D8.2.1.P2 Report: *mWater* prototype #2 analysis and design

Vicente Botti (UPV), Natalia Criado (UPV), Antonio Garrido (UPV), Juan A. Gimeno (UPV), Adriana Giret (UPV), Pablo Noriega (CSIC)

Abstract.
CSD2007-0022, INGENIO 2010
Deliverable D8.2.1.P2 (WP8, Task 8.2)

The mWater prototype #2 analysis and design is detailed in this report.
Keyword list: mWater, e-market, analysis and design

Document Identifier	AT/2008/D8.2.1.P2/v0.1
Project	CSD2007-0022, INGENIO 2010
Task	T8.2
Version	v0.1
Date	Dicember 03, 2010
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	03.12.10	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. On the other hand, this report (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.

Contents

Chapter 1

mWater structure

mWater uses a multi-tier architecture, as depicted in Fig. 1.1. In addition to the three typical tiers of presentation, business and data persistence, we have a module that represents the Electronic Institution (EI) for *mWater*. This way, the construction of *mWater* consists of four stages: i) modelling the system as an electronic institution; ii) designing the information system based on a database of the entire electronic market and basin structure (persistence tier); iii) implementing the agents (business tier); and iv) deploying the GUI for simulation tool (presentation tier), which are described next.

1.0.1 Modelling the system as an EI

We have followed the IIIA EI conceptual model [AEN+05], whereas for the actual specification and implementation we have used the EIDE platform[1]. The *mWater* institution is specified through a nested performative structure with multiple processes, as depicted in Fig. 1.2 (see Deliverable 8.2.1 for further details). There are five agents' roles: i) guests, i.e. users before entering the market; ii) water users, i.e. the guests that have valid water rights; iii) buyer/seller, thus representing the particular role the water user currently joins for the market; iv) third parties, i.e. those water users that are direct or indirectly affected by a water transfer —usually conflicting parties; and v) market facilitator and basin authority, thus representing the governing roles of the market. The top structure describes the overall market environment and includes the following elements:

- Entitlement, which represents the bootstrap routine to give access to the market

[1]EIDE is a development environment for Electronic Institutions, implemented at the IIIA (`http://e-institutor.iiia.csic.es/eide/pub`). It consists of a set of tools that support all the stages of EI engineering, namely: i) ISLANDER, a tool for EI specification; ii) aBUILDER, a tool to support the automatic generation of agent (code) skeletons from ISLANDER specifications; iii) the AMELI middleware that handles the enactment of the institution; and iv) SIMDEI, a testing and monitoring tool. We have also used the following methods during the system design [GB04a, JB04, SJR+02, GB04b, AGV+04, GB06, GJR+10, GV09, APA+07, BJC+06]

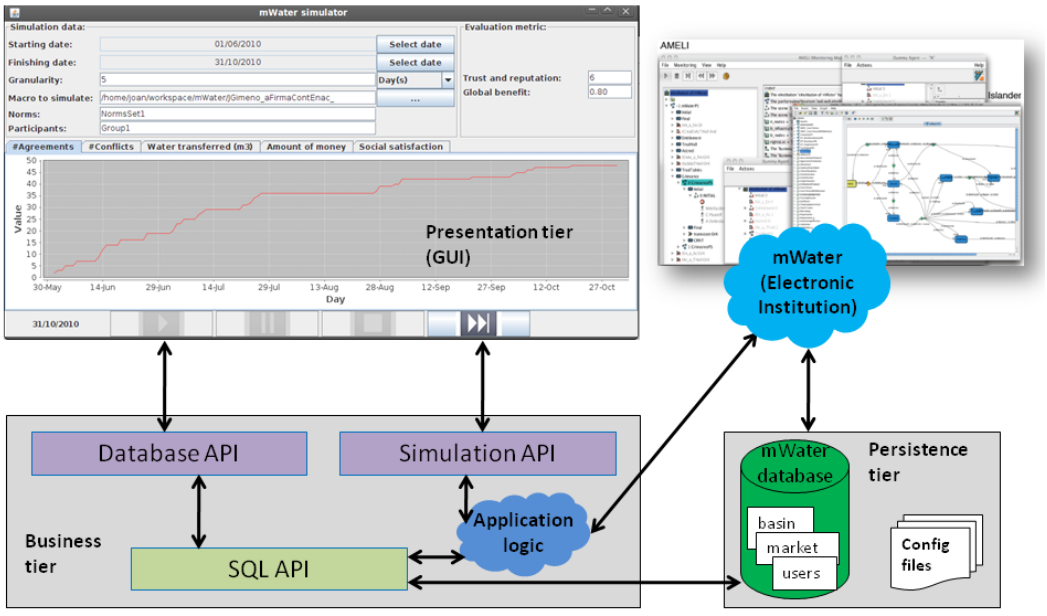

Figure 1.1: Multi-tier architecture of the *mWater* decision support tool

to those water-right holders who prove they are entitled to trade because: i) they have an existing right, or ii) a new right is created by the *mWater* authorities and an eligible holder gets it granted.

- Accreditation, which allows legally entitled water-right holders to trade by registering their rights and individual data for management and enforcement purposes.

- TradingHall, which represents a nested performative structure. It basically provides information about the market and, at the same time, allows users and trading staff to initiate trading and ancillary operations.

- TradingTables, which represent a nested performative structure and the core of our market. It allows a market facilitator to open a new trading table whenever a new auction period starts (i.e. automatically) or whenever a right-holder requests to trade a right (i.e. on demand). Our implementation accommodates different trading mechanisms and negotiation protocols, such as Dutch auction, English auction, standard double auction and blind double auction with mediator negotiation, but new negotiation protocols can be easily included.

- Agreement Validation, which validates agreements on water-right transfers according to the market regulation. More particularly, staff have to check whether the agreement satisfies formal conditions and the hydrological plan normative conventions.

- Contract Enactment, which represents the signature among parties involved in a norm-abiding agreement, thus making the agreement active.

- Grievances, which represent a nested performative structure. It allows external stakeholders to initiate a grievance and conflict resolution procedure that may overturn or modify an active agreement. Even if there are no grievances that modify a contract, parties might not fulfill the contract properly and there might be some contract reparation actions.

- Annulment, which deals with anomalies that deserve a temporary or permanent withdrawal of water rights.

The essence of our market relies on the Trading Tables and Grievances structures. The former implements the trading process itself, which entails the participation of the buyer/seller and staff agents. Since the agreement execution may eventually turn conflicting with third party agents, the grievances structure is necessary to allow normative conflicts to be solved within the *mWater* institution.

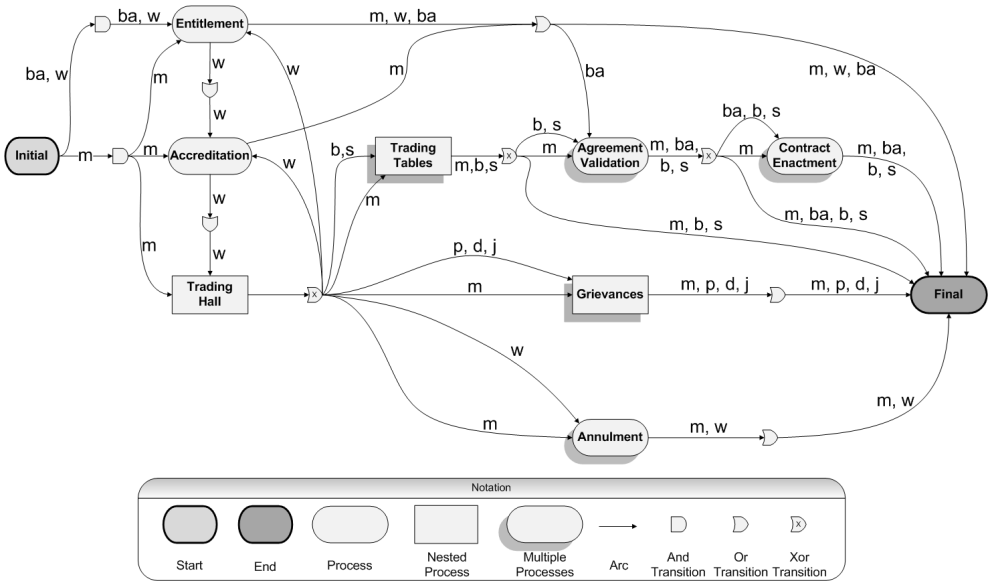

Figure 1.2: *mWater* performative structure. Participating roles: *g* - guest, *w* - water user, *b* - buyer, *s* - seller, *p* - third party, *m* - market facilitator, *ba* - basin authority

1.0.2 Storing the Information. Database Design

mWater implements the persistence tier by means of a MySQL database with over 50 relational tables in which historical data is stored (see Fig. 1.3). In essence, we have three views that comprise the basin, market and grievance structure. In the first view we model all the information about the nodes, connections, users, norms and water-right definition. In the second view we model information related to the entire market, including the trading tables and their protocols, the water rights to be traded, participants, agreements and contracts that can be signed. Finally, in the third view we model the information about the legislation and conflicts that may appear after an agreement or contract and the mechanisms for solving such a conflict, that is the negotiation stage or arbitration procedure. This way, policy makers can run the whole market with real and simulated data for drought periods, rainfall, norms and users, and analyse how they affect the final results and the number of grievances. Furthermore, all the changes in the market are registered in the database to provide statistical information and/or distributions to the policy makers, which are essential in a decision-support tool.

