A MAS-based Infrastructure for Negotiation and its Application to a Water-Right Market

Bexy Alfonso and Vicente Botti and
Antonio Garrido and Adriana Giret

Abstract This paper presents a MAS-based infrastructure for the specification of a negotiation framework that handles multiple negotiation protocols in a coherent and flexible way. Although it may be used to implement one single type of agreement mechanism, it has been designed in such a way that multiple mechanisms may be available at any given time, to be activated and tailored on demand (on-line) by participating agents. The framework is also generic enough so that new protocols may be easily added. This infrastructure has been successfully used in a case study to implement a simulation tool as a component of a larger framework based on an electronic market of water rights.

Keywords Negotiation model · MAS Infrastructure · Agents · Interactions

1 Introduction

Last decades have witnessed an increasing interest in the design and application of computational infrastructures and tools, based on intelligent agents, to virtual architectures and organizations that give support to multiple ways of negotiation. Negotiation usually involves a dynamic collection of semi-independent autonomous entities (representing heterogenous software agents or humans, departments, industries, information resources and other organizations) each of which has a range of problem solving capabilities and resources at their disposal. These entities exhibit complex behavior; they usually co-exist, collaborate and agree on some computational activity, but sometimes they compete with one another in a ubiquitous virtual scenario that is a sort of 'looking-glass reflection' of the real world.

Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València
{balfonso,vbotti,agarridot,agiret}@dsic.upv.es

Automated negotiation is essential to undertake complex behavior and architectures, including conflict identification, its management and resolution, search for consensus, assessment of agreement stability and equilibrium analysis in situations where two or more parties have opposing preferences [19]. This line of research has addressed developments for group decision support systems and meeting support systems, which can be extrapolated to automated negotiation [9,11]. Therefore, negotiation, in itself, is interesting from an application point of view but also to provide artifacts that facilitate the design, experimentation and simulation of involving agreements. In this paper we intend to profit from that experience and look at one of such artifacts: a generic negotiation MAS-based framework in which different negotiation protocols may become available. The contributions of this general framework are multiple. i) As it is defined for the Magentix2 [2] platform for open MASs, it embodies easy communication and interaction protocols among agents, roles and organizations. It also uses Jason [6] as a high-level language for programming agents, providing them with high reasoning skills. ii) Interactions among agents aim at achieving both individual and global goals, and are structured via collaboration, argumentation, negotiation and, eventually, via agreements and contracts [25]. iii) It is composed of flexible negotiation mechanisms and their supporting preparatory and ending activities. iv) As a by-product, it creates standardized negotiation modules to be grafted into larger scenarios or as plug-ins in peer to peer interactions. v) It has been used as a proof of concept in *mWater* [7, 15], a water-right market where negotiation is essential, also embedded in a decision support system where water usage is subject to conflicts whose solution may involve different types of negotiation. vi) It provides new areas of opportunities for an agreement computing solution [25], including agility, heterogeneity, reconfigurability, cooperation, argumentation, reputation and trust issues under a MAS perspective.

The paper is organized as follows. Sect. 2 introduces some related work on interaction and negotiation. In Sect. 3, a technological background is given by briefly mentioning the characteristics of the Multi-agent Platform (MAP) used to implement the generic model, and also the characteristics of the agents programming language. In Sect. 4 we present the generic negotiation model; the negotiation workflow structure is described and also the roles participating in its interactions. In Sect. 5 we describe the simulation tool, used as a case study, for implementing an electronic market of water rights. Sect. 6 shows some practical guidelines to adapt the negotiation model to particular applications. Further uses for the negotiation model are commented in Sect. 7 from the academia an industry standpoint. Finally, Sect. 8 concludes the paper, highlighting our main technical contributions.

2 Related Work

Computing has become an inherently social activity rather than a solitary one, leading to new forms of conceiving computational systems which require both

interaction and negotiation. Some proposals have been effectively developed in literature to implement a negotiation framework. That is the case of the Jade platform [1,5], which is a FIPA compliant platform that provides Java classes to handle all the FIPA interaction protocols. In this sense, the agents' interactions must be also programmed in Java by using the constructions provided by the platform, which in some cases may be a limitation. Another multi-agent platform with support for interaction protocols is Jadex [8,23]. Jadex follows a typical BDI model and can be executed alone or under other communication platforms using *adapters*. A Jadex agent is defined through an XML file and the Java classes that implement it. Jadex also owns the 'interaction protocols' capability, offering built-in support for most of the FIPA interaction protocols. However, both Jade and Jadex use Java classes for implementing FIPA interaction protocols, so the programmer can not use other specialized programming languages, such as AgentSpeak, more expressive to model and describe agents. This does not prevent us from addressing the problem using the Java approach; in fact, so far it has been broadly used. However, in MASs, it is desirable to use tools and languages that better fit with the autonomous and proactive agents' nature. In this sense, the MAS platform known as Magentix2 [2], further described in Section 3, supports a high-level language for programming conversational agents (i.e. agents whose interactions respond to interaction protocols) and the remaining of the capabilities offered by similar platforms. It also owns a conversations manager that stores and automatically adds the information required in the creation of messages during the conversation. Moreover, with Magentix2 it is possible to dynamically modify the sequence of steps in the interaction protocol in order to create more open and flexible conversations (new states and transitions between the conversation steps can be created at execution time). These features have been partially included in other platforms, whereas all of them are included in Magentix2 by default, which makes it an ideal infrastructure for a negotiation architecture.

From our point of view, the common denominator in most of the current real social systems is, interestingly, a negotiation process. Although some works have proposed the construction of formal conceptual models with some negotiation [10,24], they do not always report significant advances from a collaborative AI perspective. This is one of the main aims of this paper.

3 Technological Background

There are various technologies involved in the implementation of our MAS infrastructure. First, the MAS platform in itself, which manages agents and their interactions, allowing the information exchange among them and also with the environment. Second, a language to define the agents behavior —in this case Jason, which follows the agents' BDI model. Third, in order to support the human-software agents' interactions it is necessary to design a Graphical User Interface (GUI) and an artifact to orchestrate the communication between this GUI and the MAS.

3.1 MAS Platform

In our approach we use Magentix2 [2] as the MAS platform for implementing our simulation tool. It is FIPA compliant and it uses a middle-ware based on AMQP messages (Advanced Message Queuing Protocol)[1]. This allows heterogeneous entities to interact through FIPA-ACL messages. Specifically, what is used in Magentix2 is the Apache Qpid[2] implementation of AMQP. This way the agents use Qpid client APIs to connect to a Qpid broker and communicate with other agents in Internet. Magentix2 has been chosen as our MAS platform mainly because: i) it provides powerful techniques to facilitate agents' communication; ii) it supports interactions protocols between agents organizations/societies through conversations management; iii) it allows the use of high-level reasoning structures when programming the agents; and iv) it includes security issues for distributed systems, so it offers a dynamic and flexible model for complex systems. In short, Magentix2 gives us support at the three levels stated in [21]: organization level, interactions level and agent level.

Conversations Factory: an Artifact for Communication

A Conversations Factory [12] is mainly a Magentix2 mechanism to support FIPA interaction protocols [13]. Each conversations factory allows us to keep a complete interaction among two or more agents having an *initiator* (the one who starts the conversation) and one or more *participants* (the other agents in the conversation). The two main structures supporting conversations are *CProcessor* and *CFactory*. The former manages the sent/received messages in each step of the conversation, performing the corresponding actions, and determines the next step in the conversation. The latter creates the conversations and the *CProcessors* that correspond to a specific protocol. If the agent is playing the role of *initiator*, the conversation can start without needing an external event. On the other hand, if the agent is a *participant* an event is required for it to be part of the conversation.

3.2 Programming language

Magentix2 allows us to use a high-level language for programming agents. In this case it is Jason [6], which is an extension of the AgentSpeak language. AgentSpeak allows us to define agents in terms of beliefs, goals and plans. Beliefs represent the vision of each agent of the current state of the world in which such an agent is situated. Beliefs change frequently due to a 'perception' of the agent over its environment, because some information has been sent to it through a message, or because it explicitly modifies those beliefs as a consequence of some previous reasoning. Agents' goals represent the agents'

---

[1] AMQP: http://www.amqp.org/
[2] Apache Qpid: http://qpid.apache.org/

intentions to reach a state where they believe the goals are true, what is called 'achievement goal'. Another kind of goal is satisfied when the agents retrieve updated information from their belief base: 'test goals'. Finally, plans are just a sequence of steps that allow agents to reach some goals. The fact of adding a goal acts as a triggering event for executing the corresponding planned sequence of actions. There are other actions that act as triggering events for plan executions as it is the case of the deletion of achievement goals, adding and deletion of beliefs, and adding or deletion of test goals. If this sequence of actions does not fail, the goal is successfully reached.

