UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.dit.unitn.it

Semi-Heuristic Negotiation Protocol for Agent-based Mobile Service Application

Sameh Abdel-Naby, Paolo Giorgini and Stefano Fante

February 2007

Technical Report # DIT-07-004

Semi-Heuristic Negotiation Protocol for Agent-based Mobile Service Application

Sameh Abdel-Naby, Paolo Giorgini and Stefano Fante
University of Trento, Italy.
{sameh, paolo.giorgini, stefano.fante}@dit.unitn.it

Abstract

In this paper we present a new negotiation protocol that assists multi-agent systems to differently approach the achievement of complex tasks. In particular, we narrow our research scope to focus on the situation where a multi-agent system is serving lightweight devices through advanced communication methods (e.g., Bluetooth). Like many other schemes, our model provides agents with a monetary system and a mechanism for feedback calculation. We aim at accelerating agent's interactions while resolving end-user composite tasks. Once its efficiency is proven, our protocol can be integrated in a scenario where multipart mobile-based services are offered to holders of lightweight devices.

1. Introduction

Lightweight devices such as cellular phones and PDAs are increasingly proving their necessity and reliability. The new era of telecommunication technologies is putting normal lightweight devices in a situation to provide, virtually but not physically, portable offices. Nowadays, people can go anywhere carrying pocket devices that allow them to check their emails, exchange faxes, surf the internet, edit documents and, do shopping. Services are provided through quite simple, user-friendly and well-developed interfaces, and almost costless in regard to the value of services users are getting. Standard mobile services that never existed before are becoming a must (e.g., SMS and MMS), and advanced ones that newly existed are now highly desired (e.g., Cinemas Guides and Group Gaming).

Several efforts in literature, for example [3], tackled the scenario where a visiting professor is entering to a university and her cellular phone establishes a connection with a localized Multi-Agent System (MAS), this system synchronizes her agenda with the calendars of people interested to meet her. These people are moving within a university carrying their lightweight devices or, they have previously delegated an agent to act on their behalf. Automated system agents cooperate and negotiate available times to create a suitable agenda for everybody. Accordingly, this professor along with meeting requesters is forming a Mobile Virtual Community [7] that is location-based and MAS supported.

Another scenario (for example, [1]) that had several approaches in MAS literature is related to tourists that are making their usual sightseeing trips. Tourists turn out to be system users by simply enabling the Bluetooth functionality in their pocket devices, and using a preinstalled application that communicate with distributed MAS servers, they are able to retrieve all the useful information related to the places they are visiting. Moreover, for each single item available at a specific museum, there is a software agent that represents it and holds its information, this agent can cooperate with other agents to fulfill certain complex user desires (e.g., where this item was originally found, relevant sightseeing's opening times and how to go there?).

End-users in former scenarios are performing a set of application instructions that are accessible through their lightweight devices. These instructions lead them to create a software agent that is delegated to reflect their specific desires and characteristics. Eventually, the agent would be attached to the system and it starts wandering to fulfill user desires. Then a matchmaking process will take place; an agent that carries specific information would look for another agent that may add on extra details so a task gets completed. Sometimes, an agent may not find his completer; therefore, there are complex scenarios where cooperative agents would merge to help each other.

When software agents interact properly, an extra capability for people to cooperate is shaped. A negotiation language that is applied among distributed agents is helping them to understand each other, discuss their desires and finally achieve their objectives. Several of the negotiation protocols proposed by scholars are inspired from sociological, political and psychological studies about human negotiation in real-life situations such as Auctions, Peace agreements and Bidding theories.

We contribute to existing literature by presenting a negotiation protocol that addresses a specific situation. We focus on a MAS that delivers certain location-based service

to users of lightweight devices, relying on device communication capabilities. Although, there are some restrictions that are given by users (e.g. time to achieve) and others given by the technology used (e.g. Bluetooth data exchange rate), still the MAS architecture would ensure reliable service and increasing system usability.

