A Norm-Based Organization Management System

Natalia Criado, Vicente Julián, Vicente Botti, and Estefania Argente[⋆]

Grupo de Tecnología Informática - Inteligencia Artificial
Departamento de sistemas informáticos y computación
Camino de Vera S/N 46022 Valencia (Spain)
{ncriado,vingalda,vbotti,eargente}@dsic.upv.es

Abstract. Virtual organizations are conceived as an effective mechanism for ensuring coordination and global goal fulfilment of an open system, in which heterogeneous entities (agents or services) interact and might also present self-interested behaviours. However, available tools rarely give support for organizational abstractions. The THOMAS multi-agent architecture allows the development of open multi-agent applications. It provides a useful framework for the development of virtual organizations, on the basis of a service-based approach. In this paper, the Organization Management System component of the THOMAS architecture is presented. It is in charge of the organization life-cycle process, including the normative management. It provides a set of structural, informative and dynamic services, which allow describing both specification and administration features of the structural elements of the organization and their dynamics. Moreover, it makes use of a normative language for controlling the service request, provision and register.

Keywords: Multi-Agent Systems, Normative Language, Virtual Organizations, Web Services.

1 Introduction

A promising approach in the multi-agent systems (MAS) area is the development of open systems. The main features of open systems are: (i) they are populated by heterogeneous agents which can enter or leave the system dynamically; and (ii) they are situated in dynamic environments. Therefore, organizations are conceived as an effective mechanism for coordinating the behaviour of heterogeneous agents, imposing not only structural restrictions on their relationships, but also normative restrictions on their behaviour [1,2]. Thus, organizations describe the system functionality (i.e. roles, tasks, services), the norms that control agent behaviours, the formation of groups of agents, the global goals pursued by these

[⋆] 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.

groups and the relationships between entities and their environment. Moreover, the potential changes on the dynamic environment might require the adaptation of the organizational structure and functionality.

The "computing as interaction" paradigm [3] defines computation as an inherent social activity that takes place by means of communication between computing entities. More specifically, large distributed systems are conceived in terms of service provider or consumer entities [4]. Therefore, the relevant technological approaches of this paradigm are service oriented architectures (SOA) and MAS. On the one hand, services provide a standard infrastructure for the interaction among heterogeneous software entities. On the other hand, MAS offer a more general and complex notion of SOA; agents, due to their intelligent and social capabilities, allow the redefinition of traditional services adding new features such as dynamic service composition, negotiation about quality of service, etc. In the last years, several works have focused on the problem of integrating these two approaches, in order to model autonomous and heterogeneous computational entities in dynamic and open environments. Their main effort is directed at masking services for redirection, integration or administration purposes [5].

Taking this integrating view into account, THOMAS has been defined as an open architecture for large-scale open multi-agent systems, based on a service-oriented approach [6]. This architecture provides agents with a set of services for offering and discovering entities' functionality and for managing the organization life-cycle. With this purpose of achieving a better integration among MAS and Web Services, all the functionalities (i.e. agent and also THOMAS functionalities) are provided, required and published in THOMAS employing Web Services standards (such as OWL-S), and they can also make use of traditional Web Services.

One of the main components of the THOMAS architecture is the Organization Management System (OMS), which is responsible for the management of the organizations and their constituent entities. In order to allow this management, the OMS must provide a set of structural, informative and dynamic services for describing both specification and administration features of the structural elements of the organization and their dynamics. In this sense, the OMS provides adaptive mechanisms for creating organizational structures that allow optimizing the coordination in Virtual Organizations (VOs), taking into account the heterogeneity of agents and services.

With the aim of allowing the organization management, the OMS requires a normative language for controlling how services can be employed, i.e., when and how agents can request, provide or publish not only their services, but also these ones provided by the open architecture. In this sense, a normative language that imposes a deontic control over agents for requesting, providing or publishing services has been defined. Then the OMS provides regulatory mechanisms that guarantee a globally efficient coordination in open systems taking into account the impossibility of controlling (the majority of) the agents and services directly.

Following, a description of the proposed OMS component, in charge of the organization management by means of organizational services is detailed. Its norm management process, both with a description of its norm representation language is detailed in section 3. The implementation of this OMS component is explained in section 4. A case study is contained in section 5. Finally, discussion and conclusions are detailed in sections 6 and 7, respectively.

2 Organization Management System

As previously mentioned, the THOMAS architecture has the aim of integrating both multi-agent systems and service-oriented computing technologies as the foundation of virtual organizations. This open architecture for large-scale open multi-agent systems is composed of a range of services included on different modules or components[1]. In this sense, agents have access to the architecture infrastructure through the following components:

– **Service Facilitator** (SF). This component provides a mechanism and support by which organizations and agents can offer and discover services.
– **Platform Kernel** (PK). It maintains the basic management services for an agent platform. It integrates the FIPA AMS and the FIPA communication network layer.
– **Organization Management System** (OMS). This component is mainly responsible of the management of the organizations and their entities. Thus, it gives support to the organization life-cycle management.