Jason provides a kind of action called 'internal action'. It is a structure that allows us the execution of legacy code (Java in this case). Thanks to this, the agent has access to the structures provided by the platform [3] in order to make use of the conversations factory in a more simplified way. By using some of the Magentix2 predefined internal actions, each agent can customize what it does in those steps of the conversations on which it needs to perform some 'reasoning', delegating details such as synchronization, timeouts, errors management, etc. in the platform. Magentix2 is also responsible for updating the state of each agent when it is necessary during the conversation (by updating its beliefs) for it to make decisions, which behaves as an indirect communication.

4 Our Generic Negotiation Model

The infrastructure for a generic negotiation model can be seen as a set of entities and roles regulated by mechanisms of social order, and created in order to negotiate with some good, service or resource.

4.1 Main Structure

Our negotiation model follows a MAS specification based on conversations, and regulation on (structural) norms. It is defined as a generic organization for negotiation (see Fig. 1)[3], where any participating agent may become involved in a negotiation process.

After admission is granted, each negotiation involves first a preliminary process of invitation and filtering of parties, then the negotiation process itself and, finally, some form of settlement process through which the agreements among participants are made explicit and, if appropriate, communicated to the organization.

---

[3] At a glance, each interaction/conversation represents an atomic process and/or dialog among agents; a workflow represents complex interaction models and procedural prescriptions. The dynamic execution is modeled through arcs and transitions, by which the different participating roles of the organization may navigate.
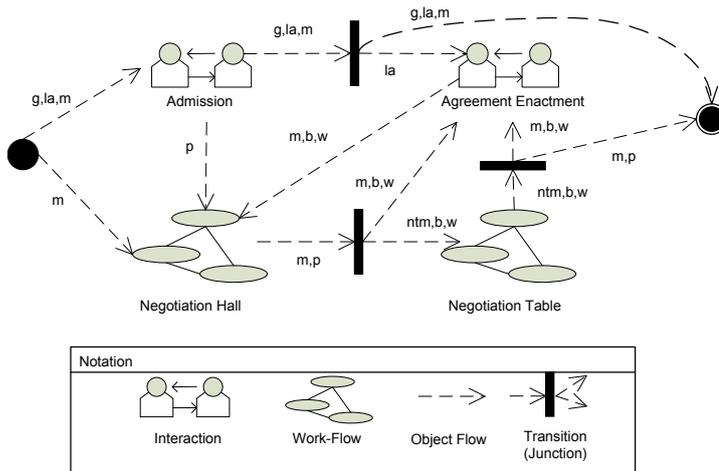
Fig. 1 Generic Negotiation workflow structure. Roles: g - guest; p - participant; b - black; w - white; m - mediator; ntm - negotiation table manager; la - legal authority.

4.2 Users and Roles

There are seven roles that interact in the model, as depicted in Fig. 1. A guest role (g) is a user that wants to enter the negotiation. The guest may be specialized into a real participant (p), and furthermore as black (b) and white (w) to differentiate the parties that are acting in a given negotiation. Finally, there are four types of staff roles. The mediator role (m) represents a negotiation facilitator agent who runs standard activities, such as managing the specific parameters of the negotiation protocols. The negotiation table manager role (ntm) represents a facilitator agent who executes activities that are specific of a given negotiation protocol, for example accept valid negotiators, tune negotiation parameters of the table, mediate in the negotiation or conflict resolution process, expel negotiators, etc. The legal authority role (la) represents an agent who is in charge of activities for agreement enactments that are executed as a result of a successful negotiation process.

The purpose of facilitators. The **mediator** *and* **negotiation table manager**

Note that, unlike other approaches, our definition introduces an explicit intelligent management into the negotiation model in the form of the mediator and negotiation table manager. Although there exist some negotiation frameworks where both parties (black and white) interact one to another with no need of a mediator (e.g. face-to-face protocol), it is clear that a mediator shows very helpful to improve and facilitate the internal behavior of the system, thus making it more appealing for further complex extensions. In particular, these

two roles demonstrate to be very helpful to improve and facilitate the internal behavior of the organization.

On the one hand, the mediator must be aware of the organizational conventions, the rules and mainly the negotiation infrastructure: the initial conditions of the negotiation, the particularities of the invitations, i.e. who is going to be invited to each negotiation and under which circumstances, and the deal to be negotiated. But more importantly, this mediator offers intelligent capabilities to help the users under two basic scenarios: i) to decide about opening a new negotiation table, and ii) to decide what user is going to be invited to join that table and why (preliminary process of invitation). Deciding whether opening a new negotiation or not seems easy, but it involves subtle details. First, the mediator must be aware of the current context of application that may allow or forbid the opening of a particular type of negotiation process. Second, the mediator needs to take into consideration the user's reputation and/or trust properties to decide about the opening [18, 22]. According to all this information, the mediator decides about accepting or declining the offer to open the new negotiation. Currently, our schema for opening is based on a yes/no answer. However, a more interesting implementation includes a pre-process advisory stage to decide which is the most adequate one. In that case, the mediator may advise the party about the best procedure to proceed (based on learnt behavior and/or conducts), the main issues of the deal (such as price, min quality criteria or max deadline), or type of negotiation to be opened. Intuitively, this comprises a more complex mediator's interaction that requires argumentation and negotiation techniques and provides more opportunities to apply agreement technologies. Moreover, the mediator also needs an intelligent behavior to send invitations to join. This means assigning a priority to each user for being invited in terms of such a priority. More specifically, the mediator sends invitations to join the table by using data mining rankings that assign a priority to each user for being invited to each table —this involves an (intelligent) deliberative process based on the user's reputation, trust or particular attributes. This is particularly relevant when there exists a constraint about the max number of users allowed. Note that this is not a simple *filtering condition* as it usually involves a process based on the user's features. Finally, the mediator decides the deadline to accept users' confirmations after the invitation message; for some users the deadline is tight but for others it may be looser. Analogously, the mediator may offer advice during the conflict resolution process, thus making it more efficient.

On the other hand, the negotiation table manager must obey the particular rules of the protocol to be used within the negotiation, and this is usually domain-dependant —different protocols require the application of different sequences of steps. Again, this

stage requires an intelligent behavior of the manager. First of all, this manager must decide the users that take part in the negotiation. In some negotiation tables, only invited parties are allowed, but in others any party may enter. The role of the manager here is to decide whether a party is accepted or not. As commented above, this usually entails a deliberative process that relies on the user's reputation or any elaborated attributes. Further, the table manager must provide this type of intelligence both before accepting any party to the table and throughout it —in some scenarios the manager may decide to expel a party if some conditions do not fulfill. Secondly, the table manager requires some intelligence [17, 20] to make the protocol more agile and/or to converge more rapidly. For instance, let us consider an argumentation-based negotiation or, simply, a double auction exchange, where the role of the mediator is essential for performing a successful negotiation.

All in all, we can conclude that having facilitators with intelligent behavior is fundamental in a generic approach as it allows us to plug-in domain-dependent knowledge rules easily to deal with particular dynamic domains. This does not only provides more expressive and realistic negotiation models, but it also improves efficiency.

4.3 Workflow

The workflow activities in the generic negotiation model of Fig. 1 are specified through a main structure which includes two other workflows: the *NegotiationHall* and *NegotiationTables*, plus two supporting interactions, *Admission* and *AgreementEnactment*.

Admission. It allows Guest agents to register to become a Party, and to 'jump start' a negotiation process. Once negotiation is open, this interaction allows Party agents to enter and negotiate by registering individual data for management and enforcement purposes (these data are domain-dependent and can be used, for example, for enforcing particular conventions and managing activities).

NegotiationHall. Actual negotiation starts here (see Fig. 2), where Party agents become aware of any activity and/or initiate concurrent activities for negotiation. There are three interactions that provide virtual scenarios for the: i) creation of, and invitation to, negotiation tables (*NTC*); ii) exchange of information about active agreements and ongoing negotiation tables (*IE*); and finally, iii) execution of specific activities in case of an anomalous/critical situation (*CS*).

