D8.2.1.P3 Report: *mWater* prototype #3 analysis and design

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

Abstract.
CSD2007-0022, INGENIO 2010
Deliverable D8.2.1.P3 (WP8, Task 8.2)

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

Document Identifier	AT/2008/D8.2.1.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 institutional structure of mWater is described in Deliverable 8.2.1: mWater Analysis and Design. Deliverable 8.2.1 defines the backbone of the market in terms of dialogical and performative structures, stating the main structural regulations, processes and roles of the system. Deliverable 8.2.1.P2 specifies the analysis and design of the constituent agents that can act in the institutional market of water rights. The main focus is on the normative design and on the deliberative components of the market staff agents and the water users. On the other hand, Deliverable 8.2.1 P3 describes the Magentix design of the market in terms of conversation structures among the participating agents.

Contents

Chapter 1

mWater prototype #3 analysis and design

1.0.1 MAS platform

In the mWater case study prototype #3 it has been used Magentix2 [1, 24, 3, 22, 4, 17] (for more details on Magentix2 see WP7 Deliverables) as the MAS platform for supporting the execution of the MAS system. The platform follows the FIPA standards [14] offering a set of useful mechanisms for the agents to communicate and also tools to allow programming agents in a high level language based on the BDI model. Magentix2 is an open system which facilitates the interaction between heterogeneous agents through FIPA-ACL messages. Also complex interactions can be carried out in a flexible an open way as conversations. The platform offers special structures to allow to use such conversations by considering a set of issues:

- In each conversation there are always two roles involved: *Initiator* and *Participant*. The first is the one who initiates the conversation, and the rest of agents play the *Participant* role.

- The conversation can be seen as a direct graph where nodes represent the actions to perform in each step of the conversation and arcs represent the transition between such states.

- Those steps allow to perform some actions and they can be of different kinds, for example: *Begin*, *Final*, *Wait*, *Send*, *Receive*, *Action*, etc.

- Conversations have a unique identifier that allows to manage them individually.

1.0.2 Agents programming language

Magentix2 allows to implement agents with high reasoning skills that follow the BDI model of agents. This is possible through the use of Jason [7] as a high level programing language. It is a Java based interpreter for an extended version of AgentSpeak(L) [21] which is an abstract logic-based language, which allows to implement agents as reactive planning systems. The agents continuously execute, react to events and execute plans for those events. The proactive nature is given by the concept of goals, which can be seen as desired states of the world. Through AgentSpeak(L) agents can be defined in terms of three main elements: **Beliefs**, **Goals** and **Plans**. **Beliefs** represent the agent's vision of the current state of the world. They change continuously depending on several factors such as the percepts observed in the environment, a new message received, or by adding "mental notes", which is a concept that allows the agents to modify its own beliefs base. **Goals** express, on the other hand, the state of the world that the agent wants to reach. When adding a goal it entails the execution of a plan if it has been defined a plan to respond to such event. Finally, **Plans** allow to reach a goal trough a sequence of actions. When an *event* is produced [1], if there is a plan for responding to such event, the corresponding sequence of actions is executed, and, if there are no fails, the goal is reached. Everything happens in a reasoning cycle that determines what to do and how to do it at every moment.

1.0.3 Conversational interactions

When designing Magentix2 agents that will be programmed in Jason it is possible to specify their interactions like conversations by using the facilities of the platform. So the agents can communicate following some interaction protocol and have multiple conversation at the same time. As it has been mentioned before, those conversations can be seen with two perspectives depending on the role of the agent on it (*Initiator* or *Participant*). The *Initiator* must always create an identifier for the conversation and notify the *Participants*, inviting them to join. Each role has, for each conversation, a set of plans: one for each step of the conversation where a reasoning or a set of actions are necessary to be performed. When one of those plans in the agent code is going to be executed, the corresponding conversation is stopped in the platform waiting for it to finish and to decide which will be the next step. The managing of fails and timeouts is made by the platform. It is possible to have nested conversations in a synchronous (the parent conversation must wait for its child to finish for it to go on) or asynchronous way (the parent conversation goes on independently of its child). If it is synchronous, the corresponding timeouts must be taken into account.

