Logic Journal of IGPL Advance Access published July 3, 2010

J-MADeM, a market-based model for complex decision problems

Francisco Grimaldo, Miguel Lozano and Fernando Barber*

Computer Science Department, University of Valencia, Dr Moliner 50, (Burjassot) Valencia, Spain

Abstract

This article presents J-MADeM, a multi-modal decision making mechanism to provide agents in a Multi-Agent System (MAS) with a market-based model for complex decision problems. The model is basically based on a set of utility functions, expressing the preferences of the agents for a specific problem, and a one-round sealed-bid combinatorial auction model as the procedure used to choose among different solutions. Thus, coordinated behaviours based on task/object passing can be evaluated to finally obtain acceptable allocations within a synthetic society. J-MADeM is able to simulate different kinds of societies (e.g. elitist, utilitarian, etc.), as well as social attitudes of their members (e.g. egoism, altruism, reciprocity, etc.). To evaluate J-MADeM, we analyse two problems: (i) the Gold Miners problem, where a new multi-agent organization is proposed to better adapt to different gold distributions; and (ii) a virtual university bar simulation, as an example of complex environment to verify the quality of the social behaviours achieved.

Keywords: Social reasoning, multi-agent resource allocation, welfare economics.

1 Introduction and Related Work

Decision making is the cognitive process leading to the selection of a course of action among a variety of them. There are several factors that influence this process, although probably the most important could be the amount of relevant information the agent manages when deciding its actions. This is specially important in Multi-agent System (MAS) simulations, where collective intelligence appears as the ability of the agents to achieve their goals by interacting with other similarly autonomous entities [8].

Social decision making models are being studied under the scope of MAS in order to regulate the autonomy of self-interested agents. Nowadays, the performance of an MAS is determined not only by its degree of deliberation but also by the degree of sociability. In this sense, sociability points to the ability to communicate, cooperate, collaborate, form alliances, coalitions and teams. Social reasoning has been extensively studied in MASs in order to incorporate social actions to cognitive agents [4]. As a result of these works, agent interaction models have evolved to social networks that try to imitate the social structures found in real life [7]. Social dependence networks allow agents to cooperate or to perform social exchanges attending to their dependence relations (i.e. social dependence/power [11]). Trust networks can define different delegation strategies by means of representing the attitude towards the others through the use of some kind of trust model (e.g. reputation [5]). In preference networks, individual preferences are normally expressed using utility functions

*E-mail: fernando.barber@uv.es

doi:10.1093/jigpal/jzq028

so that personal attitudes can be represented by the differential utilitarian importance they place on the others' utilities.

Following this preferential approach, the MADeM (Multi-modal Agent Decision Making) model [6] proposes a market-based mechanism for social decision making, capable of simulating different kinds of social welfares (e.g. elitist, utilitarian, etc.), as well as social attitudes of their members (e.g. egoism, altruism, etc.). The purpose of MADeM is to provide an MAS simulation framework with agents managing multi-modal social decisions. In this context, multi-modality expresses the ability to consider different focuses of attention coming from different sources. MADeM is based on the MARA theory [3] and it uses auctions as a basic procedure to provide the social feedback mentioned.

In this article, we describe how the MADeM model has been integrated into an agent programming language to make socially acceptable decisions available to agents eventually part of an organization. Among several languages for agent programming, we have chosen the AgentSpeak language [10] and its open source interpreter Jason [1] to program this kind of social agents. This choice was made because the language is based on the well-known BDI architecture and the interpreter can be easily customized to include the MADeM support. The coupling of MADeM with Jason is inspired in other extensions of Jason, in particular J-MOISE+ [9] and hence the name J-MADeM, as it joins Jason and MADeM.

The rest of the article is organized as follows. Section 2 reviews the definition of the MADeM model and describes its decision making procedure. Section 3 explains the J-MADeM architecture that provides Jason agents with the built-in feature of performing MADeM decisions. In Section 4, we analyse the perfomance obtained by J-MADeM in two application examples: (i) the Gold Miners problem, where a new multi-agent organization is proposed to adapt better to different gold distributions; and (ii) a virtual university bar simulation, as an example of complex environment to verify the quality of the social behaviours achieved. Finally, in Section 5 we state the conclusions of this work.

