An Agent-Based Model for Hierarchical Organizations

Luis Erasmo Montealegre Vázquez and Fabiola López y López

Facultad de Ciencias de la Computación
Benemérita Universidad Autónoma de Puebla
montealegreluis@gmail.com, fabiola.lopez@siu.buap.mx

Abstract. Hierarchical structures have been widely used by human organizations because they provide the natural means to delegate tasks, to reduce communication lines and to control the activities performed within organizations. This has motivated the development of different approaches to automatize many of the activities that take place in hierarchical organizations. Recent frameworks, such as Gaia, Aalaadin, HarmonIA and OperA, among others, have considered the agent paradigm to do so. Since organizations are dynamic entities that evolve with the time, agents in organizations must adapt to changes. Here we develop a model for flexible and open hierarchical organizations where agents can dynamically adapt to organizational changes.

1 Introduction

The deployment of Internet and the middleware available to build distributed applications provide the tools needed to automatize many of the tasks that are carried out in organizations; consequently, we need a paradigm to allow the representation of both the elements that comprise an organization and the processes that occur within it. Since humans are a key element in any organization and computational agents are conceived as software entities acting on behalf of a user, multi-agent systems (MAS) have been considered a suitable paradigm to represent any kind of social group [1].

A big effort has been made to introduce organizational concepts into the analysis and design of multi-agent systems. Examples of this can be found in models such as Aalaadin [2], Moise [3], and MaSE [4]. All of them are based on concepts such as roles, groups and structures. In addition, there are methodologies based in organizational concepts like Gaia [5]. An extension to Gaia, the GaiaExOA methodology [6], includes organizational patterns to promote the reusability of design models. Some other models such as HarmonIA [7] and OperA [8] have dealt with open systems and self-interested agent behaviour. HarmonIA is a framework to model electronic organizations from the abstract level where norms are defined to the final protocols and procedures that implement those norms [7]. OperA is a framework for the specification of multi-agent systems that distinguishes between the mechanisms through which the structure and the global behavior of the model is described and coordinated, and the aims and the behavior of the agents that populate the model [8].

Hierarchical structures have been widespread by human organizations because they provide a natural way to delegate tasks, to reduce communication

lines and to control the behavior of each member in an organization. Moreover, in hierarchical organizations, as in any type of organization, services are provided either to other organizations or to individuals, well-defined roles are established for every member and organizational objectives are set. However, organizations are complex and dynamic by nature, they can be redesigned and re-engineered, consequently the way the agents and the coordination structure adapt and change over the time may affect the organizational performance [9]. This paper presents our first results to address this problem.

Here, we have developed a model for *open hierarchical organizations* by taking a framework for normative agents [10] as the base. In our model, each member of the organization is considered as a *normative agent* [11], capable of reasoning the responsibilities and benefits it acquires acting as a member of an organization. We have used *positions profiles* [12] to represent *functional positions* (roles) and *hierarchical structures*. Because normative agents can *modify its behavior* according to the changes in the legislation by rejecting or adopting *new norms*, we have used norms in the definition of position profiles to represent *dynamical functional positions*. Thus, *agents can adapt to organizational changes at runtime* by updating the set of norms that defines its position profile. This allow to implement *flexible and dynamic organizations*. Our work also takes many concepts from different social theories such as the classic and the neoclassic administration theory [13, 14] as well as the human relationships theory [15]. We have used the *administrative process* [14] to *coordinate*, to *monitor* and to *control* the activities of the agents in the organization. The model is mainly intended *to facilitate the implementation of agent-based hierarchical organizations that may help us to automatize human organizations*. Then, in our model one or several agents may act on behalf of a human member of an organization.

The model is represented by using the Unified Modeling Language (*UML*) [16] because it has been widespread and it has become a *standard de facto* within the software development industry. This paper is organized as follows. In Section 2, we present the UML representation of the different elements of normative multi-agent systems. Section 3 describes each element of our model for agent-based hierarchical organizations. In Section 4 we develop the classical example of a conference organization [8, 5] by using our model, in order to show its applicability. Finally, in Section 5 we present our conclusions.

2 Normative Multi-Agent Systems

We have used the normative framework for agent-based systems developed by López, Luck and d'Inverno [10] as the basis of our model. In this section we present a summary of the main components, details can be found elsewhere in [11, 17–19].

2.1 Norms

According to [10], norms are the mechanisms through which societies regulate the behavior of their members. The model of a norm, which representation is shown in Figure 1, includes the following components:

- *Normative goals*. These are the goals prescribed by a norm.
- *Addressee agents*. These agents must comply the normative goals.

– *Beneficiaries agents*. These agents might result benefited from norm compliance.
– *Context*. It represents the conditions to activate a norm.
– *Exceptions*. These are the situations where the agents are not forced to fulfill a norm.
– *Punishments*. They model the penalties applied to the agents that do not satisfy the normative goals.
– *Rewards*. Rewards are given to the agents that comply the norms.

Norms may be created by the agent designer as built-in norms, they can be the result of agreements between agents, or can be elaborated by a complex legal system [11]. Here, we assume that norms have already been created.