1.0.3 Implementation of Agents

mWater implements a schema of agents that include both the internal and external roles. Broadly speaking, there is a JADE (Java Agent DEvelopment Framework[2]) definition for each class that represents the roles in the scenes. The generation of the Java classes is

[2] `http://jade.tilab.com`

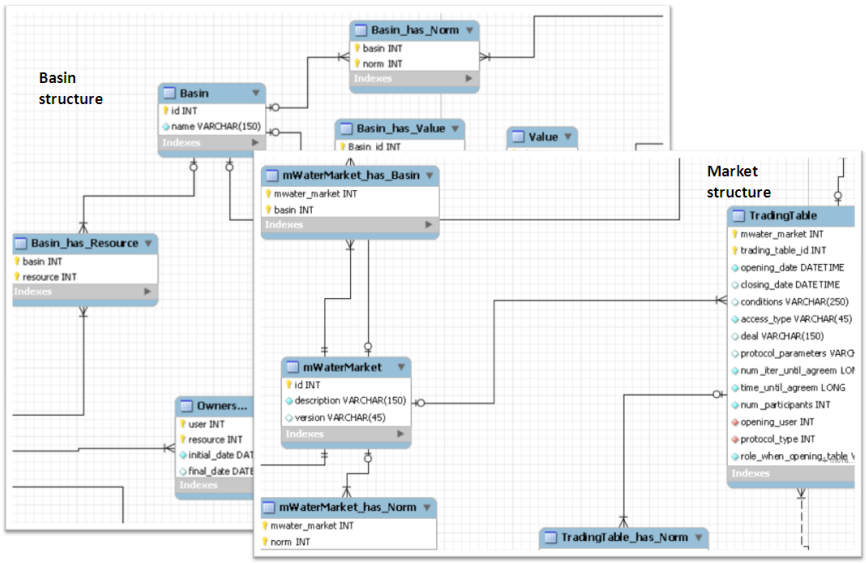

Figure 1.3: Fragment of the database: basin and market structure

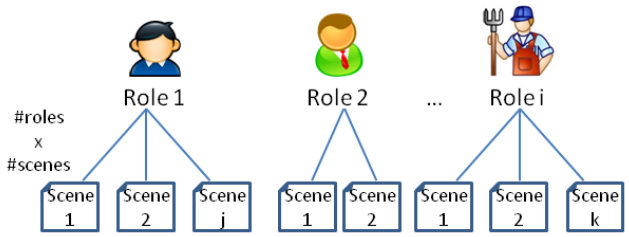

Figure 1.4: Schema of the agents implementation. The mapping proceeds by generating one Java class per role in each scene it can be involved

done in an automated way, thanks to the tools provided by the EIDE development environment. More particularly, the mapping that is used to generate the agents implementation is shown in Fig. 1.4. In particular, one Java class is created per valid role (guest, water user, buyer, seller, third party, market facilitator and basin authority) and per scene in which each role can participate. Intuitively, this can be seen as a basic template for an agent participating in a given scene. It is important to note that not all roles participate in all the scenes —see the definition of the *mWater* EI in Fig. 1.2—, so there are roles that are translated into more classes than others. The main idea with this is to offer open and flexible templates to implement different agents and norms, which provides more opportunities to the user to evaluate the market indicators under different regulations and types of agents.

Once the templates have been automatically generated, we can extend them by implementing new classes that represent different behaviours, which is interesting from a simulation perspective. Basically, we can override methods to change the original behaviour

that allows the agent to move from one state to another, i.e. to execute a transition, or send a message (interact) to other agents. For instance, in the case of the buyer/seller we have implemented a *favourable* and *unfavourable* behaviour. In the former, the agent is always in favour of achieving an agreement to trade and follow the norms of the market, whereas the latter is always against it and does not follow the rules. Note that we have also two alternatives for norm enforcement [CAG+10]. The former is to implement this reasoning process in the institution side, making it impossible for an agent to violate the norms. Although this provides a trustful and safe environment, it is less flexible and forces the implementation of the agents to be more aware of the legislation of the institution. Moreover, in real life problems, it may be difficult or even impossible to check norm compliance, specially when the violation of the norm cannot be directly observable. And perhaps, it might be preferable to allow agents to violate norms, since they may intend to improve the organization functionality, despite violating or ignoring norms. On the contrary, the second alternative moves the norm reasoning process to the agent side, thus making the system more open and dynamic. In this case, the intelligence of the agent can make it more or less law-abiding in order to obtain a higher personal benefit. If a norm is violated and a third party is affected, the grievance mechanism activates and the conflict resolution stage modelled in the EI is launched.

In the following Chapters the analysis and design of the deliberative components of these agents are detailed.

1.0.4 Simulation Tool

The interface of *mWater* as a simulation tool is simple and intuitive, as shown in Fig. 1.5. The idea is to offer a straightforward and effective way in which the user configures a given simulation with the following data: i) the starting and finishing date for the period to be simulated; ii) the water users that will participate in the market (different groups/type of water users lead to different results; e.g. a group in which water users do not trust other members of the group results in a low number of agreements and a high number of conflicts); and iii) the regulation to be applied in the current simulation. The tool outputs graphical statistical information that indicates how the market reacts to the input data in terms of the number of transfer agreements signed in the market (historical data including information about real or simulated users), volume of water transferred, number of conflicts generated, etc. Apart from these straightforward parameters, the tool also shows different quality indicators based on "social" functions in order to asses values such as the trust and reputation levels of the market, or degree of water user satisfaction, among others.

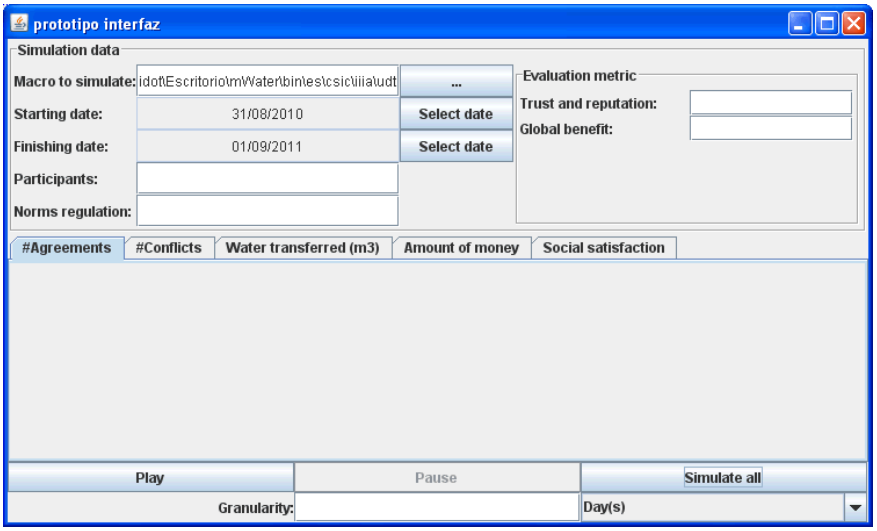

Figure 1.5: The *mWater* simulator in action.

Chapter 2

A Normative module for staff agents

In this chapter the analysis and design of *mWater* Staff Agents are presented. We focus on the normative module for *Market Facilitator* and *Basin Authority* roles, for reasoning on new and or modified norms that are not included in the structural definition of the Electronic Institution.