Negotiation Tables are created in two ways: i) by the organization itself, for example periodic negotiation tables about a set of issues, or ii) initiated on-demand by a participating agent. The negotiation tables are created in
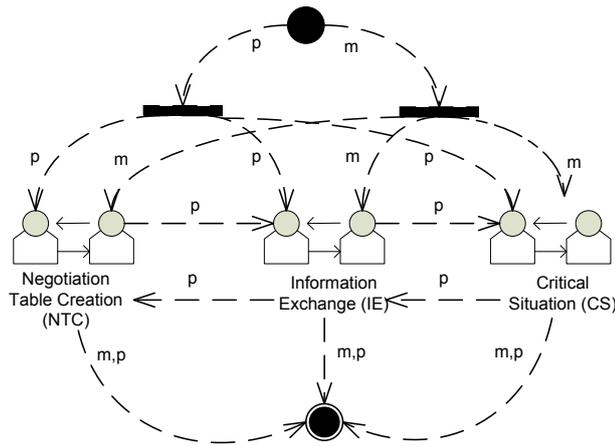
Fig. 2 *Negotiation Hall* workflow structure.
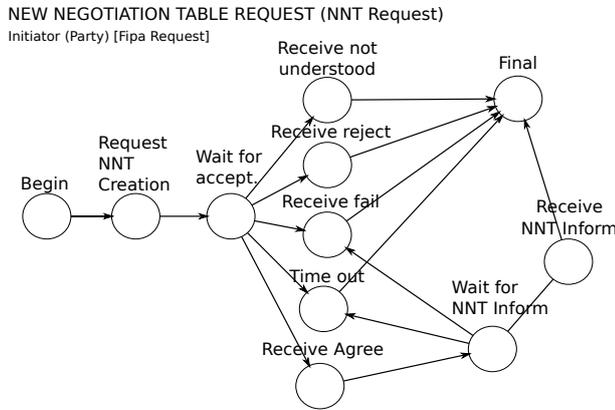

Fig. 3 Party's behavior for requesting a New Negotiation Table.

the *NTC* interaction, which responds to the *FIPA request* standard protocol
[13]. Fig. 3 and 4 show the steps of the protocol from the Party's perspective
(*initiator*) and from the Mediator's perspective (*participant*), respectively. It
issues the following illocution:

$request(p_x, m, open, protocol(params), \delta, pt, at, C)$, where the semantic is
as follows. Party agent p_x requests (see Fig. 3) to the Mediator, m, to *open*
a negotiation table with a given negotiation *protocol*. This protocol is instan-
tiated with the set of values for the parameters *params*. The table is created
to negotiate about a deal δ. The requesting party, p_x, will participate as pt
that can take one of these values: p, that is an observer Party; a Black party
b; or a White party w. at is the access type that can be *Public*, anybody can
be invited; or *Private*, only Party agents that fulfill the set of constraints C
can participate in the negotiation table.

NEW NEGOTIATION TABLE REQUEST (NNT Request)
Participant (Mediator) [Fipa Request]
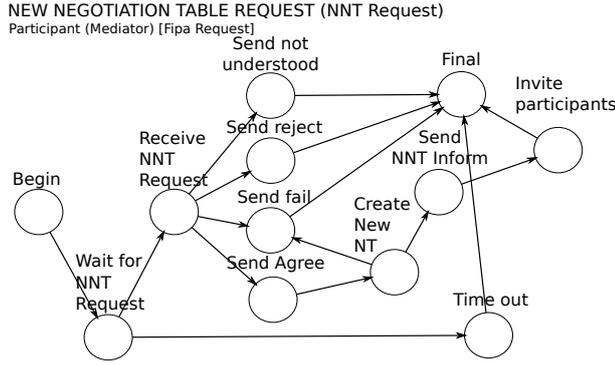
Fig. 4 Mediator's behavior during the conversation for Opening a New Table.

When the Mediator, m, receives a request to open a negotiation table (see Fig. 4), it instantiates a new Negotiation Table scenario with the requested negotiation protocol, for example a standard double auction, a face-to-face negotiation, a blind double auction, etc., and the given parameters. Moreover, a Negotiation Table Manager, ntm, is created to manage the execution of the negotiation table. Next, m issues an information illocution to the p_x agent who requested the table.

$inform(m, p_x, table_{ID}, error)$, where $table_{ID}$ is the ID of the new table if it was successfully created, or a *null* value when the table can not be created due to *error* conditions.

In order to complete the negotiation table creation, the Mediator needs to invite other Party agents to the new negotiation table. When the created negotiation table has a *Public* type of access, the m broadcasts an invitation message to all the participants:

$inviteAll(m, table_{ID}, protocol, \delta, C)$; in other words, the invitation message states the $table_{ID}$ of the negotiation table that is receiving players; the negotiation protocol *protocol* used in that table; the set of issues, δ, that is being negotiated; and the set of constraints, C, to participate in are also made public.

On the other hand, if the created negotiation table has a *Private* type of access, the m has to select first the set of possible candidates to invite, say $P_{table_{ID}}$, and then send an invitation message to every such candidate:

$invite(m, p_y, table_{ID}, protocol, \delta, C)$, where each candidate $p_y \in P_{table_{ID}}$.

NegotiationTable. It is organized in a flexible and scalable fashion in order to easily include new negotiation protocols. Each instance of a *Negotiation Table* interaction is managed by a *Negotiation Table Manager, ntm*, who knows the structure, specific data and management protocol of the given negotiation protocol. The framework provides pre-defined protocols such as face-to-face, Dutch auction, English auction, standard double auction, closed bid envelope, blind double auction with mediator, among others. Nevertheless, new negotiation protocols may be easily added provided that the new definition complies with the generic structure.
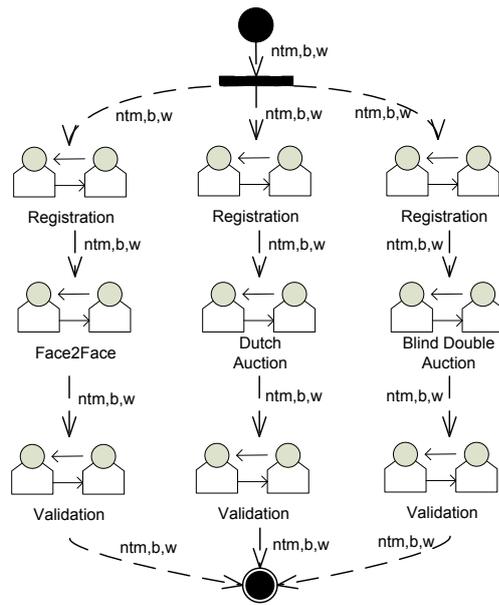
Fig. 5 *Negotiation Table* workflow structure.

Every generic negotiation table is defined as a three interaction structure (see Fig. 5). The first interaction is *Registration*, in which the *ntm* applies a filtering process to assure that only valid agents can enter a given negotiation table (recall situations when a private negotiation table is executing or only a sub-group of Party agents that fulfill a set of constraints may participate in the table). The specific filtering process will depend on the given negotiation protocol and possibly on domain specific features. The second interaction is the negotiation protocol, in which the set of steps of the given protocol are specified (see bellow for a sample negotiation protocol specification). Finally, in the last interaction, *Validation*, a set of closing activities are executed, for example registering the final deals, stating the following steps for the agreement settlement, verifying that the leaving party satisfies the leaving norms of the negotiation table, etc. The set of activities to be executed in this interaction is domain specific and will also depend on the given negotiation protocol.

AgreementEnactment. Once an agreement has been successfully reached, it is settled here according to the given conventions. This may be a rather elaborate process. First of all, the Mediator checks whether or not the agreement satisfies some formal conditions. If the agreement complies with these, a transfer contract is agreed upon and signed by the Party agents involved, and then the agreement becomes active. Once an agreement is active it may be executed and, consequently, other Party agents may initiate a grievance procedure that may overturn or modify the agreement. Even if there are no grievances that modify a contract, parties may not fulfill the contract properly and there might

be some contract reparation actions. If things proceed smoothly, the agreement
subsists until maturity.

5 Case Study: *mWater*, a Water-Right Market

5.1 *mWater* Overall Description

Water scarcity is a major concern in most countries, not only because it threatens the economic viability of current agricultural practices, but because it is likely to alter an already precarious balance among its different types of use. It has been sufficiently argued that more efficient uses of water may be achieved within an institutional framework where water rights may be negotiated more freely under different market conditions [26]. In hydrological terms, a water market can be defined as an institutional, decentralized framework where users with water rights (right holders) are allowed to voluntarily trade them, always fulfilling some pre-established norms, to other users in exchange of some compensation, economic or not [16,26]. Water-right markets allow rapid changes in allocation in response to changing demands for water and stimulate investment and employment. Because of water's unique characteristics, such markets do not work everywhere, they are not homogenous, nor do they solve all water-related issues [26]. Also, even subtle changes in the market design (allowed participants, legislation, protocols, etc.) are very costly and difficult to evaluate.

