasStartMomentOfCharge="2015.05.03"
    hasStartMomentOfDischarge="2015.05.04"
    PointOfDispatch="Sibiu"
    PointOfDestination="Bucuresti")
  :reply-with request-01
  :language sl
  :ontology lob
  :conversation-id request123)
```

Listing 4: Example of a FIPA-ACL request message

We mention that the freight is transported on pallets, a pallet can transport a maximum of 5 flat doors. To express the quantity we use the class *BoxOfFreight* that represents the wrapping of the freight. To capture the number of boxes we use the property *hasNumberOfBoxOfFreight*.

An example where an agent accepts the request is presented in Listing. 5.

```
(agree
  :sender aFBAgent
  :receiver aCAgent
  :reply-to request-01
  :language sl
  :ontology lob.message
  :conversation-id request124
)
```

Listing 5: Example of a FIPA-ACL agree message

3.2 Message Content Ontology

Ontologies play a significant role not only in the agent communication, but also in knowledge capturing, sharing and reuse. One of the main reasons why ontologies are being used is semantic interoperability. Thereby, as we specified in [13] we have decided that the request, the freight, the transport resource and the messages exchanged between agents used by the system must be semantically represented.

When agents want to communicate, an appropriate message content ontology is selected. This ontology is used by agents to make conversation about issues related to specific domain.

FIPA communication stack can be separated into seven sub-layers such as: transport, encoding, messaging, ontology, content expression, communicative act and interaction protocol.

Considering that although FIPA allows for the use of ontologies when expressing messages content, it does not specify any particular representation for ontologies or provide any domain-specific ontologies. In this section we propose to provide a messages ontology for our multi-agent system. This ontology was generated considering initial architecture presented above. In fact this proposed messages ontology is a vocabulary that help us to express the content of messages exchanged between agents in the system. For the development of the ontology we followed an engineering methodology as in [23].

First, we need to define some specific terms of scientific literature used from now such as: ontology, message, content, conversation and protocol.

An *ontology* is used to represent knowledge that is shared between different entities. It provides terms and vocabulary used to represent knowledge so that both sender and receiver can understand.

According to [17] a *message* is an individual unit of communication between two or more agents. A message corresponds to a communicative act, in the sense that a message encodes the communicative act for reliable transmission between agents. Note that communicative acts can be recursively composed, so while the outermost act is directly encoded by the message, taken as a whole a given message may represent multiple individual communicative acts.

A *content of the message* is a part of a communicative act and denotes the content of the message. In [5] the meaning of the content of any ACL message is intended to be interpreted by the receiver of the message. This is particularly relevant for instance when referring to referential expressions, whose interpretation might be different for the sender and the receiver.

A *protocol* is a special set of rules, used by end points in a connection, in order to establish and maintain communication. Protocols specify interactions between the commu-

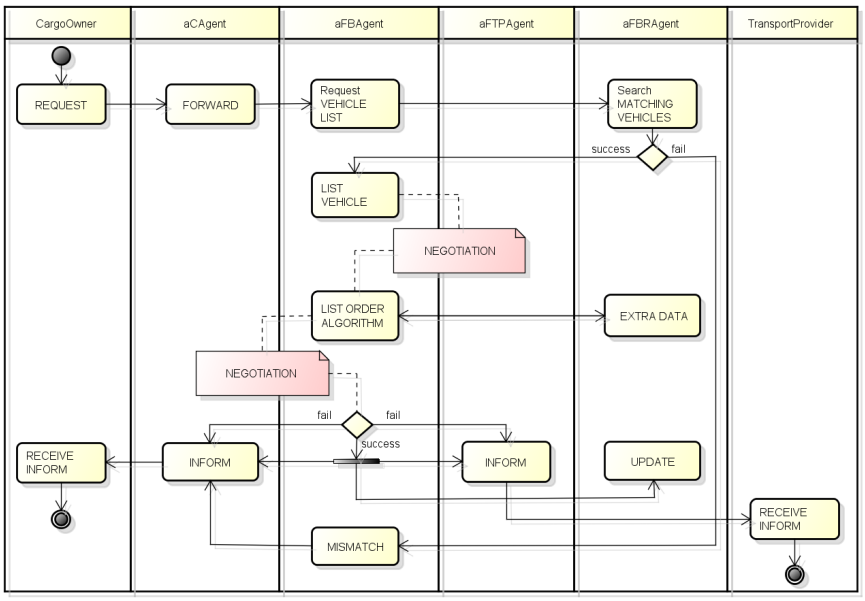

Figure 2: Communication diagram for a request made by the cargo owner in the broker's Website

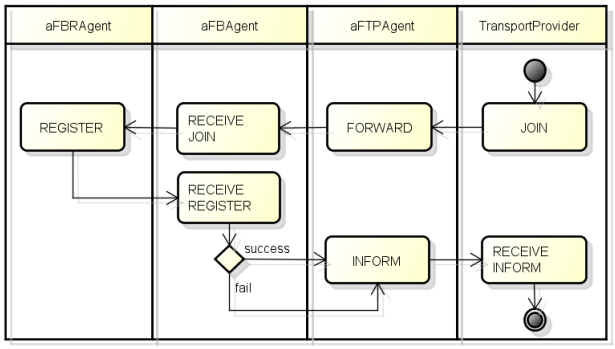

Figure 3: Sequence diagram for adding a freight provider in the broker's system

nicating entities. Further information about protocols and agents communication are defined by Amit K. Chopra's [3].