The remainder of this paper is structured as follows. Next section emphasizes our research motivation. Section 3 looks at the building blocks of our negotiation protocol. Section 4 applies the proposed negotiation protocol on a case study. Section 5 highlights the related work. Section 5 demonstrates our future work and concludes the paper.

2. Motivating Scenario

In a MAS responsible of delivering a specific service content to lightweight devices, a set of uncooperative agents that are self-interested and benefits maximizers is located; given that, this set contains two different types of agents, **BUYER AGENT** (BA) and **SELLER AGENT** (SA); each holds information related to its role in the system, a BA keeps data that helps SA increases his profit, and the data SA keeps helps BA to achieve the overall objectives of the system. A matchmaking phase is passed and a BA has found the appropriate SA, both agents try to maximize their returns through a negotiation process. If agents predefined behavior is strict and intelligent "usual case in MAS" this will lead both agents to reach a situation of disagreement.

The mentioned above refers to the fact of having two successfully matched agents and no useful results are gained. The existence of autonomous agents in MAS is necessary to increase system reliability, in the same time, interactivity between all application entities is highly desired, but an unfruitful negotiation process among involved software agents is what a complete application should avoid, and this is what we focus our research on.

In figure 1, we address the motivating scenario driving our research towards the introduction of new negotiation protocol.We assume that three different users are interested in using the same service architecture in managing their desires to obtain a certain item. This service is limited to the demand and supply of a specific product among system users only (e.g., available care seats in a carpooling system or a used book in a trade environment). These users are using their lightweight devices to communicate with the service architecture and, each is adapted to use its device-based application. One of the users may be the giver of this product and the others are the requesters. We are at the point where the number of acquisition requests is greater than the number of offered items.

Each of the involved lightweight devices is configured to utilize a specific telecommunication method to access the service, a cellular phone my exchange service requests us-

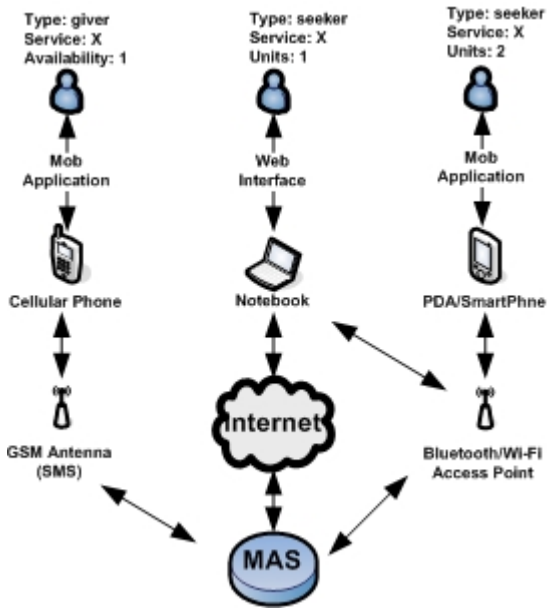

Figure 1: Different devices are using different communication methods to achieve service interaction.

ing SMSs, a notebook may access the service through a dedicated web interface or a distributed Bluetooth / Wi-Fi access points. If a user is offering a single item that more than two requesters are interested to have, also each requester may ask for more than a single unit of the same item, the managing MAS will drive these three users to a complex situation by not properly deciding upon the ownership of the offered item. In this situation, we are at the pre-agreement point where the service preferences have matched and conflict is located in the acquisition time phase.

An auction mechanism can be invoked to resolve such a complex situation, this mechanism can be restricted with a time constraints, and it can be technically adapted to ensure ultimate benefits gaining for both service supplier and demander. The invoked mechanism can be semi-heuristic by storing similar auctions results, these stored results are those who involve same system users more than a certain number of auction invocation. Following to that, the paper explains the proposed agent's semi-heuristic negotiation algorithm that helps in achieving better results in complex scenarios.