The present paper is focused on this last component. It is in charge of controlling the organizational life-cycle process. In THOMAS, organizations are structured by means of *Organizational Units* (OU) [7] which represent a set of agents that carry out some specific and differentiated activities or tasks, following a predefined pattern of cooperation and communication. An OU is formed by different entities along its life-cycle which can be either single agents or other OUs. They represent a virtual meeting point because agents can dynamically enter and leave organizational units, by means of adopting (or leaving) roles inside. An OU has also an internal topology (i.e. hierarchical, team, plain), which imposes restrictions on agent relationships (for example, supervision, monitoring or information relationships). A more detailed explanation of the OU concept and a description of different topologies can be found in [7].

The OMS is in charge of controlling how the Organizational Units are created, which entities are participating inside them, how these entities are related and which roles they are playing through time. For this reason, the OMS offers agents a set of services for organization life-cycle management, classified in:

– **Structural services**, which enable agents to request the OMS to modify the structural and normative organization specification. They comprise services for adding/deleting norms (*RegisterNorm*, *DeregisterNorm*), adding/deleting roles (*RegisterRole*, *DeregisterRole*) and creating new organizational units or deleting them (*RegisterUnit*, *DeregisterUnit*). Publishing these services enables agents to modify the organization structure through its life-time.

[1] `http://www.fipa.org/docs/THOMASarchitecture.pdf`

– **Informative services**, that provide information of the current state of the organization, detailing which are the roles defined in an OU (*InformUnitRoles*), the roles played by an agent (*InformAgentRoles*), the specific members of an OU (*InformMembers*), its member quantity (*InformQuantity*), its internal structure (*InformUnit*), and the services and norms related with a specific role (*InformRoleProfiles*, *InformRoleNorms*).
– **Dynamic services**, which allow defining how agents can adopt roles inside OUs (*AcquireRole*, *LeaveRole*) or how agents can be forced to leave a specific role (*Expulse*), normally due to sanctions. Publishing these services enables external agents to participate inside the system.

This set of services for organization life-cycle management allows defining specification and administration features of the structural components of the organization (roles, units and norms) and their dynamics (entry/exit of entities). However, a specific control on who can make use of these services and in which conditions is needed. This type of control is defined by means of norms.

3 Norm Management

Normative systems have been defined as a mechanism for enabling cooperation inside Open MAS. In this sense, norms are persuasive methods for obtaining the desired behaviour from agents. In addition, norms can be viewed as a coordination skill for organizing MAS, since they specify the desired behaviour of the society members [8]. Regarding this second conception of norms, our proposal consists on employing norms for regulating agent organizations. More specifically, this work has the purpose of applying the normative theory for defining the way in which agents may modify the structure of their organization (norms, organizational units and roles) and its execution components, in order to adapt it dynamically to environmental changes.

Recently, works on norms have focused on overcoming the gap between theoretical works on normative systems and practical MAS. They give a computational interpretation of norms that allows norm execution. However, none of them raises the normative management problem or gives an infrastructure that enables including the normative theory inside the implementation of real MAS applications. With this aim, we have developed both a normative language for controlling agent organizational dynamics and a normative management engine, which are explained in the following sections.

3.1 Norm Representation Language

Before addressing the norm controlling problem, the definition of a formal language for the representation of the normative concepts is needed. Our normative language is mainly based on previous works for defining norms inside Electronic Institutions (EI) [9,10]. These works define a normative language for controlling the communicative acts (illocutions) of agents inside an EI. In addition, they propose an extension of this language for allowing the definition of norms concerning non-dialogical actions. Our language takes these approaches as a starting point and increases them in order to give support to functional and organizational management. More concretely, it makes possible the definition of constraints on agent behaviours in terms of actions related to service controlling and organizational processes. The main contributions of the proposed language are: (i) it allows the definition of consequences of norms by means of sanctions and rewards; (ii) it gives support to organizational concepts such as roles or organizational units and; (iii) the definition of agents' functionality by means of the OWL-S standard increases norm expressiveness, as will be argued lately.

Following, some issues about the developed normative language are commented. For a more detailed description of this language see [11].

The proposed language is a coordination mechanism that attempts to: (i) promote behaviours satisfactory to the organization, i.e. actions that contribute to the achievement of global goals; and (ii) avoid harmful actions, i.e. actions that prompt the system to be unsatisfactory or unstable. Norm semantics is based on deontic logic since it defines obligations, permissions and prohibitions. Our approach conceives norms as expectations that may not be fulfilled. Thus, norms are not impositions (i.e. they are not automatically regimented on agents by their designer), but they are methods for persuading agents to behave correctly by means of the application of sanctions and rewards. For example, an agent would be expelled from the organization as sanction if it violates norms systematically. An analysis on the effectiveness of both sanctions and rewards as mechanisms for enforcing norms is over the scope of this article. However, this work assumes that agents are aware of norms, punishments and rewards. In this sense, a normative reasoning process for norm-aware agents has been proposed in [12].

Norms define agent rights and duties in terms of actions that agents are allowed or not to perform. Actions have been divided in two categories: actions related to the organizational aspects of MAS; and actions concerning service accessing. Hence, two main types of norms have been defined:

- **Organizational Norms:** related to services offered by the OMS to members of the organization. They establish organizational dynamics, e.g. role management (role cardinalities, incompatibility between roles) and the protocol by which agents are enabled to acquire roles.
- **Functional Norms:** related to services offered by the members of the organization or the SF. They define role functionality in terms of services that can be requested/provided, service requesting order, service conditions, protocols that should be followed, etc. They establish service management according to previous service results, environmental states, etc.