[1] An event is generated due to several factors like goals addition or elimination, beliefs addition or elimination etc

1.1 mWater prototype #3 general design

The design has been done according to the Negotiation Model stated in [2] considering that the characteristics of the system fit well with the requirements for using this model. In this sense the system behaves as a market where participants own some goods or services that they are interested to trade with, playing in each moment some role. The Negotiation model is based on interactions and it owns four main structures: *Admission*, *Negotiation Hall*, *Negotiation Table*, *Agreement Enactment*, where the first one and the last one are considered as simple interactions and the rest as workflows. There are also part of the model a set of roles, they are: *guest*, *participant*, *black* and *white*, that can be grouped as "users"; on the other hand we have the "staff" roles: *mediator*, *negotiation table manager* and *legal authority* to perform all managing activities. During design those roles have been simplified in three main roles: *buyer*, *seller* and *staff*, where the two first correspond to the *black* and *white* roles of the model. The possible actions to be performed are listed below grouped by the structure of the Negotiation Model to which they belong:

- **Admission**
  1. *Accreditation*: Through this action, the users can get in to the market.
- **Negotiation Hall**
  1. *Trading tables*: To know which are the current open negotiation tables in the market and which of them the agent has been invited to.
  2. *Join to a trading Table*: Allows the agent to request entering in a negotiation table to negotiate.
  3. *Create a new trading table*: Allows the agent to request the creation of a new negotiation table.
  4. *Invite participants*: To invite possible participants to the negotiation table.
  5. *Tradable water rights*: To know which are the water rights available for negotiation in the market.
- **Negotiation Table**
  1. *Negotiate*: For establishing negotiations with other agents following some interaction protocol.
  2. *Validation*: For validating agreements on water-right transfers according to the market regulations checking formal conditions.
  3. *Agreement validation*: For validating agreements on water-right transfers according to the hydrological plan normative conventions.
- **Agreement Enactment**

1. *Contract enactment*: To register both the information related to the negotiation agreement and to the agents participation on it, if it has been successful.

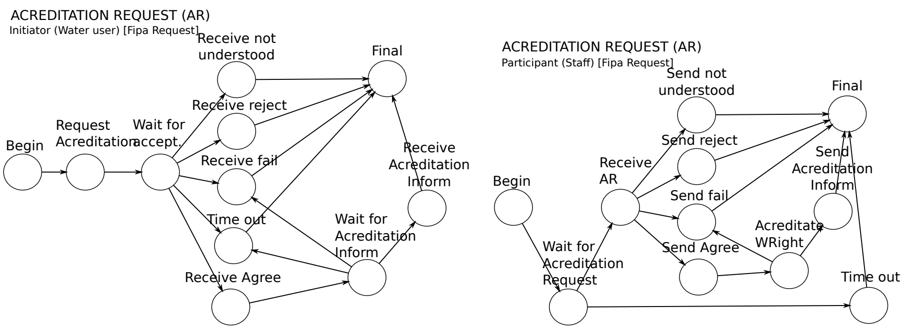
This broadly summarizes all the necessary actions in a water market. There are other actions that are also part of this kind of market, as it is the case of the grievance processes that can be included in the market; they can even cause modifications to the agreements already made. Nevertheless, this proposed actions are the foundation of a system with more features.

1.1.1 Interactions structure

For each possible feature, an interaction or also conversation is performed between *staff* and *users*. This interactions are going to be described further by showing both the *initiator* and the *participant* perspective in each one. An oriented graph where nodes represent the states in the conversation and arcs represent transitions between the states, is going to be used for showing the interactions from both perspectives.

Accreditation

The accreditation interaction follows a FIPA-Request protocol. The *user* starts requesting to the *staff* its accreditation; in response it can receive an *Agree*, *Not Understood*, *Reject*, *Fail*, or maybe it can receive no answer after a *Time out*. If the answer is an *Agree* the next message from the *staff* must be the *Accreditation information*, or, on the other hand, it can receive a *Fail* message (if during the execution something has failed) or no answer. In any other case the conversation finishes. Figure 1.1 shows this interaction.

