An Assistance Infrastructure to Inform Agents for Decision Support in Open MAS*

Pablo Almajano[1], Maite Lopez-Sanchez[2], and Inmaculada Rodriguez[2]
[1] *Artificial Intelligence Research Institute (IIIA-CSIC),* palmajano@iiia.csic.es
[2] *WAI research group, University of Barcelona,* {maite,inma}@maia.ub.es

Abstract

Organisations are an effective mechanism to define the coordination model that structure agent interactions in Open MAS. Execution infrastructures mediate agents interactions while enforcing the rules imposed by the organisation. Although infrastructures usually provide open specifications to agents, understanding this specification and participating in the organisation could result a difficult task to agents, specially when the system is hybrid (i.e participants can be both human and software agents) and its specification becomes more and more complex. In this paper we further formalise a two layered *Assistance Infrastructure* in order to enable and evaluate different *Assistance Services* to agents in MAS. We also formalise the *Information Service* and evaluate it using the case study of a water market. Experiments results show that the information service increases agents satisfaction and helps the system meets its organisational goals. In addition, different information services may support individual agents in their decision processes when they follow alternative strategies.

Keywords: Decision Support, Assistance Infrastructures, Assistance Services

1 Introduction

Usually, multi-agent systems (MAS [15]) design and implementation involves the specification of a coordination model and the development of an infrastructure in charge of enacting it. In Open MAS, systems are populated by heterogeneous agents trying to achieve particular and/or collective goals. These agents are developed by third parties so that the number and the cognitive abilities of agents that may participate in an Open MAS is unknown at development time, and varies at runtime [16]. Organisation Centred MAS (OCMAS [12]) have proven to be an effective mechanism to define the coordination model that structures agent interactions in MAS, and infrastructures give support to their execution by imposing

*This work is partially funded by EVE (TIN2009-14702-C02-01 / TIN2009-14702-C02-02), AT (CONSOLIDER CSD2007-0022) and TIN2011-24220 Spanish research projects, EU-FEDER funds.

interaction rules between participants. Although OCMAS infrastructures usually provide open specifications to agents [11] [14], understanding these specifications and participating in the organisation could result a difficult task to agents, specially as its specification becomes more and more complex. If we take the humans in the loop and consider hybrid systems, where agents may be humans or software agents, the complexity increases and facilitating agent participation becomes a mandatory issue [3] [19].

This paper focuses on the challenge of improving agents' participation in the organisation by means of an *Assistance Infrastructure*. Certain data about the organisation and its environment require complex computational processes in order to be useful for agents. Therefore, agents would improve their participation in the organisation if the infrastructure could provide them with some assistance mechanisms that facilitate such processes. Our aim is to help agents in achieving their goals, and, when they are aligned with global goals, lead to a better system's global performance [18].

In this paper we further formalise an extension of the *Assistance Infrastructure* defined by Campos et al. [7] in order to enable and evaluate assistance in Open OCMAS systems. Our framework is composed by: i) the extension of the *Organisational Layer*, which regulates the participation of the agents in the system and manages the historical information about the organisation and its execution state; and the addition of an *Assistance Layer*, populated by *Personal Assistants* which provide general *Assistance Services* to agents in the organisation. In a previous work [1], we have preliminarily formalised four main categories of *Assistance Services*. Here, we further formalise the *Information Service*, provide a specific example, and evaluate it in a prototype based on *amWater* [1], which implements a water market in the agriculture domain.

In our experiments we first execute a *base-line* configuration without assistance. Then, data from this initial configuration are used in three alternative configurations that provide information assistance about *Runtime Properties*. An *Assistance Quality of Service* measure evaluates whether the assistance is an useful decision support tool for participant agents (i.e helps agents to satisfy their goals) and helps the systems to meet its goals (when they are aligned with participants' goals).

The paper is structured in the following parts. First, section 2 summarizes the related work. Second, section 3 provides definitions and notation in order to formalise both the proposed *Assistance Infrastructure* and the *Information Service*. Next, section 4 empirically evaluates information service. Finally section 5 gives some conclusions and future work.

2 Related work

There are two main lines of active research in assistance to MAS provided by *Software Agents*: organisational assistance services [8] [5] [6], and agent assistance services [10] [17] [9]. Regarding organisational assistance, Centeno et al. [8] defined an incentive mechanism (*Incentivators*) which induces individual participants to follow organisational goals by learning their preferences and doing modifications in the environment. On the other hand, Bou et al. [4] defined an Electronic Insti-

tution with autonomic capabilities that allows it to adapt its regulations to comply with institutional goals despite varying agent's behaviours (Autonomic Electronic Institutions). In a preceding work, they also applied Case-Based Reasoning (CBR) to reason about the process of adapting the norms of an Electronic Institution when certain system-wide measures differ from the expected ones [5]. Finally, the Two Level Assisted MAS Architecture (2-LAMA [6]) also provides organisational assistance services. It is composed by two levels. In the domain-level (DL) agents perform domain specific activities. On top of it, a distributed meta-level (ML) is in charge of providing assistance to the DL. This assistance is performed by changing the norms of the organisation and it is provided to groups of not overlapped and fixed clusters of agents. Our approach goes in the line of agent assistance service, so it differs from previous ones in organisational perspectives.