2.1 General Overview

In order to allow the simulation of different regulations and to observe their effects on the behaviour of the water right market we require a deliberative module that can reason on the regulation and acts upon them. The staff roles of *mWater* that enforce norm execution are *Market Facilitator* and *Basin Authority*. In an ideal execution of these roles, they receive messages and perceive events, determine what are the legal actions to perform and finally execute one of these actions. This execution process is repeated in every scene whenever any *Water User* requests or performs an action. But, what are the legal actions that can be performed? Some of them are defined in the structure of the Electronic Institution of *mWater* (see Deliverable 8.2.1 for more details), while others will be defined in a dynamic fashion whenever a new simulation is configured. For this last type of norm processing we define the normative module depicted in Figure 2.1.

Figure 2.1 shows the general design of a staff agent in *mWater*. The agent perceives events of the environment (*Perception*) and acts upon (*Actuation*). In order to act it reasons about the perceived event, the current status of the environment (*Situational Awareness*) and the regulation defined. The *Norm Base* includes all the norms that apply in the current simulation. The *Plan Base* defines the possible action sequences that can be executed in the given scene. The *Deliberative and Decision Module* queries the *Norm Base*, the *Plan Base* and the *Situational Awareness* in order to evaluate what action to perform.

In next sections the normative module and the deliberation process carried out by the staff agents are detailed.

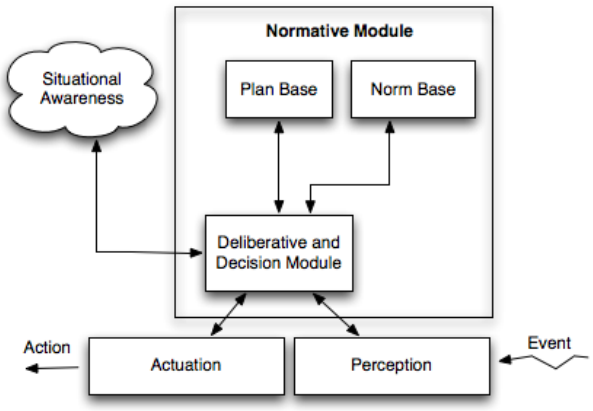

Figure 2.1: General Architecture of mWater Staff Agent

2.2 Normative Module

2.2.1 Logic Preliminaries

Let us suppose the existence of a first order language $\mathcal{L}$ whose alphabet includes: the logical connectives $\{\wedge, \vee, \neg, \rightarrow\}$; parentheses, brackets, and other punctuation symbols; and an infinite set of variables. These variables are implicitly universally quantified. In addition, the alphabet contains non-logical predicate, constant and function symbols. The set of predicate symbols is formed by *action* predicates ($\mathbb{X}$) and *state* predicates ($\mathbb{P}$), which describe properties of the world and the institution. Since we have implemented the *mWater* prototype as an Electronic Institution, we consider actions as speech acts between two or more parties. Thus, the set of action predicates $\mathbb{X}$ contains all the illocutions available to the agents in the *mWater* institution. Let us also assume the standard definition for *wffs* (well-formed formulas). Thus we make use of the standard notion of substitution (σ) of variables in a *wff*, where σ is a finite and possibly empty set of pairs Y/y where Y is a variable in a *wff* and y is a term. If the *wff* obtained from applying a substitution has no one variable then it is defined as fully grounded, and as partially grounded otherwise. Finally, the symbol Γ denotes a formal theory which is a set of sentences in the formal language $\mathcal{L}$. Γ is defined as a deductive theory since its content is based on some formal deductive system (denoted by the deductive relation $\vdash$) and that some of its elementary statements are taken as axioms. Thus, any sentence which is a logical consequence of one or more of the axioms of Γ is also a sentence of the theory Γ.

In order to represent uncertainty we define a formal language $\mathcal{L}_u$ which is formed by propositions such as (γ, ρ) where γ is a logic formula in $\mathcal{L}$, and $\rho \in [0, 1]$ is a real number representing the certainty of this proposition.

2.2.2 The situational Awareness

The market status information is defined by the data stored in the information model of *mWater* and the values of the Electronic Institution variables. In order to access to this information the staff agents execute queries to the informational model and consult the values of the EI variables. What data is retrieved at every moment will depend on the particular execution time and the specific scene in which the staff agent is in.

Thus, we assume that all staff agents have theory of beliefs ($\Gamma_B \subseteq \mathcal{L}_u$) that contains the market status information. Moreover, beliefs can also include inference rules, allowing forward chaining to lead to new beliefs. Similarly, we consider that each agent is endowed with a theory of desires (Γ_D) that represent objectives or situations that the agent would like to accomplish or bring about. Finally, staff agents are endowed with a theory of intentions (Γ_I) which are desires to which the agent has to some extent committed. This set of intentions is initially empty. When the agent selects some plan to achieve one or more of its desires then this instantiated plan becomes an intention.

2.2.3 The Plan Base

In order to define the set of possible action sequences that a staff agent can perform in the different scenes we have defined a knowledge base that includes all these complete or partial plans. When dealing with Electronic Institutions this is not a very hard task since the possible execution sequences are defined by the performative structures.

Definition 1 (Plan) *Formally, a plan p is defined as a tuple $\langle \Sigma, \gamma, \delta_c, \delta_r \rangle$ where:*

- $\Sigma = \{\alpha_1, ..., \alpha_n\}$ *is a set of actions that compose the plan. Since actions are illocutions, each α_i is a fully grounded illocution expression in $\mathbb{X}$;*
- $\gamma \in \mathcal{L}$ *is the goal achieved by the plan;*
- $\delta_c, \delta_r \in [0, 1]$ *are real values that represent the cost and the risk of the plan, respectively.*

Any staff agent has a theory of plans Γ_P, which is a set of plans $\{p_1, ..., p_n\}$ where each p_i is a plan defined as above.

2.2.4 The Norm Base

In [OPVS$^+$09], a distinction among *abstract norm* and *norm instance* is made. According to this, an *abstract norm* is a conditional rule that defines under which conditions obligations, permissions and prohibitions should be created. In particular, the

activation condition of an abstract norm defines when an obligation, permission or prohibition must be instantiated. The *norm instances* that are created out of the *abstract norms* are a set of unconditional expressions that bind a particular agent to an obligation, permission or prohibition. Moreover, a norm instance is accompanied by an expiration condition that defines the validity period or deadline of the norm instance [CAJB09, CAB09, CAG+10, CAB10b, CJBA10, HGPRG+09].

Following this proposal, our definition of both abstract norms and norm instances is provided.

Definition 2 (Abstract Norm) *An **abstract norm** is defined as a tuple $n_a = \langle D, A, E, C \rangle$ where:*

- *$D \in \{\mathcal{F}, \mathcal{O}\}$ is the deontic modality of the norm. In this work, obligations ($\mathcal{O}$) and prohibitions ($\mathcal{F}$) impose constraints on agent behaviours. We use a closed world assumption where everything is considered as permitted by default. Therefore, permissions are not considered in this work, since they can be defined as normative operators that invalidate the activation of an obligation or prohibition.*
- *A is a wff of $\mathcal{L}$ represents the norm activation condition. It defines under which circunstancies the abstract norm is active and must be instantiated.*
- *E is a wff of $\mathcal{L}$ is the norm expiration condition, which determines when the norm no longer affects agents.*
- *C is a wff of $\mathcal{L}$ represents the state of affairs or actions that are obliged, permitted, or forbidden.*

Since this work is focused on the norm compliance problem, only norms that affect the staff agents will be considered. Thus, for simplicity, we will omit the target of a norm since those norms addressed to other agents will not be taken into account by the reasoning process. Moreover, no enforcement mechanisms; i.e., sanctions and rewards, is required since the staff agents are internal agents that have been implemented as norm-oriented agents that always comply with the norms of the *mWater* institution.

For example, in case of water scarcity the *Basin Authority* should invalidate all water transfers:

$$\langle \mathcal{F}, waterScarcity \wedge validateAgr(A), \neg waterScarcity, v\_ok(A), -, - \rangle$$

In the *mWater* institutions norms are labelled with a degree that represents the priority of the norm. Therefore, all staff agents are endowed with a theory of abstract norms Γ_N which is formed by expressions such as (n, ρ) where n is a norm (according to definition 2) and $\rho \in [0, 1]$ is the priority of the norm. Thus, Γ_N contains all the abstract norms that are involved with the decisions that those agents should take.

