An Operational Approach for Implementing Normative Agents in Urban Wastewater Systems

Juan Carlos Nieves[1], Dario Garcia-Gasulla[1], Montse Aulinas[2], and Ulises Cortés[1]

[1] Knowledge Engineering and Machine Learning group (KEMLg),
Universitat Politècnica de Catalunya (UPC),
Barcelona, Spain
[2] Laboratory of Chemical and Environmental Engineering, University of Girona,
Girona, Spain

{jcnieves,dariog,ia}@lsi.upc.edu, aulinas@lequia.udg.cat

Abstract. *Water quality management policies on a river basin scale are of special importance in order to prevent and/or reduce environmental pollution caused by human sources. Industrial effluents are a priority issue particularly in Urban Wastewater Systems (UWS) that receive mixed household and industrial wastewaters, apart from rainfall water. In this paper, we present an analysis and implementation of normative agents that capture concrete regulations of the Catalan pollution-prevention policies. The implementation of the normative agents is based on Situation Calculus.*

Keywords. *Rational agents, environmental decision support systems, practical normative reasoning, situation calculus.*

Un enfoque operacional para implementar agentes normativos en sistemas urbanos de aguas residuales

Resumen. *Las políticas de gestión de la calidad del agua a nivel de cuenca hidrográfica son especialmente importantes para la prevención y/o reducción de la polución originada por el hombre en el medio ambiente. Los efluentes industriales son un elemento prioritario particularmente en los Sistemas Urbanos de Aguas Residuales (SUAR) que reciben mezcladas las aguas residuales provenientes de viviendas particulares y de industrias, así como el agua de lluvia. En este artículo, presentamos un análisis y una implementación de agentes normativos que capturan las regulaciones específicas de las políticas Catalanas de prevención de la polución. La implementación de los agentes normativos está basada en el Cálculo de Situaciones.*

Palabras clave. *Agentes racionales, sistemas de ayuda a la toma de decisiones, razonamiento práctico normativos, cálculo de situaciones.*

1 Introduction

Environmental decision-making is a complex, multidisciplinary, and crucial task. Water managers have to deal with complex problems due to characteristics of the processes that occur within environmental systems and possible consequences for the environment. In addition to this, water managers have to deal with normative regulations that have to be considered in any decision.

At the European level, [9] was developed to apply an integrated environmental approach to the regulation of certain industrial activities. This means that, at least, emissions to air, water (including discharges to sewer), and land must be considered together. It also means that regulators must set permit conditions so as to achieve a high level of protection for the environment as a whole. Several national and regional efforts are being made in order to improve water quality management as well as to comply with the European regulations. Specifically, we analyze the Catalan experience as a realistic example of adapting the European guidelines to manage water taking into account the local/regional reality.

In order to analyze the context of pollution-prevention policies in Catalonia, we consider a concrete regulation [5]. It is a regional regulation developed to follow the Catalan sanitation

program. One of the aims of this updated program is to directly link the urban wastewater treatment program with the industrial wastewater treatment program. It pays special attention to the industrial component of urban Wastewater Treatment Plants (WWTPs) in order to facilitate the connection to the public system of those industries and/or industrial parks that accomplish the requirements.

This regulation is not easy to interpret since each wastewater management case to be solved is different and has its particular complex peculiarities (e.g. flow, loads, frequency, location, polluting potential, involved agents, etc.). Water managers have to ensure that their decisions, given a particular problematic case, are taken on time and comply with specific regulations to prevent pollution and to ensure high water quality standards. For this reason, tools to simplify and shorten the experts' decision-making task, specifically those aimed at understanding and applying regulations, are required.

Following the perspective that a software agent is an active entity whose behavior is described by such mental notions as knowledge, goals, abilities, commitments, etc. [25], we have been exploring the definition of intelligent agents provided with the normative knowledge in order to manage concrete normative regulations which are in the context of UWS [2,20].

A central issue to successfully implement a normative agent is the selection of formalisms for performing practical normative reasoning. By practical normative reasoning, we mean a computable normative inference[1] able to infer a statement α from a normative knowledge base $\sum$ at a low computational cost. Although several powerful formalisms exist, finding the right one is a non-trivial challenge, as it must provide a level of expressiveness that serves the practical problems at hand in a tractable way. In the literature, one can find several approaches for performing normative reasoning such as deontic logic, temporal logic, dynamic logic, etc. [16]; however, these approaches have a high computational cost. An important feature of formal methods is that they do help in the long run to develop a clearer understanding of problems and solutions; hence, the definition of computable tractable approaches based on formal methods is relevant for performing practical normative reasoning.

Since norms in real world are usually defined at an abstract level [23], the modeling process of real norms is not straightforward. Therefore, a relevant issue for performing normative reasoning is to find an operational representation of norms. In [1], a declarative representation of norms was introduced. The main ingredient of this representation is the consideration of the norm conditions, which define a life cycle of each norm, to infer when a norm is violated. According to [1], the detection of norm violations depends on two properties of inspection to be done:

1. Observability: the conditions or actions can be checked by internal agents, given the time and resources needed;
2. Computability: the conditions of actions can be checked in a feasible and low cost manner.

Using these two properties, we can analyze their impact on the implementation of norm enforcement. The hypothesis is that by observing the items which affect the lifecycle of a norm, one can infer the state of violation of the norm. In this paper, we explore the life cycle of a norm in terms of state machines such that each state represents a state of the world. An important issue in our approach is a representation of the world in terms of states/situations. We follow the approach introduced in [14, 15] for observing the world and then inferring the state of a norm.

In this paper, we extend the work of the earlier paper [18] in order to present an implementation of normative agents based on *Situation Calculus* for performing practical normative reasoning in the domain of WWTP. The normative knowledge structure follows the approach introduced in [1]. Unlike the approach presented in [20], which extends the action language A to capture norms, in this paper we explore a norm's lifecycle by considering states of the world (situations/sets of *fluents*). This means that our main concern will be to monitor the states of norms being either *active* or *inactive* (monitoring if a norm applies to a

[1] By computable inference, we mean that there exists an algorithm which implements it. This inference is not necessary complete with respect to a logic inference; however it has to be sound.

certain agent), and *violated* or *respected* (regulating if a certain agent respects a norm's content). We will present an analysis of the existing specific Catalan regulation of providing normative knowledge to normative agents. Also, we will describe an implementation of these normative agents using *Situation Calculus*.

The rest of the paper is organized as follows. In Section 2, a realistic hypothetical scenario is described in order to illustrate the role of some regulations for managing industrial discharges. In Section 4, a brief introduction to *Situation Calculus* is presented. This section also describes how to introduce normative knowledge in a *Situation Calculus* specification. In Section 5, we explain how to implement the approach of Section 4. In Section 6, an operational execution example of the prototype is presented. Section 7 gives a summary of related work, and in the last section, we outline our conclusions.

