THOMAS: An Agent Platform For Supporting Normative Multi-Agent Systems

Natalia Criado, Estefania Argente, Vicente Botti

Grupo de Tecnología Informática - Inteligencia Artificial
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera S/N 46022 Valencia (Spain)

Abstract
Open Multi-Agent Systems (MAS) are formed by autonomous and heterogeneous agents that may pursue different and even conflicting goals. Therefore, no assumption can be a priori made about their behaviours. Thus, it is needed to constrain the potential excesses of agent autonomous behaviour by means of regulation of agent interaction. Because of this, “social” notions, such as organizations, institutions and norms, have been introduced in the MAS research as coordination and cooperation mechanisms. However, there is still a need to define frameworks which take these organizational abstractions into account. With this aim, this work describes a model of norms for agent organizations that covers both institutional and interaction levels, as well as different types of norms (i.e. constitutive, regulative and procedural norms). This model of norms has been implemented in a specific multi-agent platform, named THOMAS (Methods, Techniques and Tools for Open Multi-Agent Systems), which provides a suitable framework for the management of agent organizations.

Key words: Multi-Agent Systems, Norms, Open Systems, Virtual Organizations.

1. Introduction
Normative Theory has traditionally been developed inside research fields such as law, sociology, psychology, etc. Early approaches that tackled norms from the perspective of the Artificial Intelligence (AI) field attempted to model normative concepts in a formal way so as to evaluate and analyse normative systems in a computable way. They sought to provide a computational approach to challenges belonging to the jurisprudence domain. These proposals made a significant contribution on classical areas of AI such as expert systems, knowledge representation, case-based reasoning, argumentation and negotiation and so on [1]. In the multi-agent system (MAS)

Email address: ncriado@dsic.upv.es, eargente@dsic.upv.es, vbotti@dsic.upv.es (Natalia Criado, Estefania Argente, Vicente Botti)

1 This work is supported by TIN2005-03395 and TIN2006-14630-C03-01 projects of the Spanish government, GVPRE/2008/070 project, FEDER funds and CONSOLIDER-INGENIO 2010 under grant CSD2007-00022, FPU grant AP-2007-01256 awarded to N.Criado.
field, norms have been proposed as a formal specification of deontic statements that aim at regulating the life of software agents and the interactions among them [2]. Thus, norms allow legal issues to be modelled in electronic institutions and electronic commerce, MAS organizations, etc. Early proposals on employing normative concepts in MAS tried to establish a link between previous works on norms from a sociological, a logical and a legal perspective and Agent Theory [3]. However, research focus has evolved from studies aimed at addressing logical and representational issues to approaches focused on agent decision making, norm dynamics, legislator roles, among other issues [4].

Normative systems [5] have been redefined in the computer science field as "systems in the behaviour of which norms play a role and which need normative concepts in order to be described or specified" [6]. Therefore, "a normative multi-agent system (NMAS) is a MAS organized by means of mechanisms to represent, communicate, distribute, detect, create, modify, and enforce norms, and mechanisms to deliberate about norms and detect norm violation and fulfillment" [7]. Thus NMAS have been defined as the research field formed by the intersection between the normative system theory and the MAS area [8]; they combine models for normative systems (dealing with obligations, permissions and prohibitions, for example) with models for MAS. NMAS provide a promising model for human and artificial agent co-ordination, because they integrate norms and individual intelligence. They are a prime example of the use of sociological theories such as sociology, philosophy, economics, legal science in the MAS field [7]. Social factors in MAS are also becoming more and more important to coordinate interactions in dynamic open worlds, were to constraint agent autonomous behaviour is needed. New approaches are necessary to support the infrastructure evolution and facilitate their growth and adaptation through execution time, due to the characteristics of these open environments [9].

The main features of open agent societies are: (i) they are populated by heterogeneous agents which can enter or leave the system dynamically; and (ii) they are situated in virtual environments. The notion of virtual environment defines an electronic environment in which components come and go. Agent societies, also named Agent Organizations [10] can be understood as "complex entities where a multitude of agents interact, within a structured environment aiming at some global purpose". Agent organizations are also called Virtual Organizations (VOs) in other works, which are "a cooperation of legally independent enterprises, institutions or individuals, which provide a service on the basis of a common understanding of business. The cooperating units mainly contribute to their core competences and they act to externals as a single corporation" [11]. Thus, they are open systems formed by the grouping and collaboration among heterogeneous entities and there is a separation between form and function that requires defining how behaviour will take place. Agent societies or Virtual Organizations have been recently used as a paradigm for developing open agent systems [12, 13] (see [10] for a survey of organizational approaches to agent systems). Organisational modelling and design has to take into account not only the specification of the organisation structure and its normative behaviour, but also how the software engineer must translate the specific properties of the VOs, which are specified in the analysis and design models, into executable code for agent platforms. The main problem for obtaining executable code for VOs is the lack of agent platforms that give support to systems of this kind. Therefore, the definition and development of support platforms that allow implementing a secure and optimized management of VOs is mandatory.

The present paper represents a step forward on these open issues. In concrete, normative abstractions have been implemented by an agent platform specifically designed for the execution of VOs. This work is part of the THOMAS project (Methods, Techniques and Tools for Open Multi-
Agent Systems\textsuperscript{2}, whose main objective consists of researching and developing MAS technologies necessary for the development of virtual organizations which support open environments. In particular, a multi-agent system architecture suitable for the generation of virtual organisations in open environments has been proposed in [14]. In addition, a support platform that will allow these systems to be implemented has also been developed [14].

This paper is structured as follows: the next section contains an overview of the related proposals which constitute the background of our work. In Section 3 the developed model of VO is explained in detail. Section 4 describes how this conceptual model has been implemented in the THOMAS platform. Finally conclusions and future works are detailed in Section 5.

2. Related Work

In this section a review of proposals on NMAS is shown. These works have been classified into: formal models for the representation of norms (section 2.1); and issues related to the implementation of NMAS (section 2.2).

2.1. Formal Models of Norms

In order to employ norms to represent agent rights, duties and commitments as members of a society, a formal model of norms is needed. The work described in [15] proposes the classification of the norms that govern the behaviour of autonomous agents on three normative levels: (i) the Institutional level, with institutional norms explicitly promulgated by the institution (or a representative); (ii) the Interaction level, with agreed norms that emerge from agent interactions and negotiation processes; and (iii) the Private level, with private norms self-imposed by agents that guide their behaviour. One of the first institutional normative models was the Social Law approach [16]. It is aimed at representing norms which are imposed to the society members by an authority. Regarding norms that emerge as a result of the agent interaction, in [17] a model of agent commitment is proposed. A commitment is defined as a set of conditions that should be satisfied as a consequence of an agent agreement. Finally, in [15] the formalization of private norms following a BDI [18] representation of a mental state is proposed. More specifically, all of the norms are represented as preferences. Following on with the inclusion of normative concepts in the BDI agent architecture, the proposal contained in [19] presents an extension of the classic BDI architecture for considering norms. The main drawbacks of these initial works are: (i) they offer support to only one normative level; (ii) the set of norms or rules are defined off-line in a static way by the system designer; and (iii) they do not consider the existence of different types of norms, such as norms for describing legal aspects of facts or practical connections between a regulation and its consequences.