Other works focus on agent assistance services. Electric Elves [10] is a system that applies agent technology in service of the day-to-day activities of the Intelligent Systems Division of the University of Southern California Information Sciences Institute. Chalupsky et al. developed specific Software Personal Assistants (*SPA*) for project activities coordination and external meetings organisation. Since our proposal is general for MAS our *Assistance Layer* can include such kind of services.

These and other proposed *SPA*'s abilities were evaluated in a conceptual framework that simulated human behaviour in different MAS structures [18]. In this research, Okamoto et al. evaluated the impact that *SPA*s have on the individual performance and on the collective performance of the organisation as a whole. They built a computational model of human organisations and analysed two types of agent's organisational structures: hierarchical and horizontal. One *SPA* measured ability that is close to our proposal is the decision support (see section 3.2). They concluded that supporting decision tasks in human organisations increases the success rate (i. e., to meet the deadline with higher probability) and the speed performance average (i. e., to meet the deadline more rapidly), this is particularly the case in organisations with hierarchical structure.

A recent work presented a generic assistant agent framework in which various applications can be built [17]. As a proof of concept application, it implemented a coalition planning assistant agent in a peacekeeping problem domain. A more general framework for organising MAS [9] contains *Informative Organisational Mechanisms* and *Regulative Organisational Mechanisms*, a generalisation of the *Incentivators* [8]. As mentioned approaches, we also propose a general framework. Moreover, we propose to offer planning as an advice service in our infrastructure (see section 3.2) as in the former. The latter, *Informative Organisational Mechanisms*, is a generalisation of our *Agent's Assistance Services*.

Existing OCMAS infrastructures already offer some kind of information about the organisation to a participant. In the $\mathcal{S} - \mathcal{M}oise^+$ [14] middleware, the OrgManager provides useful information for the organisational reasoning of agents and its organisational coordination. In this model, an agent is allowed to know another agent information if their roles are linked by an acquaintance relation (defined in the social level). Moreover, the OrgManager also informs actors about the new permissions, obligations, and goals they can pursue when a new state is reached in the organisation. Our framework provides similar information services and elaborated ones, e.g., statistics on information that may be unknown for participants.

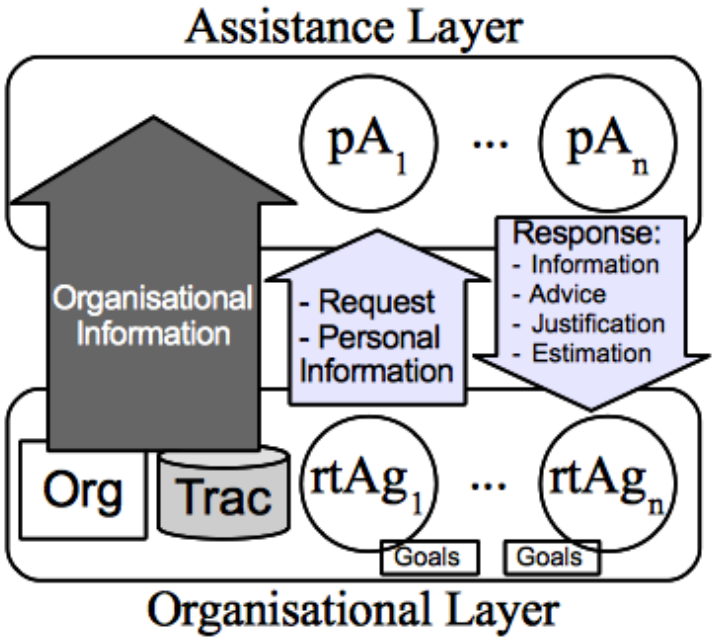


Figure 1: Assistance Infrastructure overview

3 Assistance Infrastructure formalisation

With the aim of assisting –further than enabling– agent coordination, we propose the *Assistance Infrastructure* depicted in Figure 1. It is composed by *Organisational Layer* (OL) and *Assistance Layer* (AL). The OL contains i) a set of running agents ($rtAg$), ii) the specification of the organisation (Org) and iii) historical information from previous system executions ($Trac$). The AL contains a set of personal assistants (pA) which provide support to $rtAg$. Three arrows show the private communication channel between both layers in Figure 1. First, pA use organizational information (i.e. Org and $Trac$ data) in order to provide assistance. Second, $rtAg$ may reveal personal information (e.g., their goals) and request help to pA in order to be adequately assisted. The last one is devoted for pA to provide the services (response) to $rtAg$. We illustrate the formalisation in *amWater* scenario, a water market where participants negotiate water rights. A *transaction* is the result of a negotiation where a seller settles with a buyer to reallocate (part of) the water from its water rights for a fixed period of time in exchange for a given amount of money[1].

3.1 Organisational Layer

We propose an *Organisational Layer* (OL) specification for Open MAS extended from the definition of Campos et al. [6]. Our main contributions in this layer are:

[1]We give a further explanation in section 4.1.

i) the extension of the *Organisation Specification* (Org) to include elements related to assistance and ii) the addition of the *Organisational Trace* ($Trac$).