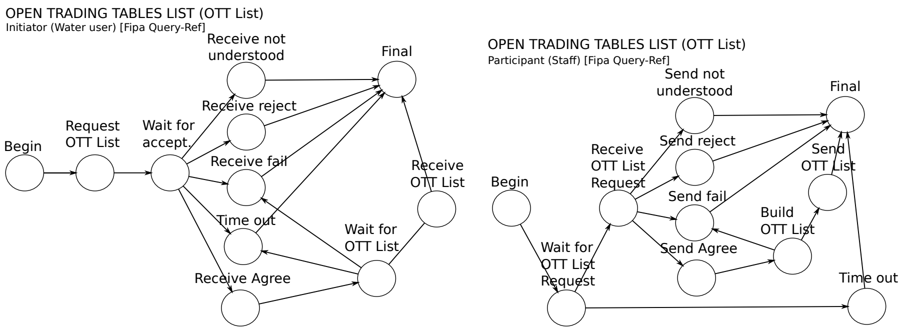


(a) (b)

Figure 1.1: Accreditation interaction from the initiator (a) and participant (b) perspective.

Trading tables

This interaction follows a FIPA-Query Ref protocol. It behaves similar to FIPA-Request protocol, but in this case, if the participant (or *staff*) sends an *Agree* to the *user*, the next step should be *Build OTT List* for building the open trading tables list and then, through the action *Send OTT List*, it sends the results to the user. The states of the interaction are shown in figure 1.2.

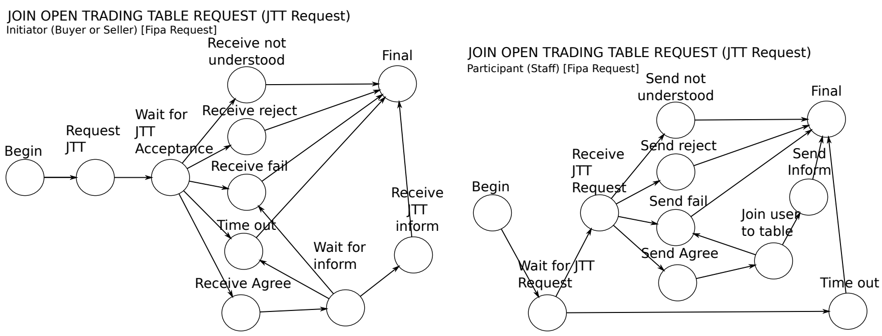


(a) (b)

Figure 1.2: Trading tables interaction from the initiator (a) and participant (b) perspective.

Join to a trading Table

This interaction follows a FIPA-Request protocol [2] . The *staff* performs the necessary actions for making the *user* to become member of the trading table. The states of the interaction are shown in figure 1.3.



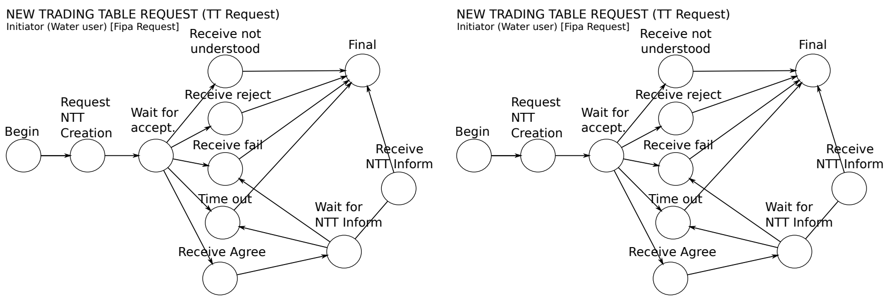
(a) (b)

Figure 1.3: Join to a trading table interaction from the initiator (a) and participant (b) perspective.

[2]For more details see "Accreditation" interaction

Create a new trading table

This interaction follows a FIPA-Request protocol [2] . The *staff* performs the necessary actions for creating a new trading table. In this case there is a variation for this protocol: after sending the information related with the creation of the table, the *staff* starts the subprotocol for inviting the participants. The states of the interaction are shown in figure 1.4.

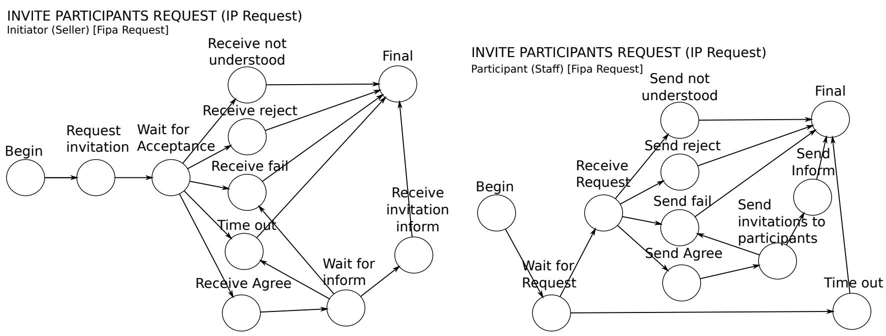


(a) (b)

Figure 1.4: Create a new trading table interaction from the initiator (a) and participant (b) perspective.