2 Realistic Scenario

In this section, a realistic hypothetical scenario is described in order to illustrate the role of some regulations for managing industrial discharges.

At the municipality of Ecopolis, a new industry called *MILK XXI* expects to be set up. As a result of its production processes, the main characteristics of its wastewater will be as follows:

- Flow: 60 l/s (5184 m^3/day),
- Suspended Solids (SS): 130 mg/l,
- Biochemical Oxygen Demand (BOD_5): 450 mg/l,
- Chemical Oxygen Demand (COD): 800 mg/l,
- Oils and greases: 275 mg/l.

The Milk factory plans to work 16 h/day (two shifts), 225 days per year. It plans to get connected to the municipal sewer system which collects wastewater from a population of 12 000 inhabitants and transports it to the municipal WWTP. WWTP complies with regulations strictly. The owner of the industry submits a request to obtain authorization to discharge into the municipal sewer system, which is compulsory by law (Decree 130/2003 establishes the public sewer systems regulations). Moreover, the Milk industry plans to apply BAT[2] in order to reduce water consumption, so this fact is also declared in the request for a final authorization decision.

Table 1. Actions and Agents involved

Action	Agent
Request authorization, exceptions in thresholds, etc.	Industry Agent
Give authorization	Water Catalan Agency Agent
Apply BAT (declare this when requesting authorization)	Industry Agent
Discharge	Industry Agent

Table 2. Agents and Norms involved

Action	Type of Norm(Article)
Request authorization, exceptions in the thresholds, etc.	Obligation (7.1)
Give authorization	Obligation (13.1) Obligation (13.2) Obligation (13.2)
Apply BAT (declare this when requesting authorization)	Obligation (8.2)
Discharge	Obligation (8.3)

The industry intends to reduce 30% on water consumption, and consequently, the increment of pollutant concentrations is projected to be as follows:

- Flow (reduction): 42 l/s (3628,8 m^3/day),
- SS: 200 mg/l,
- BOD5: 600 mg/l,
- COD: 1000 mg/l,
- Oils and greases: 357.5 mg/l.

Several rules are launched to manage this case, which are included in the regulation analyzed in this work. In Tables 1 and 2, we list the agents and the norms involved directly in the case.

[2] BAT: Best Available Techniques [4].

We omit a detailed description of each agent; however, a version of these can be found in [2]. Note that the behavior of each agent is fixed by a set of norms that the agent has to comply with. In Section 3, we describe how these norms are expressed in mental notions of an agent.

3 Normative Specification Based on Situation Calculus

This section describes our approach to introducing normative knowledge in *Situation Calculus.* We begin with a brief introduction to *Situation Calculus*.

3.1 Situation Calculus

Situation Calculus [15] is a first order language for axiomatizing dynamic worlds. Nowadays, it has been considerably extended beyond the classical language to include concurrency, continuous time, normative knowledge, etc. [6, 14, 21, 22]; however, in all cases its basic ingredients are *actions*, *situations* and *fluents*.

- Actions: Actions are first-order terms consisting of an action function symbol and its arguments. In the scenario described in Section 2, a possible action of the industry is *make_spill(spill_init,milkXXI)*.
- Situations: A situation is a first-order term denoting a sequence of actions. These sequences are represented using a function symbol *do*: *do(a,s)* denotes the sequence resulting from adding the action *a* to the sequence *s*. The constant *s0* denotes the initial situation, namely an empty action sequence.
- Fluents: Relations whose values vary from state to state are called *fluent*; they are denoted by functions and predicates symbols. For relating the values of a fluent in a given situation, the binary relation *hold(f,s)* denotes the value of the fluent *f* in a situation *s*.

As any approach for temporal reasoning, *Situation Calculus* must deal with the *Frame Problem* to make its implementation consistent [15, 22]. Being aware of that, we present a specification that fully asserts the effects of all actions on every norm.

3.2 Norm Specification

In this section, we describe how to express norms in a specification of situation calculus, that is, a modeling process of norms at a high level. Since environmental domains are dynamic, that is, truth-values change with time, the described approach must deal with the specification of norms in dynamic domains. For this purpose, we follow the approach by states for specifying the world; this means that each state of the world will be reached by a finite sequence of actions in terms of *Situation Calculus*.

Before working on how to specify norms, we will analyze them, following [1, 19, 20] and keeping Situation Calculus particularities in mind. To fully specify a norm, several aspects must be identified:

Type of norm: norms that oblige to do something, norms that allow/permit something or norms that forbid something.

Conditions and content: the norm conditions and the norm content must be separated in order to study the characteristics of situations, in which the norm is active and in which the norm is violated.

States: a set of variables the norm refers to. For each possible value of those variables, the norm has one and only one activation and violation state, e.g., if a norm is applied to an agent then it should have one variable such as *IdAgent,* and if it is applied to an agent's spill then it should have two variables, *IdAgent* and *IdSpill.*

Actions: a complete list of domain actions which may influence the activation state and the violation state, separately, for each norm.

Preconditions: preconditions for each action must be defined, that is, in which situations it can be executed and the requirements regarding its parameters.

Having all that information in mind, we can start to specify our norms. For the normative domain, we follow an adapted version of Reiter's solution to the *frame problem* presented in [21]. We propose to split the specification into two

parts corresponding to the main properties that all norms have:

- Situations in which the norm is active;
- Situations in which the norm is violated.

To specify the situations in which a norm is active or violated, we will declare a value of *fluents* that will define unequivocally a set of situations which represent those set of states. The first part of the specification is meant to contain all possible states in which the norm must be taken into consideration (it is active). The second one comprises all the states in which the norm's content is violated. In what follows, we present the first part of the specification.

In our proposal, a norm *N*, after doing an action *A* in a situation *S*, is active if and only if it fits one of three cases:

i. **N was not active before doing *A***. There is a set of conditions under which *A* changes the activation state of *N* from inactive to active. The conditions needed for *A* to activate *N* are fulfilled in *S*.

The Activation Condition: Given a certain norm in a situation where the norm is inactive, the range of *A* is a set of actions that may modify the values of the *fluents* on which the activation state of the norm depends, in a way that the resultant situation (defined by the resultant value of the *fluents*) could belong to the situations in which the norm is active.

ii. **N was active before doing *A***. There is a set of conditions under which *A* changes the activation state of *N* from active to inactive. The conditions needed for *A* to deactivate *N* are not fulfilled in *S.*

The Inertial Condition: Given a certain norm in a situation where the norm is active, the range of *A* is the actions that may modify the values of the *fluents* on which the activation state of the norm depends, in a way that the resultant situation could belong to the situations in which the norm is inactive.

iii. **N was active**. There is no set of conditions that can make *A* change the activation state of *N* from active to inactivate.