Once the activation conditions of an abstract norm hold, it becomes active and several norm instances (according to the possible groundings of the activation condition) must be created. A norm instance is defined as:

Definition 3 (Norm Instance) *Given a theory Γ, an abstract norm $n_a = \langle D, A, E, C \rangle$ is instantiated into a **norm instance** $n_i = \langle D, C' \rangle$ where:*

- $\Gamma \vdash \sigma(A)$*, where σ is a substitution of variables in A such that $\sigma(A)$, $\sigma(C)$, $\sigma(E)$ are fully grounded;*
- $C' = \sigma(C)$.

For simplicity ,we assume that once a norm is being instantiated both it is fully grounded. Defining v_A, v_E, v_C as the set of variables occurring in the different components of an abstract norm ($\langle D, A, E, C \rangle$), thus these variables may fulfil $v_A \subseteq v_E \cup v_C$ in order to ensure that all norm instances have not free variables.

Each norm instance is also labelled with a degree which might be interpreted as the salience of the norm instance. Therefore, it is assumed that each agent has a theory of norm instances Γ_{NI} which contains formulas as (n, ρ) where n is a norm instance (according to definition 3) and $\rho \in [0, 1]$ is the salience of this concrete instantiation. Thus, Γ_{NI} contains all the abstract norms that are involved with the decisions that those agents should take.

Figure 2.2 shows the logical model of the *Norm Base* , this is a fragment of the informational model described in Section 1.0.2. The definition of a norm corresponds to the notion of abstract norm previously presented (see Definition 2). Thus, it is not necessary to represent them explicitly in norm instances (see Definition 3). The NormType is a value from the enumeration: *Obligation*, and *Prohibition*. Every norm is associated to a set of ontology concepts defined in the *mWater* information base. The association relation is defined when a given norm affects to the ontology concept. For example, a norm N1 affects a basin B3 means that N1 is part of the regulation of B3.

2.3 The deliberative and decision module

Usually, proposals on agent architectures which support normative reasoning consider norms as static constraints that are blindly obeyed by agents. Thus, these norm-oriented agents are not able to learn new norms and taken into account these unforeseen norms. In order to overcome this drawback, in [CAB10a, CAB10c] the n-BDI architecture has been presented. It is an extension of classical BDI architecture with recognition and normative reasoning capabilities. Thus, n-BDI agents are capable of detecting the activation of norms and selecting those plans that obey active norms.

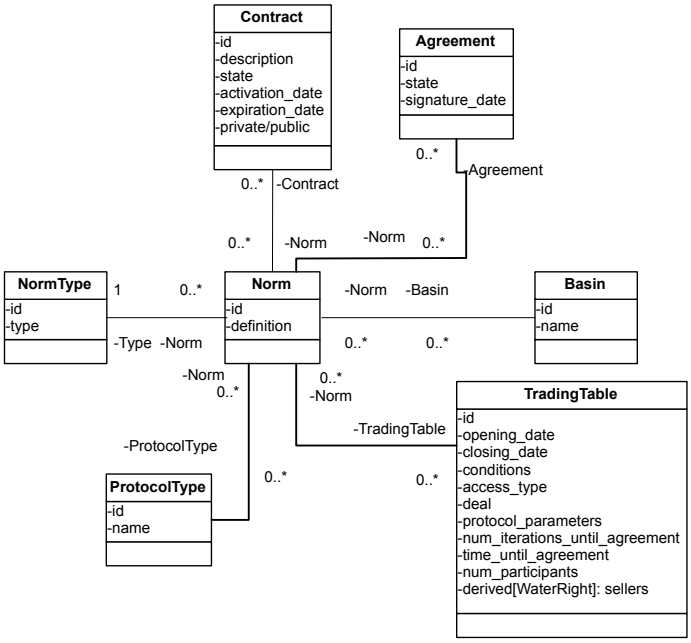

Figure 2.2: Norm Base logical model

In the *m-Water* prototype the n-BDI architecture has been considered as the basis for building the *m-Water* staff agents. Therefore, these staff agents are endowed with reasoning capabilities which consider the existence of norms in their decisions. Specifically, these *staff* agents carry out the norm reasoning process in two steps as follows:

- **Norm-based Expansion.** This first step consists in extending the theory of norm instances. Once the norm activation conditions hold, abstract norms are instantiated. Then, new desires are derived according to the current agent mental state and the norm instances. These new desires may help the agent to select the most suitable plan to be intended and, as a consequence, normative actions might be carried out by the agent.

- **Decision Making.** Once norms have been instantiated and considered for extending the desire theory, then steps corresponding to decision making are performed. In particular, both positive and negative effects of actions are considered when creating intentions for achieving goals. Then the action selection rule chooses the next action to be carried out. Then the agent executes the action.

2.3.1 Norm-based Expansion

The norm reasoning process is made up of two different phases: (i) instantiation of abstract norms; (ii) internalization of norms. Next, these phases are explained in detail.

Norm Instantiation

Once the norm activation conditions hold, then the abstract norms are instantiated and included in Γ_{NI}:

$$\frac{(\langle D, A, E, C, S, R\rangle, \rho) \in \Gamma_N, (A, \rho_A) \in \Gamma_B, (\neg E, \rho_{\neg E}) \in \Gamma_B}{(\langle D, C\rangle, fActivation(\rho_A, \rho_{\neg E}, \rho)) \in \Gamma_{NI}}$$

If an agent considers that an abstract norm ($n_a = \langle D, A, E, C\rangle$) is currently active ($A \wedge \neg E$), then a new norm instance ($n_i = \langle D, C\rangle$) is generated. The salience of a norm instance is defined by the $f_{Activation}$ function. It combines the certainty ascribed to the activation condition (ρ_A), the certainty about the non expiration of the norm ($\rho_{\neg E}$) and the norm priority (ρ). Since these three events can be defined as statistically independent, we define $f_{Activation}$ as the product among the three degrees ($f_{Activation} = \rho_A \times \rho_{\neg E} \times \rho$).

Norm Internalization

After performing the instantiation process for creating new norm instances, the agent must update its theory of desires with the new normative desires. These new goals derived from norms may help the agent to select the most suitable plan to be intended and, as a consequence, normative actions might be carried out by the agent:

- *Obligation Norm.* The rule for updating the Γ_D with the positive desires derived from obligation norms is defined as follows:

$$\frac{(\langle \mathcal{O}, C\rangle, \rho) \in \Gamma_{NI}, (C, \rho_C) \in \Gamma_D}{(C, max(\rho, \rho_C)) \in \Gamma_D}$$

  Once a norm instance corresponding to an obligation is created ($n_i = \langle O, C\rangle$), then a new positive desire will be inferred corresponding to the norm condition ($\mathcal{D}\ C, max(\rho, \delta)$). Thus, the desire degree assigned to the new proposition C is defined as the maximum between the degree of the norm instance and the previous desirability ($max(\rho, \delta)$).

- *Prohibition Norm.* The rule for updating the DC for complying with prohibitions is defined as:

$$\frac{(\langle \mathcal{F}, C\rangle, \rho) \in \Gamma_{NI}, (\neg C, \rho_{\neg C}) \in \Gamma_D}{(\ \neg C, max(\rho, \rho_{\neg C})) \in \Gamma_D}$$

  Similarly to obligation norms, a prohibition related to a condition C is transformed into a negative desire related to the norm condition ($\mathcal{D}\ \neg C, max(\rho, \delta)$).

2.3.2 Decision Making

Decision-making systems reason about future actions taking into account the agent's goals. Thus, norms restrict the range of goals to be pursued and the set of actions available for achieving them. Specifically the decision making process of staff agents is formed by two steps: (i) the generation of intentions; and (ii) the selection of actions.