Recently, new proposals on normative models have been made in order to design complex and dynamic agent societies. For example, the formal model of norms proposed in [20, 21] considers dynamic aspects of norms. This approach distinguishes between substantive norms and procedural norms. Substantive norms define the legal relationships between the members of the society and the normative system itself in terms of regulative and constitutive norms. Regulative norms define the expected contributions to the social system. More specifically, regulative norms describe the ideal behaviour and varying degrees of sub-ideal behaviour of a system by means of obligations, prohibitions and permissions [21]. These norms regulate activities which

\textsuperscript{2}http://bisite2.fis.usal.es/thomas-tin/
can exist independently of the norm. An example of a regulative norm is "It is forbidden for students to cheat". Constitutive norms introduce new classifications of abstract facts and entities, called institutional facts [22]. They do not define restrictions on behaviours, but provide an abstraction mechanism that details the ontology used for defining the behaviour of the system [20]. An example of a constitutive norm is "Looking at a book count-as cheating in an exam". Finally, procedural norms are an instrumental approach, i.e. they are aimed at achieving social order specified in terms of substantive norms. In this sense, there is no logical connection between a regulative norm and a sanction or reward. Therefore, procedural norms define a practical connection between a regulation and its consequences [21]. Procedural norms define how rewards, costs and risks are allocated within a social system. An example of a procedural norm is "Teachers are obliged to fail students in their tests whenever they have violated the cheating prohibition"; which obliges teachers to enforce the cheating prohibition by failing dishonest students as sanction. The substantive and procedural norms are complementary. The procedural norms are the mechanism by which the rights and duties, which are specified by substantive norms, are defended and enforced. One interesting application of this model of norms is the consideration of both the NMAS and agents as autonomous entities with capabilities for taking their decisions [23]. In this sense NMAS, as agents, pursue a concrete goal (i.e. the maintenance of a certain social order) and have bounded capabilities for achieving it.

Finally, one of the most complete, and most referred to, model of norms for open societies is formulated in [24]. It defines two main concepts: (i) an autonomous normative agent, which is described as an autonomous agent whose behaviour is influenced by a set of norms and; (ii) a norm, which specifies how an agent should behave in the society. More specifically, a norm defines its addresses and beneficiaries (i.e. the responsible and beneficiary agents of the fulfilment of the norm, respectively); the normative goals; the norm activation and exception states; and the related rewards and punishments. In this approach norms are classified into: general norms, which describe the ideal behaviour of agents in terms of their responsibilities, rights and duties; legislative norms, which define the way in which the normative set can be dynamically modified by agents; and enforcement norms, which include sanctions and rewards, and are defined in order to convince agents to respect the norms. The main contributions of this model are: (i) it takes dynamical features of norms into account, since it allows the definition of norms that control the norm change; and (ii) it gives support to the definition of enforcement norms in order to face with agent autonomy.

The above mentioned models present some disadvantages when supporting open MAS. As a consequence of the virtual environment, in which components may come and go, together with the heterogeneity and autonomy of the participant agents, this type of societies needs of specific mechanisms in order to deal with the inherent uncertain agent behaviours and the adaptation and modification of the normative context. In addition, the complexity of this kind of society may require a complex model of norms which covers the three different norm levels (i.e. institutional, interaction and private) and supports the different types of norms (i.e. substantive and procedural norms). Moreover, this representation of norms must allow both the individual agents and the system designer to reason about the normative system. In section 3 (see subsection 3.2) a conceptual model of norms is proposed, which tries to handle all these issues.

2.2. Norm Implementation

Most of the above mentioned proposals tackle the study of NMAS from a theoretical perspective. However, works on norms from a more practical point of view have recently arisen.
They are focused on giving a computational interpretation of norms in order to employ them in the design and execution of MAS software applications.

In the existing literature about norm implementation there are several proposals on normative languages for representing constraints on agent behaviour. In [25] a normative language of general purpose was presented. This language has been employed as a reference for other proposals on normative languages (e.g. in [26] an extension of this language to define norms in Electronic Institutions (EI) is proposed).

The codification of norms by means of the above mentioned languages is too abstract to be implemented in a society. Therefore, societies require mechanisms for ensuring norm compliance. In this sense, norms must be interpreted or translated into operational norms which are meaningful for the society [27]. In [25] authors make a study about operational aspects of norms; this work is mainly focused on analysing verifiability issues directly related to norm compliance. Methods for providing an operational interpretation to norms are classified in two categories (see Figure 1) [28, 27]: (i) regimentation mechanisms, which consist of making the violation of norms impossible, since these mechanisms prevent agents from performing actions that are forbidden by a norm; and (ii) enforcement mechanisms, which are applied after the detection of the violation of some norm, reacting upon it. In a recent work [29], a taxonomy of different techniques for implementing effectively norms is proposed. On the one hand, the regimentation of norms can be achieved by two processes: (i) mediation, in which both the resources and communication channel are accessed through a reliable entity which controls agent behaviours and prevents agents from deviating from ideal behaviour (e.g. governors in Electronic Institutions [30]); and (ii) hard-wiring, assuming that the agents’ mental states are accessible and can be modified in accordance with norms (e.g. the KAoS proposal [31] employs this compliance ensuring process). On the other hand, norm enforcement techniques are classified according to both the observer entity and the enforcer entity. Norms are self-enforced when agents observe their own behaviours and sanctions themselves. Thus, norm compliance is both observed and enforced without the need of any additional party. In situations in which those agents involved by a transaction are responsible for detecting norm compliance (i.e. second-party observability) norms can be enforced by: (i) the second-party which applies sanctions and rewards as retaliation and reciprocation, respectively; and (ii) a third entity which is an authority and acts as an arbiter in the dispute resolution process [32]. In the case of third-party observability two different mechanisms for ensuring norm compliance can be defined according to the entity which is in charge of norm enforcing: (i) social norms are defended by the society as a whole [33]; (ii) in assisted enforcement the sanctions are applied by an authority which acts as a judge in the grievance process. Finally, in infrastructural enforcement there are infrastructural entities which are authorities in charge of monitoring and applying sanctions and rewards [34].

The great majority of the works on a computational implementation of norms are based on the Electronic Institution (EI) proposal. In [35] a formalism based on rules for representing and reasoning about constraints on agent behaviours is presented. This rule-based language has been employed in order to regiment norms that govern EIs [26]. The EIs provide a framework for heterogeneous agent cooperation. However, they are not a fully open environment, because agents interact inside the institution through the infrastructure provided by the EI. The behaviour of external agents is completely controlled by the institution, which allows or forbids agents to pronounce certain illocutions. Thus, norms are regimented or pre-imposed on agents. The definition of NMAS is more similar to Partially Controlled MAS (PAC) [36]. A PAC is an open MAS in which there are two types of agents: controllable agents, which are directly controlled by the system designer; and uncontrollable agents, which are not under the designer’s direct
control. Thus, uncontrollable agents may deviate from ideal behaviour. As a consequence, there has to be an enforcing mechanism delegated into some controllable agents which act as norm supervisors. Our proposed multi-agent framework, named THOMAS (Methods, Techniques and Tools for Open Multi-Agent Systems) is more related to the notion of PAC, since it does not have any mediator layer. On the contrary, in THOMAS agents are allowed to interact freely. In this way, a conceptual model of norms has been designed (explained in the following section) that allows describing norms with enforcing mechanisms. Section 4 illustrates details related to the implementation of our conceptual model of norms into the THOMAS platform.