3.1.1 Organisation Specification

Eq. 1 presents our proposed Organisation Specification (Org) (extensions appear in bold text).

$$Org = \langle \mathbf{O}, SocStr, SocConv, \mathbf{DomP}, Goals, \mathbf{AssQoS} \rangle \quad (1)$$

$$SocStr = \langle AgP, Rol, Rel \rangle \quad (2)$$

$$SocConv = \langle Prot, \mathbf{Activ}, \mathbf{ActivRel}, Norms \rangle \quad (3)$$

$$\mathbf{Goal} = \langle OrgGoal, \mathbf{AgSat} \rangle \quad (4)$$

–**Ontology. O** is a conceptualization of the domain (actions and possible domain concepts). For example, in amWater *StartRound* corresponds an action and *Transaction* (*trans*) is a concept with some properties (e.g., *trans.quantity*).

–**Social Structure.** $SocStr$ is a *social structure* (Eq. 2) consisting of: i) a set of agents properties (AgP), common for all participants; ii) a set of roles (Rol), where each role $\mathbf{rol} \in Rol$ contains the set of properties, **RolP**, characterizing it, $\mathbf{rol} = \langle \mathbf{RolP} \rangle$ and iii) Rel, the relationships among roles.

In general, we consider a property as a triple $\langle name, type, v \rangle$ with a $name$[2], a *type* and its *visibility* (v). *Visibility* implements the natural privacy policy of any organisation. A *private* property can only be accessed by its owner. An agent $rtAg$ perceives a property defined as *restricted* whenever it deploys role rol such as $restricted = \langle RoleV \rangle$ and $rol \in RoleV$. Finally, *public* denotes it is completely visible within the organisation.

In amWater, the different roles are irrigator – with buyer and seller subroles ($inheritance \in Rel$) –, market facilitator and basin authority. The property *trans.quantity* is public, agent owned water rights are private and auction's starting price is restricted to the market facilitator role.

–**Social Conventions.** $SocConv$ stands for the *social conventions* agents should conform to and expect others to conform to. It is defined in Equation 3 and is composed by a set of *Protocols* ($Prot$), a set of *Activities* (**Activ**), their relationships **ActivRel** and a set of Norms ($Norms$).

First, a *protocol* $\mathbf{prot} \in Prot$ is defined as $\mathbf{prot} = \langle \mathbf{ProtSpec}, \mathbf{ProtP} \rangle$, where **ProtSpec** specifies the interaction mechanism between agents and **ProtP** is the set of its properties. Second, an *activity* $\mathbf{activ} \in \mathbf{Activ}$ is defined as $\mathbf{activ} = \langle \mathbf{ActivP}, \mathbf{ActivProt} \rangle$, where **ActivP** is a set of properties and $\mathbf{ActivProt} \subseteq Prot$ is a set of protocols that can perform it. One *activity* can be instantiated multiple times and each instance can have one different protocol associated. Third, **ActivRel** defines the *activities relationships* as a work-flow specifying the possible transitions between different *activities*. Finally, $Norms$ are applied by the organisation with the aim of further regulating the *activities*. $Norms$ could be specific of one *activity* or apply to the overall system. Note that norms can be violated by agents whereas protocols' rules can not. In our example scenario, we have

[2]We use the notation $x.name$ to refer to property $name$ of component x (e.g., $buyer1.credit$ denotes the property $credit$ of agent $buyer1$).

one *Auction* activity, following a *multi-unit Japanese* auction protocol to negotiate transfers. As an example of a norm is *"buyers can acquire a maximum of 40% of the total quantity of water"*. A relationship between activities in our model is: after registering rights in a *Registration* activity, a seller may move to *Waiting And Information* activity.

–**Domain properties**, defined as $\mathbf{DomP} = \langle \mathbf{OrgP}, \mathbf{EnvP} \rangle$, are differentiated between: organisation internal properties (**OrgP**), such as the list of transactions (*Trans*); and environmental properties (**EnvP**), such as the water needs of a cultivation.

–**Goals** (**Goal**) are defined in Eq. 4 as a duple containing a set of *organisational goals* (*OrgGoal*) and a method to calculate *agent satisfaction* (**AgSat**). *OrgGoal* describe the organisation design purpose in terms of desired values for certain properties. **AgSat** is a method (may be asking the agents) to obtain the participants' degree of satisfaction at runtime. In *amWater*, one *organisational goal* is to minimise the auction time and the degree of *agent satisfaction* for a buyer can be related to the quantity of water bought.

–**Assistance Quality of Service** (**AssQoS**) mainly evaluates whether the assistance layer helps agents to satisfy their goals (*AgSat*) and, when they are aligned with organisational goals (*OrgGoal*) led to a better system performance. In section 4.3, we have tested *Information Services* defined in section 3.2.2 by comparing the values that *AgSat* and *OrgGoal* take in different configured executions.

3.1.2 Organisation Historical Information

At runtime, agents enter the organisation, interact trying to achieve their goals and finally exit. As the result of these (inter)actions, the state of the organisation changes. The Organisational Layer keeps the sequence of *Execution States* (S) and the agent *Runtime Actions* (RtA) within the *Organisational Trace* ($Trac$).