The functioning of a water-right market relies on the willingness of users to buy or sell water, which depends on the difference between the price of water and net revenue each users expects to earn by (usually) irrigating. Thus, for a given price of irrigation water, a farmer would be willing to purchase water if (s)he expects a unit of water to generate more incomes than it costs. If another farmer expects a unit of water to earn less than (s)he could sell it for, (s)he might want to sell it thus originating the trading process. But it is not always a matter of price expectations, but also of regulation, seasons and time horizons. This means that there is not a unique model of water-right market, as it usually depends on many individual, social and organizational factors, such as the price, type of users, negotiation protocols, and basin, region or country laws.

In short, the negotiation process in a given market can be initiated on demand (by the buyer or seller) or under a periodic basis (once per month, semester, season, etc.) The time horizon is also flexible and can vary from one negotiation to another, although it usually does not extend for more than one year. The negotiation strategy is also variable, although simple protocols are usually preferred. The same user can participate in more than one negotiation, which means that one buyer can potentially buy rights to more than one seller and vice versa; but what regulation usually does not per-

mit is to speculate on the rights. The emphasis on regulatory aspects is motivated by the fact that the main objective policy makers have in mind is to achieve an adequate behavior of users to ensure the success of the market. And regulation is the main tool that policy makers have to modify behavior by means of: i) government laws, ii) basin or local norms, and iii) social norms. However, in practice, users are prone to achieve "order without law" or, at least, to preserve their practices within the established regulation, whereas policy makers adapt regulation to guide users in a constantly changing environmental and political media.

mWater is a particular instance of the MAS infrastructure for negotiation presented above, and it is used as a simulation tool for What-If Analysis of water-right markets policies [7,15], trying to be as flexible as possible. More specifically, *mWater* assists in designing appropriate water laws and regulate, either privately or publicly, the users' actions, interactions and their eventual trade.

5.2 *mWater* as a Simulation Tool

mWater builds on a MAS infrastructure, simulates a flexible water-right market, and includes its own ontology for dealing with water issues and both the trading and grievance processes. We have focused our model on humans' actions: agents are the crucial component in these models and our interest relies on the social aspect of the market, which is usually missing in other markets in the literature [4,16]. This simulator includes heterogeneous and autonomous intelligent agents representing the different independent entities in the market. We focus on demands and, in particular, on the type of regulatory (in terms of norms selection and agents behavior), and market mechanisms that foster an efficient use of water while also trying to prevent conflicts among parties. In this scenario, this system plays a vital role as it allows us to define different norms, agents behavior and roles, and assess their impact without jeopardizing the real-world market, thus enhancing the quality and applicability of its results as a decision support tool.

The user can configure simulation parameters such as: the group of water-users that will participate in the market[4], the norms and regulations that define the policies in the market, the negotiation protocol to be used[5], the seasons in which the water-right transfer will take place, etc. The simulation

---

[4] It is important to point out that the simulation we have developed is a mixed-initiative simulation in which there are software agents that are completely autonomous/automated and other software agents that are simple interfaces for human users. In this way, it is very easy to include complex social behavior that are hard to implement or highly time consuming.

[5] We are currently interested in checking the viability of our approach, rather than in providing a huge range of protocols. Consequently, it only implements the Japanese auction protocol. The implementation of other auction and negotiation protocols is part of our future work.

tool executes with a given configuration and the user can assess the market's behavior by means of indicators such as: number of water-right transfer agreements, volume of water transferred, amount of money, overall social satisfaction of the water-users that participated in the market, number of conflicts generated, etc.

5.3 Some Notes about Additional Intelligent Capabilities for *mWater*

In addition to the general intelligent capabilities of the facilitators discussed in Sect. 4.2, in *mWater* we have included two more domain-dependent intelligent features.

First, guiding an agent when navigating through the workflow structures. For instance, if we consider an agent that starts from the initial state and wants to reach the agreement enactment node of Fig. 1, it is now possible to provide him/her a plan (i.e. sequence of transitions) to reach such a node. This is an intelligent planning capability that can be embedded into the existing staff agents of the system or, like in our case, as additional behavior in a new *navigation-provider agent*. Similarly, inner and probably shorter plans are also possible; that is, an agent that has been admitted and simply wants to reach and agreement.

Second, we have implemented a planning constraint-based formulation within the mediator role, based upon the work of [14], which focuses on the negotiation process in terms of finding the best plan (optimization behavior) to reach the agent's goals for water and/or money. More particulary, if an agent that is trading is interested in buying some water rights, the plan will provide here the best users to trade with, i.e. those users to sell these water rights and the quantity of water to be transferred from the sellers to the buyer.

The main advantage of using planning and a constraint-based programming formulation is that we can easily integrate both the navigation through *mWater* and the optimization in the negotiation tables as a unique model in one centralized agent or in a distributed way through several staff agents. Hence, the plan will provide the actions (in form of transitions) necessary to achieve a goal, such as reaching an agreement. If this plan requires visiting the negotiation tables and a particular protocol to trade water rights, the plan will also provide the best combination of trading actions.

5.4 *mWater* in Action

Fig. 6 shows a snapshot of the *mWater* simulator in action. This interface allows the user, i.e. the water policy maker, to choose different input values

that involve simulation dates, participants, norms (in the form of protocols used during the trading negotiation) and some decision points that can affect the behavior of the participants[6].

---

[6] In our current implementation, these additional decision points rely on a random basis, but we want to extend them to include other issues such as short-term planning, trust, argumentation and ethical values.
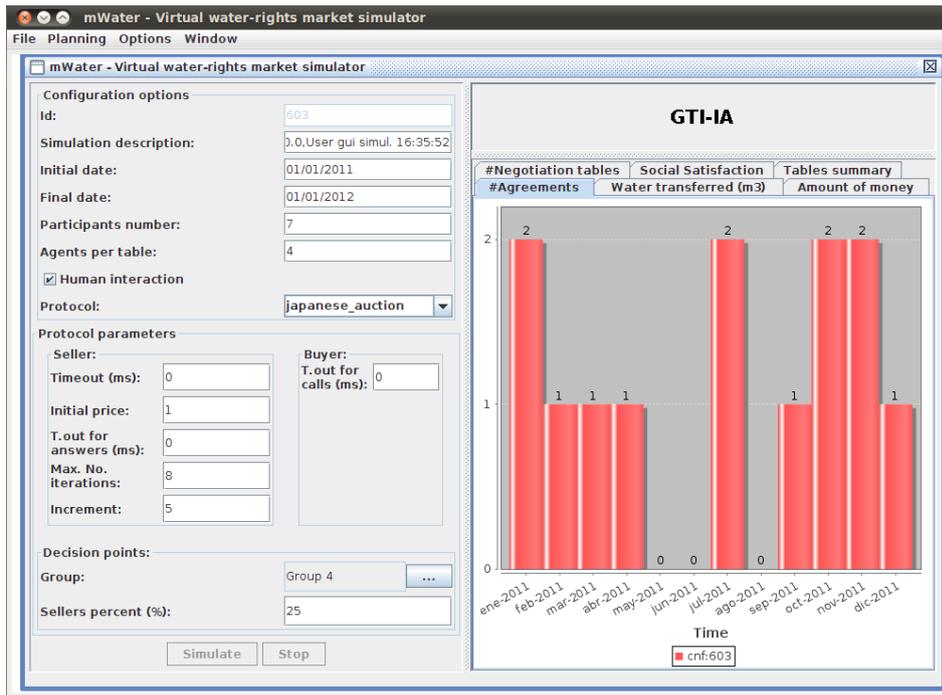
(a) An arbitrary simulation.
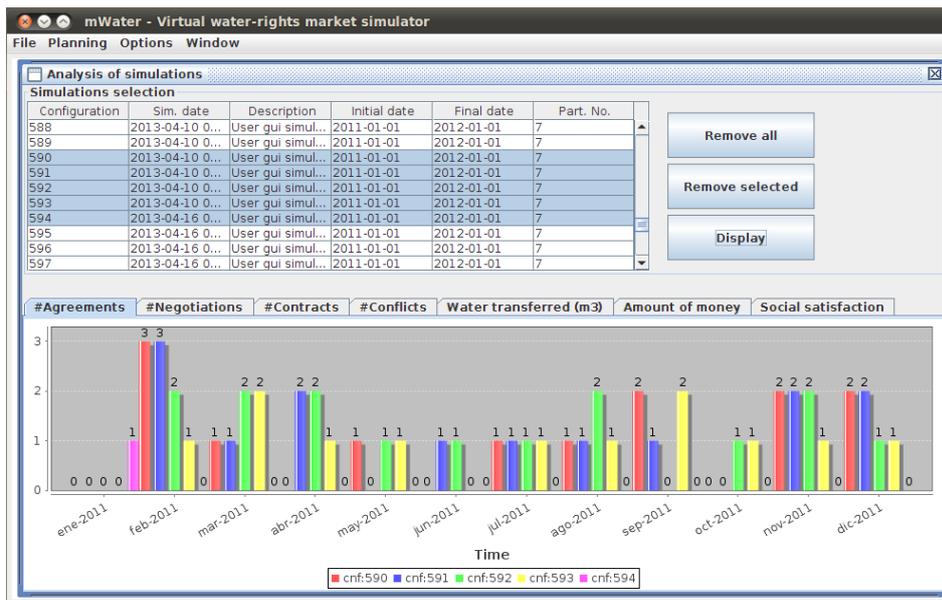
(b) Results of various simulations.