Throughout this section, a short overview of the most relevant works on normative models and norm implementation has been given. These proposals have been evaluated taking as a reference the main challenges of open MAS. After analysing the concrete challenges of open MAS and considering the current normative models, a new model of norms for VOs is proposed in the next section.

3. Virtual Organization Model

As previously stated, VOs have been employed as an abstraction for modelling open agent societies. In this sense, they allow systems to be modelled at a high level of abstraction. VOs include the integration of organizational and individual perspectives and also the dynamic adaptation of models to organizational and environmental changes [37]. Based on the Human Organization Theory [38], the work described in [39] consists of a Virtual Organization Model (VOM) which allows the main aspects of an organization to be described. The main contribution of VOM is that it is aimed at modelling open societies in which heterogeneous and autonomous agents work together, focused on the integration of both Web Services and MAS technologies. This section contains a general overview of the proposed model together with a theoretical definition of the main components of a VO (Section 3.1). In concrete, the definition of norms for controlling
the VO performance is explained in Section 3.2.

3.1. Virtual Organization Definition

The VOM approach takes into account the four main aspects of a VO: (i) the Structural dimension, which describes components of the system and their relationships; (ii) the Functional dimension, that details the system functionality; (iii) the Normative dimension, that defines mechanisms employed by a society in order to influence its member behaviour; and (iv) the Environmental dimension, which describes the environment in terms of its resources and how agents can perceive and act on them. A complete description of this model can be found in [39].

Taking the main concepts and relationships of VOM as a reference, a VO is formally defined as follows:

**Definition 3.1 (Virtual Organization).** A Virtual Organization (VO) is defined as a tuple $\langle A, R, OU, N, S, ER, Ac, F, \text{roles, plays} \rangle$ where:

- $A = \{a_1, ..., a_n\}$ is the set containing all agent labels. Agent labels ($a_i$) are constants that identify members of the VO by a unique name;
- $R$ is the set of labels identifying roles which have been defined in the VO;
- $OU = \{ou_0, ou_1, ...\}$ is the set of organizational units which has been defined in the VO, where each organizational unit is a partition of $A$ ($ou_i \in P(A)$);
- $N$ is the normative context of the VO, i.e. the set of norms that control the operations carried out in the VO (see definition 3.2);
- $S$ is the set of services defined inside the VO, which represent tasks and protocols. Each service is identified by a unique label;
- $ER$ is the set of resources located in the VO environment;
- $Ac$ is a set of primary individual actions. Since functionalities are represented as services, actions in our model are related to the service access:

$$\forall s \in S : \text{Serve}(s) \in Ac \land \text{Request}(s) \in Ac \land \text{Register}(s) \in Ac$$

where $\text{Serve}(s)$ means that service $s$ is provided, $\text{Request}(s)$ represents a request of functionality $s$ and $\text{Register}(s)$ implies that a new service $s$ is registered in a directory. In addition, actions for accessing the environment are also considered. The set $Ac$ is defined as follows:

$$\forall r \in ER : \text{Act}(r) \in Ac \land \text{Percept}(r) \in Ac$$

where $\text{Act}(r)$ implies a change in the parameters of resource $r$ and $\text{Percept}(r)$ represents that some of the parameters of $r$ has been read.

- $F$ is the set of basic facts. It covers facts related to the execution of actions $\text{Do}_m(a)$, $\text{Done}_m(a)$. More explicitly: $\forall a \in Ac \land \forall m \in A : \text{Do}_m(a) \in F \land \text{Done}_m(a) \in F$;

- $\text{roles} : OU \rightarrow 2^R$ is a function that maps each organizational unit with the set of roles that can be played by agents belonging to this unit;
• plays : \( A \times OU \rightarrow 2^R \) is a function that maps each agent and unit with the set of roles played by the agent inside this specific unit.

This theoretical definition of a VO allows defining:

• The Structural Dimension; i.e. the set of agents (\( A \)), roles (\( R \)) and organizational units (\( OU \)) which form a VO. An Organizational Unit (OU) is a basic social entity that represents the minimum set of agents that carry out specific and differentiated activities or tasks, following a predefined pattern of cooperation and communication [40, 41]. An OU includes a set of roles (\( R \)) that can be played by its members.

• The Functional Dimension; i.e. the set of services (\( S \)) defined by the VO. Services represent the functionality that entities offer to others, independently of the concrete one that makes use of it. VOM allows describing agent functionality in terms of services, by means of OWL-S [42] ontology, highly standardized and popular in the Web Service domain. Any entity, both agents and OUs, is capable of offering, publishing or requesting some specific functionality. Thus, the request (Request), publication (Register) and provision (Serve) of services are valid actions of agents in a VO.

• The Environment Dimension; i.e. the resources (\( ER \)) located in the VO environment. Our purpose is focused on describing environmental resources, perceptions and acts on these elements. Therefore, the actuation (Act) and perception (Percept) of these environmental resources belong to the set of valid actions (Ac).

• The Normative Dimension of VOM (\( N \)) is a coordination mechanism that attempts to: (i) promote behaviours that are satisfactory to the organization, i.e. actions that contribute to achievement of global goals; and (ii) avoid harmful actions, i.e. actions that prompt the system to be unsatisfactory or unstable.

This definition of VO extends the traditional models of MAS, with the normative context (\( N \)) of the society. Therefore, this theoretical model of VO is mainly focused on the definition of norms which control agent behaviours. Next, a complete definition of these normative concepts is provided.

3.2. Normative Context Definition

Along this section, the normative dimension of a VO is formally defined by means of a conceptual model of norms whose full description can be found in [43]. It is based on previous proposals of computational norm models [20, 24, 15] and takes its inspiration from human legal systems [44, 45]. Following, a formal description of its main components is provided. More concretely, it defines normative context as follows:

Definition 3.2 (Normative Context). The Normative Context (\( N \)) of a VO covers the set of norms defined by the organization itself as well as the norms derived from agent interactions. More formally, \( N = I \cup C \) where:

• \( I \) are the institutional norms (\( I = I_{Constitutive} \cup I_{Regulative} \cup I_{Procedural} \));

• \( C \) are the interactional norms, i.e. the set of norms which have been created as a consequence of the interactions among agents (\( C = C_{Constitutive} \cup C_{Regulative} \cup C_{Procedural} \)).
Taking the previously cited norm levels [15] as a reference, the normative context is formed by a set of norms belonging to institutional and interactional levels, thus they can be defined by the institution or organization (I) or derived from agent interactions (C). The private level (i.e. internal norms of agents) has not been considered in this work, since this model is focused on the social dimension of norms. Table 1 summarises the main differences among institutional and interactional norms in our approach. For example, an interactional norm is created by two agents, one of them is responsible for fulfilling the commitment and the other one benefits from it, for a specific period of time. On the contrary, an institutional norm states permanently an agent responsibility, right or duty which benefits the overall society. However, the existence of an institutional norm does not imply its acceptance by the addressee.