–**Execution States.** An state S contains current runtime (Rt) values of non-static organisational elements. Eq. 5-11 specify these elements based on Eq. 1-4. The applicable norms at runtime, $RtNorms$, specify (see Eq. 9): i) a set of active norms ($RtAct$); ii) the list of norm violations ($RtVio$); and iii) the list of consequences due to both norm applications or norm violations ($RtCon$).

$$S = \langle RtSocStr, RtSocConv, RtDomP, RtGoals, RtAssQoS \rangle \tag{5}$$

$$RtSocStr = \langle RtAg \rangle; rtAg = \langle RtAgP, RtRolP \rangle, rtAg \in RtAg \tag{6}$$

$$RtSocConv = \langle RtActiv, RtNorms \rangle \tag{7}$$

$$rtActiv = \langle RtActivP, RtProtP \rangle, rtActiv \in RtActiv \tag{8}$$

$$RtNorms = \langle RtAct, RtVio, RtCon \rangle \tag{9}$$

$$RtDomP = \langle RtOrgP, RtEnvP \rangle \tag{10}$$

$$RtGoal = \langle RtOrgGoal, RtAgSat \rangle \tag{11}$$

–**Runtime Actions**, $RtA = \{rtA_1, \ldots, rtA_n\}$, is the set of agents' actions performed at runtime, where n is the number of agents of the institution and rtA_i is the action performed by an agent i. We assign the *idle* action to rtA_i ($rtA_i = idle$) when agent i does not perform any action.

–**Organisational Trace**, $Trac$, contains the trace of the organisation (Eq. 12) specified by: i) a set of *time stamps*, T; ii) the sequence of *execution states* at each *time stamp*, S; and iii) a set of agent *actions performed* at each *state*, RtA. S_0 is the initial state, S_c is the current state and in general S_i is the organisation execution state at step i, RtA_i is the set of actions that take place at such step and T_i indicates the time at which S_i occurred. S only keeps information about changes in the organisation.

$$Trac = \{\langle T_0, S_0, RtA_0\rangle, \ldots, \langle T_c, S_c, RtA_c\rangle\} \tag{12}$$

Table 1 shows an example of $Trac$ in amWater. The property *"state"* of one auction activity at step occurred at time t ($rtActiv_i \in S_t.RtSocConc.RtActiv$) has the value *opened*, therefore the auction has not yet started. It just starts when agent $rtAg_j$ enacting market facilitator role ($RtRol_{rtAg_j} = MarketFacilitator$) performs the illocution *"StartRound"*. Then, the property *"state"* is updated and S changes at step *t+1*.

T	S	RtA
t	$rtActiv_i.state = opened$	$StartRound(rtActiv_i, rtAg_j)$
t+1	$rtActiv_i.state = running$	

Table 1: Example of amWater Organisational Trace at time t and *t+1*

3.2 Assistance Layer

The *Assistance Layer* depicted at the top in Figure 1 is in charge of providing decision support to agents in the *Organisational Layer*. It is populated by the so-called *Personal Assistants*, a set of organisational agents offering a set of *Assistance Services* to participants in the *Organisational Layer*.

We propose the Assistance Layer to provide the following services ($AsServ = \{Info, Adv, Just, Est\}$): i) refined *information* ($Info$), ii) *advice* (Adv), iii) *justification* ($Just$) of either action consequences or applied constraints when performing an action and iv) *estimation* (Est) of action consequences.

3.2.1 Personal Assistant Agents

We define PA as the set of *Personal Assistant Agents* which belong to the organisation. One personal assistant $pA_i \in PA$ provides direct and sole support to one agent[3] $rtAg_i \in RtAg$. We have formalised two assistance communication processes[4]:

$$AsServ : Org \times Trac \rightarrow Res \tag{13}$$

$$AsServReq : Req \times Org \times Trac \rightarrow Res \tag{14}$$

Since *Personal Assistants* are organisational agents, they are allowed to use organisational information[5] in order to provide assistance. However, they only provide

[3]An agent, as an autonomous entity, can freely use the given support in its decision process.

[4]The private communication channel is depicted as arrows in Fig. 1.

[5]We assume in this definition that the infrastructure sends information about the organisation specification (Org) and its execution state ($Trac$).

responses (Res) concerning properties that respect *visibility*[6] constraints. First, Eq. 13 defines the process by subscription ($AsServ$). It can be provided at different frequencies of subscriptions: each time the information changes, e.g. when a new water right has been registered to transfer; at concrete moments, e.g. the first time entering the organisation the agent receives a "welcome pack"; and never, e.g. subscription disabled. Second, Eq. 14 defines the process under request ($AsServReq$). The request, Req, can include personal information of the agent (e.g., its goals) which can decide whether to reveal it or not[7]. The specification of both Req and Res illocutions contains first the sender agent, second the receiver agent and a set of parameters in the request, $Req = \langle rtAg_i, pA_i, Par \rangle$, or a set of values in the response, $Res = \langle pA_i, rtAg_i, Val \rangle$.

As previously introduced, we propose to establish a private communication channel between a personal assistant pA and its assisted agent $rtAg$ in order to preserve the privacy of the information in the communications. To ensure the use of private information in the defined terms and conditions, a service contract may be signed between pA and $rtAg$. On one hand, this contract commit pA to keep personal information private, to not exploit it, and to offer services pursuing $rtAg$'s private information following social conventions. On the other hand, $rtAg$ is responsible for the use it makes of the services rendered based on the personal information revealed by it and can decide when the personal information should be erased by the assistants. As a pA is designed to use organisational trace and personal information in a private way, we consider that agents should have confidence using them.