In our scenario, we use the ontology to define the types of message exchange between agents, this completes our already developed ontologies. All the ontologies developed through this research project are available online[2].

The top level class of our ontology is the *message*, see Figure. 4, which in turn is also a *Thing*. We then dived the message in other 4 subclasses: *messageForward*, *messageOfResult*, *messageToDb* and *messageFromDb*. The separation was based on the the general purpose of the messages in our system.

The *messageForward* class describes messages that are being forwarded, the message content does not change, only the sender and receiver change. Three other sub classes of *messageForward* are defined here: *messageForwardRequestVehicle* related to the first scenario, *messageForwardRequest* related to both scenarios defined in section 2.2, and *messageForwardAddVehicle* related to the second scenario. E.g. *AddVehicleNotificationOfFailure* is a subclass of *messageForwardRequestAddVehicle* therefore this subclass is related to the second scenario. Further we see that this subclass is equivalent to *messageOfResultAddVehicleFailure* which is a message sent by *aFBAgent* to *aFTPAgent* when a vehicle could not be added to *aFBRAgent*, but our current subclass has as sender *aFTPAgent* and receiver *TransportProvider*. Thus *AddVehicleNotificationOfFailure* is a forward message.

The class *messageOfResult* defines messages sent by *aFBAgent* after the process of negotiation (first scenario) or after *aFBRAgent* has received the request to add a vehicle in the system data base (second scenario). Otherwise said these messages conclude the processes in both scenarios. These type of messages are then forwarded through the *messageForward* messages.

All the messages received by *aFBRAgent* are defined by the *messageToDb* class. Also this class has two other subclasses: *messageToDbQuery* which contains only queries (i.e. interrogations), and *messageToDbUpdate* comprising of messages that update the system data base. For example, after the negotiation is finished with *aCAgent* in the first scenario and a vehicle (V) is chosen *aFBAgent* sends an update message to *aFBRAgent* to set vehicle V as occupied for the time of the transport.

The class *messageFromDb* describes messages that contain data structures needed in the process of negotiation (first scenario). These data structures are either obtained from the system data base or from the negotiation process

[2]intelligentdistributedsystems.github.io/FreightOntology/

```
1 CargoOwner->aCAgent
2 aCAgent->aFBAgent: messageForwardRequest
3   aFBAgent->aFBRAgent: messageToDbQueryRequest
4     IF success in finding matching vehicles
5       aFBRAgent->aFBAgent: messageFromDbDataStructureLMV
6       aFBAgent->aFBRAgent: messageToDbQueryExtraData
7       aFBRAgent->aFBAgent: messageFromDbDataStructureLMVP
8       aFBAgent negotiation process with aCAgent:
9       IF negotiation success:
10        aFBAgent->aCAgent: messageOfResultRequestVehicleSuccess
11        aCAgent->CargoOwner: RequestVehicleNotificationOfSuccess
12        aFBAgent->aFTPAgent: messageOfResultRequestVehicleSuccess
13        aFTPAgent->TransportProvider: RequestVehicleNotificationOfSuccess
14        aFBAgent->aFBRAgent: messageToUpdateInfoVehicle
15      ELSE negotiation failed:
16        aFBAgent->aCAgent: messageOfResultRequestVehicleFailed
17        aCAgent->CargoOwner: RequestVehicleNotificationOfFailure
18        aFBAgent->aFTPAgent: messageOfResultRequestVehicleFailed
19        aFTPAgent->TransportProvider: RequestVehicleNotificationOfFailure
20    ELSE failure in finding match vehicles
21      aFBRAgent->aFBAgent: messageOfResultRequestVehicleMismatch
22      aFBAgent->aCAgent: RequestVehicleNotificationOfMismatch
23      aCAgent->CargoOwner: RequestVehicleNotificationOfFailure
```

Listing 2: First scenario of message exchange between agents

```
1 TransportProvider->aFTPAgent
2 aFTPAgent->aFBAgent: messageForwardRequest
3 FBAgent->aFBRAgent: messageToDbUpdateAddVehicle
4 aFBRAgent ->aFBAgent:
      messageFromDbRegistrationResponse
5 IF registration success:
6   aFBAgent->aFTPAgent:
      messageOfResultAddVehicleSuccess
7   aFTPAgent->TransportProvider:
      AddVehicleNotificationOfSuccess
8 ELSE registration failure:
9   aFBAgent->aFTPAgent:
      messageOfResultAddVehicleFailed
10  aFTPAgent->TransportProvider:
      AddVehicleNotificationOfFailure
```

Listing 3: Second scenario of message exchange between agents

with *aFTPAgent*, see *messageFromDbDataStructure* subclass. For the second scenario *messageFromDb* defines messages sent by *aFBRAgent* after the process of vehicle addition was tried. Let us suppose that a vehicle was not registered because it is already present in the system data base, then *aFBRAgent* should send a message to *aFBAgent* containing an error line *Update failure, (object already present...)*.

```
1 Class:
2   <lob#messageOfResultAddVehicleFailure>
3 Annotations:
4   rdfs:comment
5   "Message format:
6   Request request_id failed"
7 EquivalentTo:
8   <lob#AddVehicleNotificationOfFailure>
9 SubClassOf:
10  <lob#messageOfResultAddVehicle>
11 DisjointWith:
12  <lob#messageOfResultAddVehicleSuccess>
```

Listing 6: Example of a primitive class in the *Messages Exchange Ontology*

4 Related Work

The authors of [16] proposed ontology-services to facilitate agents' interoperability. By defining this ontology they provide a vocabulary to be used in the communication among different agents.

We found a similar approach to ours in [22], the authors present an overview of issues concerning allocation of processes in the grid through negotiations.

Paper [10] discusses the implementation of information flows and data transformations. As well as in our paper information about products are ontologically represented.

An ontology whose aim is to share information between sending and receiving agents in Multi-Agent System is presented in [1]. The challenge targeted by the authors was the interactions among agents as agent-agent and agent-user communication can be very complex.

Van Aart et.al.[21] constructed a theoretical framework for message-based communication between agents. As in the current paper, the meaning and intention of the messages is specified through a message content ontology.

5 Conclusions and Future Work

In this paper we have introduced the means of communication between the agents in our MAS. Also we detailed the way in which agents interact with each other.

One of the most important issues in the area of agent communication is the understanding of messages content

Figure 4: Infered Ontology Diagram

meaning. We proposed and introduced a message content ontology to capture the meaning of messages

As a future work, we intend on the short term to develop our messages ontology with messages exchanged during the negotiation process. Also we shall develop/adapt an algorithm to sort the list of transport resources (*lmv*) depending on collaboration history of customers (transport providers) and their reputations. On the long term we propose to develop an application-based system using several agents and collect real data for experiments.

5.0.1 Acknowledgements

This work was supported by the strategic grant POSDRU/159/1.5/2/133255. Project ID 133225(2014), cofinanced by the European Social Fund within the Sectorial Operational Program Human Resources Development 2007-2013.

References

[1] M. Arora and M. S. Devi. Ontology based agent communication in resource allocation and monitoring. *IJCSI*, page 27, 2010.

[2] F. L. Bellifemine, G. Caire, and D. Greenwood. *Developing Multi-Agent Systems with JADE (Wiley Series in Agent Technology)*. John Wiley & Sons, 2007.

[3] A. K. Chopra and M. P. Singh. *Agent communication. In G. Weiss, editor, Multiagent Systems*, chapter 3, 101-142, MIT Press, 2013.

[4] M. Drozdowicz, K. Wasielewska, M. Ganzha, M. Paprzycki, N. Attaoui, I. Lirkov, R. Olejnik, D. Petcu, and C. Bădică. Ontology for contract negotiations in an agent-based grid resource management system. In P. Iványi and B. Topping, editors, *Trends in Parallel, Distributed, Grid and Cloud Computing for Engineering*, Computational & Technology Resources, chapter 15, pages 335–354. Saxe-Coburg Publications, 2011.