The Non-Termination Condition: Given a certain norm and a situation where the norm is active, the domain of *A* is the actions that may modify the values of the *fluents* in a way that the resultant situation could change the state of the norm from active to inactive or from inactive to active.

If we analyze these three rules, we can assure that every state in which the norm is active fits into one and only one of these three rules. By checking a certain situation with a proposed action, we can assert the activation state of any norm after that action has been performed in the situation.

The second part of the specification contains the situations in which a norm is violated. In this case, we have decided to make a simpler specification. In the specification of the activation condition, we had the temporal progression integrated into it by the use of a variable that represents the action just performed (variable *A*). This variable allows us to represent temporariness by joining the current state with the past state. In the case of the violation state, we propose a static specification. In it we will omit the variable *A* and have the specification of the violation state solely based on the situation's *fluents*. It is possible to do so without losing expressivity since the temporal progression in our domain is represented as well in the *fluents* definition (whose specification looks very much like the *Activation Condition* specification), which contains the variable *A* as well. Otherwise, we would lose the concept of time. By deleting this action variable, the specification becomes much simpler as only the *fluents* that define the states where the norm is violated have to be stated.

In our proposal, a norm *N* is violated in a situation *S* if and only if it fits one of these two cases:

i. *N* is active in *S*, *N obliges* to the value of one or more *fluent,* and S does not fulfil all of those obliged *fluents*. This rule is intended to cover violations performed upon norms that oblige to something.
ii. *N* is active in *S*, *N forbids* the value of one or more *fluent*, and *S* fulfils one of those forbidden *fluents*. This rule is intended to

cover violations performed upon norms that forbid something.

With those two rules, we cover all possible violations which can come upon a norm, as norms that allow something cannot be violated. Since a norm cannot be forbidding and obliging at the same time, those two cases are mutually exclusive. Once having the two parts of the specification of each norm, we can implement them to see how they work in a real life domain.

3.3 Permission Norms

Before seeing the resultant implementation, there is a particular case that must be studied, as it implies specification and implementation particularities: it is the case of *permission norms*. Following the Deontic Logic definition of permission, permission norms cannot be violated. That is because formally the permission deontic operator specifies something that can be done, but does not implicitly specify that the opposite cannot be done. On the other hand, considering that our objective is to faithfully reproduce an existing legal framework that regulates a real world domain, our world is closed such that everything is allowed unless otherwise regulated. These two facts together make implicit knowledge of permission norms an issue, which must be dealt with specifically.

As an example of the implicit forbidding content of a permission norm, we can see that the norm *It is allowed to spill black waters into the river if one has the required authorization* implicitly includes the forbidding norm *It is forbidden to spill black waters into the river without authorization*. When that forbidding part of a norm does not appear on its own among the norms and only appears implicitly, it must be made explicit in order to capture the complete meaning of a normative system.

Once we have analyzed all implicit norms, which must be made explicit (ideally with the help of a legal expert), it is necessary to coordinate them with the rest of the norms, which cover the same situation. The permission norms which are applicable to the same situation as forbidding norms (the norm may originally contain the implicit forbidding norm or not) will work as exceptions to the generic forbidding norm. It is important to make sure that the activation states of all those norms are dependent on each other. That is, one or more permission norms will be active for a given agent when the forbidding norm is not active, and no permission norm will be active for a given agent when the forbidding norm is active. One of the norms (a permission norm or a forbidding norm) must always be active, and two (one permission norm and one forbidding norm) must never be active at the same time in all possible situations of the norm to avoid self-contradiction.

To achieve such merging of norms, it is necessary to analyze and integrate together all norms regulating the same situation. One of the advantages of the specification and implementation approach presented here is that it defines generic norms first and allows later addition of exceptions (usually represented as permission norms) to the existing norm. Therefore, it is very easy to integrate new exceptions to generic norms, thus extending and enhancing the normative knowledge base.

4 Normative Implementation Based on Situation Calculus

In the previous section, we proposed a *specification approach* for norms working on dynamic domains using *Situation Calculus* and specifying a norm's lifecycle as active, inactive, violated, and respected. Now we will show how this specification can represent real laws in standard Prolog. First, we consider and justify the code, which represents the states of a norm, and then we present an example of its application in our prototype.

We will discuss in detail only the code of one norm here, but the interested reader can find the whole Prolog domain and normative knowledge base at http://www.lsi.upc.edu/~jcnieves/software/NormativeKnowledge-PAAMS-2010.pl.

In this example, we are going to consider Decree 130/2003 of the *Catalan Water Agency*. Remember, the motivation of this decree was explained in Section 2. Norms regarding bureaucracy issues will be avoided deliberately, as we consider them to be less relevant to the

concrete domain we are trying to represent. We will formalize and work with the following set of norms:

- 7.1 For the following agents it is obligatory to obtain an authorization and to respect the restrictions of Annex I[3] and II:
  - Non-domestic users whose activity is included in C, D and E sections of the Catalan Classification of Economic Activities (Decree 97/1995) are considered as potential pollutant agents.
  - Those who generate spills > 6000 m^3/year.
- 7.2 If the agent is a domestic or similar user, such agent must comply with Annex I.
- 7.3 Only if the pertinent agent considers it best to spill to the environment, then it is possible not to spill to the sewage system.
- 7.4 If a new spill takes place, the pertinent agent will register it in a census (Article 18)
- 8.1 Prohibitions:
  - Substances of Annex I.
  - To dilute; if there is an emergency or an imminent risk, it is possible to dilute with previous warning to the competent agent.
  - To spill white waters to the public sewer system; if there is no alternative, an authorization to perform such spills must be obtained.
- 8.2 If a domestic agent spills contain substances included in Annex II, the agent must respect the established limitations.
- 10.2 If one has obtained the authorization, this agent may spill black waters to the public sewer system according to the established regulations.

In order to make the formalizing process easier to understand, we will see a commented implementation of only one of those norms; specifically, we will see the regulation included in article 7.1. Following the analysis explained in Section 4, we know that:

- Norm 7.1 is an obliging norm.
- The norm is active in the following situations:
  - A non-domestic agent is considered pollutant.
  - An agent spills > 6000 m3/year.
- The activation state of the norm can be changed by the following actions:
  - Make a spill: the execution of a spill is started by the agent thus increasing the agent's m3/year production.
  - Add substance: a certain amount of a substance is added to a spill of the agent thus increasing the agent's m3/year production.
  - Set agent type: the type (domestic, industrial...) of the agent is changed.
  - Set agent activity: the activity of the agent is changed; agents performing polluting activities are considered pollutant.
  - Set/Unset an activity as pollutant: the pollutant consideration of an activity is changed; agents performing polluting activities are considered pollutant.
  - Cancel a spill: the execution of a spill by the agent is terminated thus decreasing the agent's m3/year production.
  - Delete substance: a certain amount of substance from a spill of the agent is deleted thus decreasing the agent's m3/year production.
- Preconditions of each action will be defined by a predicate termed *poss*(*A*,*S*) following the Situation Calculus syntax, where *A* is an action and *S* is a situation.

