D8.2.1.P3 Report: *mWater* prototype #3 review

Bexy Alfonso (UPV), Vicente Botti (UPV), Antonio Garrido (UPV), Adriana Giret (UPV), Pablo Noriega (CSIC)

Abstract.
CSD2007-0022, INGENIO 2010
Deliverable D8.2.3.P3 (WP8, Task 8.2)

The mWater prototype #3 review is detailed in this report.
Keyword list: mWater, e-market, analysis and design

Document Identifier	AT/2008/D8.2.3.P3/v0.1
Project	CSD2007-0022, INGENIO 2010
Task	T8.2
Version	v0.1
Date	June 03, 2012
State	draft
Distribution	public

Agreement Technologies Consortium

This document is part of a research project funded by the Consolider Programme of the Ministry of Science and Innovation as project number CSD2007-0022, INGENIO 2010.

Spanish Scientific Research Council (CSIC)
Institut d'Investigació en Intel·ligència Artificial (IIIA)
- Coordinator
Campus UAB
08193, Bellaterra
Catalonia
Spain
Contact person: Carles Sierra
E-mail address: sierra@iiia.csic.es

Universidad Rey Juan Carlos (URJC)
Centre for Intelligent Information Technologies
(CETINIA)
Campus de Móstoles
C/ Tulipán s/n E-28933 Móstoles (Madrid)
Spain
Contact person: Sascha Ossowski
E-mail address: sascha.ossowski@urjc.es

Universitat Politècnica de València (UPV)
Departament de Sistemes Informàtics i Computació
Camino de Vera s/n
40622 València
Spain
Contact person : Vicent Botti
E-mail address: vbotti@dsic.upv.es

Work package participants

The following partners have taken an active part in the work leading to the elaboration of this document, even if they might not have directly contributed to writing parts of this document:

IIIA-CSIC, Bellaterra
Universidad Rey Juan Carlos, Madrid
Universitat Politecnica de Valencia, Valencia

Changes

Version	Date	Author	Changes
0.1	03.05.12	Adriana Giret	creation

Executive Summary

mWater is a software demonstrator developed in the Agreement Technologies Project. It is a Multi-Agent System (MAS) application that implements a market for water rights, including the model and simulation of the water-right market itself, the basin, users, protocols, norms and grievance situations.

mWater is motivated due to the fact that water scarcity is becoming 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.

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. And an institutional framework such as mWater, where water rights may be exchanged more freely and not only under exceptional conditions, leads to a more efficient use of water.

mWater is a regulated open MAS that uses intelligent agents to manage a flexible water-right market. One of the main goals of mWater is to be used as a simulator to assist in decision-taking processes for policy makers. Our simulator focuses on demands and, in particular, on the type of regulatory (in terms of norms selection and agents behaviour), and market mechanisms that foster an efficient use of water while also trying to prevent conflicts among parties.

mWater plays a vital role as it allows us to define different norms, agents behaviour and roles, and assess their impact in the market, thus enhancing the quality and applicability of its results as a decision support tool.

The mWater prototype #3 review is presented in this deliverable. The review described is a general negotiation market infrastructure for MAS systems.

Contents

Chapter 1

A Generic Negotiation Model for electronic markets

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 [23, 6, 31, 30, 28]. 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 behaviors; 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.

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 [32]. This line of research has addressed developments for group decision support systems and meeting support systems, which can be extrapolated to automated negotiation [18, 20]. 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 [24, 25]. 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 [10] 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 [36]. 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* [11, 26], 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 [36], including agility, heterogeneity, reconfigurability, cooperation, argumentation, reputation and trust issues under a MAS perspective.
The chapter is organized as follows. In Sect. 1.1 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. 1.2 we present the generic negotiation model. It is described the negotiation workflow structure and also the roles participating on its interactions. In Sect. 1.3 we describe the simulation tool, used as a case study, for implementing an electronic market of water rights. Sect. 1.5 shows some practical guidelines to adapt the negotiation model to particular applications. Further uses for the negotiation model are commented in Sect. 1.6 from the academia an industry standpoint. Finally Sect. 1.7 concludes the paper giving some data about the related work.

1.1 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.

1.1.1 MAS Platform

We use Magentix2 [2] as our MAS platform 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 [33]: organization level, interactions level and agent level.

Conversations Factory: an Artifact for Communication

A Conversations Factory [21] is mainly a Magentix2 mechanism to support FIPA interaction protocols [22]. 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.

1.1.2 Programming language

Magentix2 allows us to use a high-level language for programming agents. In this case it is Jason [10], 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' 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 [38, 8, 27, 7, 37, 31, 4] 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.