Fig. 6 Snapshots of the *mWater* simulator.
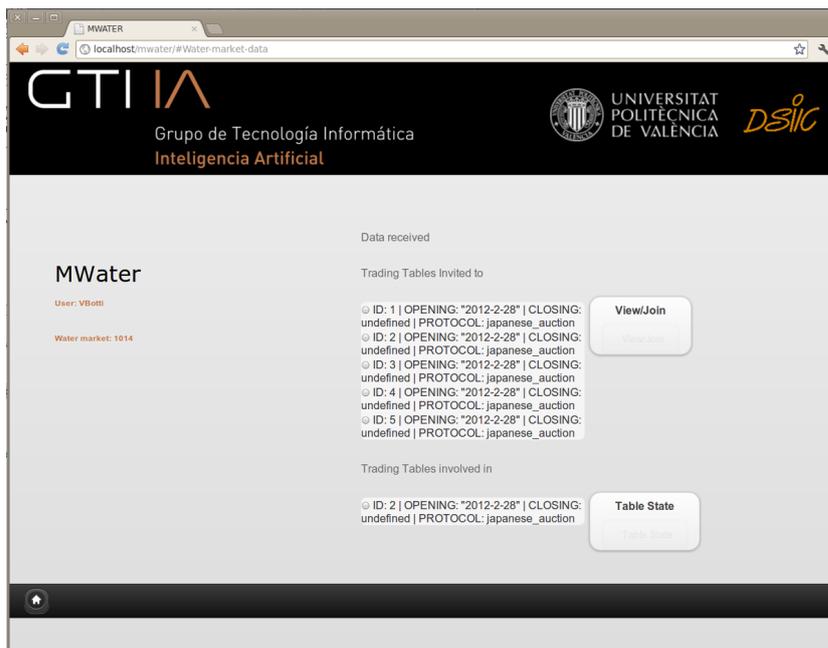
Fig. 7 Snapshot of the human-agents GUI. It shows the trading tables that the human has been invited to and that is currently involved in.

In the *mWater* simulator we want to have a tool for studying different behavior and situations and to allow an hybrid simulation in which software agents can interact with human users as well as with software agents. To do this it is necessary to create some GUIs with the required options for the human to make changes in the system and submit information to other agents at execution time. For this reason we implemented a Web page, with PHP as scripting language, and an interface application to submit all the requests from the Web page to the MAS, and all the results from the MAS to the Web page. Fig. 7 shows the state of the trading hall for this specific user by listing the trading tables he has been invited to participate and the trading tables he is involved in, either for being it's owner or for being currently participating in it. On the other hand, Fig. 8 shows how a user can participate in a Japanese Auction of a water right, by interacting with other human or automated agents.

In order to allow a human being to participate in the simulation like any other (software) agent of the system, an agent is automatically created that represents him/her and translates his/her actions in actions of an agent in the system. So the participation of the human ultimately does not modify the simulation performance in a high degree, because it behaves as yet another agent. The main difference is that, when a decision must be taken, it is necessary a feedback from the human user for resuming the simulation.

Nevertheless, since humans' actions are significantly slower than agents' actions, it is necessary to introduce some timeouts that would not be necessary without the human participation, in order to allow him/her to take decisions in a reasonable time. For this reason, the simulation time is almost determined by the human's speed in performing actions (like joining a trading table, accepting or declining a bid, etc.). In order to inform the human how much time (s)he has still left for acting, it was necessary to include some informative elements in the GUI.

As mentioned above, this capability allows us to include into a given simulation specific user populations with very complex behavior, which can be hard to implement or make its development almost prohibitive. Integrating this human behavior into the simulation allows us to analyze new situations without the need of changing the basic reasoning of the agents (e.g. to analyze how the new user's profile influences the decisions of the other users and how this affect the final market statistics).

This simulation tool allows users to analyze: i) how the conventions and negotiation protocols of the market change over time; ii) how participants in these markets (re)act to these changes; and ii) how to extrapolate the empirical outcomes of the market, in terms of economic and environmental impact, to deal with the social (welfare) aspect of the market. Our preliminary experiments shed light on the benefits that a collaborative AI perspective may bring to the policy makers, general public and public administrators.

From the experts' point of view and their advice, we can conclude that a model+tool like this provides nice advantages: i) it successfully incorporates the model for concepts on water regulation, water institutions and individual behavior of water users; ii) it formally represents the multiple interactions between regulations, institutions and individuals; iii) it puts strong emphasis on user participation in decision making; and iv) it finally provides a promising tool to evaluate changes in current legislation, and at no cost, which will surely help to build a more efficient water market with more dynamic norms. Note, that the simulator is mainly policy-maker-oriented rather than stakeholder-oriented. It focuses on the possibility of changing the norms within the market and evaluate their outcomes —which is the policy makers' labor—, but not in the participation of stakeholders to change the model of the market itself. On the contrary, the human-agents GUI is fully stakeholder-oriented, where stakeholders themselves use the system and interact constantly following the negotiation strategies. Although having real stakeholders to trade real water rights is very difficult because we require special permission from basin and local authorities, this will help validate the results, which is part of our future work. All in all, from the experts' evaluation we conclude that a tool like this
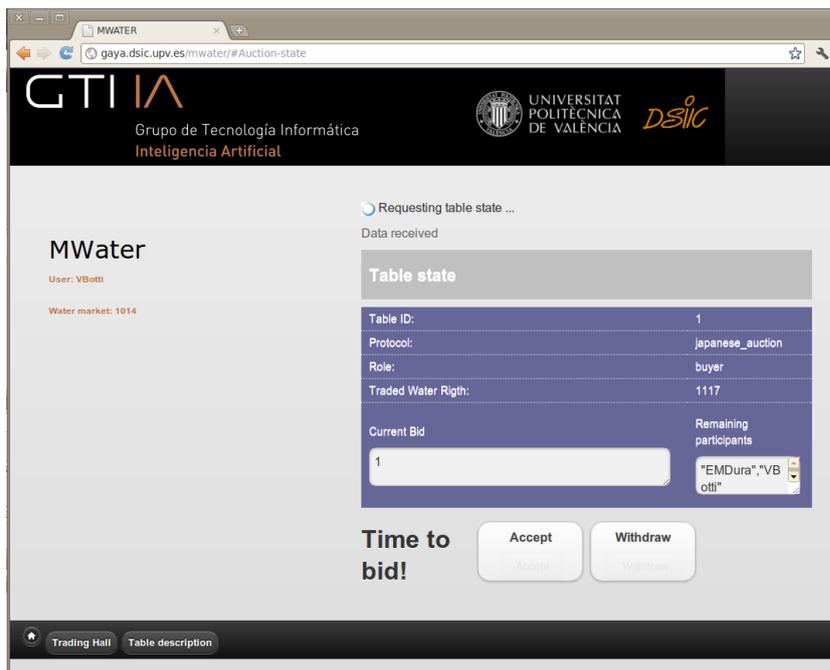
Fig. 8 Snapshot of the human-agents GUI. The user can participate in a Japanese auction with other humans and/or automated agents.

provides an advantageous approach to help build a more efficient water market with more dynamic norms.

5.5 Case Study Performance Tests and Experimental Results

In order to test the performance of the system, according to the proposed structure, we designed a set of simulation tests varying some parameters. In this particular implementation of the Case Study there is an agent called "staff" (this agent implements and re-unite the roles: mediator, negotiation table manager and legal authority of the general model) who interacts with the database and makes all the necessary queries and updates. The database manager, the Qpid broker and the *staff* agent are all running in the same computer. The rest of the agents may run in this same computer or in other computers depending on the test performed.