4.1 A Norm Implementation Example

To implement the previously analyzed norm, we start with a representation of the activation state which may be active or inactive depending on the previous state of the world and what action occurred in it. In the code given further in this section, we specify three ways for a norm to become active in a given situation after an action occurred, as an adapted version of Reiter's simple solution [21]. The code may look too extensive at first glance, but it is quite simple. After a thorough look, it can be observed that since we need to assert all the cases in which the

[3] The mentioned Annex I and II in the laws refer to Annex I and II of the Catalan Decree 130/2003 that regulates the public drainage system. Annex I includes a list of limited substances and specifies these limitations; Annex II includes a list of forbidden substances.

norm is active so that the unasserted cases can be automatically classified as inactive, understanding the meaning and purpose of each line of the code is easy. The requirement of defining unequivocally all the cases in which the norm is active generates a simple and explicit representation of each of the possible states in which the norm is active.

The resultant Prolog implementation of norm 7.1 after applying the specification schema given before for the activation state of a norm (split into the three cases; *Activation Condition*, *Inertial Condition* and *Non-termination Condition* as explained in Section 4) can be seen in Tables 3 and 4.

Table 3. Prolog code for the activation state of norm 7.1

holds(norm(71,IdAgent),do(A,S)):-	Norm 7.1 is active (holds) for an agent *IdAgent* after doing action *A* in situation *S* (*do*(*A*,*S*)) if and only if one of the following situations (Activation, Inertial or Non-termination conditions) takes place.
ACTIVATION CONDITION: Actions which could activate the norm and did so when the norm was inactive.	
A=make_spill(IdSpill,IdAgent), holds(spill_total_size(IdSpill,SizeS),S), holds(agent_spills_total_size(IdAgent,SizeA),S), Total is SizeA+SizeS,Total>6000, \+holds(norm(71,IdAgent),S), poss(A,S);	The action is *make a new spill*, and with the size of this spill, the agent's total spills are bigger than 6000 m^3/year.
A = add_substance(IdSpill,Sub,Qua), holds(agent_spill(IdAgent,IdSpill),S), holds(agent_spills_total_size(IdAgent,Size),S), 6000<Qua+Size,\+holds(norm(71,IdAgent),S), poss(A,S);	The actions is *add a substance* to a spill of the agent, and with the added substance, the agent's total spills are bigger than 6000 m^3/year.
A=set_agent_type(IdAgent,non_domestic), holds(pollutant_agent(IdAgent),S),\+ holds(norm(71,IdAgent),S) , poss(A,S);	The action is *set the agent's type* as non-domestic, and such agent is pollutant.
A=set_agent_activity(IdAgent,Activity), holds(pollutant_activity(Activity),S), holds(agent_type(IdAgent,non_domestic),S), \+holds(norm(71,IdAgent),S) , poss(A,S);	The action is *set the agent's activity* to the one considered as pollutant and the agent is non-domestic.
A=set_pollutant_activity(Activity), holds(agent_activity(IdAgent,Activity),S), holds(agent_type(IdAgent,non_domestic),S), \+holds(norm(71,IdAgent),S) , poss(A,S);	The action is *set an activity to pollutant*, this activity is of the agent, and the agent is non-domestic.
INERTIAL CONDITION: [4] Actions which could deactivate the norm but did not do it when the norm was active.	
holds(norm(71,IdAgent),S), A = set_agent_type(IdAgent,non_domestic),poss(A,S);	The norm was active for the agent, and the action is *set its type* as non-domestic.
holds(norm(71,IdAgent),S), A=set_agent_type(IdAgent,AgentType), AgentType\=non_domestic, holds(agent_spills_total_size(IdAgent,SizeA),S)	The norm was active for the agent, the action is *set its type* not as non-domestic, and the

[4] Only the most significant inertial conditions are seen here. The rest can be found in: http://www.lsi.upc.edu/~jcnieves/software/NormativeKnowledge-PAAMS-2010.pl

,SizeA>6000, poss(A,S);	agent's spills are bigger than 6000 m^3/year.
holds(norm(71,IdAgent),S), A=set_agent_activity(IdAgent,Activity), holds(pollutant_activity(Activity),S), poss(A,S);	The norm was active for the agent and the action is *set its activity* to the one considered pollutant.
holds(norm(71,IdAgent),S), A=unset_pollutant_activity(Activity),\+holds (agent_activity(IdAgent,Activity),S),poss(A,S);	The norm was active for the agent and the action is *set an activity* which is not of the agent as non-pollutant.
holds(norm(71,IdAgent),S), A = cancel_spill(IdSpill,IdAgent), holds(agent_type(IdAgent,non_domestic),S), holds(pollutant_agent(IdAgent),S), poss(A,S);	The norm was active for the agent, the action is *cancel a spill* of the agent, and the agent is non-domestic and pollutant.
holds(norm(71,IdAgent),S), A = cancel_spill(IdSpill,IdAgent), holds(spill_total_size(IdSpill,SizeS),S), holds(agent_spills_total_size(IdAgent,SizeA),S), VAR is SizeA -SizeS, VAR> 6000, poss(A,S);	The norm was active for the agent, the action is *cancel a spill* of the agent, and the agent's spills without the cancelled amount are bigger than 6000 m^3/year.

NON-TERMINATION CONDITION: Actions which could not deactivate the norm when the norm was active.

holds(norm(71,IdAgent),S), \+A=set_agent_type(IdAgent,Type), \+A=set_agent_activity(IdAgent,Activity), \+A=unset_pollutant_activity(Activity2), \+A=del_total_substance(IdSpill,Sub),\+ A=del_substance(IdSpill,Sub,Qu),\+A=delete_agent(IdAgent), \+A=cancel_spill(IdSpill,IdAgent), poss(A,S).	The norm was active for the agent, and the performed action could not deactivate the norm in any possible situation.