Table 1 details the reduced BNF syntax of this language. A norm is defined by means of a deontic operator (<*deontic_concept*>), an addressed entity and an action, that concerns organizational (<*organizational_action*>) or functional (<*functional_action*>) management. The <*temporal_situation*> field establishes a temporal condition for the activation of the norm. It can be expressed as a deadline, an action or a service result. A norm may also contain a state condition

Table 1. On the left side, BNF syntax of norms is detailed. On the right side, its semantics expressed by means of dynamic logic is given. α is an action description. β is an state description. V, $DO(\alpha)$ and $DONE(\alpha)$ are the well-known predicates for representing violation states, an action α that will be done next and an action α that has been performed. Finally, ϕ represents a norm.

<norm>::=<deontic> <entity> <action> [<temporal>] [*IF* <if_condition>] | *norm_id*	
<ext_norm>::=<norm> [*SANCTION*(<norm>)] [*REWARD*(<norm>)]	$\phi : FORBIDDEN\ \alpha \equiv [\alpha]V$ $\phi : OBLIGED\ \alpha \equiv [\neg\alpha]V$
<deontic>::=*OBLIGED* | *FORBIDDEN* | *PERMITTED*	$\phi : PERMITTED\ \alpha \equiv [\alpha]\neg V$
<entity>::=<agent>: <role> [− <unit>] | <role> [− <unit>] | <entity_id>	$\phi : \phi' SANCTION\ \alpha \equiv \phi' \wedge [V]DO(\alpha)$ $\phi : \phi' REWARD\ \alpha \equiv \phi' \wedge [\neg V]DO(\alpha)$
<agent>::=?*variable* | *agent_id* <role>::=?*variable* | *role_id* <unit>::=?*variable* | *unit_id* <entity_id>::=*agent_id* | *role_id* | *unit_id*	$\phi : \phi' BEFORE\ \alpha \equiv \phi' \vee DONE(\alpha)$ $\phi : \phi' AFTER\ \alpha \equiv [\alpha]\phi'$ $\phi : \phi' BETWEEN(\alpha_1, \alpha_2) \equiv [\alpha_1]\phi' \vee DONE(\alpha_2)$
<action>::=<functional_action> | <organizational_action>	$\phi : \phi' IF \beta \equiv \beta \rightarrow \phi'$
<temporal>::=*BEFORE* <sit> | *AFTER* <sit> | *BETWEEN*(<sit> , <sit>)	

for its activation (<*if_condition*>). It is a boolean condition expressed over variables, identifiers, failed or satisfactory states of norms or service results.

Usually, obligations and prohibitions have sanctions and rewards as persuasive methods. Sanctions and rewards are represented through norms addressed to entities that will act as norm defenders or promoters. The definition of sanctions and rewards recursively as norms can create an infinite chain of norms. Thus, not only addressed agents might be controlled by norms, but also their controllers (defenders or promoters). Following M. Luck et al. proposal [13], our normative model does not impose any restriction on this fact, so it is the norm designer who is in charge of specifying when to stop this recursive process, i.e. when a controller is trustworthy enough. For example, norm 1 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 1 contains a state condition and a temporal condition also. In this case, these conditions indicate that the agent is obliged to request *AddUnit* service before 10 seconds if it is the only *Supervisor* inside the organization.

$$\begin{aligned}
?agent : Supervisor\ OBLIGED\ REQUEST\ AddUnit \\
IF\ InformQuantity(``Supervisor'') = 1 \\
BEFORE(10'') \\
SANCTION\ oms\ OBLIGED\ PROVIDE\ Expulse \\
MESSAGE(CONTENT(?agent, ``Supervisor''))
\end{aligned} \tag{1}$$

Organizational norms are related to actions that allow agents to request organizational services (<*org_service*>) for adopting roles, registering new norms, etc. These services are provided by the OMS. Functional norms are defined in terms of actions related to the publication (REGISTER), provision (SERVE) or usage (REQUEST) of services. The BNF syntax of both organizational and functional actions is detailed in Table 2. Norm 2 contains an example of a functional norm. It obliges a service *Provider* agent to register its own *SearchService* service.

$$\begin{aligned}?agent : Provider\, OBLIGED \\ REGISTER\, SearchService \\ PROFILE \\ INPUT(ServiceDescription : String) \\ OUTPUT(ServiceID : Identifier)\end{aligned} \tag{2}$$

Table 2. BNF syntax of organizational and functional actions

```
<organizational_action>::=REQUEST <org_service> MESSAGE(<msg_cont>)
          <org_service>::=<structural_service> | <dynamic_service> | <informative_service>
   <structural_service>::=RegisterNorm | RegisterRole | DeregisterNorm |
                          DeregisterRole | DeregisterUnit | RegisterUnit
  <informative_service>::=InformUnitRoles | InformAgentRoles | InformUnit | InformMembers |
                          InformRoleProfiles | InformRoleNorms | InformQuantity
      <dynamic_service>::=AcquireRole | LeaveRole | Expulse

    <functional_action>::=<serv_publication> | <serv_provision> | <serv_usage>
     <serv_publication>::=REGISTER service_name PROFILE <profile_desc>
                          [PROCESS<process_desc>]
    <service_provision>::=SERVE service_name PROCESS <process_desc>
                          [MESSAGE(<msg_cont>)]
        <service_usage>::=REQUEST service_name MESSAGE(<msg_cont>)
```