2 The MADeM model

The MADeM model provides agents with a general mechanism to make socially acceptable decisions. In this kind of decisions, the members of an organization are required to express their preferences with regard to the different solutions for a specific decision problem. The whole model is based on the MARA (Multi-Agent Resource Allocation) theory [3], and therefore it represents each one of these solutions as a set of resource allocations. Thus, the definition domain of MADeM is composed by the following elements:

- A set of agents $\mathcal{A}=\{a_1,...,a_n\}$ where each a_i represents a particular agent involved in the decision. A vector of weights $\overrightarrow{w}=<w_1,...,w_n>$ is associated to each agent representing the internal attitude of the agent towards other individuals.
- A set of resources $R=\{r_1,...,r_m\}$ to be allocated by the agents, where each r_i represents resources in the form of *task(Slot)*, where the *Slot* is a parameter that needs to be assigned in order to execute the *task*. Then, it identifies each one of the solutions for a specific decision problem as an allocation P of elements (either agents or objects) to task-slots as follows:

$$P=\{t_1(s_1)\leftarrow e_1,\ldots,t_1(s_n)\leftarrow e_n,t_2(s_1)\leftarrow e_{n+1},\ldots,t_m(s_n)\leftarrow e_{n*m}\}$$

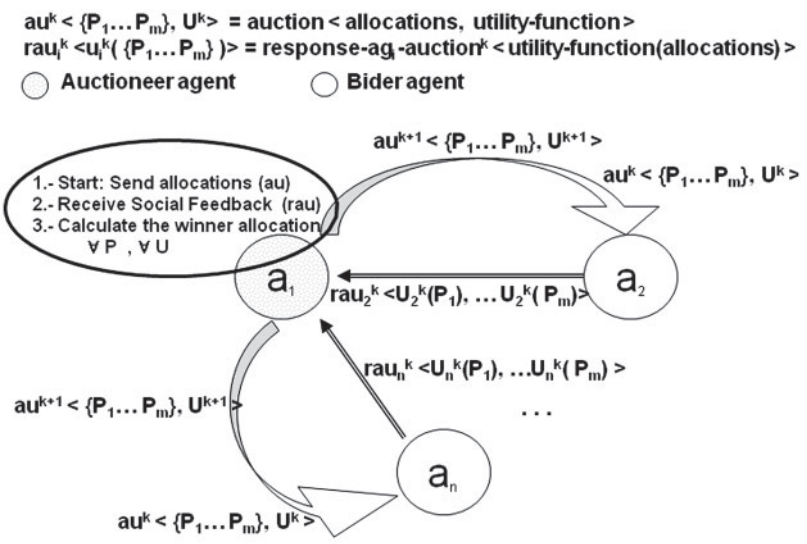

FIG. 1. MADeM procedure.

- A set of utility functions $U = \{U^1, U^2, ..., U^q\}$. These utility functions will be used to evaluate the allocations from different points of view. Additionally, each agent will have a vector of utility weights $\overrightarrow{w_u} = < w_{u_1}, ..., w_{u_q} >$ representing the importance given to each point of view in the multi-modal agent decision making.
- A collective utility function $Cuf \in \{elitist, egalitarian, utilitarian, nash\}$, representing the social welfare of the simulated society, i.e. the type of society where agents are located.

MADeM uses a market-based winner determination problem to merge the different preferences being collected according to the kind of agent or society simulated. The details of the whole decision making procedure are explained in the following subsection.

2.1 Decision making procedure

MADeM uses one-round sealed-bid combinatorial auctions to choose among different solutions to a decision problem. Auctioneer and bidder roles are not played by fixed agents throughout the simulation. Instead of that every agent can dynamically adopt each role depending on his/her needs or interests. For example, an agent would be the auctioneer when he wanted to pass a task to another agent. On the other hand, agents receiving the auction would bid their utility values provided that they were interested in the task being auctioned. Thus, MADeM lies in between centralized and distributed market-based allocation.

An overview of the multi-modal decision making procedure followed by the agents is shown in Figure 1. This procedure is mainly based on the following steps:

Auctioning phase: this phase is carried out by a single agent (a_1) who wants to socially solve a decision problem (e.g. where to sit). This agent then constructs the set of allocations representing all the possible solutions for the problem ($< P_1, P_2, ..., P_m >$). These allocations have the form of task slots assignations such as $SitAt(Obj_m) \leftarrow table_1$. Next, he auctions them

to a particular group of agents, whom we call the target agents. Each auction also includes a single type of utility function that the agent is interested in evaluating from the others ($au^k(< P_1, P_2, ..., P_m >, U^k)$). As complex decisions require to take into consideration more than one point of view, the auctioneer agent can start different auctions for the same set of allocations (au^1 through au^q).