Table 4. Prolog code for the violation state of norm 7.1

violated(norm(71,IdAgent),S) :-	Norm 7.1 is violated by an agent *IdAgent* if and only if one of the following situations takes place.
sholds(norm(71,IdAgent),S), holds(agent_spill(IdAgent,IdSpill),S), holds(spill_violates_limitation(IdSpill,Substance),S)	The norm is active for the agent *IdAgent*, and the agent produces a spill which violates a substance limitation.
holds(norm(71,IdAgent),S), holds(agent_spill(IdAgent,IdSpill),S), holds(spill_place(IdAgent,SpillPlace),S), holds(agent_associated_entity(IdAgent,IdAgentEntity),S), \+holds(spill_authorized (IdSpill,SpillPlace,IdAgent,IdAgentEntity),S) .	The norm is active for the agent *IdAgent*, and the agent produces a spill in a place not authorized by the associated entity agent.

This implementation, as justified in Section 4, represents fully and in a computable way the activation states of a norm because it includes all possible situations in which the norm is active. Table 4 presents an implementation of the violation state of the norm as explained in Section 4.

5 An Operational Example

After having seen how the code of a norm is implemented within a dynamic domain, we can consider how agents working on that domain can interact with it. To achieve this, we developed a

prototype representing our domain and its norms. Using it, we can define the events that happen and observe how the state of world and what norms change as a result. The internal operation of the prototype makes use of functions introduced in Section, i.e., *do(A,S)* and *holds(X,S)*. With them, the prototype is able to represent the world, its norms, interactions available in it, and the effects caused by the latter. Nevertheless, in order to interact with the prototype, we do not need any technical knowledge because most part of it is handled internally. To run it, we just need to define an initial situation, and based on it, a list of one or more actions to be performed sequentially on the initial situation. The prototype will then show us how the world and its norms change, affected by the actions of the user.

Next, we give an execution example based on a situation similar to the realistic scenario introduced in Section 2. In this example, we have an industry called *MILK XXI*, which intends to connect to the local WWTP. The new industry predicts the main characteristics of its wastewater to be as follows:

- Flow: 60 l/s (5184 m^3/day),
- SS: 130 mg/l,
- BOD_5: 450 mg/l,
- COD: 800 mg/l,
- Oils and greases: 275 mg/l.

An industry agent named *milkXXI* defines the initial situation. This agent is of type *non_domestic* and has no activity assigned. The activity *dairy_farming* is considered pollutant in the *Catalan Classification of Economic Activities*, and there is an agent representing the local water agency called *water_agency*. There is one spill, termed *spill_init* still not associated with *milkXXI*, containing the substances described previously. Figure 1 presents this situation.

In that initial situation, no norm is violated and only one is active[5]:

- 8.1 It is forbidden to spill forbidden substances.

[5] Norm 8.1 is always active for each agent.

In the active *fluents* of the initial state, it is declared that the spill *spill_init* does not respect the limitation for the substance *oils_and_greases* (which is a forbidden substance). In the rest of the *fluent*, certain things can also be certified, which are true in the initial situation, i.e., that the spill, its total size (5184 m^3/year), and its substances are registered in the census.

In this situation, we will set the agent's activity as *dairy_farming* which corresponds to the action:

set_agent_activity(milkXXI,dairy_farming).

As a result, we will reach the situation described in Figure 1b. In this situation, norm 7.1 is activated because the first condition is satisfied.

- 7.1 For the following agents, it is obligatory to obtain an authorization and to respect the restrictions of Annex I and II:
  - Non-domestic users whose activity is included in C, D and E sections of the Catalan Classification of Economic Activities (Decree 97/1995) and who are considered potential pollutant agents.
  - Those who generate spills > 6000 m^3/year.

If we look at the code of norm 7.1, in the part of the norm activation commented previously, the *Activation Condition* part, we can see one rule to justify this:

```
A= set_agent_activity(IdAgent,Activity),
   holds(pollutant_activity(Activity),S)
   holds(agent_type(IdAgent,non_domestic),S),
   \+holds(norm(71,IdAgent),S),
   poss(A,S);
```

As with the action *set_agent_activity*, we set the agent's activity to the one considered pollutant, and from now on, the agent is considered pollutant. Since the agent was *non_domestic*, which is the other requirement of the first condition, the norm activates.

The next step is to associate the spill to the agent. Our intent is to simulate the scenario in which the company *milkXXI* produces the spill defined as *spill_init*, in our prototype and based on the previous situation we execute the following action:

make_spill(spill_init,milkXXI).

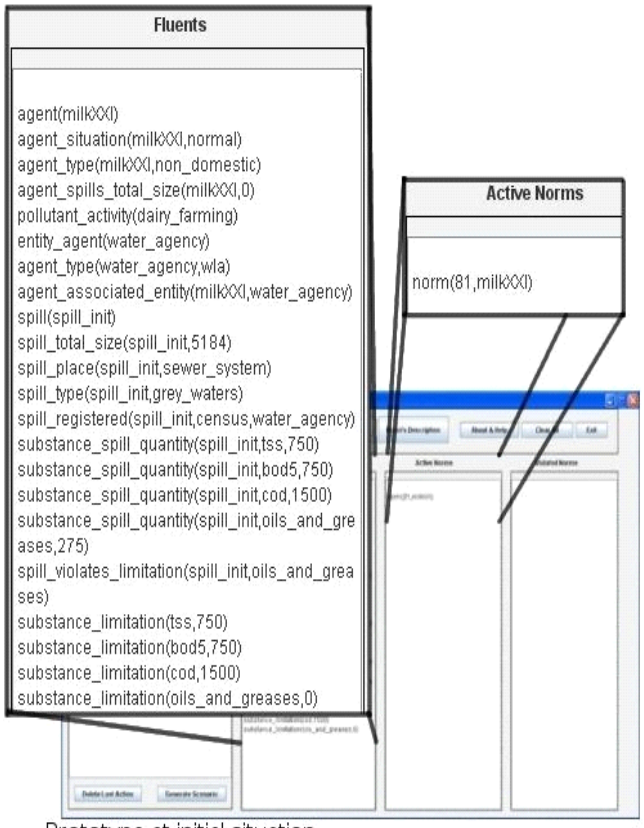

Fig. 1. a)