3.2.2 Information Services Specification

We consider 3 main reasons for agent needing of information. First, the specification of the organisation could be difficult to understand for some participants. Second, runtime data could not be perceived by participants, because, even though it is visible to them, they were not notified about it. Finally, data from previous executions may require to be processed in order to be useful for agents' decisions, e.g., the weighted average value of transactions' prices in *amWater* along a previous market execution. Therefore, *Information Service* can be divided into three subcategories of services: i) *Organisation Specification*, ii) *Runtime Organisation Specification* and iii) *Runtime Properties*. Next we define the different parameters of the request (Par) and the values provided in the response (Val) for each subcategory.

Organisation Specification (Os) is information about the organisation as defined at design time. The only parameter of the request is the name of a component[8] in the organisation specification (Org). The response contents the value of such a component at design time. For example, let imagine that in amWater an agent Pablo ($rtAg_{Pablo}, rtRol_{Pablo} = seller$) wants to participate in a registration activity following protocol $prot_j$ and require the roles allowed to participate in such a protocol to his personal assistant: $Req_{Os}(rtAg_{Pablo}, pA_{Pablo}, prot_j.Rol)$,

[6] *Properties visibility* is further explained in section 3.1.1

[7] We stress that the more relevant information revealed, the better services will be provided.

[8] It should be specified by using the access variable values operators mentioned in section 3.1.1

obtaining as response $Res_{Os}(pA_{Pablo}, rtAg_{Pablo}, \{seller, marketFacilitator\})$.

Runtime Organisation Specification ($RtOs$) is information of the organisation specification based on the current situation of the agent within the organisation at runtime. In this case, one parameter $par \in \{actions, location, destination\}$ indicates the type of specification requested: i) the allowed *actions*, ii) the current *location* or iii) the possible *destinations* in the organisation. The values in the response can be i) a set of permitted *actions*; ii) a *location loc* expressed as an activity identifier or an activity's relationship, $loc \in \{Activ, ActivRel\}$; or iii) a set of *destinations*, where a destination *des* is expressed in the same way as a location ($des \in \{Activ, ActivRel\}$). Following with the example in amWater, once Pablo has joined the registration activity $rtActiv_j$, he could ask for the actions he is allowed to perform at current state. The request will be $Req_{RtOs}(rtAg_{Pablo}, pA_{Pablo}, actions)$ and one response could be $Res_{RtOs}(pA_{Pablo}, rtAg_{Pablo}, \{register, exit\})$.

Runtime Properties (Rt) is (processed) information about historical values of the observable properties. The parameters of the request are the name of a property in S, and two timestamps defining a period: t_{ini}, indicating the beginning, and t_{fin}, indicating the end. The response will content the corresponding values of such a property at runtime between the specified timestamps. Continuing with the previous example, Pablo can request for a low transactions price in order to decide the starting price of his water rights in the negotiation. One possible request could be about transactions of a previous auction activity k between rounds with time stamp t_1 and t_{80}: $Req_{Rt}(rtAg_{Pablo}, pA_{Pablo}, \langle rtActiv_k.Trans.lowPrice, t_1, t_{80}\rangle)$. One possible response could be $Res_{Rt}(pA_{Pablo}, rtAg_{Pablo}, 9.00)$.

4 Assistance Evaluation

This section is devoted to evaluate our proposed *Assistance Infrastructure* by testing different information service configurations in a water market case study.

4.1 amWater: a Demonstrator of Assistance Scenario

As a case study, we have enacted an *a*ssisted electronic *m*arket of *Water* rights (*amWater*[9]) based on *mWater* [13]. The *Organisational Layer* corresponds to an *Electronic Institution* (EI [2]) and the *Assistance Layer* improves the market by providing *Runtime Properties Information Service*. Considering an agricultural context, *amWater* is associated to an irrigation basin which is divided in geographical areas of interconnected lands (and their associated water rights). Water transfers between lands are possible by using available basin's infrastructures.

External agents join this water market enacting either *buyer* or *seller* subroles while *market facilitator* and *basin authority* are designated for staff agents (*Rol* and *Rel* in Eq. 2). Specifically, agents can join three activities (*Activ* in Eq. 3): *Registration*, where the market facilitator is in charge of registering sellers' rights; *Waiting and Information*, where irrigators can ask for information about auctions to the market facilitator; and *Auction*, where the negotiation of water rights takes place. We have selected a multi-unit Japanese Auction protocol (*Prot* in Eq. 3)

[9]Notice that this domain is naturally structured and requires organisation.