Table 1: Norm Levels

<table>
<thead>
<tr>
<th>Beneficiaries</th>
<th>Institutional Level (I)</th>
<th>Interaction Level (C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addressee</td>
<td>Society</td>
<td>Grantee Agent</td>
</tr>
<tr>
<td>Legislator</td>
<td>Institution</td>
<td>Agents involved by the agreement</td>
</tr>
<tr>
<td>Validity</td>
<td>Undefined</td>
<td>Known Period of Validity</td>
</tr>
</tbody>
</table>

In addition, these two set of norms are classified into three other categories [21]: Constitutive, Regulative and Procedural norms. The constitutive norms \( N_{\text{Constitutive}} = I_{\text{Constitutive}} \cup C_{\text{Constitutive}} \) allow giving an abstract meaning to facts, environmental elements, etc. They introduce new classifications of facts and terms, called abstract facts. The regulative norms \( N_{\text{Regulative}} = I_{\text{Regulative}} \cup C_{\text{Regulative}} \) describe the ideal behaviour of a system by means of obligations, prohibitions and permissions. Finally, the procedural norms \( N_{\text{Procedural}} = I_{\text{Procedural}} \cup C_{\text{Procedural}} \) are an instrumental approach, i.e. they define the enforcing mechanisms in terms of punishments and rewards carried out by representative agents of the VO. Therefore, procedural norms define a practical connection between a regulation and its consequences [21]. Next, the formal definition of each type of norm is described in detail.

3.2.1. Constitutive Norms

This kind of norms is employed inside the VO at the interaction and institutional levels for the definition of new categories, concepts and facts in terms of which the ideal behaviour of the system is defined.

**Definition 3.3 (Constitutive Norm).** A constitutive norm \( n_c \in N_{\text{Constitutive}} \) is defined as a tuple \( n_c = (x, y, c) \) where:

- \( x \) represents a basic concept or set of basic concepts. It is an expression formed as a conjunction of literals belonging to the set of basic facts \( F \);
- \( y \) represents the new abstract concept.
- \( c \) represents the context in which the basic term \( x \) can be abstracted as the concept \( y \).

The set of constitutive norms is formed by the set of constitutive norms formulated by the VO as well as those defined in the context of an agreement among agents \( N_{\text{Constitutive}} = I_{\text{Constitutive}} \cup C_{\text{Constitutive}} \).
Constitutive). Therefore, two set of facts are defined through the constitutive norms: institutional facts \( (I_{\text{Constitutive}}) \) and agreed facts defined in the context of an agreement among agents \( (C_{\text{Constitutive}}) \).

For example, the next expression contains a constitutive norm \( (n_{\text{constitutive}} \in N_{\text{constitutive}}) \) which defines what is considered as valid register inside a chat unit:

\[
n_{\text{constitutive}} = \langle \text{Done}_{\text{id}}(\text{Serve}(\text{logging}(\text{NickName}))), \text{register}(\text{NickName}), \text{registerAuthority} \in \text{plays}(\text{chat}, \text{Id}) \rangle
\]

It defines that if the logging service is provided by any agent \( (\text{Done}_{\text{id}}(\text{Serve}(\text{logging}(\text{NickName})))) \), then it will be considered as a valid register \( (\text{register}(\text{NickName})) \) if, and only if, the provider agent (identified as \( \text{Id} \)) plays the \( \text{registerAuthority} \) role in the \( \text{chat} \) unit \( (\text{plays}(\text{chat}, \text{Id})) \). As defined above, \( \text{plays} \) is a function that maps each agent and unit with the set of roles played by the agent in this specific unit.

### 3.2.2. Regulative Norms

In this work, we take as a basis the formalization of norms made in [46]. In this proposal a regulative norm is a rule that defines an action as obliged, permitted and prohibited. The conditions under which regulative norms are instantiated and enforced are defined in terms of procedural norms (see Section 3.2.3). Regulative norms allow the definition of the legal behaviour of the system at different levels: from an institutional perspective or from the interaction level. Next, the formalization of regulative norms belonging to each one of these two levels of norms is described.

- **Institutional Level:** they are the normative sentences through which the organization itself determines rights and responsibilities of the society members. This approach distinguishes between the general norms, which control the behaviour of all agents belonging to the NMAS; and rules addressed at a specific set of agents which are playing a certain role inside a group.

  - **Definition 3.4 (General Norm).** A General Norm \( (gn \in GN) \) is defined by a tuple \( \langle D, C \rangle \) where:

    * \( D \in \{F, P, O\} \) is the deontic modality [47], which define the norm type. In this work obligations \( (O) \) and prohibitions \( (F) \) impose constraints on agent behaviours, whereas permissions \( (P) \) are operators that define exceptions to the activation of more general obligations or prohibitions;
    * \( C \) is a logic formula that represents the norm condition; i.e. the actions \( (Ac) \) that must be carried out in case of obligations, that must be avoided in case of prohibitions or that can be freely carried out in case of permissions.

For example, the following norm \( (n_{gn} \in GN) \) defines as mandatory for any agent to provide some functionality:

\[
n_{gn} = \langle O, \text{Register}(\text{ServiceId}) \rangle
\]

Therefore, all agents are obliged to register themselves as providers of one service \( (\text{ServiceId}) \) at least.

  - **Definition 3.5 (Rule).** A rule \( (r \in \text{Rule}) \) is defined as a tuple \( r = \langle D, C, T \rangle \), where:
Next an example of an rule (n_rule ∈ Rule) is provided:

\[ n_{\text{rule}} = (O, \text{Register(logging)}, (\text{registerAuthority, chat})) \]

It defines that all agents that play the registerAuthority role inside the chat are obliged to register their own logging service.

In summary, the set of institutional regulative norms is formed by the general norms and the rules which deal with controlling a specific role (\( I_{\text{Regulative}} = GN \cup \text{Rule} \)).

- **Interaction Level:** they are normative prescriptions that emerge as a result of the interaction among agents. Therefore, this level of norms is formed by the commitments reached by agents along their execution. This proposal distinguishes two types of commitments: those which have been informally defined (agreements) and those which have been formalized at the institutional level (contracts). Therefore, those agreements which have been formalized by some legal representative of the VO are named contracts.

  - **Definition 3.6 (Agreement).** An agreement (\( c_{\text{Regulative}} \in C_{\text{Regulative}} \)) is defined as

    \[ c_{\text{Regulative}} = (D, C, i, j) \]

    where:

    - \( D \in \{F, P, O\} \) is the deontic modality;
    - \( C \) is the norm condition;
    - \( i, j \in (A \cup OU) \) represent the two agents or organizational units involved in the agreement.

    Thus, they are deontic expressions which define the rights and responsibilities derived of the agreement. In particular, rights are defined as permissions whereas responsibilities are formalized by means of obligations and prohibitions.