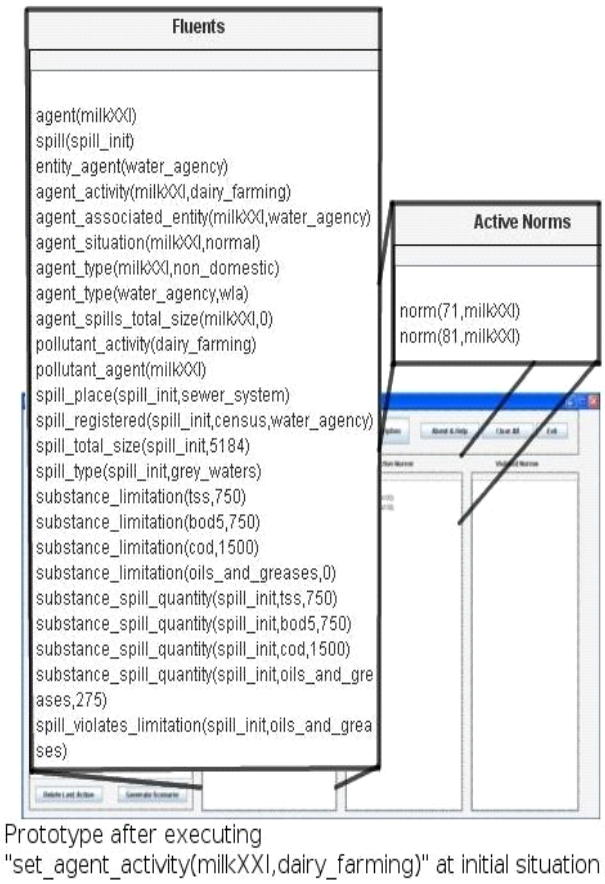

Fig. 1. b)

By doing this, we reach the situation of Figure 2a, where there are five norms active, two of which are violated. The only active norms are the following:

- 7.31. Active for agent *milkXXI* and *spill_init*: only if the pertinent agent considers it best to spill to the environment, then it is possible to not spill to the sewage system[6].
- 8.1.21. Active for agent *milkXXI* and *spill_init*: it is allowed to dilute in order to approach best levels; if there is an emergency or an imminent risk, it is possible to dilute with a previous warning given to the competent agent[7].
- 10.21. Active for agent *milkXXI* and *spill_init*: if one has obtained the authorization, this agent may spill black waters to the public sewer system according to the established regulations[8].

The violated norms are the following:

- 8.1 Violated for agent milkXXI and substance oils_and_greases: forbidden substances must not be spilled.

[6] Since this is a permission norm and the pertinent agent does not consider it best to spill to the environment, it is forbidden to not spill "spill_init" to the sewage system.

[7] Since this is a permission norm, the activated unit here is the forbidding part of the norm. For this agent, it is forbidden to dilute "spill_init" because the situation is normal.

[8] Since this is an permission norm and "milkXXI" does not have authorization, it cannot spill black waters.

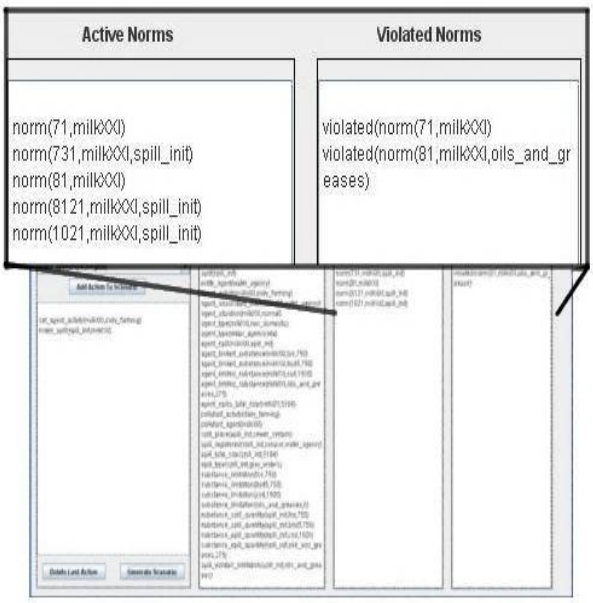

Fig. 2. a)

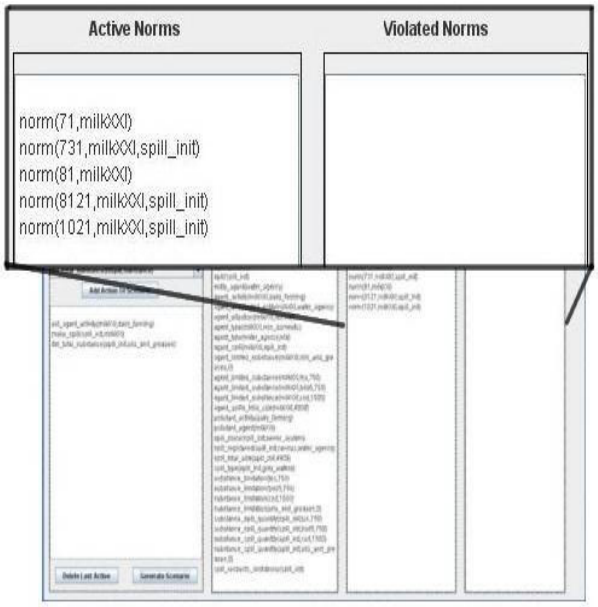

Fig. 2. b)

- 7.1 Violated for agent milkXXI: for the following agents, it is obligatory to obtain an authorization and to respect the restrictions of Annex I and II:
  - Non-domestic users whose activity is included in C, D and E sections of the Catalan Classification of Economic Activities (Decree 97/1995) are considered potential pollutant agents.
  - Those who generate spills > 6000 m^3/year.

We will focus on norm 7.1 because we have already analyzed its code. Norm 7.1, after being set active in the previous situation, is active again in the situation obtained after the execution of the action *make_spill*. We can find the reason for that in the activation state code, specifically, in the *Non-termination condition*:

```
holds(norm(71,IdAgent),S),
\+A=set_agent_type(IdAgent,Type),
\+A=set_agent_activity(IdAgent,Activity),
\+A=unset_pollutant_activity (Activity2),
\+A=del_total_substance (IdSpill,Sub),
\+A=del_substance(IdSpill,Sub,Qu),
\+A=delete_agent(IdAgent),
\+A=cancel_spill(IdSpill,IdAgent), poss(A,S).
```

Since the norm was previously active and the executed action could not deactivate the norm (which is why make_spill does not appear in the Non-termination condition code), the norm will be active after the action is performed. Regarding the violation state, we can see that one of the conditions is fulfilled, specifically, the following:

The norm is active for the agent *IdAgent* and it produces a spill which violates a substance limitation.

```
holds(norm(71,IdAgent),S),
holds(agent_spill(IdAgent,IdSpill),S),
holds(spill_violates_limitation
(IdSpill,Substance),S);
```

This is why the norm is active and violated in that state.

As it can be noted in the parameters of the violated norms, both refer to agent *milkXXI*. If we analyze both norms, we will realize that they regulate the same issue, and therefore, both violations can be solved by a single action. Such action deletes the amount of substance causing the violation. The action to achieve that would be: del_total_substance(spill_init,oils_and_greases).