As previously mentioned, the specification of functionalities by means of OWL-s standard allows defining functionality more expressively: representing service preconditions and effects; global functionalities are described as complex services that are composed of atomic services, so a complex service specification describes how agent behaviours are orchestrated; and functionality is detailed in two ways: services that entities perform and services that entities need. Thus, Service Oriented Computing (SOC) concepts such as ontologies, process models, choreography, facilitators, service level agreements and quality of service measures can be applied to MAS. Our proposal of normative language offers support for specifying knowledge about a service following OWL-s ontology. The profile (*<profile_desc>*) for advertising and discovering services contains input and output parameters of the service and its preconditions and postconditions. The process model (*<process_desc>*) gives a detailed description of a service's operation. It details the sequence of actions carried out by the service. These actions are linked through each other by means of different control constructs: CONNECTS indicates a sequential ordering between two actions; JOIN indicates a concurrence between actions and a final synchronization of them; IF-THEN-ELSE and

Table 3. BNF syntax of service profile and process

```
 <profile_desc>::=[INPUT(<param_list>)] [OUTPUT (<param_list>)]
                  [PRE(<cond_exp>)][POST(<cond_exp>)] | profile_id
<process_desc>::=process_id | ?variable | <action> CONNECTS <process_desc> |
                  <action> JOIN <process_desc> |
                  IF <cond_exp> THEN(<process_desc>) [ELSE (<process_desc>)] |
                  WHILE <cond_exp> DO(<process_desc>)
    <msg_cont>::=[SENDER(<entity>)] [RECEIVER (<entity>)]
                  [PERFORMATIVE (performative_id)] CONTENT (<args>)
      <action>::=task_id(<param_list>) | <service_usage>
  <param_list>::=variable : type [,<param_list>]
```

WHILE-DO define the classical control structures. Finally, the grounding provides details on how to interoperate with a service, via messages <*msg_cont*>). Table 3 contains syntax of service processes, profiles and requesting messages.

In this section, a general language for controlling agent service access has been briefly described. For further details and examples see [11].

3.2 Norm Management Process

This section describes aspects related to the management of organizational norms. As previously mentioned, our formalism allows representing constraints over organizational dynamics. Thus, the controlled norms define access to the organizational services provided by our architecture.

Recently, the line of research on computational implementation of norms is based on the Electronic Institution (EI) proposal [9,10]. The EIs provide a framework for heterogeneous agent cooperation. However, they are not an open environment in its broadest sense, because agents interact inside the institution through the infrastructure provided by the EI. Therefore, the behaviour of external agents is completely controlled by the institution, which allows or not agents to pronounce certain illocutions. Thus, norms are pre-imposed on agents. The institutional mediation prevents agent behaviour from deviating from desired behaviour. Moreover, the fact that all communications are made through the institution allows an easy regimentation and enforcement of norms. In this sense, the existence of a middle-ware for mediating the agent communication avoids the need to take into consideration the limitations that exist in open environments. Such limitations are related to the detection of fact occurrence and the extra capabilities needed in order to impose norms upon other agents.

Our proposed virtual organization architecture is completely different since it does not have any mediator layer. On the contrary, agents are allowed to interact freely. In this sense, our architecture is more related to the notion of *Partially Controlled MAS* [14], in which agents may deviate from ideal behaviour. As a consequence, there has to be a control mechanism for motivating agents to obey the norms. However, our architecture offers a set of services for the management of the organizations life-cycle. In this sense, the OMS is not a centralised entity that controls all the interactions performed by agents. On the contrary, it is a *controllable* entity (which is directly controlled by the system designer [14]) which offers a set of organizational services in order to give support to agent cooperation. One of these coordination mechanisms is the definition of norms that regulate agent behaviour. The OMS does not control the agent communications, it is only in charge of implementing the norms that regulate access to OMS services. The verifiability issue of a norm is a crucial aspect in the normative management process since there is not any mediator middle-ware that controls all the communications.

Norm Verifiability. Works on norm implementation conceive norms as a mechanism for enabling coordination and cooperation inside open MAS. Nevertheless, none of them faces with one of the main problems inside open systems, which is the existence of limitations. The term *limitation* refers to the fact that an entity needs extra information and capabilities in order to act as a norm supervisor or controller. Therefore, an analysis of normative verifiability is needed, before dealing with norm implementation.

The OMS can control norms which are related to the provision of organizational services. Therefore, the set of OMS verifiable norms is a subset of the organizational norms. Verifiable norms are *regulative* norms that define ideal behaviour by means of obligations, prohibitions and permissions. More specifically, permissive and prohibition norms concerning the access to OMS services are controllable, since the OMS checks whether the client agent is allowed to perform such request before providing it. On the other hand, obligation norms can not be imposed by the OMS as it has not capabilities to force agents to carry out an action. However, the OMS can detect the violation of an obligation norm related to an OMS service and perform the associated sanction. On the contrary, if the norm has been fulfilled then the OMS will carry out the actions specified by the reward. Logically, obligation norms should be active for a certain period of time, i.e. normative activation and deactivation events must be defined in order to allow the OMS to determine the norm fulfilment.