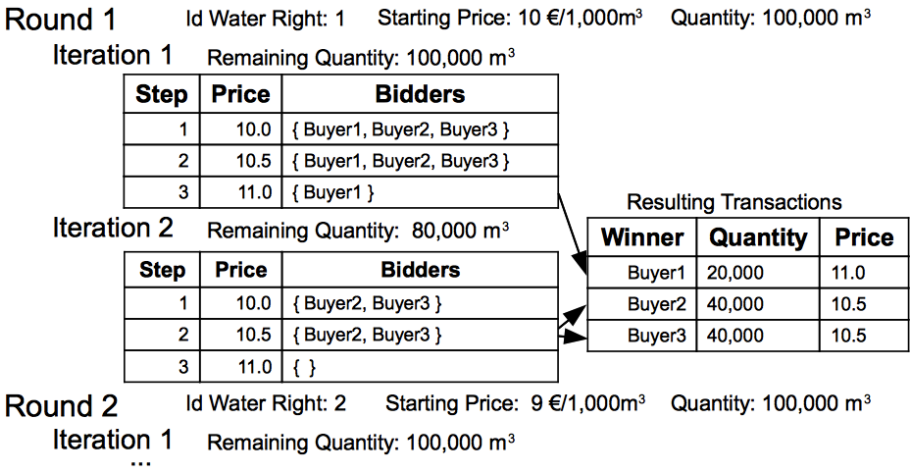


Figure 2: An example execution of a *Multi-unit Japanese* protocol

because it is suitable for divisible and perishable goods, in our case, water. In this protocol, the *market facilitator* conducts the auction of registered water rights –composed by several m^3 of water– in a *round-iteration-step* schema. Figure 2 shows an example execution of such a protocol. *Rounds* are divided in several *iterations* which are further divided in several *steps*. The *market facilitator* starts a new *round* for each registered water right at the starting price established by the *seller*. Subsequent *iterations* follow these four rules. First, one *iteration* is composed by several *steps* where the price increases in regular increments (0.5€ in Fig.2). Second, *buyers* are allowed to place bids for consecutive *steps*. Third, the *iteration* ends when i) just one buyer bids at current *step* and becomes the winner or ii) no bids are performed, so the winners are determined to be the *buyers* that bid at previous *step*. Last, winners request the amount of water desired (e.g. *Buyer1* buys 20,000m^3). If there is more than one winner (e.g. *Buyer2* and *Buyer3* in second iteration), then the water is assigned by following a proportional allocation algorithm (i.e. 40,000m^3 for each buyer). Once an *iteration* is finished, the *basin authority* validates the result(s) and announces the resulting transaction(s). Afterwards, if it remains water in the right, then a new *iteration* starts. No bids then would imply the ending of both the *iteration* and the *round*. The negotiation is over when all rights have been traded.

4.2 Experiment configuration

In order to asses the benefits (in terms of system performance and quality of service) of our information service, we have configured four alternative experiments. The first one does not use it and, thus, acts as a base-line for comparison purposes. The other configurations use the service (which gathers trace information from the base-line) by issuing different information requests.

Agents in the *base-line* configuration include staff agents, which follow a predefined and fixed behaviour, and an heterogeneous population of 100 buyers and 100

sellers. All buyers and sellers ($rtAg_i, rtRol_i = buyer|seller$) aim at buying/selling the same fixed water quantity ($rtAg_i.quantity = 100{,}000$). The variation in their behaviour is modelled in terms of the purchasing/sale price of water rights. Specifically, we assume buyers have an inner maximum purchasing price whose value is normally distributed, $\mathcal{N}(\mu, \sigma^2)$, with $\mu = 12$ (i.e. 12 €/*1,000m*^{3}) and $\sigma^2 = 2$. As for sellers, their starting price is low enough to ensure sales and follows a normal distribution $\mathcal{N}(4, 1)$. In order to preserve similar market conditions, just sellers'

Configuration	**Parameters of request** (Par)	**Value of response** (Val)
base-line	no information	–
low	$\langle Trans.lowPrice, t_1, t_{80}\rangle$	$Trans.minPrice - k$
medium	$\langle Trans.medPrice, t_1, t_{80}\rangle$	$Trans.wAvePrice - k$
high	$\langle Trans.highPrice, t_1, t_{80}\rangle$	$Trans.maxPrice - k$

Table 2: Request performed in each configuration and the subsequent response

price strategies are changed among the other test configurations. As Table 2 shows, sellers request for a different information service in different configurations. These services represent a decision support tool for setting seller's starting price. Thus, if the seller strategy is to set low starting prices, the corresponding information service will provide the minimum transaction price in the trace with a k decrement. Similarly, the seller can request a medium or a high starting price (Par), and the values of the responses (Val) will be a value k below the weighted average ($wAvePrice$) or the maximum historical price respectively. Equation 15 details the $wAvePrice$ computation over a set of transactions ($Trans$), where $trans_i.quantity$ and $trans_i.price$ denote the water quantity and price of transaction i.