Intention Generation

The set of preferred formulas which are reachable by some existing plan will derive the intended formulas of the agent:

$$\frac{\begin{array}{c}(\Sigma,\gamma,\rho_r,\rho_c)\in\Gamma_P,\Sigma=\{\alpha_1,...,\alpha_n\},(\gamma,\rho)\in\Gamma_D\\ (\alpha_1,\rho^+_{\alpha_1})\in\Gamma_D,(\neg\alpha_1,\rho^-_{\alpha_1})\in\Gamma_D,...,(\alpha_n,\rho^+_{\alpha_n})\in\Gamma_D,(\neg\alpha_n,\rho^-_{\alpha_1})\in\Gamma_D,\\ (\rho+\sum_{k=1}^n\rho^+_{\alpha_k})\geq\sum_{k=1}^n\rho^-_{\alpha_k}\end{array}}{(\Sigma,f_I(\rho+\sum_{k=1}^n\rho^+_{\alpha_k},\sum_{k=1}^n\rho^-_{\alpha_k},\rho_r,\rho_c))\in\Gamma_I}$$

In concrete, those desired propositions (γ,δ) which can be achieved by an action belonging to a plan $(\Sigma,\gamma,\rho_r,\rho_c))$ will generate a new intention $(\Sigma,f_I(\rho+\sum_{k=1}^n\rho^+_{\alpha_k},\sum_{k=1}^n\rho^-_{\alpha_k},\rho_r,\rho_c))$ if the desirability degree of both the actions and the state $(\sum_{k=1}^n\delta^+_{\alpha_k}+\delta)$ is greater than the sum of the negative effects of the actions $(\sum_{k=1}^n\delta^-_{\alpha_k})$. Finally, f_I is a function that combines both positive and negative effects of a plan and the risk (δ_r) and cost (δ_c) of a plan that achieves it. In this case it is defined as:

$$f_I(\rho^+,\rho^-,\rho_r,\rho_c))=(\delta^+-\delta^-)\times(1-\rho_r)\times(1-\rho_c)$$

Action Selection

The intention which has the maximum degree will define the next action to be performed by the agent. The problem of how agents carry out actions and perceive if they have been executed correctly has not been considered. In this sense, an agent is able to check action execution by direct observation, information from witnesses or analysing the later evolution of the system. However, it is a complex problem which has been omitted in the current version of the *mWater* prototype.

Chapter 3

A deliberative module for water users: a constraint programming formulation for planning in electronic institutions

This chapter focuses on the water users of *mWater*. In particular, we provide a double perspective for users that participate in the water market. The former presents a constraint programming formulation to navigate through the electronic institution that models *mWater*. The latter presents an optimization process, also based on constraint programming, that aims at the optimization stage that a water user may require during the trading process, i.e. it can be seen as a tool to assist the user on the negotiation process and reaching the best result. These two perspectives can be integrated in order to generate the plan that a water user has to follow to reach his/her goals, in terms of required amount of water, money or both.

3.1 General overview and motivation

mWater is modelled as an Electronic Institution (EI), as extensively described in Deliverable 8.2.1: mWater Analysis and Design. The underlying idea with EIs is to provide a computational counterpart of conventional institutions [Nor97, RA01, Est03]. Formally, institutions are, in an abstract way, a set of conventions that articulate agent interactions [Nor90]. EIs implement these conventions in such a way that autonomous agents may participate, their interactions are supported by the implementation, and the conventions (standard practices, policies, guidelines and norms) are enforced by the system on all participants.

EIs are engineered as regulated open MAS environments. These MAS are open in the sense that the EI does not control the agents' decision-making processes and agents may enter and leave the EI at their own will. This motivates the necessity of a mechanism

to facilitate *movement* through the EI. Since the MAS is open, the agents can represent many types of different users and not all of them have the entire knowledge about how to navigate within the EI. Therefore, calculating in advance what transitions should be followed, subject to the norms, conditions on the transitions and contextual information, shows very appealing to make a more effective use of the EI. In other words, a mechanism for planning in electronic institutions will improve the openness of these systems, will promote a higher level of users' participation and, eventually, a greater degree of success for users in achieving their preferences and goals.

An EI consists of: (i) a *dialogical framework* which fixes the context of interaction by defining roles and their relationships; (ii) *scenes* that establish interaction protocols of the agents playing a given role in that scene, which illocutions are admissible and under what conditions; (iii) *performative structures* that, like the script of a play, express how scenes are interrelated and how agents playing a given role move from one scene to another, and (iv) *rules of behaviour* that regulate how commitments are established and satisfied. From a more practical point of view, an EI can be seen as the union of performative structures and (structural) norms. A performative structure defines a workflow in the form of scenes and networks of scenes, whereas the norms define the rules of the game and the (un)feasible transitions that agents may follow within this workflow. Clearly, this gives us an effective way for norm enforcement and also to verify some correctness aspects, but unfortunately mainly focused on structural norms that do not consider dynamic norms, e.g. emergent norms that may appear in response to several exogenous events. Again, this dynamic behaviour makes it necessary the application of a planning method to guide the agent when moving from one scene to the next one, as can be seen in Fig. 3.1. Let us assume in that figure one agent wants to reach the `goal` from an `init` state. Since the EI encapsulates different routes, the agent may not know a priori if a particular route is feasible according to the norms in the institution and, more interestingly for the agent, what is the best route —depicted in thick arrows in Fig. 3.1. Also, it is possible that an agent wants not to obey the norms if this produces a better gain for the agent. And, obviously, this is out of the scope of the institution and highly depends on the agent's behaviour, interests and optimization criteria to be considered. For instance, an agent may violate a norm if this entails a higher economic benefit, but another agent whose objective is to increase his/her reputation may decide to be absolutely norm-abiding. In such situation, a planning method and, more specifically, an optimization algorithm becomes very valuable. Summing up, it is not only important to find a valid route but also to be the optimal one according to the agent's interests, while also considering all the (static+dynamic) norms included in the institution.

3.2 Planning in electronic institutions

As introduced in the previous section, planning technology can be very useful in EIs. Planning, within the field of AI, consists of an intelligent search process to generate a

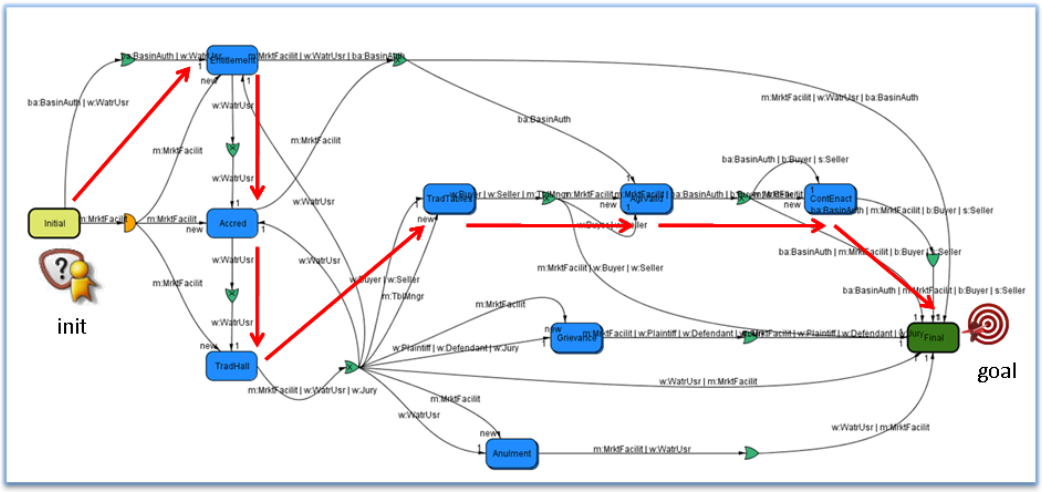

Figure 3.1: Navigation plan within the *mWater* EI

sequence of actions [GNT04]. Hence, it is concerned with selecting a set of actions whose execution will lead from a particular initial state to a state in which a goal condition is satisfied. Intuitively speaking, planning decides which actions need to be executed and their orderings in such a way that the logical executability of the plan is guaranteed.

3.2.1 Basic background on planning

Planning aims at finding a set of actions which allows an executive agent to transform an initial state into another state that satisfies some goals. Basically, a classical planning problem is usually defined as a tuple $\langle \mathcal{I}, \mathcal{G}, \mathcal{A} \rangle$, where $\mathcal{I}$ and $\mathcal{G}$ represent the initial state and the goals to be satisfied, respectively, and $\mathcal{A}$ is a finite set with the available actions that can be applied. Each action in $\mathcal{A}$ specifies, at least, some preconditions and effects. All preconditions need to hold before applying the action, whereas the effects represent the results of the action application in terms of new information that is asserted and other removed. This way, each action can be metaphorically considered as a transition that transforms one initial state into a resulting state provided that the action's conditions are satisfied in the initial state. And this transition is equivalent to the information that is stored in the workflow defined in an EI. Additionally, it is also possible to define a metric to optimize. This means that actions in $\mathcal{A}$ have a positive, or negative, impact in this metric and the planning process needs to find a plan that optimizes (or at least tries to optimize) such a metric.