Verifiable conditions are related to informative services provided by the OMS such as role cardinalities, information about roles played by agents, etc. Regarding detectable events, they are the request and provision of services offered by the OMS. Both verifiable condition and detectable events syntax are a refinement of the general condition syntax proposed in [11]. Finally, sanctions and rewards can be defined by means of norms that oblige the OMS to request or provide a specific service, as shown in norm 2.

In this section, issues concerning the formal language for representing norms and syntax of verifiable organizational norms have been detailed. Not only an abstract component in charge of the management of organizations has been proposed, but also an implementation of the OMS component has been made. Next, this implementation is described.

4 Organization Management System Implementation

As previously argued, the OMS is a *controllable* entity which offers a set of organizational services. This system maintains a fact base representing the

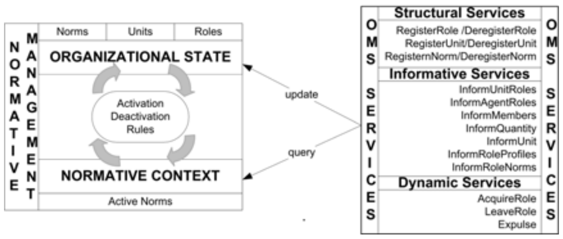

Fig. 1. OMS Implementation

organizational state, and it is also in charge of controlling verifiable norms. The implementation has been included in a prototype of the THOMAS architecture[2]. As Figure 1 illustrates, the OMS implementation is composed by two main elements: the implementation of the organizational services and the implementation of the normative management process, which are described next.

4.1 OMS Services Implementation

As previously stated, the set of services provided by the OMS are those ones related to the organizations life-cycle management, i.e. the management of their structural components (unit, roles and norms) and the organizational dynamics.

The service request management process consists of:

- (i) Firstly, the normative context must be analysed in order to determine whether the client agent is allowed to request this service according to the organizational state.
- (ii) If so, then the OMS provides the requested service.
- (iii) If the organizational state changes as a result of the service provision, then this state is updated.

Mainly, the service provision process consists of an input verification phase and the updating of the organizational state according to the produced changes. As an example, the process of registering of a new norm involves:

- (i) Verification of input data, i.e. checking the adequacy of norm syntax as well as the uniqueness of the provided norm identifier.
- (ii) Norm verifiability detection, i.e. determine if the norm control should be carried out by the OMS. Verifiability checking is done by means of an interpreter, which has been automatically built from the BNF syntax of our organizational normative language employing the JavaCC[3] tool.
- (iii) Inconsistency checking. Taking [15,16] as a starting point, an inconsistent situation is that one in which an action is forbidden and permitted or forbidden and obliged simultaneously. The off-line detection of all norm inconsistencies is not possible, since the norm activation conditions are based

[2] `http://www.dsic.upv.es/users/ia/sma/tools/Thomas/`
[3] `https://javacc.dev.java.net/`

on the detection of events and facts that may occur during execution. For the moment, the inconsistency detection mechanism is restricted to the determination of *static inconsistencies*, which are situations in which the same action is defined unconditionally as permitted and forbidden to the same entity. As future work, we will employ results of theoretical works on norm change[4] as well as conflict resolution techniques for solving *dynamic inconsistencies* among norms.
- (iv) The fact corresponding to the new norm is registered in the normative system.

Due to lack of space, only the organizational service for registering a new norm has been explained in detail. However, all services needed for the management of units, roles, norms and organizational dynamics, have been implemented in a similar way.

4.2 Norm Management Implementation

This process covers the maintenance of the organizational state and the determination of the allowed actions according to the normative context and the organizational state. These two functionalities have been implemented as internal services of the OMS, by means of a rule-based system in Jess[5] that maintains a fact base representing the organizational state and it also detects norm activation. Thus, the *update* service is in charge of adding and deleting facts into the rule system in order to register the organizational current state.

The determination of the allowed actions is made by means of the analysis of the normative context. It has been performed as an internal *query* service offered by the normative manager to the OMS services. This checking is performed by the OMS each time it receives a new service request. Then the OMS checks whether it exists a norm addressed to the client agent that forbids such service request. Thus, an action is considered as allowed when there is not any norm that forbids it explicitly. Norms can be addressed to any agent that plays a specific role or they can affect a specific agent also. Consequently, a criterion for norm precedence is needed. In this case, a rule known as *lex specialis* has been employed [17]. Therefore, the normative analysis begins with checking agent addressed norms. If there is not any norm, norms related to the roles played by the agent are considered.