We used 7 computers for performing the tests. One of them stores the database, runs both the *staff* agent and the Qpid broker; it is an Intel(R) Core(TM) 2 Duo @ 3.16 GHz, 4 GB of RAM memory and it runs the Ubuntu 10.04 LTS Linux operating system with kernel 2.6.32-31. The other computers have an Intel(R) Core(TM) 2 Duo

@ 2.60 GHz processor and 2 Gb of RAM. The other characteristics are listed in Table 1.

PC	OS	JDK version	Linux kernel
1	Ubuntu 11.10	JDK 1.7	3.0.0-15
2	Ubuntu 11.04	JDK 1.7	2.6.38-11
3	Ubuntu 11.04	JDK 1.7	2.6.38-16
4	Ubuntu 11.10	JDK 1.7	3.0.0-15
5	Ubuntu 8.10	JDK 1.7	2.6.27-17
6	Ubuntu 11.10	JDK 1.7	3.0.0-15

Table 1 PCs' technical description.

5.5.1 Experiments

In the experiments performed our main goal is to assess the performance of the system when agents negotiate. The kind of negotiation performed follows a Japanese auction protocol. This kind of auction starts with an initial bid proposed by the table's owner. Each participant can accept or reject the proposal. In each iteration the bid is always incremented in the same quantity, and it may finish when:

- There is only one participant that agrees with the proposal, or
- Nobody agrees with the proposal, or
- The maximum number of iterations is reached.

The parameters used for all the experiments, to facilitate the comparisons, where:

1. *Initiator:* 4 seconds for waiting for participants to join the auction and 15 seconds as the time limit for waiting acceptances in each iteration.
2. *Participants:* up to 50 seconds for waiting for initiator bid calls.
3. *Protocol:* 8 iterations at most; 1 as the initial bid and with increments of 5 in each round.

4. *Buying likelihood:* each agent decides whether to accept a bid or not according to its buying likelihood. In these tests the same value of 0.5 was used for all agents, so generally, in each round, around half of the agents will accept a bid and the other half will not.

According to this, the computation time must be determined by the fixed time given by the parameter 1 and the time derived from the data base access and the amount of messages sent. This,

Experim No.	Tables No.	Total No. of agents	No. of PCs	Max agents per table	Max agents per table & PC
1	1	10,15,...,450	1	10,15,...,450	10,15,...,450
2	1	70,140,210, 280,350	7	70,140,210, 280,350	10,20,...,50
3	7	70,105,...,560	7	10,15,...,80	10,15,...,80

Table 2 Combinations for the parameters values.

in turn, is determined by the number of iterations performed and the number of participants.

The experiments take as parameters: the maximum number of agents per trading table, the maximum number of agents per trading table and computer, the number of trading tables and the number of computers. In all cases, the number of agents per table remains constant. The combinations of the values used for these parameters are listed in Table 2.

5.5.2 Analysis of the results

The results obtained from evaluating the average time for negotiation according to the parameters of Table 2 are summarized in Figs. 9 and 10. Fig. 9 shows the average time for negotiating when there is one table and **7** tables in the system and the negotiation is taking place in one PC. It is important to point out that we used **7** tables in order to have comparable results with those of the tests performed later with **7** tables distributed in the **7** computers available. In both cases the time gets higher when incrementing the number of participant agents. Fig. 9(b) shows better times for the same amount of agents than Fig. 9(a). This makes sense considering that **7** negotiations, taking place simultaneously with fewer agents per table, produces less overhead that only one table with more agents. The big difference for 450 agents in a system with just one table in one computer and for 490 agents in a computer with **7** negotiation tables is produced by limitation in the version of Magentix2 used in this Case Study when managing a high number of objects.

In addition to scalability, we also wanted to test the influence of distribution for the same experiments described before. Fig. 10 shows the total time for negotiating for a single table in a computer and a single table by more that one computer (7 in this case). Results show that distribution improves the performance considerably, specially when the number of agents is high. It also allows for managing more agents interacting in the platform.

All in all, the main factor that determines how long the execution takes is the number of agents (this is a quite obvious result, but with these tests we were able to quantify the limits for the used version
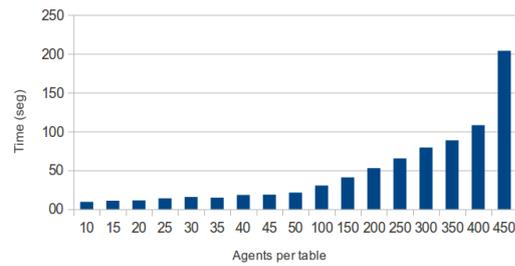
(a) 1 Table.
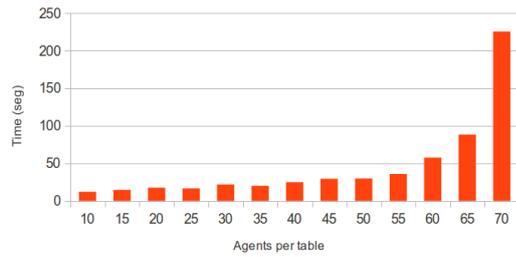
(b) 7 Tables.

Fig. 9 Time for negotiating in 1 PC.
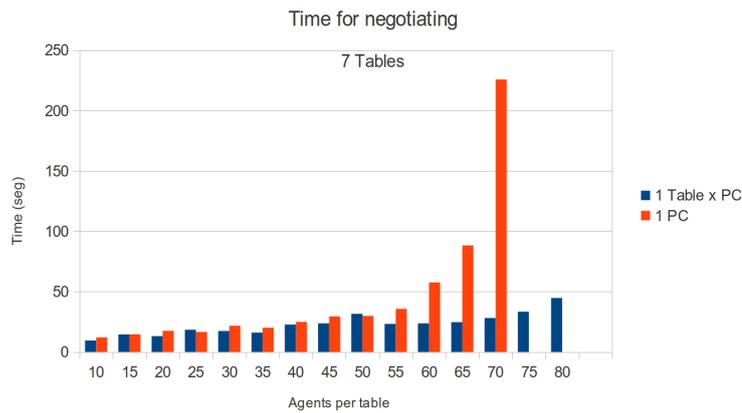
Fig. 10 Results for 1 PC and more than one trading table.

of Magentix2 and the current implementation of the Case Study). As it grows, more operations are executed and more messages are exchanged, so the time increases. The limitations of the underling platform is because the version of Magentix2 currently available only allows one Qpid broker for managing the messaging in the system.

At the same time, these tests helped us observe the influence or overhead of the "staff" in the overall performance. In this particular
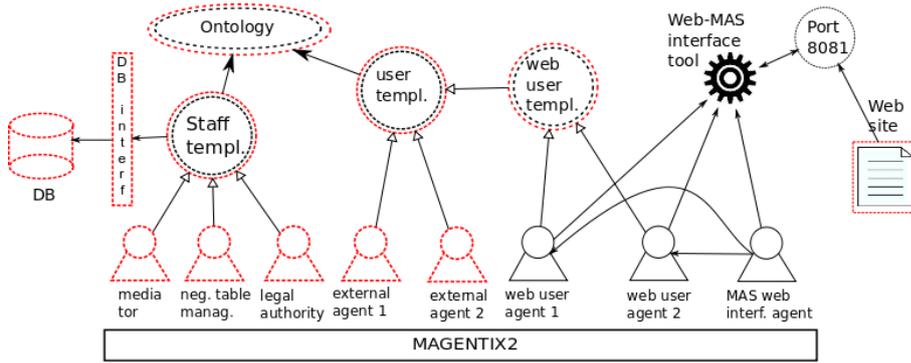
Fig. 11 Global design of the generic model. The elements drawn with solid lines do not need to be extended, the ones with a simple dashed line need to be completely implemented and the ones with double dashed line can be modified or extended with new functions for them to fit the domain.

implementation of *mWater* we used only one agent with the responsibility of all staff roles. This fact introduces an important overhead when there are a lot of agents and the negotiation is taking place in the system, because the staff agent must look for the good functioning of the market at all levels. Specially if periodic updates on the data base must be done, or also, if interactions are performed where the agents remain waiting for a the staff answer. In this case the execution time would be determined, to a great extent, by the time of this agent performing its actions. In turn, distribution of theses responsibilities in more than one physical agent, would relieve the overhead and would improve the performance.

The importance of these roles in the model justifies its existence even if the execution time becomes slightly higher. That is why it is convenient not to use centralized approaches in this case. It is neither a good choice to use a centralized approach in relation to the messaging platform if the system works with a big number of agents. In any case, the general model is still valid but, a distributed implementation of the staff roles must be performed in situations when big data and a high number of executing agents is required.

6 From the Generic Model to Specific Applications: Guidelines

After presenting our generic model for negotiation and how this can be used within a concrete example, i.e. *mWater*, we now propose some guidelines to adapt it to particular applications.