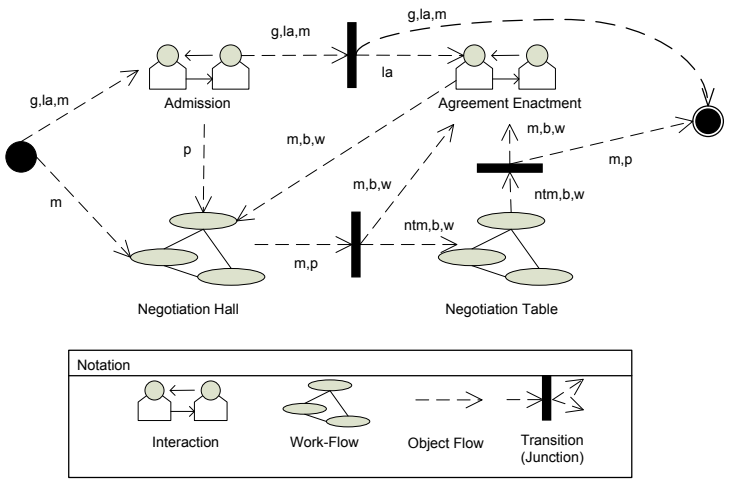

Figure 1.1: Generic Negotiation workflow structure. Roles: *g* - guest; *p* - participant; *b* - black; *w* - white; *m* - mediator; *ntm* - negotiation table manager; *la* - legal authority.

1.2 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.

1.2.1 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. 1.1)[1], 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.

1.2.2 Users and Roles

There are seven roles that interact in the model, as depicted in Fig. 1.1. A guest role (g) is a user that wants to enter the negotiation. The guest may be specialized into a

[1] 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.

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 an 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.

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. These two roles have demonstrated 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 of the market and the negotiation structure. 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.

1.2.3 Workflow

The workflow activities in the generic negotiation model of Fig. 1.1 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. 1.2), 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 the *NTC* interaction, which responds to

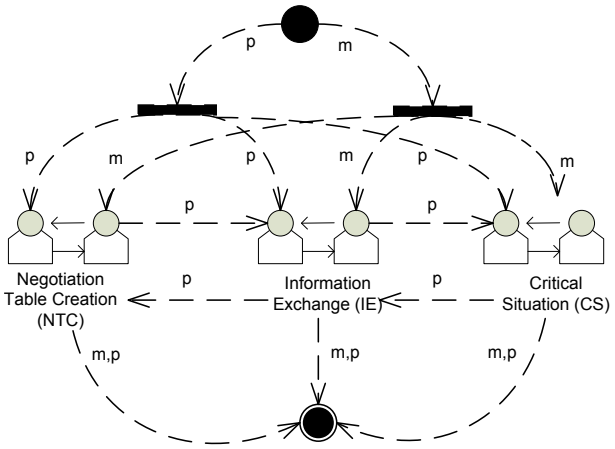

Figure 1.2: *Negotiation Hall* workflow structure.

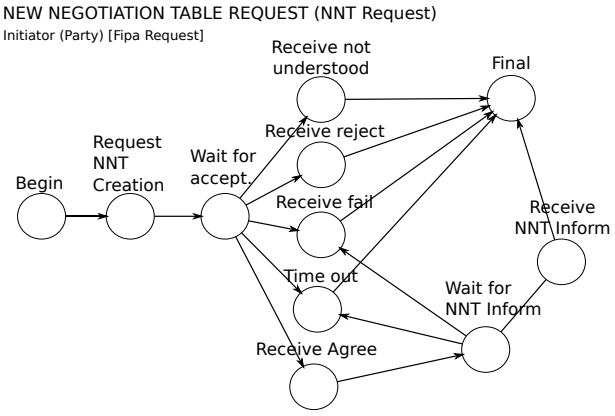

Figure 1.3: Party's behavior for requesting a New Negotiation Table.

the *FIPA request* standard protocol [22]. Fig. 1.3 and 1.4 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. 1.3) to the Mediator, m, to $open$ a negotiation table with a given negotiation $protocol$. This protocol is instantiated 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.

When the Mediator, m, receives a request to open a negotiation table (see Fig. 1.4), 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,

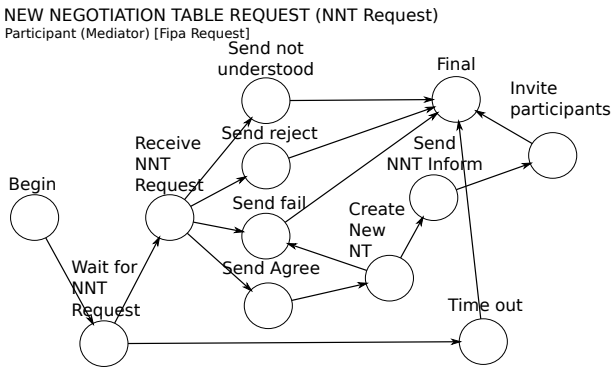

Figure 1.4: Mediator's behavior during the conversation for Opening a New Table.

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 predefined 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.

Every generic negotiation table is defined as a three interaction structure (see Fig. 1.5). The first interaction is *Registration*, in which the ntm applies a filtering process to

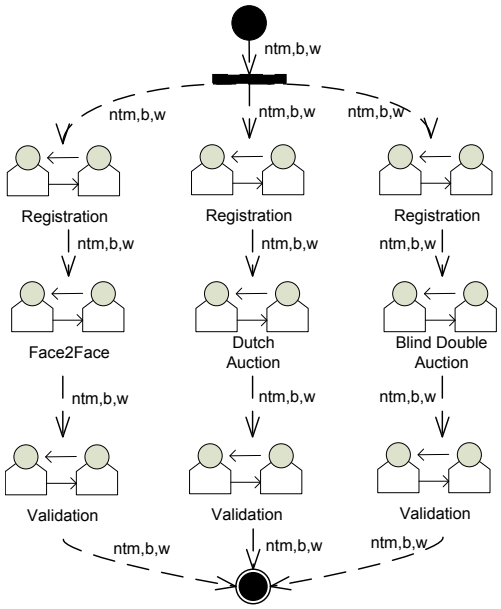

Figure 1.5: *Negotiation Table* workflow structure.