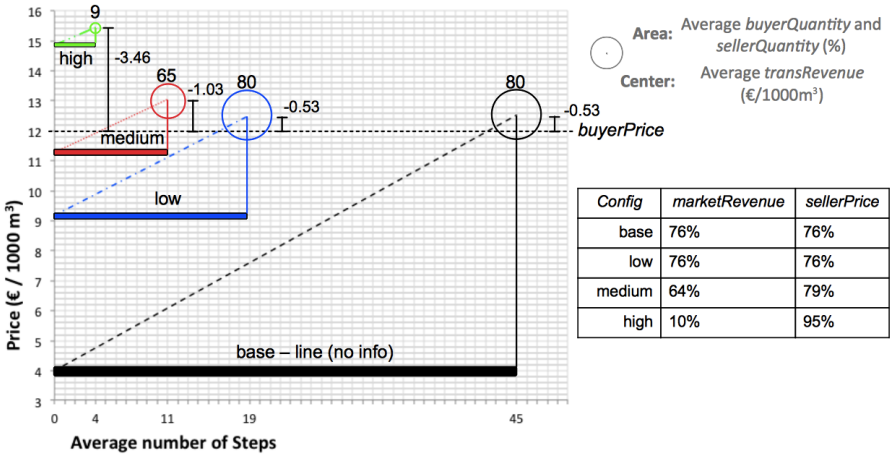
$$wAvePrice(Trans) = \frac{\sum_{i=1}^{|Trans|} (trans_i.quantity \times trans_i.price)}{\sum_{i=1}^{|Trans|} trans_i.quantity} \tag{15}$$

4.2.1 Goals

We have defined three *Organisational Goals*, and four *Agent Satisfaction* parameters ($OrgGoal$ and $AgSat$ in Eq. 4), two for buyers and two for sellers:

$$OrgGoal = \{transPerform, transRevenue, marketRevenue\}$$
$$AgSat = \{buyerQuantity, buyerPrice, sellerQuantity, sellerPrice\}$$

On one hand, $OrgGoal$ contains the following components: i) *Transaction Performance* ($transPerform$), inverse to the average number of steps to complete a transaction (the more steps, the worse performance); ii) *Transaction Revenue* ($transRevenue$), transactions' average price (€) per water unit (*1,000* m^3) (computed as the $wAvePrice$ in Equation 15); and iii) *Market Revenue* ($marketRevenue$), percentage of the actual revenue (i.e. total transacted water quantity times weighted average price) over the maximum possible revenue (i.e. total auctioned quantity times maximum transactions' price).



Config	marketRevenue	sellerPrice
base	76%	76%
low	76%	76%
medium	64%	79%
high	10%	95%

Figure 3: Average *OrgGoal* and *AgSat* values of ten executions

On the other hand, *AgSat* is composed by: i) *buyerQuantity*, percentage of transacted water quantity over the total quantity that buyers aim to acquire (i.e. buyer satisfaction increases with water acquisitions); ii) *buyerPrice*, a price value which indicates the difference of the average of buyers' maximum purchasing price ($rtAg.maxPrice$) with respect to the actual average price ($wAvePrice$) (i.e. buyer satisfaction decreases when the price on the market exceed its inner maximum purchasing price); iii) *sellerQuantity*, the percentage of water quantity actually transacted over the total quantity registered by sellers (i.e. seller satisfaction increases with water sales); iv) *sellerPrice*, the percentage of $wAvePrice$ with respect to the maximum transaction price (i.e. sellers' satisfaction increases when the price on the market gets closer to the maximum historical price).

4.3 Assistance Quality of Service

In order to evaluate our information assistance infrastructure ($AssQoS$ in Eq. 1), we have conducted the experiments defined in section 4.2. Figure 3 shows the averaged results of executing ten times each of the four configurations[10]. The graphic on the left contains thick horizontal bars which correspond to the number of steps required to close a transaction (the inverse of $transPerform$). Thin vertical bars go from the starting prices (responses of each information request) to the weighted average transaction prices ($transRevenue$). The latter becomes the centre of a circle whose area represents both, buyers' and sellers' quantity satisfaction (*buyerQuantity* and *sellerQuantity*)[11]. The area value labels the circle. *buyerPrice* value is represented by a vertical bar just on the right of each circle. It starts in $transRevenue$ and ends in the average buyers' maximum purchasing price, in our case 12. Additionally, the table on the right completes the results by providing the $marketRevenue$ and

[10]For these experiments, we have fixed k to the 10% of the respective $Trac$ value

[11]Note that we have $buyerQuantity = sellerQuantity$ because $rtAg_i.quantity = 100{,}000$ for all buyer and seller agents.

sellerPrice. Since the information services under evaluation target seller agents, we will focus on their satisfaction and the overall system performance.

As we can observe, the values obtained for *low* configuration get a *sellerPrice*, *sellerQuantity*, *marketRevenue* and *transRevenue* that are as high as the *baseline* configuration, with the advantage that they are reached in far less time (see the average number of steps). Thus, we can argue that the *low* information service drastically improves performance ($transPerform$). Accordingly, if we assume that system and agent goals are aligned, a seller agent can use this service to reduce the time of its sales without affecting any of its other satisfaction attributes.

On the other hand, seller pricing strategies may be more aggressive and pursue a higher *sellerPrice*. The results of the *medium* and *high* configurations show us that if the seller agent follows the corresponding services' advice when setting the starting price, then the *sellerPrice* can be increased close (95%) to the maximum sold price. Nevertheless it takes the risk of not being the one who is actually selling at the desired higher price (since the proportion of transactions, *sellerQuantity*, decreases down to 9 and the *marketRevenue* moves down to 10%). Moreover, taking higher risks has the additional positive effects of further improving $transPerform$ and *transRevenue* (i.e. transactions are performed faster and at higher prices).

Overall, we can conclude that these experiments show that system performance and agent satisfaction (and thus, the quality of service) increase with the addition of information services. Furthermore, different services can be useful for decision support processes being carried out by following alternative strategies.