3. The Semi-Heuristic Negotiation Protocol

In this section we propose our negotiation protocol that assists agents in a mobile environment to establish proper communications, achieve better results, and learn from frequently repetitive task.

```
Seller_Agent_procedure()
1:  bestValue = 0;
2:  numLoop = 0;
3:  auctionIsOpen = true;
4:  askBAToStartAuction(bestPreOffer);
5:  while (auctionIsOpen) do
6:    waitForOffers();
7:    val = calculateBestValueOfFunction();
8:    if (numLoop == 0) then
9:       bestValue = val;
10:      numLoop++;
11:   else
12:      if (val > bestValue) then
13:         bestValue = val;
14:         requireNewOfferToBuyers(bestValue);
15:         numLoop++;
16:      else
17:         if (val <= bestValue) then
18:         quitAuction();
19:         informWinners();
20: end while
21: quitAuction();
```

Algorithm: 1 The procedures taken by the Seller Agent.

Given a set of lightweight devices that are capable of communicating a specific service request with central **Service Oriented Architecture** (SOA) via distributed access points, and given that these access points and central servers are providing mobile users with location-based service. The lightweight devices are used here as a tool to clarify users' preferences and, consequently a **Mobile-to-Server Link Agent** (MSLA) is configured. This particular agent carries specific user desires and it is capable to move from lightweight device, through the nearest access point, to end at server side. When the MSLA arrives to one of the central service servers, its carried desires are forming an autonomous software agent that reflects certain characteristics. This MSLA is a configuration file that is produced by the lightweight device service application and links together the device used with the SOA.

Eventually, a phase where system verification occurs is placed. The arrival of a new agent to the server side requires the running MAS to verify whether this agent is new and to be bootstrapped or, it already exists and it meant to update the behavior of a previously running agent.

A group of autonomous agents that are delegated by several users to achieve varied tasks in different times is formed at the server side of the architecture. Given that some of the tasks to be achieved are complex and require multi-agent coordination, thus a negotiation scenario that requires agent-to-many is formed. Still some of tasks are simple and require only agent-to-agent cooperation. In agent-to-agent situations, the negotiation protocol applied is simple and efficient; it is the same as the market demand and supply. When the supplier and the demander are matched, a mutual benefits exchange is achieved. This usually occurs because only one demander and one supplier are located within the service range of each. Unsurprisingly, in agent-to-many it is more complex.

In Algorithm 1, we show the algorithm used on the **Seller Agent** (SA) side to invoke and manage a specific auctioning situation. From `line 1` to `line 3`, both SA variables, `bestValue` and `numLoop`, are initially set to `"0"`. In `line 4`, the seller agent requests the **Buyer Agent** (BA) to start the auction by sending the value of the best offer previously obtained during the pre-offer session. From `line 5` to `line 7`, the SA waits to receive new offers from all involved BAs, and a `"val"` is created as a function to calculate the currently obtained best-offer-value. From `line 8` to `line 10`, if the algorithm had its first round and a `"val"` is gained, the `"bestvalue"` in `line 1` is now updated with the value of `"val"` and the number of loops `"numLoop"` is gradually incremented.