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.

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

1.3.1 *mWater* Overall Description

Water scarcity is a major concern in most countries. It has been sufficiently argued that more efficient uses of water may be achieved within an institutional framework where water rights may be negotiated under different market conditions [39]. In hydrological terms, a water market can be defined as an institutional, decentralized framework where users with water rights are allowed to voluntarily trade them, always fulfilling some pre-established norms, to other users in exchange of some compensation [29, 39]. Because of water's unique characteristics, such markets do not work everywhere, they are not homogenous, nor do they solve all water-related issues [39]. Also, even subtle changes in the market design (allowed participants, legislation, protocols, etc.) are very costly and difficult to evaluate.

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 [11, 26]. More specifically, *mWater* assists in designing appropriate water laws and regulate, either privately or publicly, the users' actions, interactions and their eventual trade.

1.3.2 *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. 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[2], the norms and regulations that define the policies in the market, the seasons in which the water-right transfer will take place, etc. The simulation

[2]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 behaviors that are hard to implement or highly time consuming.

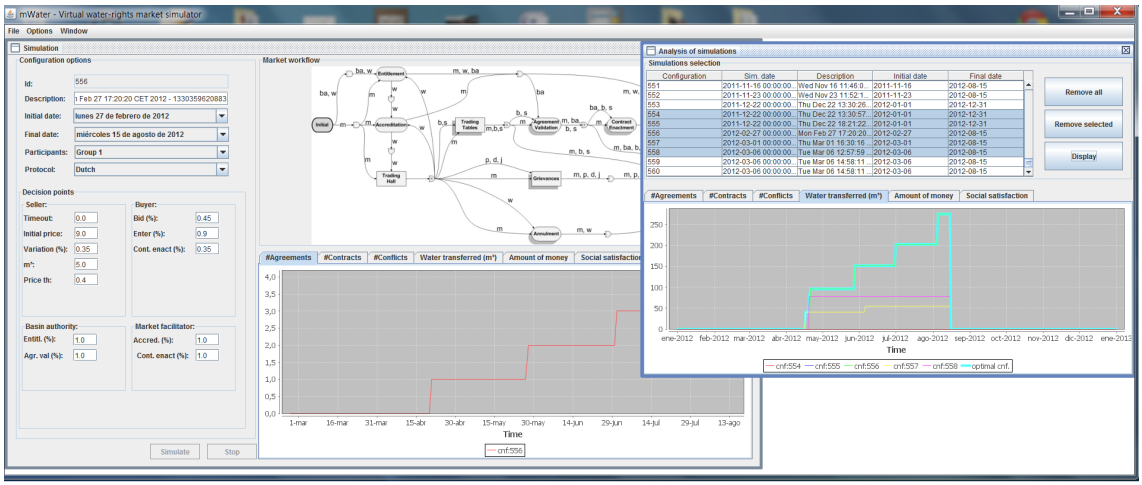

Figure 1.6: Snapshot of the *mWater* simulator.

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.

1.3.3 *mWater* in Action

Fig. 1.6 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[3].

To implement human-agents interactions, in order to have a tool for studying different behaviors and situations, it was 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. This makes possible to count on a MAS composed by a mixture of automated agents and humans, and even a system completely based on automated agents. Fig. 1.7 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 on it. On the other hand, Fig. 1.8 shows how a user can participate in a Japanese Auction of a water right, by interacting with other human or automated agents.

[3]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.

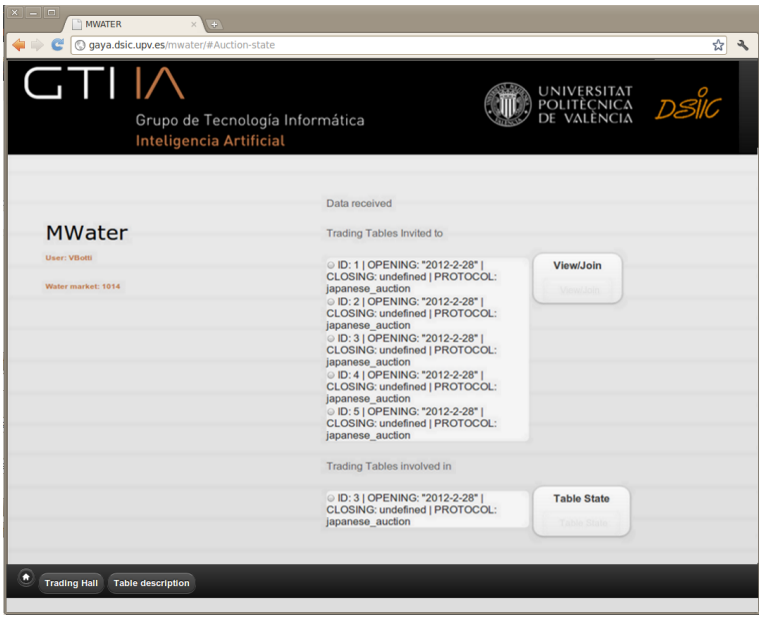

Figure 1.7: Snapshot of the human-agents GUI. It shows the trading tables that the human has been invited to and that is involved.