After executing the action *del_total_substance*, we reach the situation of

Figure 2b. In this situation, even though all five norms are still active for *milkXXI*, none of them is violated. This can be proved by the fact that the fluent representing the violation *spill_violates_limitation(IdSpill,Substance)* is not true in that situation, thus, the deactivation of the norm is justified. Since the situation and the action performed satisfy the *Non-termination condition* again, the norm is still active.

This example demonstrates how to represent the world and its norms with *fluents*, how actions performed on the world change those *fluents*, and how a norm's state is altered as a result of these changes. The presented prototype was developed with the objective of making it intuitive for the user. UWS managers, potential users of the tool, would require a very short introduction to the software because they are experts on the specific subject it works on. The prototype would help in the decision making process by considering, thanks to its computational power, all involved variables and all existing normative details. In particular, if the chosen actions to be performed on a UWS are tested with the prototype before their execution, potential unexpected violations of norms can be detected, undesired side effects can be avoided, and future situations can be analyzed.

6 Related Work

In the literature, different approaches for performing normative formalization can be found [7, 8, 13, 17, 24]. Papers more related to our work are those which use a state machine to represent the world and its norms, as we do. Such approaches come from the use of logic formalisms like *Situation Calculus* and *Event Calculus*, because their *actions* and *fluents* support that kind of representation. *Event Calculus* is formalism similar to *Situation Calculus*. The former uses actions (or events) happening on the domain as its main representing element, while the latter involves mostly with the world states (or situations). In [3], a normative formalization based on *Event Calculus* with the main objective to detect conflicts between policies is proposed. It is an interesting approach, and it deals with contradictory norms – one of the problems we came across when formalizing the laws regulating WWTPs. Also based on *Event Calculus*, Fornara and Colombetti [10] develop a more agent-orientated approach to deal with normative frameworks. They consider methods of communication between agents and work with such organizational elements as agent's institutions. They define a norm's state similar to our norm's life cycle, although in our case, we focus on the world situation to define it, while Fornara & Colombetti use mostly roles and their available actions for the same purpose (which is explained by differences between *Situation Calculus* and *Event Calculus*).

Even though the system presented here could be used for both normative monitoring and reasoning, we consider normative reasoning as its main use. In particular, we consider of a special interest the practical reasoning process for analyzing the scope of a set of norms with respect to a sequence of actions. In fact, this is one of the main objectives of the prototype described in Section 5. Concerning that, practical normative reasoning can be applied to many scenarios, from a human society regulating the behavior of groups of people, like the WWTPs presented in this article, to a digital interaction unit controlling, for example, the interaction of Web Services. Kagal *et al.* [12] use their own specification language to define and manage the policies and constraints regulating the interaction of Web Services.

7 Conclusions and Future Work

We discussed an integrated approach towards building *real* normative systems to be deployed in scenarios were decision-making is constrained by the norms in use. The main contribution of this paper is that our research is a *proof of concept* for the advantages of using a normative system to make decisions in complex real life environments. In this sense, the most relevant principle is *Accountability*: agent-based services should *know* what they are doing and why, and they should be able to explain their actions or recommendations. Also, the separation of the logic layer from the user-interface and the dialogue layer is important. Since norms in real world are usually defined at

an abstract level [23], modeling real norms is not a straightforward process. Some authors have already pointed out that an instantiation of norms in a context domain helps to represent norms in a normative knowledge base [23].

In order to capture the scope of a norm in a dynamic domain like UWS, we have shown that one can fix the observable items that affect the lifecycle of a norm (see Section 4). In particular, the representation of these items in terms of *fluents*/*predicates* can help to infer the state of a norm. Since the state of a norm will be affected by changes in observable items, one can analyze the lifecycle of a norm in parallel to the changes of the observable items (see Section 4). We considered the use of *Situation Calculus* for implementing our approach. Note that the context domain can be clearly delimited by a set of *fluents* (a situation). This fact has been one of the main reasons for us to use *Situation Calculus*. As a running example, we analyzed the Catalan Decree 130/2003. It is a realistic example for managing UWS (see Sections 2 and 5).

In order to incorporate normative knowledge in a *Situation Calculus* specification, we proposed to split the specification of norms into two parts: 1) situations in which a norm is active and 2) situations in which a norm is violated.

The first part of the specification is meant to include all possible states in which the norm must be taken into consideration (the norm is active). The second one comprises all the states in which the norm's content is violated. Since the norms are represented in terms of the *fluents* in a given domain, the proposed specification represents a natural extension of a *Situation Calculus* specification. Although the integrated framework has not been completely realized, we expect our work to lead to a methodology of systematic development of normative systems for decision-making in complex real life environments.

Here are some open issues we will pursue in future:

1. **Lifecycle of actions**: at the moment, we have assumed actions as atomic events. This assumption has its limitations in capturing temporal aspects as deadlines. Preliminary results with respect to this issue are presented in [11].
2. **Conflicts between norms**: to consider this issue, we will explore a partial order of norms.

Acknowledgements

Numerous discussions with J. Vázquez-Salceda helped us to clarify our ideas. This work has been partially supported by the FP7 European project ALIVE IST-215890. The views expressed in this paper are not necessarily those of the ALIVE consortium.

References

1. **Aldewereld, H. (2007)**. *Autonomy vs. Conformity: an Institutional Perspective on Norms and Protocols*. PhD thesis, Utrecht University, Utrecht, Netherlands.
2. **Aulinas, M. (2009).** *Management of industrial wastewater discharges through agents' argumentation.* PhD thesis, University of Girona, Girona, Catalunya, Spain.
3. **Bandara, K., Lupu, E.C., & Russo, A. (2003).** Using Event Calculus to Formalise Policy Specification and Analysis. *4th IEEE Workshop on Policies for Distributed Systems and Networks,* Lake Como, Italy, 26–40.
4. **Baral, C. (2003).** *Knowledge Representation, Reasoning and Declarative Problem Solving*. New York: Cambridge University Press.
5. **Decree 130/2003**, Reglament dels serveis públics de sanejament. DOGC, 3894:11143–11158, 2003.
6. **Demolombe, R. (2004).** From belief change to obligation change in the situation calculus. *European Conference on Artificial Intelligence,* Valencia, Spain, 991–992.
7. **Demolombe, R. & Pozos, P. (2006).** Integrating state constraints and obligations in situation calculus. *Latin American Workshop on Non-Monotonic Reasoning*. Retrieved from http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-217/.
8. **Digmun, F., Dignum, V., Padget, J., & Vázqez-Salceda, J. (2009).** Organizing Web Services to develop Dynamic, Flexible, Distributed Systems. *11th International Conference on Information Integration and Web-based Applications and Services- iiWAS2009*, Kuala Lumpur, Malaysia, 225–234.