Bidding phase: since the auctioneer informs about both the task slot allocations and the utility functions being considered, the a_i bidder agents simply have to compute the requested utility functions and return the values corresponding to each auction back to the auctioneer ($rau_i^k =< U_i^k(P_1), ..., U_i^k(P_m) >$).

Winner determination phase: in this phase, the auctioneer selects a winner allocation for each launched auction. To do this, he solves a classical winner determination problem. Afterwards he chooses one final winner allocation among these auction winners using a multi-modal decision making process. Thus, the final winner allocation will represent an acceptable decision for the society being simulated. The details of these calculations are fully described in [6].

3 J-MADeM architecture

This section describes how the MADeM model [6] has been integrated into Jason [1] as an open source library named J-MADeM. The J-MADeM is built upon the *Jason Communication Infrastructure*, thus extending the communication-level options available in Jason with a set of modules that provide the agents with the built-in feature of performing MADeM decisions. Figure 2a illustrates how these components are integrated into Jason. The J-MADeM basically offers to the AgentSpeak programmer: (i) an agent architecture that Jason agents can use to carry out their own MADeM decisions, (ii) an interface to develop utility functions that can be used along with the MADeM model and (iii) a set of internal actions to manage the parameters of these kinds of decisions. For a complete description of the utility function interface and the set of internal actions refer to the J-MADeM documentation available at the Jason web site [2].

The *J-MADeM Agent Architecture* extends the *Jason Agent Architecture* in order to incorporate all the necessary modules that allow MADeM decisions to be automatically carried out. The main components of the *J-MADeM Agent Architecture* are shown in the Figure 2b, where we can identify the following elements:

MADeM Parameters: this data storage contains the MADeM context currently defined for the agent. Essentially, it stores the personal weights, the utility weights, the collective utility function and the bid timeout to be used in future MADeM decisions.

Decision Launcher: this module starts the MADeM process for a particular decision. First, it stores the MADeM context for this decision into the *Decision Data* storage, thus allowing other decisions to be concurrently performed with different MADeM parameters. Second, it auctions each of the allocations being considered as solutions to the target agents.

Decision Data: this data storage holds all the information related to the MADeM decisions still in process. Therefore, it contains their MADeM context, their considered allocations and the preferences received for each of them.

MADeM Communication Module: this module extends the Jason agent communication module in order to deal with MADeM messages. When it receives a MADeM auction, it invokes the *Bidder Module* to get the agent's preferences over the considered allocations. On

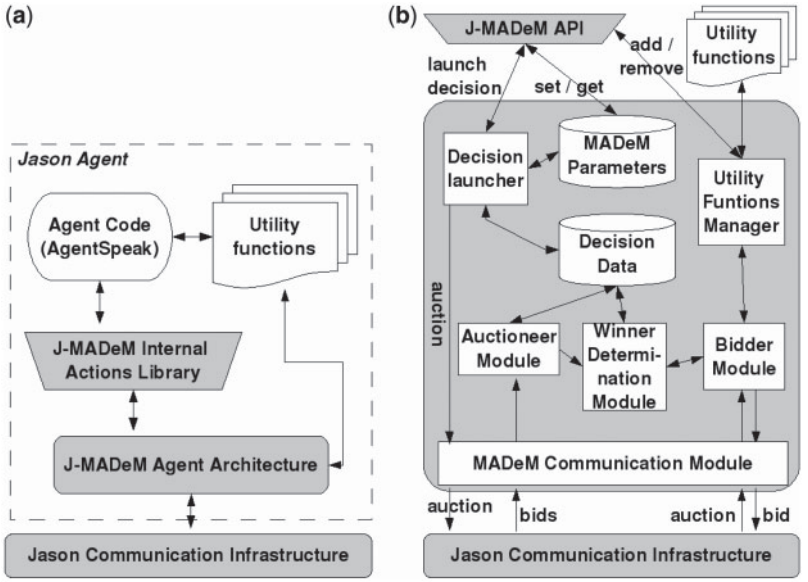

FIG. 2. (**a**) Overview of the J-MADeM architecture and (**b**) detailed view of the J-MADeM Agent Architecture.

the other hand, when it receives a MADeM bid, it informs the *Auctioneer Module* about the received preferences.

Bidder Module: this module manages the reception of a MADeM auction. It extracts the considered allocations and bids for them according to the agent's preferences. To express these preferences it relies on the utility values provided by the *Utility Functions Manager*.