Otherwise, since it is not the first loop, from `line 11` down to `line 19`, the SA checks whether the `"val"` function is increasing in comparison with the previously obtained best value or not. At this point, two scenarios may occur, if `"val"` is greater, the value obtained from the concerned BA is communicated with other BAs and, they are asked to communicate new offer if applicable, then the algorithm is restarted, `line 14` and `line 15`. If the `"val"` is less or equal to the best value previously obtained, the auction is suspended and the BA currently bid the `"bestValue"` wins, `line 17` to `line 19`. Finally, the kernel of the algorithm is terminated and the auction scenario is ended, `line 20` and `21`. Later to that, we explain the Buyer Agent behavior in response to SA.

```
Buyer_Agent_procedure()
1:  BABestOffer = 0;
2:  sent = false;
3:  while(auctionIsOpen) do
4:    sent = false;
5:    bestOffer = waitForRequest(bestPreOffer);
6:    decision = decideIfAcceptOrRefuse();
7:    if (decision == accept) then
8:       while(modificationsArePossible && !sent) do
9:          newVal = reviewParameters();
10:         if (newVal>BABestOffer&&newVal>bestOffer) then
11:            BABestOffer = newVal;
12:            sendOffer(BABestOffer);
13:            sent = true;
14:         if (!sent && !modificationsArePossible) then
15:            sendOffer(BABestOffer);
16:      end while
17:   else
18:      if (decision == refuse) then
19:         quitAuction();
20: end while
21: quitAuction();
```

Algorithm: 2 The procedures taken by the Buyer Agent.

In Algorithm 2, we show the algorithm used on the **Buyer Agent** (BA) side to determine the significance of its role in the impending auction. In `line 1` and `line 2`, a variable `"BABestOffer"` that carries the BA best offer value is created

and set to "0". A variable "sent" is initially set to "false" and it changes to "true" only after a BA has communicated his offer. From line 3 to line 6, while the auction is open, BA holds its offer transfer until a communication was received from the **Seller Agent** (SA) asking for an auction participation decision. The BA puts the results from the evaluation function into the "decision" variable.

From line 7 to line 9, if the BA accepts the call for auction, a self-revision for the holding parameters is made. This revision refers to the BA insistent to obtain the auctioned item; therefore, it is made with the intention to show extra negotiation flexibility. The part from line 10 and down to line 13 refers to the comparison made by the BA to put together the newly obtained value and the existing one. If the new value obtained is greater than the previous one and, greater than the "bestOffer", the future offered value "BABestOffer" is set to be new one "newVal", and the offer is sent to the concerned SA.

Line 14 to line 16, if the self-revision made by the BA has yielded a disappointing result and the value gained is the same as the previous one, this specific BA does not send the previous value if "modificationArePossible" is "true". The BA continues to review the carried parameters until "modificationArePossible" becomes "false" or it communicates new best offer. If "modificationArePossible" stays on "false" and parameters are not sent, BA communicates same previous offer.

However, from line 17 to 19, if BA refuses the auction call, the algorithm terminates and the auction involves this specific agent ends. If the user has an inflexible behavior, the algorithm passes the first condition on if (decision == accept) but the condition of the successive while (modificationArePossible && !sent) return "false". The method "decideIfAcceptOrRefuse" return "refuse" if for instance, a BA has a lot of time before the deadline to achieve the task; therefore, it decides to refuse auction participation. Finally, the kernel of the algorithm is terminated and the auction scenario is ended, line 20 and 21.

Auctioning among agents requires high level of agent-to-user interactivity and increased level of network resources consumption; therefore, agents' intelligence appears when a repetitive scenario occurs. If system user is configuring the mobile-based application to repeat the same service request on daily or weekly basis (e.g., common in mobile news exchange service or carpooling), the created demanding agent would participate in system auctions only if needed. Once an agreement is settled between a specific supplier and a demander at a certain price, the next time this demander agent will first look-up the very exact supplier agent, which has potential agreement than others in early agreement. This is due to learning agent behavior that maintains an array that saves last successful agreement details.

4. Case Study

ToothAgent [2] is an example of a Multi-agent system (MAS) that allows students within a university main sections (e.g. library, main hall or classroom) to use any of their lightweight devices to exchange used books requests and offers. Once an agreement is reached, the system helps students to agree on meeting places and times. This is all done through normal Bluetooth communications that take place between both, user and distributed servers. Agent-oriented programming techniques are used to form a Mobile Virtual Community [7]. This helps the system, including its *Intelligent*, *Mobile* and *Autonomous* agents to accomplish the matchmaking and exchange of requests and, support the price negotiation phase.