5 Conclusions and future work

In this paper we have formalised both an *Assistance Infrastructure* and an *Information Service* to support agents participation in Open MAS. The *Assistance Infrastructure* is composed by two layers. First, the *Organisational Layer* is composed by a set of *Running Agents*, an extension of an *Organisation Specification* and the addition of the *Organisational Trace*, which keeps historical information of previous organisation executions. Second, the *Assistance Layer* has been populated with one *Personal Assistant* per each *Running Agent*. *Information Services* offers, in addition to basic organisational information, more elaborated information, e. g. specific statistics on data that might be unknown to participants.

In order to illustrate and evaluate our approach we use an OCMAS example scenario that implements an electronic market of water rights. In the tests performed, we have compared the values that different agent satisfaction parameters and system goals take when agents request for different information services, using as a base-line a configuration without enabling assistance services. The experiments show that system performance and agent satisfaction (and thus, the quality of assistance service) increase with the addition of information services. Furthermore, individual agents following alternative strategies can use different services as an useful decision support tool. As future work, we plan to enable and evaluate both the *Os* and *RtOs* information services, the rest of services (advice, justification and estimation) as well as include and evaluate assistance for human participants by means of a 3D Virtual World interface.

References

[1] P. Almajano, M. Lopez-Sanchez, M. Esteva, and I. Rodriguez. An assistance infrastructure for open mas. In *CCIA '11*, volume 232, pages 1–10. IOS Press, 2011.

[2] J. Arcos, M. Esteva, P. Noriega, J. Rodríguez-Aguilar, and C. Sierra. An integrated development environment for electronic institutions. In *Software Agent-Based Applications, Platforms and Development Kits*, pages 121–142. 2005.

[3] A. Bogdanovych, M. Esteva, S. Simoff, C. Sierra, and H. Berger. A methodology for developing multiagent systems as 3d electronic institutions. volume 4951 of *LNCS*, pages 103–117. Springer Berlin / Heidelberg, 2008.

[4] E. Bou, M. López-Sánchez, and J. Rodríguez-Aguilar. Towards self-configuration in autonomic electronic institutions. volume 4386 of *LNCS*, pages 229–244, 2007.

[5] E. Bou, M. López-Sánchez, and J. A. Rodríguez-Aguilar. Autonomic electronic institutions' self-adaptation in heterogeneous agent societies. In *Organized Adaption in Multi-Agent Systems*, volume 5368, pages 18–35. Springer, 2009.

[6] J. Campos, M. Esteva, M. López-Sánchez, J. Morales, and M. Salamó. Organisational adaptation of multi-agent systems in a peer-to-peer scenario. *Computing*, 91:169–215, 2011.

[7] J. Campos, M. López-Sánchez, and M. Esteva. Coordination support in multi-agent systems. In *AAMAS'09*, pages 1301–1302, 2009.

[8] R. Centeno and H. Billhardt. Adaptive regulation of open mas: an incentive mechanism based on online modifications of the environment (extended abstract). In *AAMAS'11*, pages 1243–1244, 2011.

[9] R. Centeno, H. Billhardt, R. Hermoso, and S. Ossowski. Organising mas: a formal model based on organisational mechanisms. In *Proceedings of the 2009 ACM symposium on Applied Computing*, pages 740–746, 2009.

[10] H. Chalupsky, Y. Gil, C. A. Knoblock, K. Lerman, J. Oh, D. V. Pynadath, T. A. Russ, and M. Tambe. Electric elves: Applying agent technology to support human organizations. In *IAAI'01*, pages 51–58. AAAI Press, 2001.

[11] M. Esteva. *Electronic institutions. from specification to development.* PhD thesis, Universitat Politecnica de Catalunya, 2003.

[12] J. Ferber, O. Gutknecht, and F. Michel. From agents to organizations: An organizational view of multi-agent systems. In *Agent-Oriented Software Engineering IV*, volume 2935, pages 443–459. 2004.

[13] A. Garrido, A. Giret, and P. Noriega. mwater: a sandbox for agreement technologies. In *CCIA'09*, pages 252–261, 2009.

[14] J. Hübner, J. Sichman, and O. Boissier. $\mathcal{S}-\mathcal{M}oise^{+}$: A middleware for developing organised multi-agent systems. In *Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems*, volume 3913, pages 64–77. 2006.

[15] N. R. Jennings, K. Sycara, and M. Wooldridge. A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*, 1:7–38, 1998.

[16] I. J. Jureta, M. J. Kollingbaum, S. Faulkner, J. Mylopoulos, and K. Sycara. Requirements-driven contracting for open and norm-regulated multi-agent systems. Technical report, 2007.

[17] J. Oh, F. Meneguzzi, K. Sycara, and T. J. Norman. Prognostic agent assistance for norm-compliant coalition planning. In *ITMAS'11*, pages 126–140, 2011.

[18] S. Okamoto, K. Sycara, and P. Scerri. Personal assistants for human organizations. In V. Dignum, editor, *Organizations in Multi-Agent Systems*. 2009.

[19] T. Trescak, M. Esteva, and I. Rodriguez. Vixee an innovative communication infrastructure for virtual institutions. In *AAMAS'11*, pages 1131–1132, 2011.