Invite participants

This interaction follows a FIPA-Request protocol [2] . The *staff* performs the necessary actions for inviting the participants of the trading table to join to it. The states of the interaction are shown in figure 1.5.

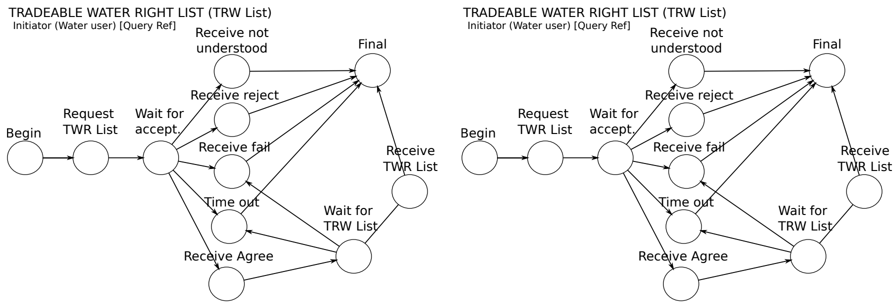


(a) (b)

Figure 1.5: Invite participants interaction from the initiator (a) and participant (b) perspective.

Tradable water rights

This interaction follows a FIPA-Query Ref protocol [3] . In this interaction the *staff* performs the necessary actions for building the list of tradable water rights in the market and send it to the requesting *user*. The states of the interaction are shown in figure 1.6.

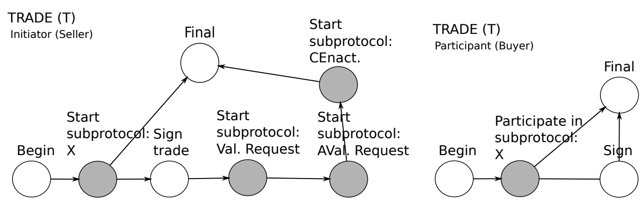


(a) (b)

Figure 1.6: Tradable water rights interaction from the initiator (a) and participant (b) perspective.

Negotiate

This interaction doesn't follow a standard protocol. Instead it contains other sub interactions that can follow some protocol specification. In this case the *seller* basically trades with the participants starting the sub interaction for doing it, then, if the trading has been successful, it signs the contract, or, on the other hand, the conversation finishes. Later it starts interactions with the *staff* for validating the contract according to the market regulations and also according to the hydrological conventions. Finally it starts the sub interaction for the contract enactment also with the *staff*. In this interaction the *buyer* is one of the participants. It interacts with the *seller* in the subprotocol for negotiate, and, if the trade was successful it signs the contract. The *staff* as a participant merely responds to the interactions with the *seller*. Figure 1.7 shows the interaction from the *seller* and *buyer* perspective.



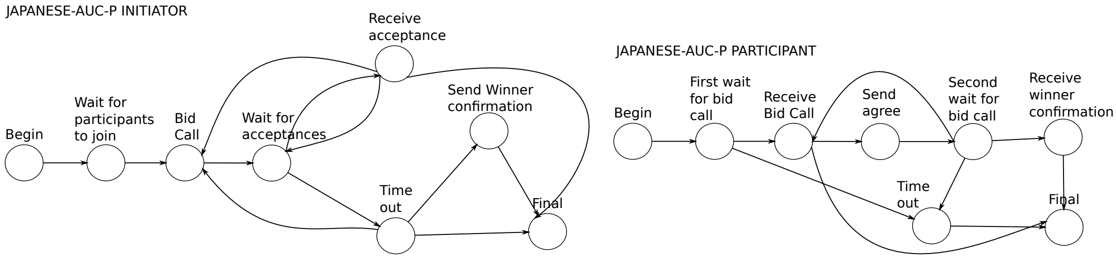
(a) (b)

Figure 1.7: Interaction for trading from the initiator (a) and participant (b) perspective.

[3]For more details see "Trading tables" interaction

Figures 1.8 and 1.9 represent the two possible interactions to trade that have been considered in this version of the system. The first one represents a Japanese auction. In this case the *initiator* (or *seller*) makes a *Bid Call* to the participants (or *buyers*) proposing a price and waiting then for them to accept the price. After the timeout stated for each round, if there is more than one who has accepted, the *initiator* starts another round with an incremented price. On the contrary, if there was an only one *participant* who has accepted, the *initiator* sends it the *Winner confirmation*; if nobody accepts, the conversation finishes. There is also a limit for the number of rounds to perform.