There are many planners that reason on the causal relationships created between preconditions and effects. Planners usually apply planning graph techniques, heuristic estimates and decomposition methods, among others, to find a plan in an efficient way. But when there are actions unknown a priori or complex norms to satisfy, modelled as

constraints on the execution of the actions, and these norms have a significant impact on the planning metric, they do not behave as efficiently and do not always show expressive enough [GAO09]. Actually, modelling and reasoning under scenarios with complex features of planning (e.g. actions that have unknown or variable parameters) entail a much higher complexity of the resolution process. To face this limitation, some researchers have focused on the use of constraint satisfaction techniques and have delegated the new sort of constraints to a CSP (Constraint Satisfaction Problem) solver, as proposed in [GAO09, VG06]. The underlying idea with this is double. First, to create a general formulation that includes reasoning mechanisms to manage causal relationships, orderings and threats, i.e. the classical planning mechanisms to support preconditions and subgoal preserving, as well as any other type of constraint such as those that promote ethical values and norms, which include a full support for obligations, prohibitions, permissions and recommendations. Second, to model a planning problem as a CSP, with all its variables and constraints, and solve it by applying traditional CSP techniques. More intuitively, this approach consists of a knowledge engineering stage that can be automatized. The navigation rules in EIs that define the causal relationships (cause-effect properties) are automatically extracted from the EI definition together with the constraints/norms that are also subsequently translated into preconditions/effects. This has the clear advantage of separating the modelling stage from the solving stage and their algorithms. Consequently, once the constraint programming model is formulated, it can be solved by any CSP solver, making this approach solver-independent as well.

3.2.2 A general constraint programming formulation for planning

Let us assume a general action a used for planning with a set of preconditions $\{Pre(\phi_i, a)\}$, which need to be supported (by other actions) before a can be executed, and that produces some changes in $\{\phi_i\}$ —that is, a updates the values of $\{\phi_i\}$ after its execution[1]. In a constraint programming setting, a problem is represented as a set of variables, a domain of values for each variable, and a set of constraints among the variables. A solution to this problem is an assignment of one valid value to each variable, while all the constraints are satisfied. In our general representation we also include the formulation of branching situations to expand alternative partial solutions. Now, we present the mapping necessary to automatically create a constraint programming formulation (in terms of variables, domains, constraints and branching situations) for each action.

Variables and domains

Variables are basically used to define actions and the preconditions required by actions, along with the actions that support these conditions and the time when these conditions

[1] In many cases, preconditions and effects are modelled as boolean variables, but we present here a more general model that subsumes both logic and numeric variables.

occur (time is modelled in $\mathbb{R}$). For simplicity, we also include two dummy actions $Start$ and End; $Start$ provides the information of the initial state, whereas End requires the problem goals. These variables, their initial domains and their descriptions are organized into two blocks:

- **Block 1**. Variables necessary for any action:
  - $S(a) \in [0, \infty[$ represents the start time of action a (clearly, $S(Start) = 0$).
  - $dur(a) \in [dur_{\min}(a), dur_{\max}(a)]$ represents the duration of action a within two positive bounds. Although this is useful to model actions with different duration, we assume the duration of all actions is 1, so this variable can be removed for the sack of simplicity. Clearly, $dur(Start) = dur(End) = 0$.
  - $InPlan(a) \in [0, 1]$ encodes a binary variable that indicates the presence of a in the solution plan. Clearly, $InPlan(Start) = InPlan(End) = 1$ (***true***).
- **Block 2**. Variables for reasoning on the preconditions and effects of each action:
  - $Sup(\phi, a) \in \{b_i\} \mid b_i$ supports ϕ for a. This symbolic variable encodes the supporter b_i of ϕ.
  - $Time(\phi, a) \in [0, \infty[$ represents the time at which the action b_i selected as a value for variable $Sup(\phi, a)$ updates ϕ.
  - $V_{actual}(\phi, a) \in]-\infty, \infty[$ represents the actual value of ϕ at time $S(a)$, which must satisfy the precondition $Pre(\phi, a)$, defined by the action itself.
  - $V_{updated}(\phi, a) \in]-\infty, \infty[$ represents the value that a updates for ϕ. This variable is only necessary if a modifies the value of ϕ.

Although the model may include many variables, not all of them are decision variables. This means that only $S()$, $InPlan()$ and $Sup()$ need to be considered in the search process, whereas the others variables ($Time$, V_{actual} and $V_{updated}$) are bounded after the assignation of the former ones.

Constraints

Constraints represent relations among variables and correspond to assignments and bindings of the values to variables, supporting and other constraints. Constraints are defined for each variable that involves action a and each precondition/effect ϕ for a, and are organized into three blocks:

- **Block 1**. Constraints between each action and $Start$/End:
  - $S(Start) < S(a)$ binds any action a to start after $Start$.

  - $S(a) < S(End)$ binds any action a to start before End.

- **Block 2**. Constraints necessary for the preconditions/effects of the action:

  - $Time(\phi, a) \leq S(a)$ forces to satisfy ϕ before a starts.
  - If $Sup(\phi, a) = b_i$ then $\{Time(\phi, a) =$ time when b_i updates ϕ (i.e. $S(b_i) + 1$, as the duration is always 1), and $V_{actual}(\phi, a) = V_{updated}(\phi, b_i)\}$, which binds the time to satisfy the supporting variable $Sup(\phi, a)$, and determines how $V_{updated}(\phi, b_i)$ is propagated throughout the model.
  - $Pre(\phi, a) = V_{actual}(\phi, a)$ `comp-op` $Expression$, where `comp-op` $\in \{<, \leq, =, \geq, >, \neq\}$ and $Expression$ is any combination of variables and/or values that is evaluated in $\mathbb{R}$, which represents the precondition that ϕ must accomplish for a in $S(a)$.

- **Block 3**. Additional constraints:

  - Var_i `comp-op` $Var_j + x$, where `comp-op` $\in \{<, \leq, =, \geq, >, \neq\}$ and $x \in \mathbb{R}$, which represents any type of binary constraint.
  - $Constraint(Var_i, Var_j \ldots Var_n)$, which represents a general customized n-ary constraint that encodes more complex constraints among several variables of the model and may depend on each particular problem or EI.

Constraints in block 1 are straightforward as they provide the insights of an action within a plan, always between $Start$ and End. Block 2 includes similar constraints for the preconditions to satisfy the supporting condition and the assignments that are calculated by simple propagation. In particular, $V_{actual}(\phi, a)$ is assigned to a possible value of $V_{updated}(\phi, b_i)$. Also $V_{actual}(\phi, a)$ must satisfy the precondition $Pre(\phi, a)$, as modelled in the EI. Finally, block 3 includes additional constraints in the form of extra features such as precedences, quantitative temporal deadlines that involve two or more variables, etc. Although these three blocks may impose a high number of constraints in the model, it is important to note that not all of them need to be considered; only the constraints that involve an action a with $InPlan(a) = 1$ are applicable and, consequently, need to be satisfied in the solution.

Branching situations

Branching is used to generate the search space of different partial solutions that appear when supporting a precondition, or when solving a mutex (mutual exclusion relationship) and threat between actions. Particularly, in the former case a partial plan is created for each possible alternative (action b_i) in the support. In the latter case, a distinct constraint is posted to solve the mutex, thus preventing two actions from updating the same variable simultaneously (effects interference):

- Branching for the preconditions/effects of the action:
  - $Sup(\phi, a) = b_i \wedge Sup(\phi, a) \neq b_j \mid \forall b_i, b_j \ (b_i \neq b_j)$ that supports ϕ for a, which represents all the possibilities to support (update the value of) ϕ while $|Sup(\phi, a)| > 1$.
  - Let $time(b_i, \phi) = S(b_i) + 1$ and $time(b_j, \phi) = S(b_j) + 1$ be the time when actions b_i and b_j ($b_i \neq b_j$) modify the fluent ϕ. In that case, $\forall b_i, b_j$: $time(b_i, \phi) \neq time(b_j, \phi)$ must hold, which represents the mutex resolution by preventing two actions from updating ϕ simultaneously.
  - Let $time(b_i, \phi) = S(b_i) + 1$ be the time when b_i modifies ϕ. In that case, $\forall b_i$ that threats causal link $Sup(\phi, a)$ with $b_i \neq a$: $(time(b_i, \phi) < Time(\phi, a)) \vee (S(a) < time(b_i, \phi))$ must hold, thus solving the threat.