    As an example, \( n_{\text{agreement}} \in C_{\text{Regulative}} \) defines that an agent \( (a) \) could be authorized to employ other’s resources (\( \text{Act}(r_b) \)) as a consequence of a negotiation process among them:

    \[ n_{\text{agreement}} = (P \text{Act}(r_b), a, b) \]

  - **Definition 3.7 (Contract).** A contract is defined as an agreement recognized by the organization.

    Therefore, the set of contracts (Contract) is a subset of agreements (\( \text{Contract} \subseteq C_{\text{Regulative}} \)).

    As an example of a contract agreed by agents, \( n_{\text{contract}} \in \text{Contract} \) defines that a seller agent \( (s) \) is obliged to deliver certain item (\( \text{Provide(delivery(item))} \)) to a buyer agent \( (b) \):

    \[ n_{\text{contract}} = (O, \text{Provide(delivery(item))}, s, b) \]

In conclusion the set of regulative norms is defined as the union between institutional and interactional regulations (\( N_{\text{Regulative}} = I_{\text{Regulative}} \cup C_{\text{Regulative}} \)). Next, Table 2 summarizes the main features of each type of regulative norms defined in the proposed model. These features of norms...
are based on the well known works on norm typologies [48] [49]. The main features of a norm are related to its formalization, distribution, evaluation, enforcement and learning. A norm is formalized if it has been recognised as a norm by the institution. Regarding the distribution feature of a norm, it represents the extent of knowledge of the norm; i.e. whether the norm is distributed to all the society members, to some group of them or if it is not distributed. The learning process by which an agent internalizes a norm can be: primary, if the agent should know the norm as a result of being a society member; secondary, if the agent should know the norm when it assumes a concrete position inside the society; and the result of an agent interaction. The evaluation feature of a norm determines whether norm compliance can be observed by all the society members (third-party) or only by those agents affected by an interaction (second-party). Finally, the enforcement property has been divided into two attributes: the enforcement mechanisms employed for achieving norm fulfilment; and the enforcement entity, i.e. the type of entity in charge of norm enforcing, explained in previous section 2.2.

<table>
<thead>
<tr>
<th>General Norm</th>
<th>Formalized</th>
<th>Distribution</th>
<th>Learning</th>
<th>Evaluation</th>
<th>Enforcement Mechanism</th>
<th>Enforcement Entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule</td>
<td>Yes</td>
<td>Group</td>
<td>Primary</td>
<td>Third-party</td>
<td>Sanction</td>
<td>Authority(Judge)</td>
</tr>
<tr>
<td>Contract</td>
<td>Yes</td>
<td>No</td>
<td>Interaction</td>
<td>Second-party</td>
<td>Sanction</td>
<td>Authority (Arbiter)</td>
</tr>
<tr>
<td>Agreement</td>
<td>No</td>
<td>No</td>
<td>Interaction</td>
<td>Second-party</td>
<td>Sanction</td>
<td>Second-party</td>
</tr>
</tbody>
</table>

Table 2: Characterization of Regulative Norms

For example, a legal contract is a norm which is not necessarily known by all the organization members. Contracts are stated when a representative of the institution recognises an agreement among agents. In addition, the fulfilment of a contract is evaluated by the affected agents privately, not by the whole society. Regarding enforcement mechanisms, contracts can define both sanctions, for punishing agents which do not respect the agreement; and rewards, as an incentive for the accomplishment of the contract. Since contracts have been given an institutional meaning by a representative of the legal system (authority); then an authority of the legal system would be in charge of applying the defined punishments in case of contract violation.

3.2.3. Procedural Norms

This kind of norms determines the way in which regulative norms are applied in the VO. More concretely, they allow agents in charge of controlling a norm to detect its violation and to carry out the appropriate actions in response.

**Definition 3.8 (Procedural Norm).** A Procedural Norm \( n_{Procedural} \in N_{Procedural} \) links a regulative norm with the mechanisms that allow its application. Thus, it is defined as: \( n_{Procedural} = (n_{Regulative}, A, E, S, R) \) where:

- \( n_{Regulative} \in (N_{Regulative} = I_{Regulative} \cup C_{Regulative}) \) represents the substantive norm controlled by the procedural norm.
• A, E are boolean conditions defined over terms belonging to F, which is the set of basic facts (see definition 3.1). They define the activation and expiration conditions of a regulative norm, respectively.

• S, R ∈ N_{Regulative} are normative expressions which describe the actions that will be carried out in case of norm violation or fulfilment, respectively. This type of expressions is defined as regulative norms.

Regarding an example of a procedural norm, n_{procedural} ∈ N_{Procedural} defines that a seller agent (s) which is obliged to deliver certain item (Provide(delivery(item)) to a buyer agent (b) will be paid 100 euros ⟨⟨O, b, Provide(Payment(100)))⟩⟩:

\[ n_{procedural} = ⟨⟨O, Provide(delivery(item)), s, b⟩, −, −, ⟨O, b, Provide(Payment(100))⟩⟩ \]

3.2.4. Summary of Norms

Next, Table 3 contains a summary of the normative concepts defined before. It details the main elements which characterise each category and the norm types belonging to each category of norms. For example, constitutive norms define a new abstract concept in terms of a conjunction formed by basic facts. Constitutive norms are classified into: legislative norms, which define the process by which the normative system is dynamically modified by agents; and norms for defining new categories of institutional and agreed facts (i.e. \textit{Institutional Definition} and \textit{Agreed Definition}).

<table>
<thead>
<tr>
<th>Norm Category</th>
<th>Norm Elements</th>
<th>Norm Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regulative</td>
<td>Deontic Modality&lt;br&gt;Norm Condition&lt;br&gt;Beneficiaries&lt;br&gt;Target</td>
<td>Institutional Regulations&lt;br&gt;Agreements</td>
</tr>
<tr>
<td>Constitutive</td>
<td>Abstract Fact&lt;br&gt;Basic Fact</td>
<td>Legislative Norm&lt;br&gt;Institutional Definition&lt;br&gt;Agreed Definition</td>
</tr>
<tr>
<td>Procedural</td>
<td>Regulative Norm&lt;br&gt;Activation&lt;br&gt;Expiration&lt;br&gt;Sanction&lt;br&gt;Reward</td>
<td>Enforcement Norm&lt;br&gt;(Sanction and Reward Norms)</td>
</tr>
</tbody>
</table>

Table 3: Summary of Norms

In this section a formal definition of VO for supporting the design of open MAS has been given, which includes normative descriptions that can be employed in enforcement mechanisms that help ensuring global coordination in open systems, taking into account the impossibility of controlling (the majority of) agents directly. This formal definition takes the previous proposals (Section 2.1) as a starting point and extends them by adding dynamical features to the normative model and covering both the different norm levels and types. The next section illustrates how this theoretical model has been implemented inside the THOMAS platform.
4. The THOMAS Approach

The aim of the THOMAS project is to advance and contribute solutions in the development of a MAS architecture suitable for the generation of VOs in open environments, as well as a support platform which allows these systems to be implemented. More specifically, our aim is to develop mechanisms based on organizational structures and VOs that optimise and regulate the coordination of services in open multi-agent systems.

This section contains an overview of the THOMAS approach together with a detailed description of how the normative model proposed in the previous section has been applied into the THOMAS platform.