This simulation tool allows users to analyze: i) how the conventions, norms 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,

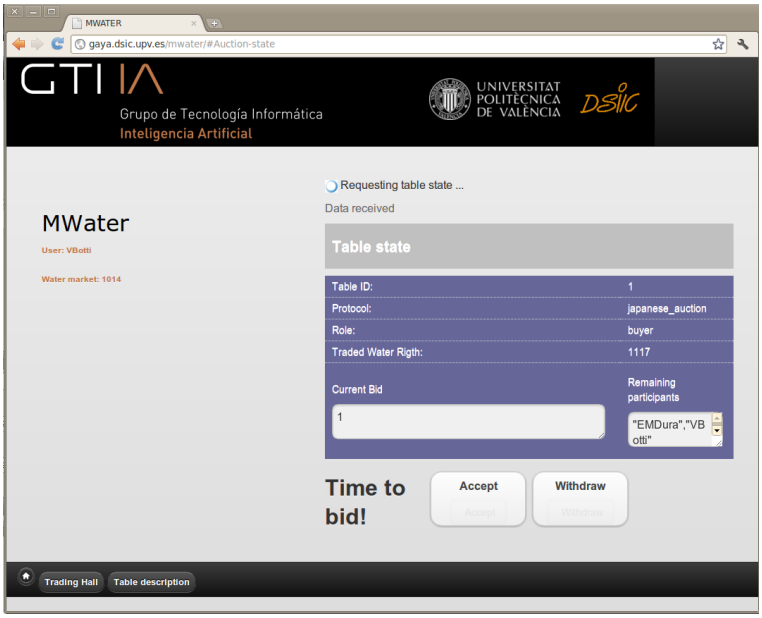

Figure 1.8: Snapshot of the human-agents GUI. The user can participate in a Japanese auction with other humans and/or automated agents.

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. Also, from the experts' evaluation we can conclude that a tool like this provides an advantageous tool to help build a more efficient water market with more dynamic norms.

1.4 Performance evaluation of the generic negotiation model

In this section the runtime performance evaluation of the generic negotiation model is presented. The evaluation is performed with a prototype implementation of the mWater case study. Nevertheless the measurement presented is independent of the application domain. The prototype has been developed using the Java language so we used java applications to launch agents in the platform. The MAS platform is Magentix2 whose agents use Qpid client APIs to connect to the Qpid broker and to communicate with other agents. The relational database management system used is MySQL. There is an agent called "staff" 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 live in this computer too or in other computers depending on the test performed.

We used 7 computers for performing the tests. One of them stores the database, and also the *staff* agent and the Qpid broker are running on it; 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 1.1.

PC	OS	JDK version	Linux kernel
1	Ubuntu 11.10	OpenJDK 1.6	3.0.0-15
2	Ubuntu 11.04	OpenJDK 1.6	2.6.38-11
3	Ubuntu 11.04	OpenJDK 1.6	2.6.38-16
4	Ubuntu 11.10	OpenJDK 1.6	3.0.0-15
5	Ubuntu 8.10	JDK 1.6	2.6.27-17
6	Ubuntu 11.10	OpenJDK 1.6	3.0.0-15

Table 1.1: Pcs' technical description.

1.4.1 Experiments

Basically it was tested the performance of the system when agents negotiate. Our aim is to measure the system load due to the generic negotiation model specific features, so we are interested in the communication performance of the key structural components of the

model such as negotiation tables, staff agents, and common configuration functions of the negotiation protocols. In this evaluation we used Japanese auctions, nevertheless other negotiation protocols can be evaluated following the same steps. The Japanese auction starts with an initial bid proposed by the table owner. Each participant can accept or reject the proposal. In each iteration the bid is always incremented in the same quantity, and it finishes when:

- An only one participant is agree with the proposal.
- Nobody is agree with the proposal.
- A maximum number of iterations is reached.

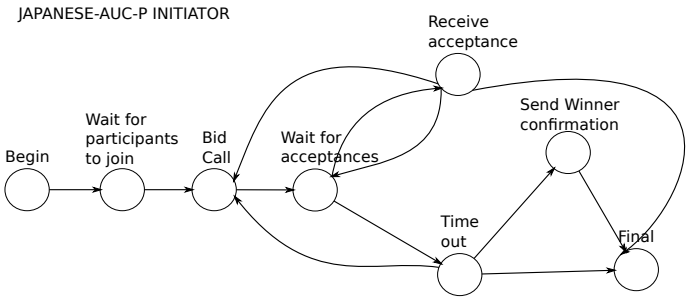

Figure 1.9: Steps of the Japanese Auction protocol from the *initiator* perspective.