Again, the constraints to represent branching situations are only necessary for those actions that belong to the plan, i.e. $\forall a : InPlan(a) = 1$.

Main properties

This constraint programming formulation for planning offers very interesting properties:

- The formulation shares the advantages of other CSP-like approaches, including the expressiveness of the modelling language, which shows very appealing for specifying complex planning and EI models.
- The formulation is a purely declarative representation of the planning problem and is independent of any CSP solver. Consequently, any systematic or local search-based solver that uses its own heuristics can interpret and handle this constraint model.
- The whole formulation can be automatically derived from the EI definition, without the necessity of specific hand-coded domain knowledge, just by following the mapping presented above for actions, variables, domains and constraints.

3.3 Planning in mWater

The general constraint programming formulation presented above can be directly used in the *mWater*'s EI. The field of application comprises two parts. First, guiding the agent when moving from one scene to the next one, i.e. in the navigation process of the EI. For instance, if we consider an agent that starts from the initial state and wants to reach the contract enactment scene, it is now possible to provide him/her a plan to reach the goals (see Fig. 3.1). Similarly, inner and probably shorter plans are also possible, that

is an agent that has been accredited and simply wants to reach and agreement. Second, the constraint formulation can have a more specific application, thus focusing just on the trading process in terms of finding the best plan (optimization behaviour) to reach the agent's goals for water and/or money. For instance, if an agent that is trading in the trading table scene is interested in buying some water rights, the plan will provide here the best users to trade with, i.e. those users to sell these water rights and the quantity of water to be transferred from the sellers to the buyer.

3.3.1 Navigating through *mWater*

Navigation through an EI is a simple task, as each possible transition defines: (i) the causal relationship between two scenes or networks of scenes, and (ii) the norms, in form of constraints that must hold to apply such a transition. And this can be transformed into a constraint programming formulation. To make this translation easy, in the *mWater* setting we use a simple mapping, as described by algorithm 1. Each scene is modelled as a possible action in the plan (steps 1–2), and the transitions that reach such a state are modelled as its preconditions (steps 4–6) which are supported by the predecessor scenes. Note that this is a simplification of the variables presented in block 2 of section 3.2.2 as no V_{actual}, $V_{updated}$ variables are necessary here. The reason for this is that the only value that is updated (and later required by another scene) is the fact that the transition has been $done$, which indicates whether the transition has been reached or not. More particularly, the preconditions that support the *execution* of scene scn_i are modelled by means of $Sup(trn_j\_done, scn_i)$, where each trn_j is a transition from scn_j to scn_i. Analogously, the effect of the scene scn_i is modelled by $Sup(trn_i\_done, scn_k)$, where trn_i is a transition from scn_i to any scn_k defined as a valid successor of scn_i in the EI. This way, we are representing the flow of transitions followed in the workflow and what paths of scenes are necessary (causal chains) to reach a given goal scene. The constraints and branching situations for these variables are created as defined in section 3.2.2. It is important to highlight that special constraints can be defined if additional norms are attached to the transitions (see Constraints, Block 3 in section 3.2.2). For instance, if we want to include a prohibition (deadline) for both buyer and seller to register the final deal or agreement no more than 15 days later than the agreement date, we will include this constraint '$S(Register(buyer, seller)) \leq S(Agreement(buyer, seller)) + 15$'. This is a simple, but still very effective way to deal with norm reasoning without requiring a special reasoner, because norms are managed as one more constraint that involves variables of the model.

3.3.2 Optimization in the trading table

In *mWater*, the trading tables performative structure is a network of scenes that represents the different alternatives to trade water rights. No matter the protocol that is used to trade,

1: **for all** scene $scn_i \in$ ***mWater*** EI (including the `init` and `final` scenes as $Start$ and End dummy actions) **do**
2: create $InPlan(scn_i) \in [0, 1]$
3: create $S(scn_i) \in [0, \infty[$
4: **for all** transition $trn_j \in$ that reaches scn_i from scene scn_j **do**
5: create $Sup(trn_j\_done, scn_i)$ and add scn_j to its domain of values
6: create $Time(trn_j\_done, scn_i) \in [0, \infty[$

Algorithm 1: Mapping to generate all the variables for the constraint programming formulation. Constraints are generated as presented in section 3.2.2

e.g. blind double auction, Dutch or English protocols, the idea is to perform a trading action that involves a buyer and a seller. Obviously, the same buyer/seller could trade with others participants at the same time but the atomic element is a trading action that includes the buyer, seller, water quantity to be traded, thus representing the amount of water rights that will be transferred, and money that needs to be paid in this transaction. Following the constraint programming formulation presented above, but now giving special emphasis to the optimization of the trading actions, the model for an action $trade(buyer, seller)$ is formulated as:

- **Variables**. $Water\_Qty \in [0, \infty[$ and $Money\_Qty \in [0, \infty[$ that represent the amount of water and money, respectively, for the action. If $Water\_Qty = 0$ there is no trading action between this pair $\langle buyer, seller \rangle$.

- **Constraint**. Relation between $Water\_Qty$ and $Money\_Qty$: $Money\_Qty = Water\_Qty * price(buyer, seller)$, which also allows us to define different prices according, for instance, to the distance between the $buyer$ and $seller$.

In addition to the variables and the constraint that appears for each trading action, there are also two additional constraints:

- Water required: $(WaterBought - WaterSold) >= (WaterGoal - WaterInit)$, where $WaterBought = \sum Water\_Qty\_user_i$ for all $user_i$ that is playing the role of $buyer$ and $WaterSold = \sum Water\_Qty\_user_j$ for all $user_j$ that is playing the role of $seller$. $WaterGoal$ and $WaterInit$ represent the required and initial values, respectively, of the water for each user.

- Money required: $(MoneyReceived - MoneySpent) >= (MoneyGoal - MoneyInit)$, where $MoneyReceived = \sum Money\_Qty\_user_i$ for all $user_i$ that is playing the role of $seller$ and $MoneySpent = \sum Money\_Qty\_user_j$ for all $user_j$ that is playing the role of $buyer$. $MoneyGoal$ and $MoneyInit$ represent the required and initial values, respectively, of the money for each user.

- Metric to optimize, which can be defined as any type of simple or complex expression. However, it is usually measured in terms of maximizing the expression $MoneyGoal - MoneyInit$ for a given user.

As can be seen, the constraint formulation for optimizing the trading actions is significantly simpler than the general model. The reason for this is that we do not need a very complex model here; we just focus on finding the best combination of $buyer$, $seller$ and $quantity$ variables rather than on other constraints. However, the formulation is still open to include other types of constraints. For instance, it is common to have a prohibition for both buyer and seller to agree to a price lower than a given value x, where x is defined by the Basin Authority with respect to the maximum economic compensation for water right interchange: $Water\_Qty(buyer, seller) * price(buyer, seller) \leq x$.

3.3.3 Integrating navigation, optimization and execution

The main advantage of using a common constraint programming formulation is that we can easily integrate both the navigation through *mWater* and the optimization in the trading tables as a unique model. This way, the plan will provide the actions (in form of transitions) necessary to achieve a goal, such as reaching an agreement or a contract. If this plan requires visiting the trading tables performative structure and a particular protocol to trade water rights, the plan will also provide the best combination of trading actions with tuples $\langle buyer, seller, Water\_Qty, Money\_Qty \rangle$.