4.1. THOMAS Platform

THOMAS [14] is an open agent platform that employs a service-based approach as the basic building blocks for creating a suitable platform for intelligent agents grouped in VOs. It feeds on the FIPA\(^3\) architecture, but extends its main components —the Agent Management System (AMS) and the Directory Facilitator (DF) — into an Organization Management System and a Service Facilitator, respectively. Therefore, the two main components of the THOMAS approach are:

- **Service Facilitator** (SF) [50], which is a service manager that registers services provided by external entities and facilitates service discovery for potential clients. Thus, the SF copes with the limitations of the traditional DF, considering semantic information, service composition as well as service roles, goals and durations, following Service Oriented Architectures guidelines.

- **Organization Management System** (OMS) [51], which is responsible for the management of VOs, taking control of their underlying structure, the roles played by the agents inside the organization and the norms that rule the system behaviour. More specifically, the OMS component is in charge of controlling how organizational units (OUs) are created; which entities are participating inside; how these entities are related to each other and which roles they play through time.

4.2. Norm Representation in the THOMAS Architecture

As described in section 3.2, our VO model classifies norms into three main groups: constitutive, regulative and procedural norms. Below, the deontic language for representing these kinds of norms inside the THOMAS platform is explained in detail. The motivation for the explicit representation of norms is to prevent a too general notion of norms. Calling every requirement a norm makes the concept empty and useless [4]. In addition, the explicit representation of norms might allow agents to reason about them and even propose changes of those norms.

4.2.1. Representation of Constitutive Norms

In general, constitutive norms are used to create new abstract concepts which allow regulative norms to be defined in a general way. Following the contextual ontology proposal [52]; in THOMAS constitutive norms are represented as COUNT-AS [53]. Next, syntax, semantics and an example of a constitutive norm are provided.

\(^3\)http://www.fipa.org
**Constitutive Norm Syntax.** Table 2 shows the BNF notation of the grammar for representing constitutive norms in THOMAS. On the one hand, \(<\text{fact}>\) represents both the set of basic facts and the new category defined by the norm. On the other hand, a \(<\text{condition}>\) is defined as a boolean condition defined over state predicates.

\[
<\text{constitutive\_norm}> ::= <\text{fact}> \text{COUNT-AS} <\text{fact}> \text{IN} <\text{condition}>
\]

*Figure 2: BNF syntax of constitutive norms*

**Constitutive Norm Example.** As an example of an abstract concept agreed by agents, Norm 1 defines which action is considered as a contract fulfillment. It claims that if a seller agent \((s)\) carries out the action corresponding to the delivery \((\text{deliver}(\text{item}, d))\) of an item \((\text{item})\) in a specific date \((d)\) before the committed delivery date \((d <= cd)\), then it will be considered as a fulfilled contract \((\text{done}(\text{fulfillContract}))\).

\[
do(s, (\text{deliver}(\text{item}, d)) \text{COUNT-AS} \text{done}(\text{fulfillContract}) \text{IN} \exists cd : \text{done}(\text{signContract}(\text{item}, cd)) \land d <= cd
\]

*Equation 1*

**Constitutive Norm Semantics.** The meaning of a constitutive norm expressed by means of first order logic [54] is:

\[
x \text{COUNT-AS} y \text{IN} c \equiv c \land x \rightarrow y
\]

The rule represents that if the condition \(c\) holds then the basic concept \(x\) can be redefined as the abstract concept \(y\).

**4.2.2. Representation of Regulative Norms**

In our approach, agent actions are related to service controlling and resource management processes. Thus, a normative language which makes possible the definition of constraints on agent behaviours in terms of this type of actions has been described in [55]. The main contributions of the proposed language are: (i) it gives support to organizational concepts such as roles or organizational units; and (ii) the definition of agents’ functionality as standard semantic web services increases norm expressiveness since the actions controlled by norms are described in a more elaborated way.

**Regulative Norm Syntax.** Table 3 shows the normative language syntax which allows regulative norms to be expressed. A regulative norm is defined by means of a deontic operator \(<\text{deontic\_modality}>\), an addressed entity and an action. These actions may be related to service or resource access. Regarding service access norms \(<\text{service\_action}>\) they are related to the publication, provision or usage of services. Resource control norms \(<\text{resource\_action}>\) define authorizations for perceiving and acting over resources.

As previously mentioned, the specification of functionalities by semantic services allows functionality to be defined in a more sophisticated way. In this sense, it allows representing service inputs, outputs, preconditions and effects. Furthermore, there could be additional parameters in a service description, which are independent of the service functionality (non-functional parameters), such as quality of service, deadlines and security protocols. Taking into account

15
that THOMAS works with semantic services, another important data is the ontology used in the service. Thus, when the service description is accessed, any entity will have all needed information in order to interact with the service and make an application that can use this service. Such a description can also be employed for pre-compiled services, in which the process model of the service is composed of the sequence of the elementary services that will be executed, instead of the internal processes of this service [56]. The global functionalities are described as complex services that are composed of atomic services, so a complex service specification describes how agent behaviours are orchestrated. Moreover, functionality is detailed in two ways: services that entities perform and services that entities need. Thus, Service Oriented Computing concepts such as ontologies, process models, choreography, facilitators, service level agreements and quality of service measures can be applied to MAS. OWL-S [42] is the chosen language to represent semantic services in the SF component of the THOMAS framework.

Regulative Norm Examples. As an example of an institutional general norm, Norm 2 authorizes any agent to request acquisition of the Subordinated role.

\[
\text{PERMITTED ADDRESSSEE ANY REQUEST AcquireRole} \quad \text{MESSAGE(CONTENT(Rol} \text{ID"Subordinated")}}
\]

(2)

Norm 3 contains an example of a norm which has been created as a result of an agreement between two agents; agent\(_a\) and agent\(_b\), agents, respectively. This norm determines that the addressee agent is obliged to provide the SearchService service to the beneficiary agent.

\[
\text{OBLIGED ADDRESSSEE agent}_a \text{ BENEFICIARY agent}_b \quad \text{SERVE SearchService} \quad \text{MESSAGE((S ENDER agent}_b \text{) CONTENT(QueryInfo ?qi))}}
\]

(3)

Regulative Norm Semantics. Regulative norms have been defined in the previous section as rules that constrain allowed actions of agents. In this sense, actions play a crucial role in the semantics of regulative norms. Next, their semantics is given by means of dynamic logic [47]:

\[
\begin{align*}
\text{FORBIDDEN } a a & \equiv [\text{Done}_a(a)]^V \\
\text{OBLIGED } a a & \equiv [\neg \text{Done}_a(a)]^V \\
\text{PERMITTED } a a & \equiv [\text{Done}_a(a)]\neg^V
\end{align*}
\]