An example of a MAS implementation that provides its users with a possibility to utilize their lightweight devices to offer/look-up shared car rides is Andiamo [11]. To understand the Rideshare service or "Carpooling" as stated in literature; it is a method to reduce the use of cars in a specific town or area, this usually takes place by having a car owner who uses his/her car to move from a place to another, and another person who is interested to go somewhere along the car owner's way to destination, and at the same time the ride seeker is willing to share the ride cost with the car owner. Based on the use of location and available car seats, Andiamo allows a substantial number of people to share car rides, using their cellular phones. This system would, among other advantages, rationalize energy consumption, save money, and decrease traffic jams and human stress, and eventually make a significant improvement in human life.

In figure 2, we explain the mechanism of the proposed algorithm using a mobile-based rideshare service architecture. In our example, we primarily assume that a car ride giver (Seller Agent) - SA Started - has communicated and submitted his offer details to the Multi-Agent System, this MAS is managing the service requests exchange among connected lightweight devices, and we assume that only one available car seat is given by the car owner, and a matching phase has resulted three or more interested ride seekers that are all willing to share the given ride cost with the car owner.

As shown in figure 2, from this point on - SA Ready, the agent acting on behalf of the ride-giver is responsible of resolving this complex situation by: 1) according to the parameters given by agents of the ride seekers, a calculation process is performed and each agent is assigned a value, 2) a comparison between the yielded values is made and sent to ride seekers, then a call for auction is made, 3) a request to all interested agents to send new offers is communicated.

Each agent acting on behalf of a ride seeker is free to choose whether to participate in the auction or not. Hence, an agent has decided to skip an auction, the negotiation process involving both parties is ended - SA Ending - BA Ending. This may happen because another ride giver was located by this seeker agent, the user himself has changed his mind or restricted behavior was applied to this agent. But once a seeker agent has decided to go through the auction - BA Ready, a self-revision process for the carried parameters is made - BA Computation, and then a value calculation is reached and compared by the one sent by the giver at the algorithm invocation phase. The seeker agent tries to modify some of the carried parameters in accordance with users' interests, so a new compromise is reached.

Results reached after parameters modification - BA Ready to Respond will indicate if a new auction winning potential is created or not. Then, new or the same old parameters are communicated back with the ride giver agent (SA), depending on the

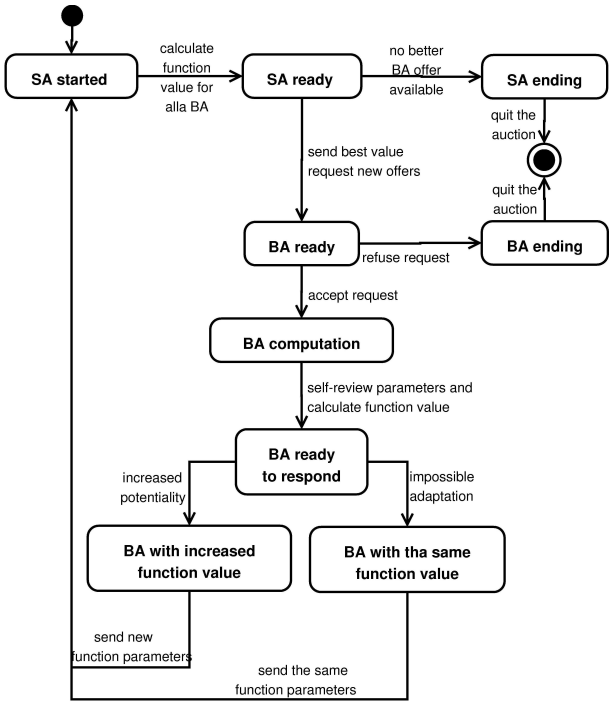

Figure 2: Auction call and termination in a Rideshare MAS service architecture.

potentiality found earlier to this step - `BA with Increased Function Value` or `BA with Same Function Value`. The ride giver agent re-evaluates the received parameters, in addition to the parameters newly received from seeking agents that were not involved from the beginning, if any. Then, announcement is made for the only available car seat winner. Accordingly, the auction terminates and the entire negotiation process ends. The mechanism can be repetitive only if no agreement situation was found and the time to achieve the actual car ride is yet not approaching.