Once the plan has been generated, the user needs to execute it to reach the goals. Clearly, the execution of the plan can differ from the expected results. For instance, some failure/delay in the validation of the water user within the institution, or perhaps some buyers/sellers that fail to participate in the trading process —e.g. one of the water sellers that appears in the plan is not interested in selling water anymore. In this case a replanning stage becomes necessary to fix the plan. And here the constraint formulation can be very useful again, as the model remains nearly identical, keeping the same variables and constraints, as presented in section 3.2.2. Actually, the only difference will be in the domain of actions that will be available and in how many of those are now relevant. For instance, if a particular seller $s1$ is not available, we will make $InPlan(trade(buyer, s1, Water\_Qty, Money\_Qty)) = 0$ for all actions that involve $s1$. This forces not to use this action in the plan. On the contrary, if we want to maintain some actions in the plan we will make their $InPlan()$ variable to 1. If we want the CSP solver to decide whether an action will be in the plan, we keep its domain to the original [0,1] values. Consequently, we are not limited to complete input plans but to semi-complete or even empty plans. Also, we can encode new norms and requirements to allocate actions in time by means of the variables $S()$, to establish ordering between actions $S(a_i) < S(a_j)$ and fix deadlines on the execution of the plan $S(End) < deadline$.

All in all, using a constraint programming formulation facilitates the modelling stage of the EI, including its scenes, transitions and norms, into a unique formulation that com-

prises variables for the transitions/scenes that need to be executed to satisfy the water and money user's goals. From a solving perspective, a CSP algorithm solves (and tries to optimize) this model while also reasoning with norms, modelled as typical constraints. This is helpful to assist in the multiple trading process in *mWater*, but also in any generic goods market. Finally, from a multi-agent perspective the CSP model can also be solved in a distributed way by using distributed CSPs, as pointed out in [SOGA08].

Chapter 4

Conclusions and Future Work

This paper has contributed with mWater, a rather sophisticated regulated open MAS-based simulator to assist in decision taking and policy makers; we simulate and test how regulations and norms modify the usersÕ behaviour and how it affects the quality indicators of the market. The core component of mWater is an agent-based virtual market for water rights that intends to grasp the components of an electronic market, where rights are traded with flexibility under different price-fixing mechanisms and norms. In addition to trading, mWater also simulates those tasks that follow trading, namely, the negotiation process, agreement on a contract, the (mis)use of rights and the grievances and corrective actions taken therein. These ancillary tasks are particularly prone to conflict albeit regulated through legal and social norms and, therefore, they represent a crucial objective in policy-making and a natural environment for the application of agreement technologies. In summary, this type of MAS has a vital importance for decision support as it provides the foundations for the study of that interplay among agents, rule enforcing and performance indicators.

mWater constitutes a rather sophisticated regulated open MAS-based simulator. The core component of mWater is an agent-based virtual market for water rights that intends to grasp the components of an electronic market, where rights are traded with flexibility under different price-fixing mechanisms and norms. In this prototype analysis and design we have put special emphasis on the normative standpoint. Thus, we have dealt with a particular module for the staff agents of the institution (Chapter 3), and provided a constraint programming formulation for the water users (Chapter 4) that guarantees the norm abiding when navigating through such an institution. It is important to note that although we focus on a water-right market, the system is open to other types of markets, such as energy or stock markets. In addition to trading, mWater also simulates those tasks that follow trading, namely, the negotiation process, agreement on a contract, the (mis)use of rights and the grievances and corrective actions taken therein. Moreover, the constraint formulation allows the user to optimize this trading process. These ancillary tasks are particularly prone to conflict albeit regulated through legal and social norms and, therefore, they represent a crucial objective in policy-making and a natural environment for

the application of agreement technologies. In summary, this type of MAS has a vital importance for decision support as it provides the foundations for the study of that interplay among agents, rule enforcing and performance indicators.

Our current works addresses the following issues. First, to develop a richer normative regulation in order to allow us to simulate more complex types of norms and to observe what are the effects of a given regulation when different types of water users interact in the market. Second, to design specific heuristic estimations to improve the performance of the CSP solving algorithms used within these tasks. Third, to elaborate more expressive performance measures to evaluate social issues in the market behaviour in order to asses values such as trust, reputation, and usersÕ satisfaction. We believe that this type of measures will provide the policy makers with extra valuable data for decision making about new regulation. Fourth, although we now consider mWater as a simulation tool for decision-support taking, as a long-term research we are also interested in it as an open environment to human users for conducting social and participatory simulations. This would allow us to: i) let stakeholders use directly the system, ii) apply this approach to a specific basin and particular regulation, and iii) see how this is able to reproduce some real data. In such situations, human subjects will take part in the simulation to see the effects of their interaction with virtual agents, applicable norms and their adaptation. Finally, although we focus on a water-right market, the MAS framework is open to other types of (virtual or real) markets, such as energy (electricity) or stock markets. In this line, it would be interesting to compare whether this agent-based water-right market would differ from electricity trading and the systematic effect on the market outcomes.

Apart from the normative components analyzed in this deliverable we will deal in future works with the argumentation module for conflict resolution and grievances. A trust and reputation model will be also included to add more realistic behaviour to the market and the agents playing in the system.

Bibliography

[AEN+05] Josep Arcos, Marc Esteva, Pablo Noriega, Juan Rodriguez-Aguilar, and Carles Sierra. Engineering open environments with electronic institutions. *Engineering Applications of Artificial Intelligence*, (18):191–204, 2005.

[AGV+04] 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.

[APA+07] 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.

[BJC+06] Javier Bajo, Vicente Julian, Juan Manuel Corchado, Carlos Carrascosa, Yanira de Paz, Vicente Botti, and Juan Francisco de Paz. An execution time planner for the artis agent architecture. In *Engineering Applications of Artificial Intelligence*, volume 21, pages 769–784, 2006.

[CAB09] 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.

[CAB10a] N. Criado, E. Argente, and V. Botti. A BDI Architecture for Normative Decision Making (Extended Abstract). In *9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, pages 1383–1384, 2010.

[CAB10b] 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.

[CAB10c] 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 of *LNAI*, pages 52–63. Springer, 2010.

[CAG+10] Natalia Criado, Estefania Argente, Antonio Garrido, Juan A. Gimeno, Francesc Igual, Vicente Botti, Pablo Noriega, and Adriana Giret. Norm enforceability in electronic institutions? In *Proceedings of the MALLOW Workshop on Coordination, Organization, Institutions and Norms in Agent Systems in On-Line Communities (COIN@MALLOW)*, pages 49–64, 2010.

[CAJB09] 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.

[CJBA10] 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.

[Est03] M. Esteva. Electronic Institutions: from specification to development. *IIIA PhD Monography*, 19, 2003.

[GAO09] A. Garrido, M. Arangu, and E. Onaindia. A constraint programming formulation for planning: from plan scheduling to plan generation. *Journal of Scheduling*, 12(3):227–256, 2009.

[GB04a] A. Giret and V. Botti. Holons and agents. In *JOURNAL OF INTELLIGENT MANUFACTURING*, volume 15, pages 645–659, 2004.

[GB04b] A. Giret and V. Botti. Towards an abstract recursive agent. In *INTEGRATED COMPUTER-AIDED ENGINEERING*, volume 11, pages 165–177, 2004.

[GB06] 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.

[GJR+10] 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.

[GNT04] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning. Theory and Practice*. Morgan Kaufmann, 2004.

[GV09] A. Giret and V.Botti. Engineering holonic manufacturing systems. In *Computers in Industry*, volume 60, pages 428–440, 2009.

[HGPRG+09] 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.

[JB04] V. Julian and V. Botti. Developing real-time multi-agent systems. In *INTEGRATED COMPUTER-AIDED ENGINEERING*, volume 11, pages 135–149, 2004.

[Nor90] D.C. North. *Institutions, institutional change, and economic performance*. Cambridge Univ Pr, 1990.

[Nor97] P. Noriega. Agent-mediated auctions: The fishmarket metaphor. *IIIA Phd Monography*, 8, 1997.

[OPVS+09] N. Oren, S. Panagiotidi, J. Vázquez-Salceda, S. Modgil, M. Luck, and S. Miles. Towards a formalisation of electronic contracting environments. *Coordination, Organizations, Institutions and Norms in Agent Systems IV*, pages 156–171, 2009.

[RA01] J.A. Rodrıguez-Aguilar. On the design and construction of agent-mediated electronic institutions. *IIIA Phd Monography*, 14, 2001.

[SJR+02] 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.

[SOGA08] O. Sapena, E. Onaindia, A. Garrido, and M. Arangu. A distributed CSP approach for collaborative planning systems. *Engineering Applications of Artificial Intelligence*, 21(5):698–709, 2008.

[VG06] V. Vidal and H. Geffner. Branching and pruning: an optimal temporal POCL planner based on constraint programming. *Artificial Intelligence*, 170:298–335, 2006.