Figure 1.9 represents a Contract Net interaction. The first thing the *initiator* does is a *Call for proposals* for the *participants* to bid up. From each *participant* it can receive a *Not understood*, *Refuse* or a *Propose*, or maybe no answer after a *Timeout*. After receiving all proposals or after a timeout, it evaluates them following some criteria and sending the corresponding *Acceptances* and *Rejections*. In the *participant* side, if it receives an acceptance, it performs some *Task* associated to it sending the confirmation to the *initiator* if it succeeds or a *Failure* on the other hand. Then the initiator receives a *failure* or an *Inform* depending on the case.



(a) (b)

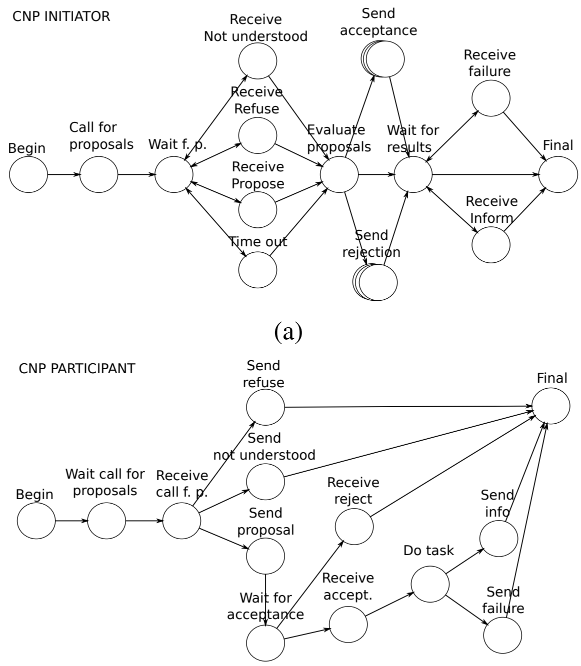
Figure 1.8: Japanese auction from the initiator (a) and participant (b) perspective.

Validation

This interaction follows a FIPA-Request protocol [2] . The *staff*) performs the necessary actions for validating the contract according to the market regulations. There is a small variation in the protocol: in this case, after trying to validate the contract, the *staff* can send an acceptance or a rejection. The states of the interaction are shown in figure 1.10.

Agreement validation

This interaction follows a FIPA-Request protocol [2] . The *staff* performs the necessary actions for validating the contract according to the market regulations. We found the same variation in the protocol: after trying to validate the contract, the *staff* can send an acceptance or a rejection. The states of the interaction are shown in figure 1.11.



(a)

(b)

Figure 1.9: Contract Net interaction from the initiator (a) and participant (b) perspective.

Contract enactment

This interaction doesn't follow a standard protocol. Basically, the *staff*, instantiated now as "Basin authority", sends to *seller* the contract to be signed. If it receives the confirmation it goes on doing the same thing with the *buyer*. Instead of the confirmation, the *staff* could receive *Not understood*, *Reject*, *Fail* or maybe no answer after a *Timeout*. In any of those cases the interaction finishes. Figure 1.12 shows the steps of the interaction from both perspectives.

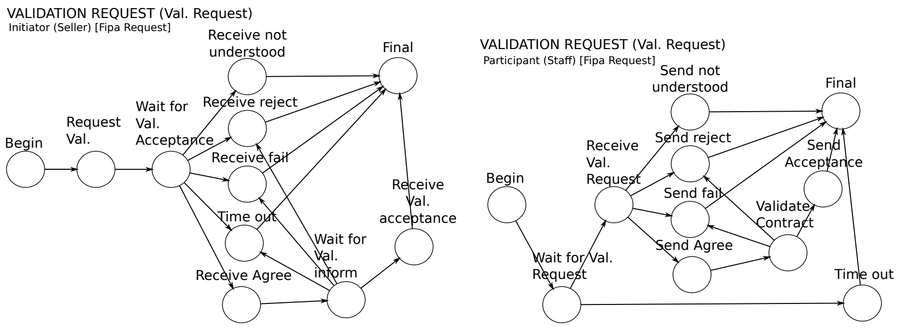
1.2 mWater prototype

1.2.1 Design description

For designing the prototype there have been selected a subset of the functionalities listed in 1.1 as part of the general design. In this case "Contract enactment" has been renamed as "Register transfer agreement" but has the same purpose. This functionalities are:

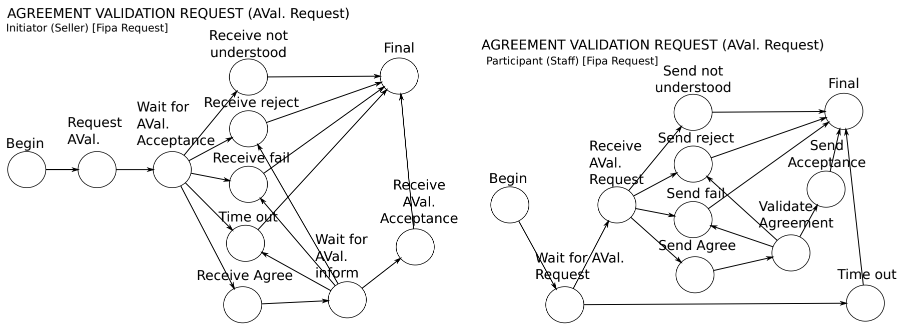
- **Admission**

    1. *Accreditation*.



(a) (b)

Figure 1.10: Interaction for validating the contract according to market regulations from the initiator (a) and participant (b) perspective.

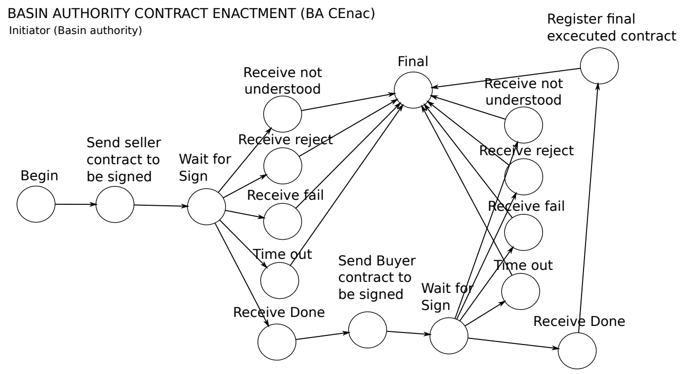


(a) (b)

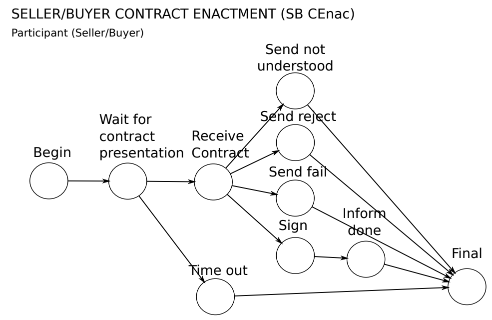
Figure 1.11: Interaction for validating the contract according to hydrological plan conventions from the initiator (a) and participant (b) perspective.

- **Negotiation Hall**
  1. *Negotiation trading tables*.
  2. *Join to a Negotiation Table*.
  3. *Create a new negotiation table*.
  4. *Invite participants*.
- **Negotiation Table**
  1. *Negotiate*.
- **Agreement Enactment**
  1. *Register transfer agreement*.

Further it will be shown a model that links those interactions in a model for generating source code.



(a)



(b)

Figure 1.12: Interaction for trading from the initiator (a) and participant (b) perspective.

1.2.2 Interactions model

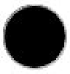
Taking into account the features mentioned in 1.2.1 it has been used the modeling tool presented in [13] to create the model for the prototype and to obtain the corresponding code. Figure 1.13 shows the model. It allows the code generation for the *staff* agent, so the elements in the model represent the actions to be performed by the *staff* when interacting with the "users" agents. For the comprehension of the model it is necessary to know some main issues related to it, but for more details refer to [13]. The elements of the Negotiation Model presented in [2] have been mapped into the modeling tool and some other have been added. Those elements are described in table 1.1:

The first process once the prototype starts is a *SingleAction* called "createConfiguration". In this action a new configuration is created. It allows to explore, for an specific execution, all the results and operations performed given a set of parameters. This parameters are a requirement to create the configuration which is associated to all the operations to be performed, so they are part of the conditions of the *SequenceLink* from the initial state to *createConfiguration* action. Next the agent can be accredited either as a *buyer* or *seller*. For doing this, it will initiate a Request interaction with the *staff*. In this interaction the *staff* performs some checks in the corresponding steps of the protocol as participant of the conversation. As result, the agent gets the states "accredited" and "inTradingHall"

SingleAction: Represents an action to be performed. During the code generation it is transformed in a peace of code that forms a plan.