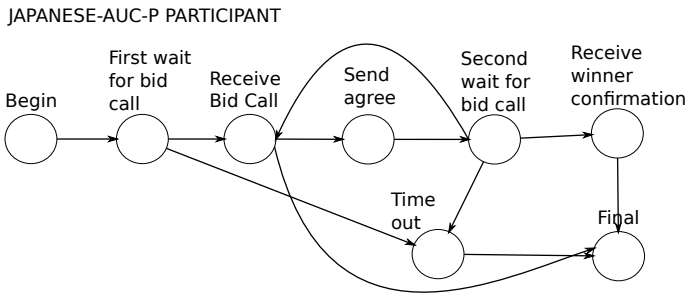

Figure 1.10: Steps of the Japanese Auction protocol from the *participant* perspective.

Figures 1.9 and 1.10 show the steps of the Japaneses auction protocol from the *initiator* and *participant* perspectives respectively. The following parameters where used for all the experiments in order to obtain comparable results:

- *Initiator:* 4 seconds for waiting for participants to join the auction and 15 seconds for waiting acceptances in each iteration.
- *Participants:* 50 seconds for waiting for initiator bid calls.

Experim No.	Tables No.	Total No. of agents	No. of PCs	Max agents per table	Max agents per table & PC
1	1	50,100,...,450	1	50,100,...,450	50,100,...,450
2	1	70,140,210, 280,350	7	70,140,210, 280,350	10,20,...,50
3	7	70,105,...,315	1	10,15,...,45	10,15,...,45
4	7	70,105,...,315	7	10,15,...,45	10,15,...,45
5	5,6,...,11	50,90,140, 200.270,350, 440	5	10,15,...,45	2,3,4,...,8
6	14,21,28, 35,42	140,210,280, 350,420	7	10	10
7	14,20,30	140,200,300	1	10	10
8	14	420	1	30	30

Table 1.2: Combinations for the parameters values.

- *Protocol:* 8 iterations max, value 1 as initial bid and increments of 5 in each round.

According to this, if the maximum number of iterations is reached and there are no delays from the initiator side, the auction takes 124 seconds $(4 + 8 * 15)$ excluding the time for the database updates and for the notifications to all agents of the final results. This time may be less if there is a winner before the 8^{th} round.

The experiments performed take as parameters: the maximum number of agents per trading table, the maximum number of agents per trading table and computer, number of trading tables and number of computers. In all cases there is always the same number of agents in each table. The combinations of the values used for these parameters are listed in table 1.2.

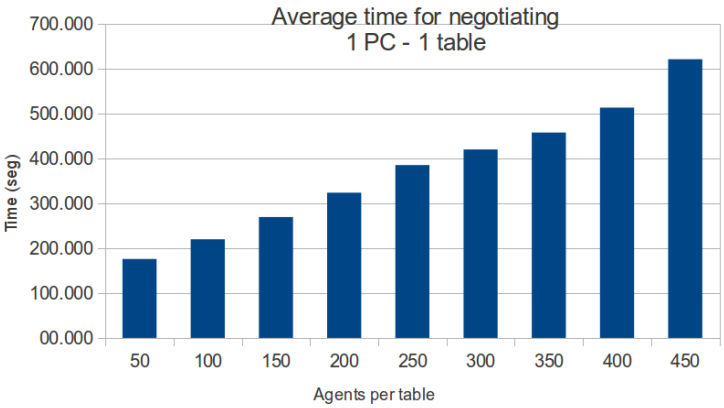

Figure 1.11: Results for 1 PC and 1 trading table.

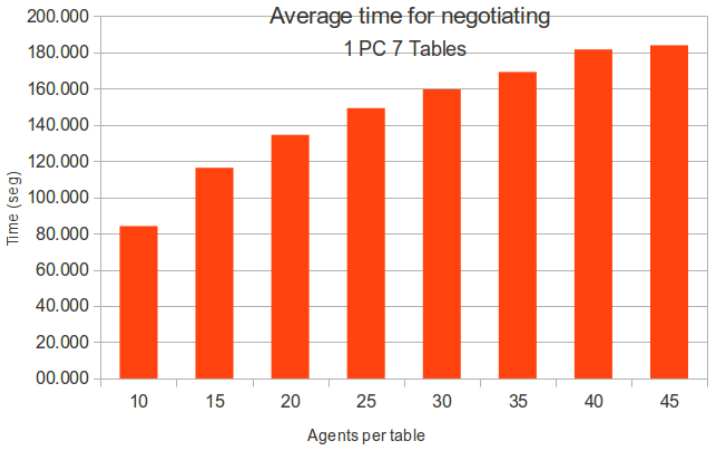

Figure 1.12: Results for 1 PC and 7 trading tables.

1.4.2 Evaluation Results

The results obtained from evaluating the average time for negotiatig according to the parameters of table 1.2 are summarized in figures 1.11, 1.12, 1.13, 1.14 and 1.15. Figures 1.11 and 1.12 show 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. In both cases the time gets higher when incrementing the number of participant agents in the table in a more or less linear way no matter the number of open trading tables. Figure 1.13 shows the same results but all in the same graphic, and adding some more results with other number of tables. It confirms the previous result where the number of tables is not a critical factor.