[5] FIPA. Fipa acl message structure specification. http://www.fipa.org/specs/fipa00061/SC00061G.html#_Toc26669708.

[6] FIPA. Fipa communicative act library specification. http://www.fipa.org/specs/fipa00037/SC00037J.html#_Toc26729689.

[7] FIPA. Fipa sl content language specification. [online].

[8] N. Fornara, D. Okouya, and M. Colombetti. Using owl 2 dl for expressing acl content and semantics. In *Proceedings of the 9th European Conference on Multi-Agent Systems*, EUMAS'11, pages 97–113. Springer-Verlag, 2012.

[9] M. Fowler. *UML Distilled: A Brief Guide to the Standard Object Modeling Language (3rd Edition)*. Addison-Wesley Professional, 2003.

[10] M. Gawinecki, M. Ganzha, P. Kobzdej, M. Paprzycki, C. Bădică, M. Scafes, and G.-G. Popa. Managing information and time flow in an agent-based e-commerce system. In *ISPDC*, pages 352–359. IEEE Computer Society, 2006.

[11] M. Gawinecki, M. Gordon, P. Kaczmarek, and M. Paprzycki. The problem of agent-client communication on the internet. *Scalable Computing: Practice and Experience*, 6(1):111–123, 2005.

[12] S. Ilie, C. Bădică, A. Bădică, L. Sandu, R. Sbora, M. Ganzha, and M. Paprzycki. Information flow in a distributed agent-based online auction system. In D. D. Burdescu, R. Akerkar, and C. Bădică, editors, *Proceeding of 2nd International Conference on Web Intelligence, Mining and Semantics, WIMS'12*, page 42. ACM, 2012.

[13] L. Luncean, A. Becheru, and C. Bădică. Initial evaluation of an ontology for transport brokering. Proceeding of 19th IEEE International Conference on Computer Supported Cooperative Work in Design – CSCWD 2015, 2015.

[14] L. Luncean and C. Bădică. Semantic modeling of information for freight transportation broker. In *Proceeding of 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing – SYNASC 2014*, pages 527–534. IEEE Computer Society, 2014.

[15] L. Luncean, C. Bădică, and A. Bădică. Agent-based system for brokering of logistics services - initial report. In *Intelligent Information and Database Systems - 6th Asian Conference, ACIIDS 2014, Bangkok, Thailand, April 7-9, 2014, Proceedings, Part II*, pages 485–494. Springer, 2014.

[16] A. Malucelli and E. da Costa Oliveira. Ontology-services to facilitate agents' interoperability. In *Intelligent Agents and Multi-Agent Systems*, volume 2891 of *Lecture Notes in Computer Science*, pages 170–181. Springer Berlin Heidelberg, 2003.

[17] S. Poslad. Specifying protocols for multi-agent systems interaction. *ACM Trans. Auton. Adapt. Syst.*, 2(4), Nov. 2007.

[18] N. S. Talukdar. Collaboration rules for autonomous software agents. *Decision Support Systems*, 24(3-4):269–278, 1999.

[19] F. Tim, R. Fritzson, D. McKay, and R. McEntire. Kqml as an agent communication language. In *Proceedings of the third international conference on Information and knowledge management, CIKM*, pages 456–463, 1994.

[20] C. van Aart, G. Caire, F. Bergenti, and R. Pels. Creating and using ontologies in agent communication. In *Proceedings of the Workshop on Ontologies in Agent Systems, 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS'02*, 2002.

[21] van Aart, Chris and Wielinga, Bob and Schreiber, Guus Message Content Ontologies Ontologies for Agents: Theory and Experiences, Whitestein Series in Software Agent Technologies, Tamma, Valentina and Cranefield, Stephen and Finin, TimothyW. and Willmott, Steven Birkhuser Basel, pages 169-200.

[22] K. Wasielewska, M. Ganzha, M. Paprzycki, M. Drozdowicz, D. Petcu, C. Bădică, N. Attaoui, I. Lirkov, and R. Olejnik. Negotiations in an agent-based grid resource brokering system. In P. Iványi and B. Topping, editors, *Trends in Parallel, Distributed, Grid and Cloud Computing for Engineering*, Computational & Technology Resources, pages 355–374. Saxe-Coburg Publications, Stirlingshire, UK, 2011.

[23] Natalya F. Noy and Deborah L. McGuinness. Ontology Development 101: A Guide to Creating Your First Ontology. Stanford University, Stanford, CA, 94305. http://protege.stanford.edu/publications/ontology_development/ontology101.html, 2014