9. ***Directive 96/61/EC*** *of 24 September 1996 concerning integrated pollution prevention and control.* Official Journal L, 257(10), 10, 1996.
10. **Fornara, N. & Colombetti, M. (2008).** Formal Specification of Artificial Institutions Using the Event Calculus. (Technical Report 5). Italy: Institute for Communication Technologies, Università della Svizzera Italiana, Retrieved from http://doc.rero.ch/lm.php?url=1000,42,6,20090402172858-HW/ITC_TR05.pdf
11. **Garcia, D., Nieves, J. C., & Cortés, U. (2010).** Reasoning about Actions for the Management of Urban Wastewater Systems using a Causal Logic. *International Congress on Environmental Model and Software 2010, Ottawa, Canada,* Retrieved from http://www.iemss.org/iemss2010/papers/S21/S.21.04.Reasoning%20about%20Actions%20for%20the%20Management%20of%20Urban%20Wastewater%20Systems%20using%20a%20Causal%20Logic%20-DARIO%20GARCIA%20GASULLA.pdf
12. **Kagal, L., Finin, T., & Joshi, A. (2004).** Declarative Polices for Describing Web Services Capabilities and Constraints, *W3C Workshop on Constraints and Capabilities for Web Services*, Redwood Shores, USA, Retrieved from http://ebiquity.umbc.edu/paper/html/id/193/Declarative-Policies-for-Describing-Web-Service-Capabilities-and-Constraints.
13. **Kaponis, D. & Pitt, P. (2007).** Dynamic specifications in norm-governed open computational societies. *Engineering Societies in the Agents World VII, Lecture Notes in Computer Science,* 4457, 265–283.
14. **Lesperance, Y., Levesque, H. J., & Reiter, R. (1999).** A Situation Calculus approach to modeling and programming agents. In Michael Woodridge & Anand Rao (Eds.) *Foundations and theories of rational agents* (275–299).The Netherlands: Kluwer Academic Publishers.
15. **McCarthy, J. & Hayes, P.J. (1969)**. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie (Eds.) *Machine Intelligence, vol. 4* (463–502), Edinburgh: Edinburgh University Press.
16. **Meyer, J.J.C. & Wieringa, R. J. (1993).** *Deontic Logic in Computer Science: Normative System Specification.* Chichester: John Wiley and Sons Ltd.
17. **Modgil, S., Faci, N., Meneguzzi, F.R., Oren, N., Miles, S., & Luck, M. (2009).** A framework for monitoring agent-based normative systems. *Eighth International Conference on Autonomous Agents and Multi-Agent Systems –AAMAS 2009*, Budapest, Hungary, 153–160.
18. **Nieves, J.C., Garcia, D., Aulinas, M., & Cortés, U. (2010).** Using Situation Calculus for Normative Agents in Urban Wastewater Systems. *8th International Conference on Practical Applications of Agents and Multi-Agent Systems, Advances in Soft Computing, vol.70,* Salamanca, Spain, 247–257.
19. **Oren, N., Panagiotidi, S., Vázquez-Salceda, J., Modgil, S., Luck, M., & Miles, S. (2008).** Towards a formalisation of electronic contracting environments. *Coordination, Organization, Institutions and Norms in Agent Systems IV, Lecture Notes in Computer Science,* 5428, 156–171.
20. **Panagiotidi, S., Nieves, J.C., & Vázquez-Salceda, J. (2009).** A framework to model norm dynamics in answer set programming. *Multi-Agent Logics, Languages, and Organisations Federated Workshops* (MALLOW-FAMAS'09). Retrieved from http://ceur-ws.org/Vol-494/famaspaper8.pdf.
21. **Reiter, R. (1991).** The frame problem in situation the calculus: a simple solution (sometimes) and a completeness result for goal regression. *Artificial intelligence and mathematical theory of computation: papers in honor of John McCarthy,* Boston: Academic Press, 59–380.
22. **Shanahan, M. (1997).** *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia*. Massachusetts: Massachusetts Institute Technology Press.
23. **Vázquez-Salceda, J. (2003).** *The Role Of Norms And Electronic Institutions In Multi-Agent Systems Applied To Complex Domains The Harmonia Framework.* Ph. D. Thesis, Universitat Politècnica de Catalunya , Barcelona, Spain.
24. **Vázquez-Salceda, J., Aldewereld, H., Grossi, D., & Dignum, F. (2008).** From human regulations to regulated software agents' behavior. *Journal Artificial Intelligence and Law,* 16(1), 73–87.
25. **Wooldridge, M. (1999).** *Intelligent Agents. In G. Weiss (Eds), Multiagent Systems A Modern Approach to Distributed Artificial Intelligence* (27–78), Massachusetts: Massachusetts Institute Technology Press.

Juan Carlos Nieves is a postdoctoral researcher in the Knowledge Engineering and Machine Learning Group at the Department of *Llenguatges i Sistemes Informàtics* (LSI) of the Technical University of Catalonia (UPC). He explores reasoning approaches for defining reasoning skills of individual agents in such domains as Wastewater Treatment Plants. His general research domain is knowledge representation and reasoning, mainly using the argumentation theory and answer set programming, a declarative logic programming approach.

Dario Garcia-Gasulla is a Ph. D. student at the Technical University of Catalonia (UPC) and a junior researcher in the Knowledge Engineering and Machine Learning Group of the Department of *Llenguatges i Sistemes Informàtics* (LSI). He is interested in dynamic domain representation, normative reasoning and knowledge discovery in semantic networks.

Montse Aulinas has been a partial time associate teacher of the University of Girona (UdG) since 2008. Since 2004, she has been working on several areas of environmental engineering: solid waste management, water and wastewater management. Her interests include intelligent environmental decision support systems and use of knowledge representation techniques for environmental data and knowledge. She has a Ph. D. in Environmental Sciences, particularly, in multi-agent systems and argumentation processes to improve water management in river basins.

Ulises Cortés has been a professor of the Technical University of Catalonia (UPC) since 2007. Since 1982, he has been working on several areas of Artificial Intelligence: knowledge acquisition and concept formation in knowledge-based systems, machine learning, and autonomous intelligent agents. He was awarded with the CLUSTER chair at the *École Polytecnique Fédérale de Laussane* (EPFL) for 1998-1999. He has been a guest editor of special issues in several international journals: AiCommunications, Journal of Computer-aided and Civil Engineering, Computación y Sistemas, Environmental Modeling & Software, Applied Intelligence, Neural Networks, and Complex Problem-Solving Technologies, International Journal of Approximate Reasoning. From July 2002 to 2008, he was a member of the European Coordinating Committee for Artificial Intelligence (ECCAI).

Article received on 27/03/2010; accepted on 14/12/2010.