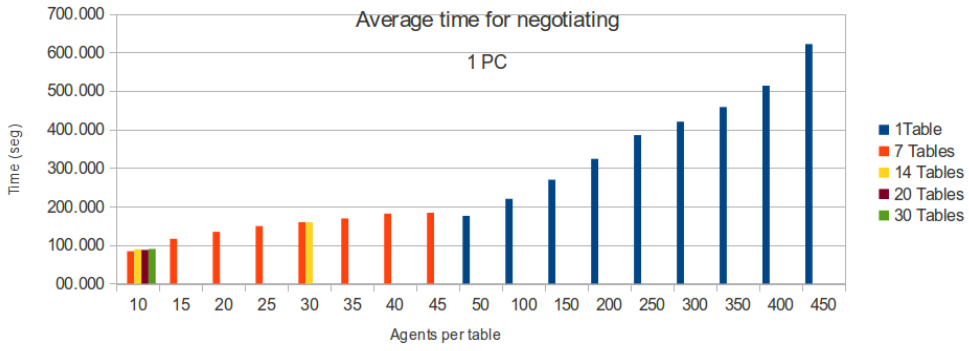

Figure 1.13: Results for 1 PC and more than one trading table.

Figure 1.14 show results when the agents are distributed in 7 computers and the number of negotiations in each computer vary. There are similar results when we have the same number of agents per trading table no matter if the number of tables per computer gets higher (what means a higher number of total trading tables), despite of there are

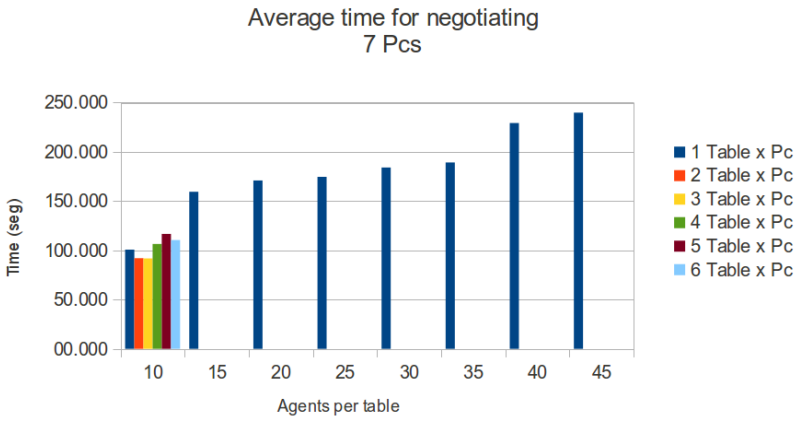

Figure 1.14: Results for 7 PCs and more than one trading table.

small growths. Having the same amount of tables in each computer but but increasing the number of agents, the time also grows.

Finally figure 1.15 confirms what has been observed in previous results: when varying the number of tables the results are similar; when the number of agents per table (and consequently the total number of agents) is incremented, the process of negotiation takes longer in a more or less linear behavior.

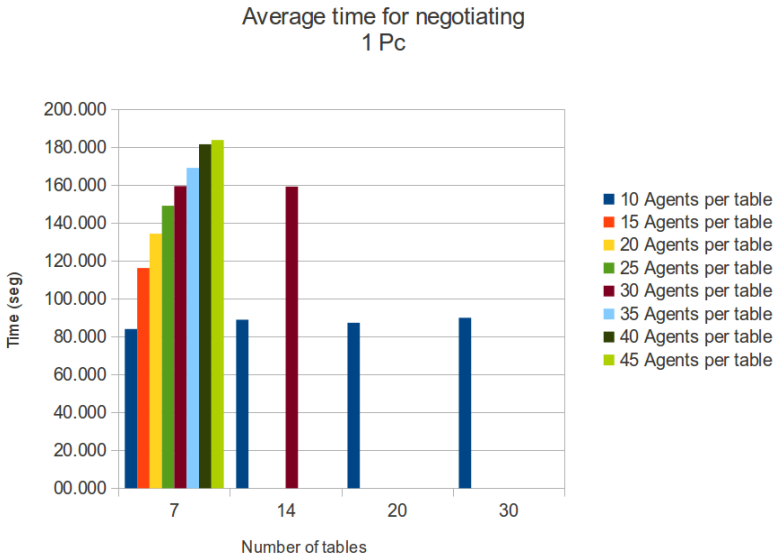

Figure 1.15: Results for 1 PC and more than one trading table varying the agents number per table.

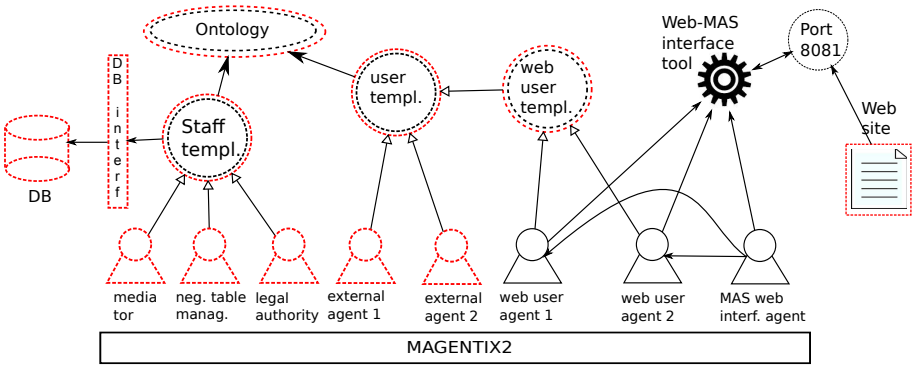

Figure 1.16: Global design of the generic model. The elements drawn with solid lines do not need to be extended, the ones with a simple dashed line need to be completely implemented and the ones with double dashed line can be modified or extended with new functions for them to fit the domain.