\(V\) is the well-known predicate for representing violation states; \(a\) represents an action; \(a\) represents an entity to which the norm is addressed to. The predicate \(\text{Done}_a(a)\) represents an action \(a\) that has been performed by an entity \(a\) (see Definition 3.1). The interpretation of this predicate depends on the entity which is affected by the norm. In case of a general norm, which controls behaviours of all the members of the VO, \(\text{Done}_a\) is defined as \(\forall m \in A (\text{Done}_m(a))\). In case of a rule that controls agents which are playing a specific role \(r\) inside a unit \(u\), the predicate \(\text{Done}_a\) represents \(\forall m \in A (r \in \text{plays}(u, m) \land \text{Done}_m(a))\); where \(\text{plays}\) is a function which maps an agent to the roles it plays inside an organizational unit to which it belongs.
\[
<\text{regulative\_norm}> ::= <\text{deontic\_modality}> <\text{addressee}> [<\text{beneficiary}>]
\]
\[
<\text{addressee}> ::= \text{ADDRESSEE} <\text{entity}>
\]
\[
<\text{beneficiary}> ::= \text{BENEFICIARY} <\text{entity}>
\]
\[
<\text{deontic\_modality}> ::= \text{OBLIGED} | \text{FORBIDDEN} | \text{PERMITTED}
\]
\[
<\text{entity}> ::= <\text{agent}> | <\text{role}> - <\text{unit}> | \text{ANY}
\]
\[
<\text{agent}> ::= ?\text{variable} | \text{agent\_id}
\]
\[
<\text{role}> ::= ?\text{variable} | \text{role\_id}
\]
\[
<\text{unit}> ::= ?\text{variable} | \text{unit\_id}
\]
\[
<\text{action}> ::= <\text{resource\_action}> | <\text{service\_action}>
\]
\[
<\text{resource\_action}> ::= \text{ACT(resource\_id)} | \text{PERCEPT(resource\_id)}
\]
\[
<\text{service\_action}> ::= <\text{service\_publication}> | <\text{service\_provision}> | <\text{service\_usage}>
\]
\[
<\text{service\_publication}> ::= \text{REGISTER service\_name PROFILE <profile\_desc>}
\]
\[
<\text{service\_provision}> ::= \text{SERVE service\_name PROCESS <process\_desc>}
\]
\[
<\text{service\_usage}> ::= \text{REQUEST service\_name MESSAGE(<msg\_cont>)}
\]

Figure 3: BNF syntax of regulative norms
4.2.3. Representation of Procedural Norms

Returning to the definition of procedural norms given in the previous section (see Definition 3.8), they define the way in which the VO implements norms, i.e. they establish a connection between a norm and its consequences. In order to represent procedural norms, our normative language has been extended with syntactical rules for representing the violation and exception conditions, the validity interval and sanctions as well as rewards.

**Procedural Norm Syntax.** Figure 4 defines BNF syntax of procedural norms. More specifically, a procedural norm is defined by a regulative norm, whose syntax is contained in Figure 3; and a time interval which specifies its validity period ([START <event>] [END <event>]). Usually, procedural norms define sanctions and rewards as persuasive methods; they are represented through regulative norms which oblige some empowered agent to act as norm defender or promoter.

```
<procedural_norm> ::= <regulative_norm> [START <event>] [END <event>] [SANCTION <regulative_norm>] [REWARD <regulative_norm>]
```

Figure 4: Procedural norm BNF syntax

**Procedural Norm Example.** For example, norm 4 obliges a Supervisor agent to request AddUnit service; if the Supervisor agent does not respect the norm, then it will be expelled from the organization by the OMS as sanction. Norm 4 contains an exception state condition and a validity condition also. In this case, these conditions indicate that the agent is obliged to request AddUnit service before 10 seconds (END(10′′)) if it is the only Supervisor inside the organization (Quantity("Supervisor") > 1).

```
?agent : Supervisor OBLIGED REQUEST AddUnit
EXCEPTION Quantity("Supervisor") > 1
END(10′′)
SANCTION oms OBLIGED PROVIDE Expulse
MESSAGE(CONTENT(?agent,"Supervisor"))
```

(4)

**Procedural Norm Semantics.** As previously argued, procedural norms define the process by which norms are implemented by the VO. The representation of a procedural norm has the following form:

```
regulative_norm START start_event END end_event
SANCTION sanction_norm REWARD reward_norm
```

where regulative_norm is the regulation controlled by the procedural norm; start_event and end_event define the activation interval of the norm; and finally sanction_norm and reward_norm represent the norms which define responsibilities of the norm enforcer entities. The semantics of procedural norms given by means of inference rules [54] is described as:

18
The first inference rule defines a regulative norm \( (\text{regulative}_{\text{norm}}) \) as active after occurrence of the start event \( (\text{start}_{\text{event}}) \). The regulative norm determines the occurrence of the violation condition. Finally, the last two inference rules detect norm deactivation \( (\text{end}_{\text{event}}) \) and retract the regulative norm from the active norm set \( (\neg\text{regulative}_{\text{norm}}) \). If the norm has not been violated, then the reward norm is activated. On the contrary, the sanction will be active if the norm has been violated \( (\text{sanction}_{\text{norm}}) \). These inference rules provide a basic interpretation of procedural norms and their relationship with regulative norms. The following section illustrates how each kind of norms has been implemented in the THOMAS platform.

### 4.3. Norm Control in the THOMAS Architecture

Once the general characteristics of our proposed deontic language have been described, some aspects related to the management of norms are commented in this section. In Open NMAS, internal states of agents are not accessible. Therefore, norms cannot be imposed as agent beliefs or goals. On the contrary, norms have to be implemented in the society. Therefore, this section focuses on the implementation mechanisms employed in THOMAS in order to let norms have an effective influence on agent behaviours.

#### 4.3.1. Operational Interpretation of Norms

As previously argued, the THOMAS platform does not have any mediator layer. Thus, there is not a centralized entity that controls all the interactions carried out by agents. Therefore, only the norms that regulate access to the THOMAS services \( (\text{i.e.} \) the services provided by the SF and the OMS components) are controlled by the platform.

The set of controllable norms in THOMAS is the set of norms which define permissions, prohibitions and obligations concerning the access to THOMAS services. On the one hand, permission and prohibition norms are controllable, since the THOMAS services check whether the service client agent is allowed to perform such request before providing it. More specifically, these norms have been regimented. On the other hand, obligation norms cannot be imposed, since the SF and OMS components have not capabilities to force agents to carry out any action. However, they can monitor the agent behaviour in order to detect the violation and fulfillment of obligation norms and react to them. Thus, controllable obligation norms have been enforced in the THOMAS platform. The rest of norms that can be defined in THOMAS \( (\text{i.e.} \) all those ones that do not concern specific THOMAS services) cannot be controlled directly by the platform, since it is an open system in which heterogeneous and unknown agents can interact freely without any mediation. As a consequence, they must be controlled by additional mechanisms such as: social enforcement, arbitration, etc.

Next, some details about the operational of controllable norms are explained \[51]. In particular, operational semantics of controllable norms have been expressed as Event-Condition-Rules \( (\text{ECA-rules}) \) \[57] since they can be used in rule engines for implementing knowledge-based systems and expert systems. Therefore, ECA-rules have been selected for providing an operational
semantics to norms since they can give a better understanding of how norms have been effectively implemented by means of a rule-based system (this implementation will be described in Section 4.3.2).