Regarding detection of norm activation and deactivation, this functionality has been implemented through a set of rules that detect the occurrence of the activation and deactivation events. This implementation of norms by means of rules is based on a previous work aimed at implementing norms for Electronic Institutions [9]. Next, some details about the implementation of each type of norm are commented:

[4] `http://icr.uni.lu/normchange07/`

[5] `http://herzberg.ca.sandia.gov/`

– *Service access norms.* As previously mentioned, these norms allow the definition of prohibitions and permissions concerning the use of organizational services. More formally, the semantics of permission norms, expressed as Event-Condition-Action rules (ECA-rules), is:

$$\begin{aligned}\textbf{on } event_{start} \textbf{ if } if_{condition} \textbf{ do } \oplus permitted(a)\\ \textbf{on } a \textbf{ if } permitted(a) \textbf{ do } \oplus provided(a)\\ \textbf{on } event_{end} \textbf{ do } \ominus permitted(a)\\ \textbf{if } not(if_{condition}) \textbf{ do } \ominus permitted(a)\end{aligned} \tag{3}$$

According to this, a general service access norm is controlled by means of the definition of four new rules in the expert system. The first rule detects norm activation and asserts the permission ($\oplus$*permitted*(a)). If the action (a) occurs and it is allowed, then the service is provided ($\oplus$*provided*(a)). The last two rules retract the norm from the expert system when the norm is deactivated, i.e. when the condition ($if_{condition}$) is not true or the completion event is detected ($event_{end}$). In case of prohibition norms, a prohibition ($\oplus forbidden(a)$) is asserted. Thus, if an action a is forbidden, then a *notAllowed* fact is asserted.

– *Obligation norms.* They cannot be directly imposed by the OMS, since it is not able to force another agent to carry out a specific action. However, it might persuade agents to behave correctly by performing sanctions and rewards; i.e. the OMS acts as norm enforcer. Following, a formal description of obligation semantics is shown:

$$\begin{aligned}\textbf{on } event_{start} \textbf{ if } if_{condition} \textbf{ do } \oplus expected(a)\\ \textbf{on } a \textbf{ if } expected(a) \textbf{ do } \ominus expected(a) \bullet reward\\ \textbf{on } event_{end} \textbf{ if } expected(a) \textbf{ do } \ominus expected(a) \bullet sanction\end{aligned} \tag{4}$$

Thus, the implementation of obligation norms consists on controlling the activation of the obligation. Then the OMS waits for the fulfilment of the expected action (a), i.e. the OMS asserts a new expectation ($\oplus$*expected*(a)). If the action is performed before the deadline ($event_{end}$) then the reward is carried out. Otherwise the OMS will perform the sanction.

This section has illustrated some details of the implementation of the OMS entity. The next section illustrates a case study which describes how an untrusted external agent is prevented from participating in the agent society.

5 Case Study

In order to illustrate the performance of the OMS services and the normative management process carried out by the OMS, a case-study example of a customised information system has been employed in this section. This example, named *InformationSystem*, aims at distributing relevant information of different topics in an efficient way, trying to guarantee the maximum quality of this information. Thus, users might ask the system for information about some specific topic punctually or they might request to be a member of the system. As a member, a user can supply its own information for a specific topic, he can also create new topics and even classify or evaluate the quality of the information given by others.

In this section, this system is modelled as an Open MAS in which external agents can participate as information requesters, providers or even information evaluators. Following, a description of the organizational structure of the *InformationSystem* organization is explained, and also a dynamical usage of the organization is detailed, providing an execution scenario.

5.1 Organization Structure

Three roles have been identified in the *InformationSystem* example: (i) *Consultant*, that requests information to the system; (ii) *Provider*, that is allowed to provide new information (documents); and (iii) *Evaluator*, who is in charge of evaluating the documents and controlling member behaviours. Two domain services have been identified: *ProvideInfo* service, that allows *Providers* to add new documents; and *EvaluateInfo* service, which is offered by *Evaluators* in order to assess the quality of a given document.

5.2 Dynamic Usage

This scenario shows how a *provider* agent, which is not providing good documents, is expulsed permanently from the *InformationSystem* (Figure 2). P1, E1, C1 have been previously registered as a *provider*, an *evaluator* and a *consultant* inside the *InformationSystem*, respectively. Then C1 requests P1 to give information related to hotels (Figure 2 message 1). P1 returns a *Document* as a result of the *ProvideInfo* service (message 2). Then C1 requests agent E1 to provide the *EvaluateInfo* service so as to known quality of this information (message 3). After evaluating the provided document, E1 answers by informing C1 about the low quality of this information (message 4). Then, E1 decides to expel P1 from the *InformationSystem*, since its provided documents are of low quality, by requesting the OMS to execute the expulsion service (messages 5 and 6). Then the OMS informs P1 about its expulsion as an information *provider* (message 7). Next E1 creates a new norm in order to prevent P1 acting as a *provider* in the future (messages 7 and 8). Therefore, when P1 tries to become an information *provider* inside the *InformationSystem* (message 9) the OMS will not allow P1 to acquire the *provider* role (message 10), since there is a norm which forbids it explicitly.

Due to lack of space, a detailed application example illustrating all the new features of the proposal has not been included here. However, this example illustrates how the OMS services are employed in order to change the normative context dynamically. The normative context defines the way in which agents behave inside THOMAS. For further details a simple demo example can be found together with an available prototype of the THOMAS architecture[6]. Following, conclusions and a discussion on related works are presented.