1.5 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. 1.16 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 [10] 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 en 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. 1.16, 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. 1.16.

3. Specify the conceptual model for storing the information of the problem (**DB** in Fig. 1.16). 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. 1.1 and 1.2. 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. 1.5, 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. 1.18).

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. 1.18).

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 behaviour 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 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

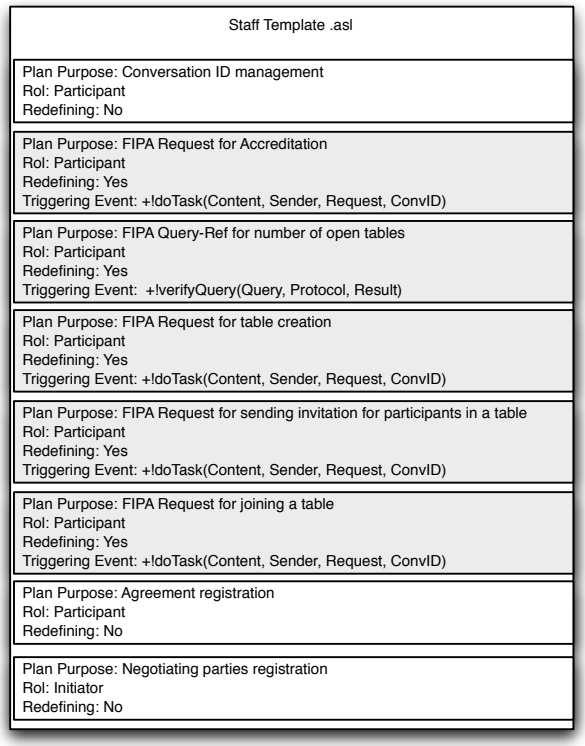

Figure 1.17: The staff template. The gray boxes represent the plans that can be redefined.

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 behaviour according to the new needs of the problem. Fig. 1.17 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 [10]. 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. 1.17 represent the plans that may require modifications.

8. Specify the automated software agents that implement the system participants. When doing this, we need to explicitly implement the different roles a participant can play: guest or black and white participant, as defined in Fig. 1.1. 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 behaviours arise. Fig. 1.18 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 [10]. 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

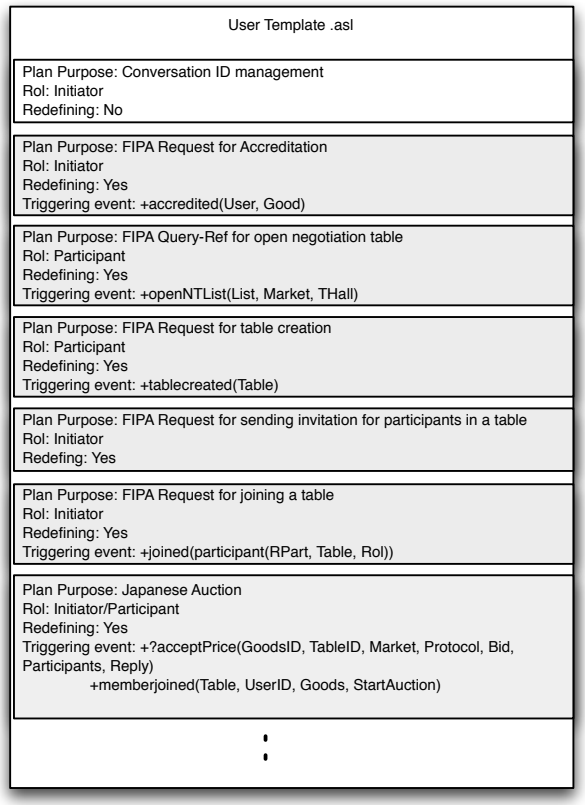

Figure 1.18: The user template for the automated software agents participants. The gray boxes represent the plans that can be redefined. The figure shows an incomplete list of the plans for negotiation protocols because new ones can be included as required.

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. 1.18 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 [10]. 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 in-

dicators (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.

1.6 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 (`http://www.agreement-technologies.org`). 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 [36, 16, 17, 13], specially in grievances and conflict resolution.
- Collective decision-making, reconfigurability, cooperation, social issues and coordination. What is the impact of argumentation [30], 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 1.3.
- 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? [15, 14, 5] 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 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, fac-

tors 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 behaviour 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 [6], 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.

1.7 Conclusions through 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, 9], 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. Another multi-agent platform with support for interaction protocols is Jadex [12, 34]. 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 inter-

action 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, Magentix2 [2] supports a high-level language for programming conversational agents (i.e. agents whose interactions respond to interaction protocols) and the rest 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, which makes it become 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 [19, 35], they do not always report significant advances from a collaborative AI perspective. 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 [11, 26]. 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.

Bibliography

[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 DSM'11, TMC'11, AGERE!'11, AOOPES'11, NEAT'11, & VMIL'11*, SPLASH '11 Workshops, pages 221–226, New York, NY, USA, 2011. ACM.