- **Service access norms.** They allow the definition of prohibitions and permissions concerning the use of both OMS and SF services. Next, the semantics of both a permission and prohibition norm are given:

\[
\text{on start event do } \oplus \text{permitted}(a) \\
\text{on a if permitted(a) do } \oplus \text{provided}(a) \\
\text{on end event do } \oplus \text{permitted}(a) \\
\text{on start event do } \oplus \text{forbidden}(a) \\
\text{on a if forbidden(a) do } \oplus \text{notAllowed}(a) \\
\text{on end event do } \oplus \text{forbidden}(a)
\]

(5)

The first three ECA-rules (5) describe how a permission norm has been regimented, whereas the last three ones represent a prohibition norm semantics (6). According to this, a general service access norm is controlled by means of the definition of three new rules. The first rule detects norm activation and asserts the permission (\(\oplus \text{permitted}(a)\)) or prohibition (\(\oplus \text{forbidden}(a)\)). If the action (\(a\)) occurs and \(a\) is allowed, then the service is provided (\(\oplus \text{provided}(a)\)). On the contrary, if \(a\) is forbidden, then \(\text{notAllowed}\) fact is asserted. The last rule retracts the norm from the expert system when the norm is deactivated, i.e. when the completion event is detected (end_event).

- **Obligation norms.** These norms are enforced by the THOMAS platform. Thus, it might persuade agents to behave correctly by performing sanctions and rewards. Following, a formal description of obligation semantics is shown:

\[
\text{on start event do } \oplus \text{expected}(a) \\
\text{on a if expected(a) do } \ominus \text{expected}(a) \bullet \text{reward} \\
\text{on end event if expected(a) do } \ominus \text{expected}(a) \bullet \text{sanction}
\]

(7)

Thus, the implementation of obligation norms involves controlling the activation of the obligation. Then the THOMAS infrastructure waits for the fulfilment of the expected action (\(a\)), i.e. a new expectation (\(\oplus \text{expected}(a)\)) is asserted. If the action is performed before the deadline (end_event) then the reward is carried out. Otherwise the sanction will be performed.

4.3.2. Norm Implementation

The operational interpretation of norms, described above, provides a computational meaning to norms. However, these operational norms are too abstract for being translated into a software implementation. Next, a description concerning the computational implementation of norms inside the THOMAS platform is given.

The process of normative management covers: i) maintenance of the organizational state; ii) performance of the enforcement actions (sanctions and rewards); and iii) determination of the
allowed actions according to the normative context and the organizational state. These functionalities have been implemented as internal services of THOMAS, by means of a rule-based system implemented in Jess⁴ that maintains a fact base representing the organizational state and it also detects norm activation. Figure 5 shows a schematic view of this process. This rule-based system is accessed by the THOMAS services through two different services: a query service which allows determining whether an agent is allowed to request a service; and an updating service which is in charge of adding and deleting facts into the rule system in order to register the organizational current state. In addition, this system monitors agent behaviours and carries out sanctions and rewards related to controllable obligation norms. Each one of its four main components is briefly explained as follows [58]:

1. **Norm Interpretation** (Figure 5 Item 1). The formal specification of a norm (described with our BNF syntax defined in section 4.2) is translated into a normative fact or norm instance (defined with ECA-rules) which can be managed by a rule-based system. As stated before, constitutive norms create new categories of facts which make up the (institutional) ontology employed when general norms are defined. Therefore, it is necessary to translate abstract norms into concrete norms according to the normative ontology in order to make norms computable.

2. **Norm Activation Detection** (Figure 5 Item 2). A new fact corresponding to the new norm is asserted into the belief base. As previously argued, norms are not active at all times. In fact, the activation of norms might be defined in terms of state and temporal conditions. As a consequence, a set of rules for detecting the activation and deactivation of norms are needed [26, 59]. These rules are the codification in Jess of the operational semantics of norms provided in Section 4.3.1.

---

⁴http://herzberg.ca.sandia.gov/
3. **Norm Enforcement** (Figure 5 Item 3). This norm management process is informed by the THOMAS services of both the organizational changes and the occurrence of events related to the usage of THOMAS functionalities through the update service. Therefore, it is able to monitor the way in which agents interact with the THOMAS infrastructure. Consequently, the THOMAS platform acts as a norm promoter and defender by carrying out sanctions and rewards related to controllable obligation norms.

4. **Norm Regimentation** (Figure 5 Item 4). When a new request for a THOMAS service is sent to the OMS or the SF, they register this fact in the organizational rule-based system using the update service. Then, using the query service, the normative context is analysed in order to determine whether the client agent is allowed to request this service according to the organizational state. If so, then the requested service is provided. The determination of the allowed actions to a client agent that requests a certain THOMAS service offered by the OMS or the SF is made through the query functionality provided by the rule system. This function follows a criterion for norm precedence known as *lex specialis* [60], which states that the most specific norm precedes the rest of norms. Therefore, the query service begins with checking permissive norms addressed to the agent. If there is not any norm, prohibition norms are taken into consideration, since permissive norms are defined as exceptions to a more general prohibition norm. In the last case, permissive and prohibition norms addressed to the roles played by the client agent are checked. If no norm is found, the service is considered as permitted by default.

Along this section, details concerning the implementation of our normative model for VO (proposed in Section 3.2) have been described. This implementation of norms covers: i) the definition of a normative language for allowing an explicit representation of norms; ii) the software implementation of the normative control in the THOMAS platform.

5. **Conclusions**

VOs have been employed as a support in the design of open multi-agent systems. These complex scenarios require an extended concept of norm in order to face with the heterogeneity and dynamic features of both the participants and the environment. Therefore, current agent architectures and platforms must integrate these concepts to allow designers to employ higher abstractions when modelling and implementing these complex systems. With this aim, the THOMAS platform has been implemented as a support platform suitable for the generation of VOs.

Contributions of this paper can be summarised as follows:

- The definition of a model of norms for controlling VOs (Section 3.2). The main advantages of our normative model for VOs are: (i) it covers the different norm levels which have a social meaning, i.e. interaction and institutional levels; (ii) the different types of norms are considered, i.e. constitutive, regulative and procedural norms; (iii) it is a general normative model, since it does not define the logical constitution of norms; (iv) it allows a structured definition of the normative system; and (v) it offers mechanisms for the creation and derogation of norms, by means of constitutive norms which define the dynamics of the legal system.

- In addition, this work describes how norms have been given a computational interpretation in the THOMAS platform (Section 4). In order to allow both agents and the system...
designers to represent each type of norms a formal deontic language has been described in Section 4.2. However, norms defined through this language are too abstract to be computationally implemented. In this sense, the THOMAS platform has been extended with a normative management process (Section 4.3) which consists of a rule-base system for implementing norms. This rule-based implementation allows norms which regulate an agent organization to be modified at run-time, allowing the VO performance to be adapted in response to the environmental changes.

As future work, we plan to extend this proposal by providing a BDI agent architecture [18] in which the different types of norms defined by our normative model are represented. Therefore, normative agents will be able to consider their adopted norms in their decision making processes. Preliminary results of this ongoing work can be found in [61, 62]. The possibility of developing agents which are able to reason about norms will allow some legal responsibilities of the VO to be delegated into certain agents which will act as legislators or judges in the agent platform.

References