The operating MAS will take the responsibility of storing the auction initiator and the auction winner. The invocation of an auction and the exchange of messages among involved agents consume time and network resources. To rapidly resolve future complex situations, if an auction result is repeated certain times, the system would automatically count the winning ride seeker agent as future auctions winner, or one of the winners if more available seats were given. This happens only if exact auction scenario (e.g. same ride preferences, same participants) is about to be invoked.

5. Related Work

A significant part of the research conducted in *Distributed Artificial Intelligence* (DAI) focuses on the coordination among objects located in distributed environments. Thus far, a particular research topic under DAI, which is *Distributed Problem Solving* (DPS), has proposed several negotiation strategies that mostly aimed at the construction of what we call **Distributed Objects Communication Language** (DOCL). These negotiation languages are, among other advantages, helping to form a cooperative environment that successfully achieve multipart tasks and deliver refined services, and it is also allowing objects to heuristically improve their negotiation behavior.

In Multi-Agent Systems (MAS) scholars have tried to address the problem of negotiation by reflecting real human negotiation in a computing background [4, 5, 6]. These studies were mainly carried out because of two abstract reasons, 1) a software that entirely acts on behalf of its holder is the ultimate goal that many researchers' visions are chasing, and that is exactly the main characteristic of a Software Agent. And 2) for numerous applications to automatically interact and achieve complex tasks is up till now another vision to chase, and this has raised the need for a negotiation language to be used among applications to facilitate their interactions. These two prior reasons are forming together the need to design the *negotiating agents* that are able to meaningfully interact and talkatively negotiate to maximize their user's benefits.

In their book [9], J. S. Rosenschein and G. Zlotkin are doing what they call **Social Engineering**; they have dedicated part of their research on how designers of software agents would react to the development process of Multi-agent systems and, the use of certain design steps regarding the accomplishment of suitable negotiation protocol, which in return will lead to appropriate interactions among several MASs. They emphasized the urgent need to look at agents as the new era of human "*surrogates*", and this is because of the nowadays speed taken to approach full system and machines delegation.

In Game Theory [12], a clear approach was taken to study the rational behavior among self-interested agents. Different software designers are working on the development of several software agents; these development processes will only produce an agent that is reflecting designer's personal behavior. Although the agents produced are self-interested and autonomous, they are going to interact with different agents that are designed differently and contain different level of autonomous performance and complexity. An agent that is rationally driven within a system entities will make goals and procedures to achieve them clear for all system actors, but it will apply an atmosphere of firmness and inflexibility in formed interactions. This earlier discussion has raised the confrontation of two important design aspects, would it be more appropriate to design an agent that is deterministic or an agent that is flexible?

When Distributed Artificial Intelligence (DAI) started to have its own structure as independent research area, Reid G. Smith has contributed significantly to this structure formation by having his PhD thesis defense, in 1978, discussing a new perspective in achieving proper negotiation and interactivity among multiple automated network-nodes. Later to that, an important contribution was added to literature regarding the same topic, which is **Contract Net** [10]. When applied to multi-agent systems, the **Contract Net** protocol assumes that each node in the network is an agent that is seeking another completer-agent that may, together, form a coalition to resolve a complex task. This coalition can yield some results that can not be achieved if each agent is operating separately.