In order to get a better understanding of the implementation structure of the generic model here we summarize the main Magentix2 components and agents we provide. Fig. 11 shows the general design for a generic negotiation framework in Magentix2. It contains different kind of elements. The ones drawn

with a single solid line are implementations that are domain independent and do not need further functionalities, or in any case, just small adaptions. The ones with a single dashed line need to be completely implemented. Finally, the ones with a double dashed line are implementations already made but can be modified or extended in order to fit the particular features of the domain. In more detail, these elements are:

- **Ontology:** File in '.asl' format [6] that contains the domain ontology elements.
- **DB:** Data base.
- **DB interf.** Java class to mirror the changes of the belief base of an agent (the *staff* in this case) in the data base and vice versa.
- **staff templ.** Template for implementing the *staff* functions. The implemented functions are domain independent.
- **user templ.** Template for implementing the *user* agent functions.
- **web user templ.** Template for implementing a *user* agent representing a user in the web (a human user).
- **mediator, neg. Table manag, legal authority** Agents that behave as *staff*, each one specialized in their respective roles.
- **external agent 1, external agent 2** Generic agents that represent the final users of the system. They own a customized reasoning process according to their individual reasoning methods.
- **web user agent 1, web user agent 2** Generic agents that are executed in the MAS system, the actions of their respective human users.
- **MAS web interf. Agent** Agent whose main functions are to create the agents representing the human users in the system when they are accredited and to destroy them when they are not in the system anymore.
- **web-MAS interface tool** Tool for the interaction between the agents of the MAS and the web site through a communication port.
- **web site** Web interface to allow human users to interact with the agents in the MAS.

In order to adapt the generic structure of Fig. 11, the designer must complete, modify and redefine (override) some of its elements. From our experience, the right identification of the following issues is essential for a successful use and adaptation of the negotiation model:

1. Identify the type of application to be implemented, e.g. a simulation module, a decision-support tool, a virtual electronic market, a grievance resolution process, etc. To determine the way in which agents interact and the characteristics of the system, (e.g. if it behaves as an electronic institution or as a conversational system) as well as the expected size of the application, in terms of number of agents and interactions, is useful to find out which MAS platform must be used.
2. Identify the ontology of the problem domain. There are many questions to be answered in order to find out the main ontology concepts. Some of the most important ones are the following. What deal, in the form of a

product or service, will take part in the negotiation (e.g. a water-right, a by-product to support the re-use of waste in industry, some raw material, etc.)? How many negotiation processes will be implemented? In case of multiple processes, is there any interaction among them; and do they share the same information system and agents? Finally, what attributes will define the participants? This is important as it can have a significant impact in the ontology complexity. All these concepts should be included in the **Ontology** part of Fig. 11.

3. Specify the conceptual model for storing the information of the problem (**DB** in Fig. 11). The ontology identified in the previous step will define the conceptual model of the negotiation process, which can be easily implemented by using a database. We can store in the database the information about the users, negotiation tables, interactions and agreements, which are very valuable in the different workflow structures, such as those depicted in Fig. 1 and 2. In particular, in the *mWater* problem we have over 60 relational tables implemented in a mySQL database to keep trace of all the interactions that happen in the market. Additionally, the use of a database also offers a flexible way to make the model more complex by simply adding new tables and relate them to the new workflows.

4. Identify what negotiation protocols will be used. Here we can select the desired negotiation protocol from the Magentix2 library, such as face-to-face, Dutch auction, English auction, etc., or create a new one. The only restriction for doing this in our generic model is that the new protocol must meet the three interaction structure given in Fig. 5, which comprises Registration, Negotiation and Validation. Every negotiation protocol should be included as a new set of plans when implementing the **user templ.** (see Guideline #8 and Fig. 13).

5. Determine the negotiation parameters for each negotiation protocol: min/max number of participants and the possible conditions they have to satisfy, number of max number of interactions, interaction deadlines, pre and post conditions for agreements, etc. All these parameters are necessary for the *request* illocution. These elements must be included when implementing the negotiation protocol plans of the **user templ.** (see Guideline #8 and Fig. 13).

6. Analyze the features of the system participants and their possible roles. Also, will the participants be automated software agents (see Guideline #8), human agents (see Guideline #9) or a combination of both (as in our *mWater* example)? It can be also important to define the min and/or max number of participants, and the possibility to model internal and/or external participants. These values may affect the behavior of the system. For instance, if we are implementing a simulation module the type and number of participants can make the process more complex, though the results will be more useful.

7. Identify the necessity of a facilitator (either a mediator or a negotiation table manager) and the intelligent capabilities he has to provide in every negotiation protocol. In a simple approach, the intelligence capabilities

Fig. 12 The staff template. The gray boxes represent the plans that can be redefined.

may be null but in others, such as in *mWater*, these expert capabilities
help the users under two basic scenarios: i) to decide about opening a new
negotiation table, and ii) to decide what participant is going to be invited
to join that table and why (preliminary process of invitation). In both
cases, the facilitator must be aware of the current context of application
and the current norms. Therefore, it is important in this stage to focus on
the expert knowledge, and intelligent deliberative process, the facilitator
should implement. It is also possible to have a human agent that plays this
role. In our implementation, it is played by an agent and, in this case, new
checking can be added by customizing its behavior according to the new
needs of the problem. Fig. 12 shows the template for the facilitator agent.
This template implements the main functions of agents playing the *staff*
role. It has a '.asl' format and hence it is written using the AgentSpeak
language [6]. It mainly contains plans for answering users requests. For
each new interaction in which it must participate, the corresponding plans
according to the kind of interaction (or conversation), must be added. The
gray boxes in Fig. 12 represent the plans that may require modifications.
8. Specify the automated software agents that implement the system partici-
pants. When doing this, we need to explicitly implement the different roles

Fig. 13 The user template for the automated software agents participants. The gray boxes represent the plans that can be redefined. The figure shows an incomplete list of the plans for negotiation protocols because new ones can be included as required.

a participant can play: guest or black and white participant, as defined in Fig. 1. When doing this, individual intelligence capabilities for taking decisions in any required situation must be specified in such a way that they can be easily replaced if new behavior arise. Fig. 13 shows the template for the user agent. This template implements the main functions of agents playing the *user* role. It has a '.asl' format and hence it is written using the AgentSpeak language [6]. It mainly contains plans for start conversations with the *staff* requesting her/him an action or information, and it also contains plans for interacting with other users. For each new interaction in which it must participate, the corresponding plans according to the kind of interaction (or conversation) and according to the role in the conversation (*initiator* or *participant*), must be added [3]. In Fig. 13 the key plans and methods that may require modifications are represented. Moreover, the list of the plans for the negotiation protocols is incomplete as it can be extended with new protocols as required.

9. Analogously to the previous point, we need to specify the software agents that will simulate the human participants by implementing the specific functions. The **web user templ.** implements the main functions of web users. It has a '.asl' format and hence it is written using the AgentSpeak language [6]. It includes all plans of '**user templ.**' because a web user behaves also as a *user*. It also includes plans for receiving requests from an agent in the '**Web-MAS interface tool**' when the human user wishes to perform any action in the system.

10. Execute validation tests in order to evaluate the quality of the model. Obviously, this highly depends on the type of application that is being implemented. For instance, in the case of a simulation module we may be interested in some performance indicators (e.g. volume of water that is transferred in the *mWater* problem). On the other hand, in the case of a decision-support tool to help in policy design, we may be more interested in finding out which set of rules/norms will be incorporated in the final legislation.

7 Further Uses for the Generic Negotiation Model

The infrastructure for generic negotiation that we have presented here has several application uses, from both the academia and industry point of view. From the academia standpoint, it can be used as a testbed for other developments within the agreement technologies paradigm[7]. In particular, there are several challenging questions on:

— Organization and roles. How beneficial is the inclusion of collective, heterogeneous roles, their collaboration (and trust theories) and how the policies for either flat or hierarchical group formation affect the system behavior? To answer this we need to capture all those roles currently recognized by legislation that have any impact on negotiation and agreement management, specially in grievances and conflict resolution.
— Collective decision-making, reconfigurability, cooperation, social issues and coordination. What is the impact of argumentation, judgement aggregation, reputation, prestige and multi-party negotiation in the system performance? The answer to this question is not straightforward and requires simulation tools for performance assessment, as seen in section 5.
— Institutional limitations. What type of enforcement mechanisms are necessary and how they change *w.r.t* the evolution of regulation? This is highly related to the definition, adoption and compliance of (emerging) norms and, more particularly, how to model and reason on them? To solve this, we need to face the problem of expressiveness: the type of norms we have dealt with so far has a formal representation, but other types of representation may be more complex to handle. Finally, ensuring norm compliance is not always possible (or desired), so norm violation and later detection via grievances

---

[7] http://www.agreement-technologies.org

usually makes the environment more open, dynamic and realistic for taking decisions.