ActionsFlow: Represents a workflow in the system. It's a kind of action and it can be defined by a model with all the elements in the diagram, even other workflows, so it can be contained within itself.

InitialState: The starting point to get into the container workflow.



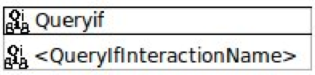
FinalState: The ending point to get out of the container workflow.

State: It is associated to actions. It can represent the state of the agent to execute an action or, on the other hand, the resulting state for the agent after executing the action, depending on the direction of the link between the action and the state.

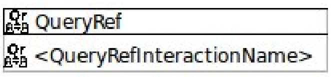
StateLink: For representing the relations *state-action* and *action-state*.

SequenceLink: For representing the relations *action-action*, *InitialState-Action* and *Action-FinalState*.

RequestInteraction: Represents a request interaction according to FIPA standard interaction protocols.



QueryIfInteraction: Represents a Query-If interaction according to FIPA standard interaction protocols.



QueryRefInteraction: Represents a Query-Ref interaction according to FIPA standard interaction protocols.

Table 1.1: This table shows some data

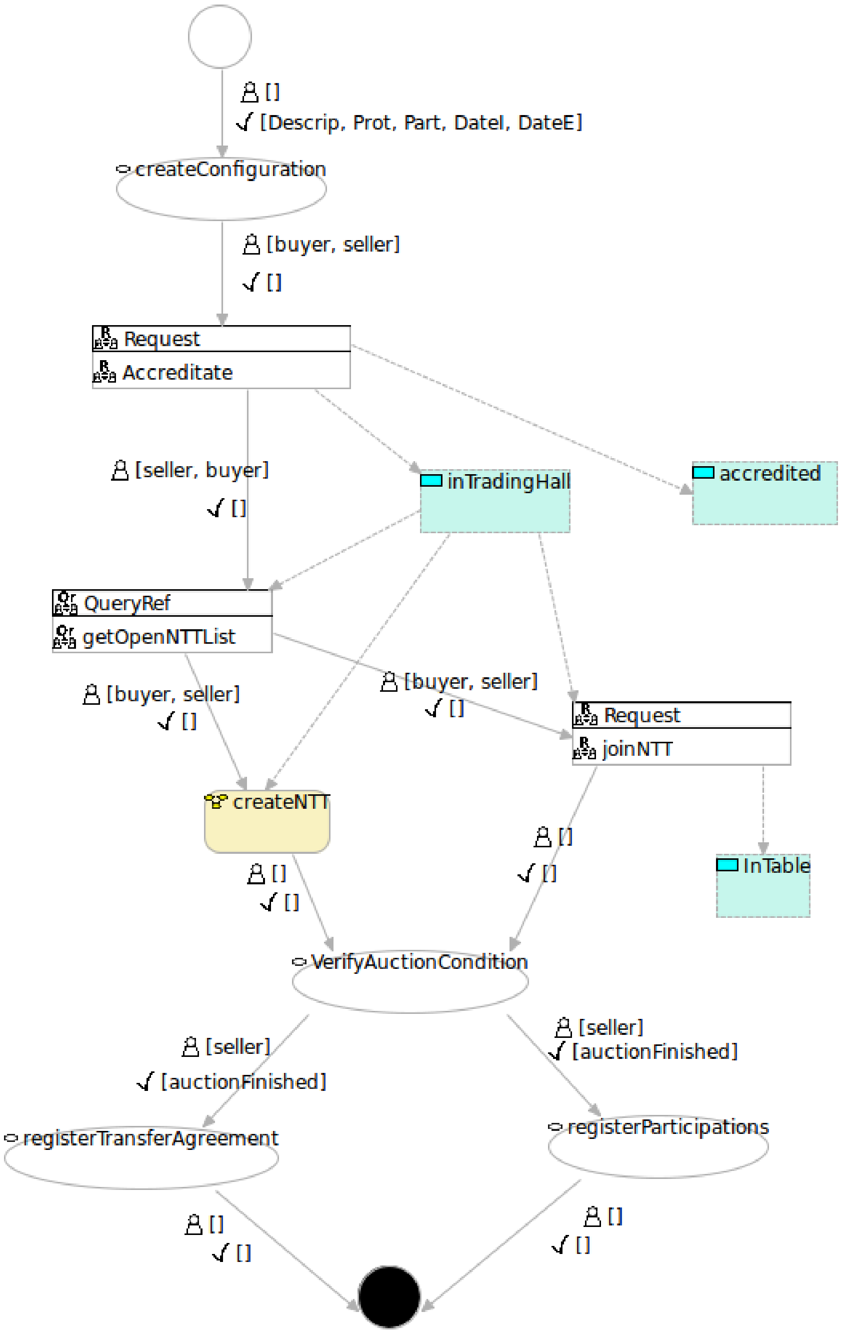


Figure 1.13: Model for mWater workflow.

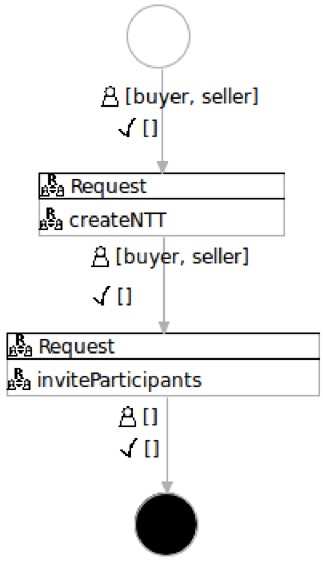


Figure 1.14: Model for createNTT workflow.

that are also represented in figure 1.13. As shown, this last state is a requirement to initiate the interaction for getting the list of active negotiation trading tables (getOpenNTTList), and also for joining a negotiation table (joinNTT) and to create a new negotiation table (createNTT). In every interaction the agent who starts it, can participate both as *buyer* and *seller*. Once the agent knows the open negotiation tables it can decide to join a table to which it has been invited or to create a new one. If it joins a table, it gets the state "InTable". Further the *staff* will check if the conditions to start a negotiation are fulfilled (VerifyAuctionCondition); it executes this action each time a new table is created or a participant joins to an existing table, performing the necessary actions to notify the table owner that the trading must begin. When the negotiation finishes the *seller* provides the *staff* the information related to the trading and to the participants. the actions to register this information are performed through the actions: "registerTransferAgreement" and "registerParticipations".