When the exact rare resources are to be used by several agents, an **Auction** [8] is formed between these agents so that system re-

sources are utilized at the highest possible value, and certain negotiation language that perfectly applies in this situation is used. Due to issues related to equality, ordering and planning, Auctions have gained a wide range of applications in multi-agent systems. Four major auction types that are widely recognized; 1) English, 2) Dutch 3) First-price Sealed-bid, 4) Vickrey's Mechanism or Second-price Sealed-bid. These auctions are reflecting real human behavior in different auction styles and similarly apply it to agents.

6. Conclusions and Future Work

Negotiation among agents that are serving computer based applications differs from these used when lightweight devices are involved, and because we are rapidly approaching the era of lightweight services, a great focus and immediate redirection is realized towards the achievement of cooperative agents in mobile-based service architectures. In this paper we explained the motivation behind our interests to develop new agents' negotiation protocol that serves mobile-based applications. We demonstrated the research conducted in reaching cooperative MAS architectures, and the negotiation protocols previously proposed by scholars. We proposed our semi-heuristic negotiation protocol, and we applied it on Rideshare service architecture.

Our future research aims at increasing the usability of agent-based mobile service application, and accelerating the mobile service content delivery process. This would take place by: (1) Integrating the newly proposed negotiation protocol with different-purposes architectures that supply lightweight devices with certain mobile service. (2) Simulating agent's behavior in achieving complex tasks while applying our negotiation protocol, and performing the same task using existing negotiation protocols. This will help us observe differences in overall application performance and refine the proposed protocol. (3) Outlining the software agent design aspects that combine between both, the new negotiation protocol and the nature of the content offered to end users mobile devices. (4) We intend to study the possibility to let developers of software agents able to customize the negotiation protocol inputs so it fits into different services modules.

7. Acknowledgement

This work has been partially supported by different projects involvements: EU-SERENITY, PRIN-MEnSA, PAT-MOSTRO, PAT-STAMPS, and PAT-UNIQUIQUE SUUM. We also thank ArsLogica for the unabated cooperation and support given to innovative and creative ideas.

References

[1] M. Bombara, D. Cali, and C. Santoro. Kore: A multi-agent system to assist museum visitors. In *Proceedings of the Workshop on Objects and Agents (WOA2003)*, pages 175–178, Cagliari, Italy, 2003.

[2] V. Bryl, P. Giorgini, and S. Fante. Toothagent: A multi-agent system for virtual communities support. In *Proceedings of The Eighth International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2006)*, Hakotade, Japan, May 8-12, 2006.

[3] O. Bucur, P. Beaune, and O. Boissier. Representing context in an agent architecture for context-based decision making. In *Proceedings of the Workshop on Context Representation and Reasoning (CRR'05)*, Paris, France, 2005.

[4] K.-M. Chao, R. Anane, J.-H. Chen, and R. Gatward. Negotiating agents in a market oriented grid. In *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 436–437, IEEE Computer Society, 2002.

[5] N. Jennings, P. S., N. P., and S. C. On argumentation-based negotiation. In *Proceedings of the IWMAS, MIT Endicott House*, pages 1–7, Dedham, Massachusetts, USA, October 12-15, 1998.

[6] S. Kraus. Negotiation and cooperation in multi-agent environments. *Artificial Intelligence journal, Special Issue on Economic Principles of Multi-Agent Systems*, 94(1-2):79–98, 1997.

[7] A. Rakotonirainy, S. W. Loke, and A. Zaslavsky. Multi-agent support for open mobile virtual communities. In *Proceedings of the International Conference on Articial Intelligence (IC-AI 2000)*, Las Vegas, Nevada, USA, pages 127-133, 2000.

[8] K. Reynolds. A survey on auction types. Agorics, Inc., 1996.

[9] J. S. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*. The MIT Press, 1994.

[10] R. G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113, December, 1980.

[11] F. Sottini, S. Abdel-Naby, and P. Giorgini. Andiamo: A multi-agent system to provide a mobile-based rideshare service. Technical report, Informatica e Telecomunicazioni, University of Trento, 06-097.

[12] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1980.