From the industry standpoint, there exist further applications in the form of simple tools that can be embedded within our MAS framework:

– A decision-support tool for policy simulation. Policy-making is a hard task. Designing and taking legal decisions involves a complex balance among different factors, such as economic, social, administrative or environmental aspects. Also, factors usually change throughout time due to variations in economic situation, population distribution and physical conditions. Consequently, a decision-support tool that allows policy-makers to easily predict, analyze and measure the suitability and accuracy of modified regulations for the overall system, before using other operational tools for the real floor, shows very important. Our experiments with *mWater* shed light on the benefits that a collaborative AI perspective for a water-right market may bring to the policy-makers, general public and public administrators. The generic negotiation model presented in this paper could be the base of decision-making tools that can improve the capacity of policy regulators in modelling and evaluating new or modified policies in human markets. After all, in this context a policy maker has little control over the hydrographical features of a basin but (s)he has legal power to regulate water user's behavior to a larger extent by means of: i) government laws, ii) basin or local norms, and iii) social norms to design appropriate water laws that regulate users' actions. And these can be simulated easily in a decision-support system.
– A GUI tool for human negotiation that facilitates the human interaction with software agents. Particularly, our GUI provides a simple, though effective way to set up parameters and dynamic changes, which affect the performance of the system, during the negotiation process (and also while simulating this process). Moreover, it intuitively provides the results generated after such an interaction process, which can be used as an analysis tool to evaluate protocols.
– A general tool open to other negotiation processes, such as other electronic markets; the workflow structure, roles and negotiation interaction remains the same. Our experiences show that our negotiation framework is general enough and can be valid for other markets. Particularly, we are applying these ideas to a by-product exchange market to boost the re-use of waste, thus being part of our current work.

8 Conclusions

In this paper we have established the infrastructure foundations for the specification of a multi-agent-based negotiation framework as the basis for modeling virtual scenarios, and put it into practice within a water-right market, where negotiation plays a vital role. The work presented in this paper is based on the

lessons learned in [7,15]. But now, the generic negotiation framework has been implemented in Magentix2 to offer a flexible and easy way to adapt to applications in which autonomous features in regulated environments are required. Thus, the technical contributions of this work are:

– Design a generic MAS infrastructure that captures the main steps that happen in an agent-based scenario, including mechanisms for exchanging information, negotiating and dealing with the critical situations that may appear thereafter.
– Introduce the users and intelligent roles that are necessary within an agent-based setting. Differently to existing approaches, we introduce the roles of intelligent mediators, which are very valuable for the process.
– Provide multiple negotiation strategies that are managed in a three-step unified way: registering, negotiating and validating the reached agreement. This also allows us to include different protocols in a flexible fashion.
– In order to test the applicability of this generic framework, we have put these ideas into practice with *mWater*. This water market is very illustrative and has allowed us to explore the influence that the repetitive interaction of participants exerts on the evolution of the market. Also, it has given us enough evidence that the generic framework for negotiation provides a solid foundation for complex markets.

Acknowledgments

References

1. Jade. http://jade.tilab.com.
2. J. M. Alberola, J. M. Such, A. Espinosa, V. Botti, and A. García-Fornes. Magentix: a Multiagent Platform Integrated in Linux. In *EUMAS*, pages 1–10, 2008.
3. B. Alfonso, E. Vivancos, V. Botti, and A. García-Fornes. Integrating jason in a multi-agent platform with support for interaction protocols. In *Proceedings of the compilation of the co-located workshops on AGERE!'11*, SPLASH '11 Workshop, pages 221–226, New York, NY, USA, 2011. ACM.
4. J. Andreu, J. Capilla, and E. Sanchis. AQUATOOL, a generalized decision-support system for water-resources planning and operational management. *Journal of Hydrology*, 177(3–4):269–291, 1996.
5. F. Bellifemine, G. Caire, and D. Greenwood. *Developing Multi-Agent Systems with JADE*. John Wiley and Sons, 2007.
6. R. H. Bordini, J. F. Hübner, and M. Wooldridge. *Programming Multi-agent Systems in Agent Speak Usign Jason*. John Wiley & Sons, 2007.
7. V. Botti, A. Garrido, J. A. Gimeno, A. Giret, and P. Noriega. The role of MAS as a decision support tool in a water-rights market. In *AAMAS 2011 Workshops, LNAI 7068*, pages 35–49. Springer, 2011.

8. L. Braubach, A. Pokahr, and W. Lamersdorf. Jadex: a BDI agent system combining middleware and reasoning. In M. C. M. K. R. Unland, editor, *Software Agent-Based Applications, Platforms and Development Kits*, pages 143–168. Birkhäuser-Verlag, 9 2005.

9. G. B. DeSanctis and B. Gallupe. A foundation for the study of group decision support systems. *Knowledge based systems*, 33(5):589–609, 1987.

10. P. Eckersley. Virtual markets for virtual goods, 2003. Available at http://www.ipria.com/publications/wp/2003/IPRIAWP02.2003.pdf (accessed April 2012).

11. J. Fjermestad and S. Hiltz. Group support systems:a descriptive evaluation of case and field studies. *Journal of Management Information Systems*, 17(3):115–161, 2001.

12. R. L. Fogués, J. M. Alberola, J. M. Such, A. Espinosa, and A. García-Fornes. Towards Dynamic Agent Interaction Support in Open Multiagent Systems. In *Proceedings of the 13th International Conference of the Catalan Association for Artificial Intelligence*, volume 220, pages 89–98. IOS Press, 2010.

13. Foundation for Intelligent Physical Agents. *FIPA XC00025E: FIPA Interaction Protocol Library Specification*.

14. A. Garrido, M. Arangu, and E. Onaindia. A constraint programming formulation for planning: from plan scheduling to plan generation. *Journal of Scheduling*, 12(3):227–256, 2009.

15. A. Giret, A. Garrido, J. A. Gimeno, V. Botti, and P. Noriega. A MAS decision support tool for water-right markets. In *Proceedings of the Tenth International Conference on Autonomous Agents and Multiagent Systems (Demonstrations@AAMAS)*, pages 1305–1306, 2011.

16. J. Gomez-Limon and Y. Martinez. Multi-criteria modelling of irrigation water market at basin level: a Spanish case study. *European Journal of Operational Research*, 173:313–336, 2006.

17. N. K. Janjua, F. K. Hussain, and O. K. Hussain. Semantic information and knowledge integration through argumentative reasoning to support intelligent decision making. *Information Systems Frontiers*, 15(2):167–192, 2013.

18. J. Y. jen Hsu, K.-J. Lin, T.-H. Chang, C. ju Ho, H.-S. Huang, and W. rong Jih. Parameter learning of personalized trust models in broker-based distributed trust management. *Information Systems Frontiers*, 8(4):321–333, 2006.

19. G. Kersten and H. Lai. Satisfiability and completeness of protocols for electronic negotiations. *European Journal of Operational Research*, 180(2):922–937, 2007.

20. N. Lee, J. K. Bae, and C. Koo. A case-based reasoning based multi-agent cognitive map inference mechanism: An application to sales opportunity assessment. *Information Systems Frontiers*, 14(3):653–668, 2012.

21. M. Luck and AgentLink. *Agent technology: computing as interaction: a roadmap for agent-based computing. Compiled, written and edited by Michael Luck et al.* AgentLink, Southampton, UK, 2005.

22. J. Ma and M. A. Orgun. Formalizing theories of trust for authentication protocols. *Information Systems Frontiers*, 10(1):19–32, 2008.

23. A. Pokahr, L. Braubach, A. Walczak, and W. Lamersdorf. *Developing Multi-Agent Systems with JADE*, chapter Jadex-Engineering Goal-Oriented Agents, pages 254–258. Wiley and Sons, 2007.

24. C. Ramos, M. Cordeiro, I. Praça, and Z. Vale. Intelligent agents for negotiation and game-based decision support in electricity markets. *Engineering intelligent systems for electrical engineering and communications*, 13(2):147–154, 2005.

25. C. Sierra, V. Botti, and S. Ossowski. Agreement computing. *KI - Künstliche Intelligenz*, 25(1):57–61, 2011.

26. M. Thobani. Formal water markets: Why, when and how to introduce tradable water rights. *The World Bank Research Observer*, 12(2):161–179, 1997.