[6] `http://www.dsic.upv.es/users/ia/sma/tools/Thomas/index.html`

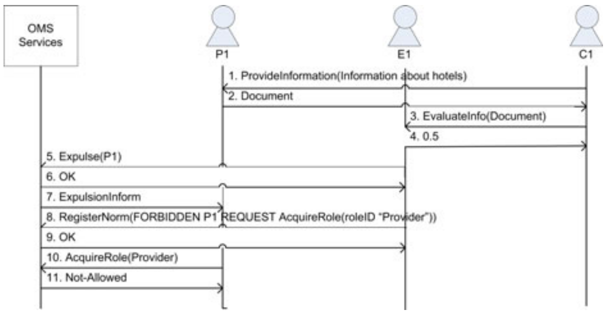
OMS Services
P1
E1
C1
1. ProvideInformation(Information about hotels)
2. Document
3. EvaluateInfo(Document)
4. 0.5
5. Expulse(P1)
6. OK
7. ExpulsionInform
8. RegisterNorm(FORBIDDEN P1 REQUEST AcquireRole(roleID "Provider"))
9. OK
10. AcquireRole(Provider)
11. Not-Allowed

Fig. 2. Case Study Dynamics

6 Discussion

Organizations represent an effective mechanism for activity coordination, not only for humans but also for agents. They have recently been used in agent theory to model coordination in open systems and ensure social orders [18].

Virtual Organizations include the integration of organizational and individual perspectives, and the dynamic adaptation of models to organizational and environmental changes [1]. The necessity of managing the organizational life-cycle has been taken into account in several approaches such as the S-Moise+ middleware [19] (based on Moise+ [20] organizational model), the Brain system [21], the ORA4MAS proposal [22] and the Rade system [23]. In our architecture the OMS entity acts in a similar way; it offers dynamical services for allowing agents to take roles at runtime, accordingly to the established organizational norms. However, the main contributions of our proposal are: (i) all this functionality has been designed following a Service Oriented approach. The OMS services have been implemented as Web Services independently of the agent platform. Thus, the OMS gives support for the coordination among agents which belong to different platforms. In addition, the OMS services are registered and published by the architecture in order to allow agents to discover them and to know how to make use of them. (ii) The OMS also offers structural services for allowing the organizational structure (i.e the roles, units and norms) to be modified at runtime. As argued previously, the characteristics of open environments make mandatory supporting the organizational structure evolution and facilitating its growth and update through execution time.

Open systems need the existence of normative mechanisms that coordinate agent behaviours. Regarding works on norms, they traditionally have a theoretical point of view. Recently, norms have received a more practical conception in order to allow the regulation of open distributed systems. More specifically, traditional approaches based on deontic logic have evolved to a more operational conception of norms that allows their employment inside the design and execution of real MAS applications as a coordination mechanism. These approaches must allow reasoning about the global system performance as well as the agent individual reasoning. The *normative learning* term has been defined as the process by which agents take norms into consideration inside their decision making process [24]. Works concerning the normative reasoning are related to the norm emergence [25,26] and the norm acceptance [27,28]. Taking these later works as a starting point, the OMS component described in this paper is aware of the existence of norms that regulate access to its services. On the other hand, proposals for computational operationalization of norms are based on the e-institution metaphor [9,10]. They assume the existence of an "special" entity, which is the institution itself, that acts as a middle-ware for agent communications. Therefore, the institution has an unlimited knowledge about occurrence of facts as well as extra capabilities for controlling norms.

In order to allow the definition of MAS as Open Systems a new normative framework that gives support to agent normative reasoning is needed. With this aim, we have proposed a general normative language for expressing norms and a normative implementation that allows agents to take into consideration the existence of norms and their limitations as norm controllers. This proposal of normative language is a more general and expressive formalism mainly focused on controlling service registering and usage, making it possible a better integration of both MAS and Web Service Technologies. The main and new aspects of the proposed language are: (i) it allows the definition of norms that cover organizational dynamics; and (ii) norms define agent functionality in terms of services requested and provided by each role. Both the organizational performance and functionality of the system are established by means of norms.

7 Conclusions

This work belongs to a higher project whose goal is to develop models, frameworks, methods and algorithms for constructing large-scale open distributed computer systems. Our aim is to employ this architecture for building demonstrators on e-procurement, e-healthcare and water conflict resolution. Thus, the Organizational Management System implementation has been included in a prototype of the THOMAS architecture. This component is mainly responsible for the management of organizations and their entities. The OMS allows the creation and management of both norms and organizations. Therefore, our normative implementation has been employed for controlling access to the organizational services. In this sense, norms can be conceived as a method for regulating the dynamical organizational adaptation to the environmental changes.