Figure 1.13 shows that the action "createNTT" is an *ActionsFlow*. This means that this action contains a sequence of actions similar to the one represented in the model in which it is contained and that they in turn define a new model. Figure 1.14 shows this model. It includes basically two interactions: one for creating a negotiation table and the other one for inviting the specified participants. This models don't take into account the possible negotiations to be performed between the agents because this is implemented in the individual reasoning of each agent. Nevertheless some templates have been created in order to be used by the programmers of the agents for them to negotiate following both the FIPA Contract Net protocol and the Japanese Auction in this first version. Section 1.1.1 shows in more detail the structure of this kind of interactions.

1.3 Conclusions

In this deliverable it has been presented the analysis and design of mWater prototype #3. The new features of prototype #3 were motivated by the need to scale and adapt mWater to the results of Task 7.2, and to include new requirements from Task 4.5. In summary the work presented here is a Magentix2 design of mWater based on conversations. The backbone of mWater prototype #1 is maintained, also the features added in prototype #2. All this prototype where designed and developed applying our previous research [10, 9, 12, 8, 11, 5, 15, 18, 16, 6, 19, 20, 23]

Bibliography

[1] 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.

[2] B. Alfonso, V. Botti, A. Garrido, and A. Giret. A mas-Based Infrastructure for Negotiation and its Application to a Water-Right Market. In *Third International Workshop on Infrastructures and Tools for Multiagent Systems*, Valencia, 2012.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] B. A. Espinosa. Aagentes bdi Bajo Interacciones Reguladas: Un Enfoque Basado en Flujos de Trabajo. Master's thesis, Universitat politécnica de València, 2012.

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

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

[16] 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.

[17] 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.

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

[19] 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.

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

[21] A. S. Rao. AgentSpeak(L): BDI Agents Speak Out in a Logical Computable Language. In R. van Hoe, editor, *Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, Eindhoven, The Netherlands, 1996.

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

[23] 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.

[24] 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.