Utility Functions Manager: this component acts as an interface between the built-in MADeM mechanism and the user-defined *Utility Functions*. Thus, it is in charge of locating and invoking them in order to calculate the agents' utilities for the set of considered allocations.

Auctioneer Module: this module manages the reception of MADeM bids. It extracts the sender's preferences and stores them into the *Decision Data*. As soon as the preferences from all the target agents have been received, it calls the *Winner Determination Module* to solve the decision.

Winner Determination Module: this module solves the MADeM winner determination problem using the information stored into the *Decision Data* for the decision being resolved (i.e. considered allocations, agents' preferences, personal weights, utility weights, social welfare, etc.). Once resolved, it notifies the agent about the winner solution.

4 Experiments and results

This section summarizes the results obtained in two application examples developed to test J-MADeM agents. First, we revisit the Gold Miners problem [1], a classical simulation scenario where agents must compete for the resources (gold) located at the environment. This example allows us to evaluate the efficiency of the auction-based method proposed and to experiment with dynamic organizations. Second, in order to test the sociability features

TABLE 1. Number of successful auctions in the original Jason Gold Miners implementation

Gold density (total golds)	0.05 (51)	0.1 (102)	0.15 (153)	0.2 (204)
4 agents	9.3	23.5	33.2	37.2
8 agents	10.6	28.4	47.4	58.3
12 agents	10.8	26.2	41.2	56
16 agents	10.4	21.5	33.4	53.6

provided by J-MADeM, we have created a virtual university bar simulation. In this scenario, waiter agents serve the orders placed by customer agents. According to the model parameters and the society being simulated, waiters are able to combine social behaviours (i.e. chatting) and efficiency at work.

4.1 The Gold Miners

In this example a team of gold-mining agents has to find a set of chunks of gold, randomly scattered in a grid-like territory, in order to carry them to a depot. The original Jason team consists of a leader that assigns each miner to a quadrant of the grid. The miners then explore and pick up the pieces of gold they find in the environment. To be more efficient, when an agent finds a gold in his way and cannot pick it up, he can use an auction-based model to inform the others about the new gold location and assign the gold to the winner agent.

Following the original Jason implementation, we have created an equivalent MAS with J-MADeM agents to be able to compare both. We have implemented an utility function (*goldDistance*), equivalent to the distance function used by the original miners, so that J-MADeM miners can express their preferences with regard to the gold units. Then, any agent can launch a J-MADeM decision as follows:

```
jmadem.launch_decision(LAgents,Alloc,[goldDistance],DecisionId);
```

where `LAgents` is the list containing all the agents apart from the announcer, since the original Jason implementation uses a broadcast auction model. A full version of this example can be downloaded from the Jason web site [2].

We first analyse the original Jason gold-miners performance to estimate the importance of the original auctioning process and to perform a fair comparison. To evaluate the efficiency of the auction model, we have measured the number of gold chunks picked up as a result of an auction. We identify this situation as a successful auction, as it is planned and completely executed. Table 1 shows the average of successful auctions obtained over 10 simulations with different number of agents and gold densities. To calculate the number of gold chunks involved in each simulation we just multiply the gold density by the grid size (in these results world size is 32×32). Surprisingly, Table 1 indicates that only the 25% of the gold are captured as a result of an auctioning process. To the authors knowledge, the reason behind this behaviour is that the centralized auction model implemented requires the leader to be able to manage parallel auctions during the simulation. However, the leader fails when dealing with concurrent auctions of the same pieces of gold.

On the other hand, Table 2 shows the number of successful auctions obtained by the equivalent J-MADeM gold-miners implementation. J-MADeM agents offer a robust auction controller able to handle different auctions at the same time. Thus, this feature allows the

TABLE 2. Successful auctions in the J-MADeM Gold Miners implementation

Gold density (total golds)	0.05 (51)	0.1 (102)	0.15 (153)	0.2 (204)
4 agents	24.4	40.9	69.5	75.8
8 agents	31.5	67.9	108.6	109.3
12 agents	30.9	70.6	98.6	121.1
16 agents	33.6	71.7	107.2	131.2

J-MADeM miners to solve the same problem but with a higher number of succesful auctions, around 65%. We notice that some planned allocations can still not be completed, since new free agents might happen to pick up gold chunks previously auctioned.