References

1. Dignum, V., Dignum, F.: A landscape of agent systems for the real world. Tech. report 44-cs-2006-061, Inst. Information and Computing Sciences, Utrecht University (2006)
2. Boissier, O., Gâteau, B.: Normative multi-agent organizations: Modeling, support and control. In: Normative Multi-agent Systems, Dagstuhl Seminar (2007)

3. Luck, M., McBurney, P., Shehory, O., Willmott, S.: Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing). AgentLink (2005)
4. Luck, M., McBurney, P.: Computing as interaction: Agent and agreement technologies. In: IEEE SMC Conference on Distributed Human-Machine Systems, pp. 1–6 (2008)
5. Greenwood, D., Calisti, M.: Engineering web service - agent integration. In: IEEE Int. Conf. on Systems, Man and Cybernetics, vol. 2, pp. 1918–1925 (2004)
6. Carrascosa, C., Giret, A., Julian, V., Rebollo, M., Argente, E., Botti, V.: Service Oriented MAS: An open architecture. In: AAMAS, vol. 2, pp. 1291–1292 (2009)
7. Argente, E., Palanca, J., Aranda, G., Julian, V., Botti, V., García, A., Espinosa, A.: Supporting agent organizations. In: Burkhard, H.-D., Lindemann, G., Verbrugge, R., Varga, L.Z. (eds.) CEEMAS 2007. LNCS (LNAI), vol. 4696, pp. 236–245. Springer, Heidelberg (2007)
8. Boella, G., van der Torre, L., Verhagen, H.: Introduction to the special issue on normative multiagent systems. Journal of. Auton. Agents Multi-Agent Syst. 17, 1–10 (2008)
9. García-Camino, A., Noriega, P., Rodríguez-Aguilar, J.A.: Implementing norms in electronic institutions. In: EUMAS, pp. 482–483 (2005)
10. Torres, V.: Implementing norms that govern non-dialogical actions. In: Sichman, J.S., Padget, J., Ossowski, S., Noriega, P. (eds.) COIN 2007. LNCS (LNAI), vol. 4870, pp. 232–244. Springer, Heidelberg (2008)
11. Argente, E., Criado, N., Julian, V., Botti, V.: Designing Norms in Virtual Organizations. In: CCIA, vol. 184, pp. 16–23. IOS Press, Amsterdam (2008)
12. Criado, N., Julian, V., Argente, E.: Towards the Implementation of a Normative Reasoning Process. In: 7th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS 2009), pp. 319–328. Springer, Heidelberg (2009)
13. López, F., Luck, M.: Modelling norms for autonomous agents. In: ENC, pp. 238–245. IEEE Computer Society, Los Alamitos (2003)
14. Brafman, R.I., Tennenholtz, M.: On Partially Controlled Multi-Agent Systems. Journal of Artificial Intelligence Research 4, 477–507 (1996)
15. Kollingbaum, M.J., Vasconcelos, W.W., García-Camino, A., Norman, T.J.: Conflict resolution in norm-regulated environments via unification and constraints. In: Baldoni, M., Son, T.C., van Riemsdijk, M.B., Winikoff, M. (eds.) DALT 2007. LNCS (LNAI), vol. 4897, pp. 158–174. Springer, Heidelberg (2008)
16. Vasconcelos, W., Kollingbaum, M.J., Norman, T.J.: Resolving conflict and inconsistency in norm-regulated virtual organizations. In: AAMAS, pp. 632–639 (2007)
17. Boella, G., van der Torre, L.: Permissions and obligations in hierarchical normative systems. In: ICAIL (2003)
18. Dignum, V., Meyer, J., Weigand, H., Dignum, F.: An organization-oriented model for agent societies. In: RASTA: 31–50 (2002)
19. Hubner, J., Sichman, J., Boissier, O.: S-moise+: A middleware for developing organised multi-agent systems. In: EUMAS, pp. 64–78 (2006)
20. Hubner, J., Sichman, J., Boissier, O.: MOISE+: towards a structural, functional, and deontic model for MAS organization. In: AAMAS, pp. 501–502 (2002)
21. Cabri, G., Leonardi, L., Zambonelli, F.: BRAIN: A Framework for Flexible Role-Based Interactions in MAS. In: CoopIS/DOA/ODBASE, pp. 145–161 (2003)
22. Kitio, R., Boissier, O., Hbner, J.F., Ricci, A.: Organisational artifacts and agents for open multi-agent organisations: "Giving the power back to the agents". In: Sichman, J.S., Padget, J., Ossowski, S., Noriega, P. (eds.) COIN 2007. LNCS (LNAI), vol. 4870, pp. 171–186. Springer, Heidelberg (2008)

23. Xu, H., Zhang, X.: A Methodology for Role-Based Modeling of Open Multi-Agent Software Systems. In: ICEIS, pp. 246–253 (2005)
24. Verhagen, H.J.E.: Norm Autonomous Agents. PhD thesis, The Royal Institute of Technology and Stockholm University (2000)
25. Walker, A., Wooldridge, M.: Understanding the emergence of conventions in multi-agent systems. In: ICMAS, pp. 384–390 (June 1995)
26. Savarimuthu, B.T.R., Cranefield, S., Purvis, M., Purvis, M.: Role model based mechanism for norm emergence in artificial agent societies. In: Sichman, J.S., Padget, J., Ossowski, S., Noriega, P. (eds.) COIN 2007. LNCS (LNAI), vol. 4870, pp. 203–217. Springer, Heidelberg (2008)
27. Castelfranchi, C., Dignum, F., Jonker, C.M., Treur, J.: Deliberative normative agents: Principles and architecture. In: Jennings, N.R. (ed.) ATAL 1999. LNCS, vol. 1757, pp. 364–378. Springer, Heidelberg (2000)
28. Dignum, F., Morley, D., Sonenberg, L., Cavedon, L.: Towards socially sophisticated BDI agents. In: ICMAS, pp. 111–118. IEEE Computer Society, Los Alamitos (2000)