[4] E. Argente, V. Botti, C. Carrascosa, A. Giret, V. Julian, and M. Rebollo. An abstract architecture for virtual organizations: The thomas approach. In *Knowledge and Information Systems*, pages 1–35, 2011.

[5] E. Argente, V. Botti, and V. Julian. Gormas: An organizational-oriented methodological guideline for open mas. In *Agent-Oriented Software Engineering X- 10th Int. Workshop AOSE 2009 - Revised Selected Papers. LNCS, Volumen 6038*, pages 32–47, 2011.

[6] E. Argente, A. Giret, S. Valero, V. Julian, and V. Botti. Survey of mas methods and platforms focusing on organizational concepts. In *Recent Advances in Artificial Intelligence Research and Development*, volume 13, pages 309–316, 2004.

[7] E. Argente, J. Palanca, G. Aranda, V. Julian, V. Botti, A. Garcia-Fornes, and A. Espinosa. Supporting agent organizations. In *H.D. Burkhard et al. (Eds.). Multiagent Systems and Applications. LNAI 46696*, pages 236–245, 2007.

[8] J. Bajo, V. Julian, J. M. Corchado, C. Carrascosa, Y. de Paz, V. Botti, and J. F. de Paz. An execution time planner for the artis agent architecture. In *Engineering Applications of Artificial Intelligence*, volume 21, pages 769–784, 2006.

[9] F. Bellifemine, G. Caire, and D. Greenwood. *Developing Multi-Agent Systems with JADE*. John Wiley and Sons, 2007.

[10] R. H. Bordini, J. F. Hübner, and M. Wooldridge. *Programming Multi-agent Systems in Agent Speak Usign Jason*. John Wiley & Sons, 2007.

[11] 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.

[12] 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.

[13] N. Criado, E. Argente, and V. Botti. A normative model for open agent organizations. In *International Conference on Artificial Intelligence (ICAI)*, pages 101–108, 2009.

[14] N. Criado, E. Argente, and V. Botti. A bdi architecture for normative decision making (extended abstract). In *Proc. 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, volume I, pages 1383–1384, 2010.

[15] N. Criado, E. Argente, and V. Botti. Normative deliberation in graded bdi agents. In *Eighth German Conference on Multi-Agent System Technologies (MATES-10)*, volume 6251, pages 52–63, 2010.

[16] N. Criado, E. Argente, V. Julian, and V. Botti. Designing virtual organizations. In *7th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS2009)*, volume 55, pages 440–449, 2009.

[17] N. Criado, V. Julian, V. Botti, and E. Argente. A norm-based organization management system. In *AAMAS Workshop on Coordination, Organization, Institutions and Norms in Agent Systems (COIN). LNAI 6069*, pages 19–35, 2010.

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

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

[20] 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.

[21] 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.

[22] Foundation for Intelligent Physical Agents. *FIPA XC00025E: FIPA Interaction Protocol Library Specification*.

[23] A. Giret and V. Botti. Holons and agents. In *JOURNAL OF INTELLIGENT MANUFACTURING*, volume 15, pages 645–659, 2004.

[24] A. Giret and V. Botti. Towards an abstract recursive agent. In *INTEGRATED COMPUTER-AIDED ENGINEERING*, volume 11, pages 165–177, 2004.

[25] A. Giret and V. Botti. From system requirements to holonic manufacturing system analysis. In *International Journal of Production Research*, volume 44, pages 3917–3928, 2006.

[26] 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.

[27] A. Giret, V. Julian, M. Rebollo, E. Argente, C. Carrascosa, and V. Botti. An open archtecture for service-oriented virtual organizations. In *L. Braubach, J.P. Briot and J. Thangarajab (Eds.): Programing Multiagent Systems. LNAI 5919*, pages 118–132, 2010.

[28] A. Giret and V.Botti. Engineering holonic manufacturing systems. In *Computers in Industry*, volume 60, pages 428–440, 2009.

[29] 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.

[30] S. Heras, J. A. Garcia-Pardo, R. Ramos-Garijo, A. Palomares, V. Botti, M. Rebollo, and V. Julian. Multi-domain case-based module for customer support. In *Expert Systems with Applications*, volume 63, pages 6866–6873, 2009.

[31] V. Julian and V. Botti. Developing real-time multi-agent systems. In *INTEGRATED COMPUTER-AIDED ENGINEERING*, volume 11, pages 135–149, 2004.

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

[33] 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, U.K. :, 2005.

[34] 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.

[35] 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.

[36] C. Sierra, V. Botti, and S. Ossowski. Agreement computing. In *Kunstliche Intelligenz)*, number 25, pages 57–61, 2011.

[37] J. Soler, V. Julian, M. Rebollo, C. Carrascosa, and V. Botti. Towards a real-time multi-agent system architecture. In *Workshop: Challenges in Open Agent Systems. AAMAS 2002*, pages 1–11, 2002.

[38] J. M. Such, A. Espinosa, A. Garcia-Fornes, and V. Botti. Partial identities as a foundation for trust and reputation. In *Engineering Applications of Artificial Intelligence*, volume 24, pages 1128–1136, 2011.

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