To evaluate global performance estimators for this problem, we have compared the total number of steps taken by miners in both implementations. This experiment shows a similar efficency for both approaches regardless of the number of agents and the gold density. This similarity also appears in other variables we have measured, such as the simulation time or the standard deviation of the pieces of gold picked by the agents.

Finally, to test the flexibility of J-MADeM, we have changed the initial uniform gold distribution. Now, the gold chunks will be concentrated in a single quadrant of the grid, thus representing a non-uniform gold distribution from a global perspective. Besides, we have modelled a new dynamic multi-agent organization to face this new situation while trying to increase the performance previously obtained. The new MAS has been organized in two roles: miners and bosses. First, the new miner agents are similar to the original Jason miners but they modify the announcing process. That is, instead of broadcasting gold locations to the rest of the agents, they just inform their corresponding boss. Second, the bosses are new J-MADeM agents devoted to organize the work of a group of miners within a quadrant of the environment. The bosses are not allowed to directly pick up pieces of gold but they can allocate them to its subordinated miners. Initially a balanced miner-boss assignment is performed, so that all bosses manage the same number of miners. However, they can dynamically change the organization by borrowing miners from other bosses when the number of gold chunks found in the quadrant they control increases. The altruist bosses modelled will always prefer to lend one of their miners than to keep it idle (i.e. without utility). Therefore, the bosses launch MADeM decisions both to allocate golds to miners and to ask other bosses for additional miners. For this new social decision, the bosses will use an utility function that returns the number of free miners to express their preference, thus trying to keep the teams well balanced.

Figure 3 compares the total number of steps followed by the three gold miners implementations considered. The first group corresponds to the performance obtained by the original Jason implementation (GM) when facing environments with non-uniform gold distribution. Then we have included the equivalent J-MADeM implementation (JM) and finally the new dynamic organization (JM+O). The figure shows how the new organization obtains higher performance regardless of the number of miners being simulated. Similar results have been also obtained for the simulation time, where the new organization carries out the same work but reducing significantly the required time.

4.2 Virtual university bar

This social scenario represents a virtual university bar where waiters take orders placed by customers. Both waiters and customers carry out tasks in the virtual bar and they use

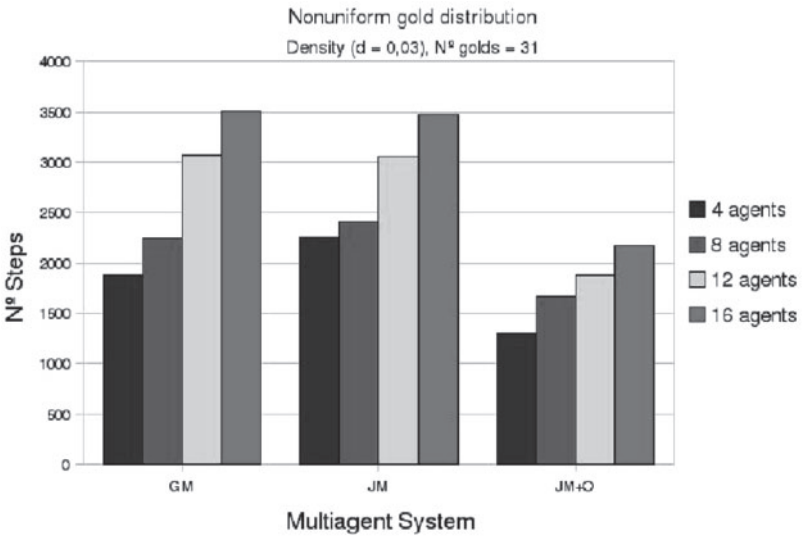

FIG. 3. Number of steps in the Gold Miners problem.

TABLE 3. Results for different personal weights.

	Coordinated		Social		Egalitarian	
Actitud	$\sigma_{Favours}$	$\overline{Favours}$	$\sigma_{Favours}$	$\overline{Favours}$	$\sigma_{Favours}$	$\overline{Favours}$
Indifference	7.57	6.9	3.52	8.7	7.58	13.6
Reciprocity	1.15	8.8	1.76	7.8	2.4	15.5
Altruism	5.94	17	6.66	12.7	4.44	17.9
Egoism	1.41	0.7	0.81	0.4	0.47	0.1

J-MADeM to decide among different task slot assignments. Next, we summarize the main results obtained. For a full description of the simulated scenario see [6].

Table 3 shows the results obtained for three kinds of waiters (i.e. *coordinated*, *social* or *egalitarian*) using different models of attitude. For instance, agents using *indifference* do not apply any modification over the utilities received, and therefore we consider the results of this attitude as the base values to compare with for each type of waiter. *Reciprocity* weights utilities attending to the ratio of favours already done between the agents. This attitude produces equilibrium in the number of favours exchanged as it can be seen in column $\sigma_{Favours}$. Altruism has been implemented in such a way that the weight given to oneself utilities is 0.25 whereas the weights for the rest of the agents is 0.75. As expected, altruist agents do more favours, since the importance given to the other's opinions is three times the importance given to their own opinion (see high values for the mean number of favours exchanged $\overline{Favours}$). On the other hand, egoism weights are 0.75 to oneself and 0.25 to the others, thus, agents rarely do favours (see low values in column $\overline{Favours}$).

Agent's preferences can sometimes go against personal attitudes. For example, *egoism* applied to *egalitarian* waiters produces that no task at all is passed among the agents ($\overline{Favours}$=0.1). However, agent's preferences can also empower personal attitudes.

For instance, *altruism* applied to *coordinated* waiters produces a high level of specialization. This type of agents produces big values for $\sigma_{Favours}$ as the agents already using a dispenser (e.g. a juice machine) keep on getting products from the dispenser following both an altruist and a coordinated behaviour that reduces collisions for the use of an exclusive resource. In spite of that, personal weights have demonstrated to produce similar effects on the agents regardless of the kind of waiter being considered.

5 Conclusions

This article has presented J-MADeM, a new open source library oriented to create different types of social simulation agents. J-MADeM agents are able to merge several points of view received from other agents. This social feedback is modelled via utility functions that express the preferences of each agent for every solution considered. The J-MADeM architecture and its integration into Jason have been reviewed. The application examples presented demonstrates the robustness and flexibility to create complex market-based social simulations. First, the Gold Miners example can use J-MADeM to maximize the efficiency of the MAS. Second, the virtual university bar example offers a more complex environment where several utilities (personal and collective) and weights (personal and utility based) allow to design elaborated social simulations.

Acknowledgements

This work has been jointly supported by the Spanish MEC and MICINN, as well as European Commission FEDER funds under grants Consolider Ingenio-2010 CSD2006-00046 and TIN2009-14475-C04-04.

References

[1] R. H. Bordini, J. F. Hübner, and M. Wooldrige. *Programming Multi-Agent Systems in AgentSpeak using Jason*. Wiley, 2007.

[2] R. H. Bordini and J. F. Hübner. Jason. Available at http://jason.sourceforge.net/ (Last accessed 25 June 2010), 2010.

[3] Y. Chevaleyre, P. E. Dunne, U. Endriss, J. Lang, M. Lemaitre, N. Maudet, J. Padget, S. Phelps, J. A. Rodriguez-Aguilar, and P. Sousa. Issues in multiagent resource allocation. *Informatica*, **30**, 3–31, 2006.

[4] R. Conte and C. Castelfranchi. *Cognitive and Social Action*. UCL Press, 1995.

[5] R. Falcone, G. Pezzulo, C. Castelfranchi, and G. Calvi. Why a cognitive trustier performs better: Simulating trust-based contract nets. In *Proceedings of AAMAS'04: Autonomous Agents and Multi-Agent Systems*, pp. 1392–1393. ACM, 2004.

[6] F. Grimaldo, M. Lozano, and F. Barber. MADeM: a multi-modal decision making for social MAS. In *Proceedings of AAMAS'08: Autonomous Agents and Multi-Agent Systems*, pp. 183–190. ACM, 2008.

[7] H. Hexmoor. From inter-agents to groups. In *Proceedings of ISAI'01: International Symposium on Artificial Intelligence*, 2001.

[8] L. M. Hogg and N. Jennings. Socially intelligent reasoning for autonomous agents. *IEEE Transactions on System Man and Cybernetics*, **31**, 381–393, 2001.

[9] J. F. Hübner, J. S. Sichman, and O. Boissier. Developing organised multi-agent systems using the Moise+ model: programming issues at the system and agent levels. *Journal of Agent-Oriented Software Engineering*, **1**, 370–395, 2007.

[10] A. S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. Vol. 1038 of *Lecture Notes in Computer Science*, pp. 42–55. Springer, 1996.

[11] J. Sichman and Y. Demazeau. On social reasoning in multi-agent systems. *Revista Ibero-Americana de Inteligencia Artificial*, **13**, 68–84, 2001.

Received 1 December 2009