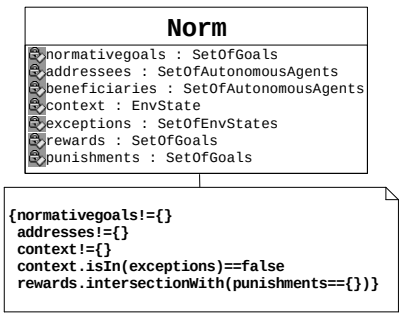

Fig. 1. Class `Norm`

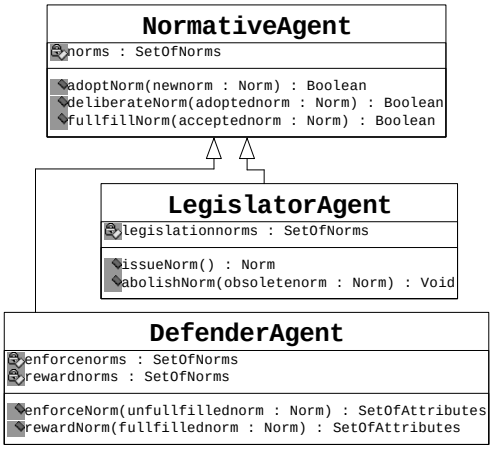

Fig. 2. Types of agents in a NMAS

2.2 Normative agents

A *normative agent* is an autonomous agent able to adopt, to deliberate and to comply with norms on the basis of its own goals and preferences. A better description of this kind of agents can be found in [19]. Due to space constraints we do not present the UML specification for attributes, goals, environmental states, and autonomous agents, but it can be found elsewhere in [20]. Figure 2 describes a normative agent which is supposed to inherit from an autonomous agent class. In a Normative Multi-Agent System (NMAS) there are agents entitled to *legislate* and therefore to create new norms (see class `LegislatorAgent`), and there are agents entitled to give *rewards* or to apply *punishments* to other agents according to the compliance of norms (observe class `DefenderAgent`). The UML representation of these agents is also shown in Figure 2.

A NMAS [11] must include the following elements (illustrated graphically in Figure 3).
– A set of agent *members* able to reason about the norms.
– A set of *legislator* agents.
– A set of norm *defender* agents.
– A set of *norms* directed to regulate the behavior of the agents.

– A set of norms whose purpose is to *enforce* and to determine the fulfillment of the most recent set of norms.
– A set of norms directed to promote the fulfillment of norms through *rewards*.
– A set of emitted norms to allow the *creation* and the *abolition* of norms.

When systems regulated by norms are populated by autonomous agents, neither can all norms be considered in advance (since *new conflicts among agents may surge, and consequently new norms may be needed*), nor can compliance with norms be guaranteed (*since agents can decide not to comply norms*). Thus, these systems must include mechanisms to deal with the modification and creation of norms as well as with the unpredictable normative behavior of agents. That is the case of the last three sets of norms described above, which are intended to be used by legislators and defenders to address such problems. Further details are given in [10].

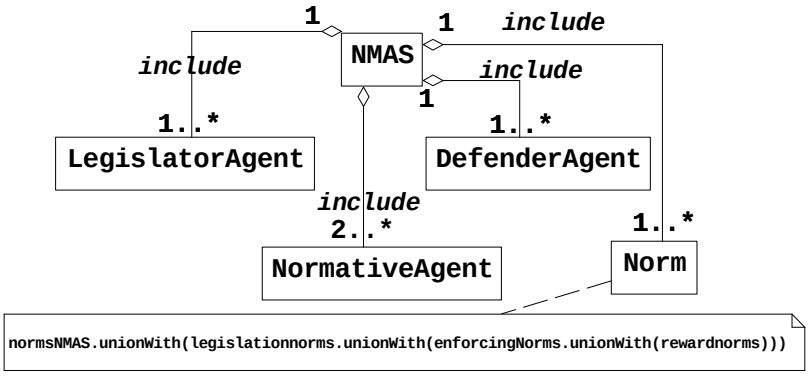

Fig. 3. Class `NMAS`

3 An Agent-Based Hierarchical Organization

In this section we use the model of NMAS and some concepts taken from administrative theories to develop a model for hierarchical organizations. A *human organization* is a goal-oriented social entity consisting of a group of persons and designed to obtain results, to generate utilities and to provide social satisfaction. It is structured deliberately because its structure suggests the division of labor in such a way that it can be used to assign the execution of tasks among its members [13]. The neoclassic administrative theory [14] adds some elements to the formal concept of organization and defines it as *a set of functional and hierarchical positions oriented to the production of goods and services.*

Translating these definitions to an *agent-based hierarchical organization* we can define it as a NMAS where agent's activities are coordinated with the purpose of reaching organizational goals and, in this way, to offer some services. In addition, an agent-based hierarchical organization has a set of functional positions which describes a hierarchical structure. This structure is used to coordinate the activities of the agents within the organization. Before providing a model for hierarchical organizations we describe some organizational concepts that will be used later on.

3.1 Resources

In order to provide services and achieve goals *a human organization* needs to use several resources. Similarly, an *agent-based hierarchical organization* requires a set of resources to operate according to its objectives. The representation of a resource includes a name, a type, a location, and its availability as shown in Figure 4.

3.2 Organizational goals

Every *human organization* exists not only to reach objectives and to produce results (lending some service) but also to obtain profits [14]. In a similar way, an *agent-based hierarchical organization* is designed to reach goals, to offer services and to obtain profits. Viewing the organization as a unit, the goals of an agent-based hierarchical organization are defined as a set of desired states. Each organizational goal has associated some elements, such as a plan to reach that goal, and a leader whose mission is to *coordinate* and consequently to *ensure* the achievement of the organizational goal. This leader must be a legislator, because it must make changes in legislation in the case of *recurrent and unexpected conflicts among agents* or due to *effectiveness reasons*. Its representation is shown in Figure 5. The plans related to goals will be further described in Section 3.7. Position profiles will be described in Section 3.5.

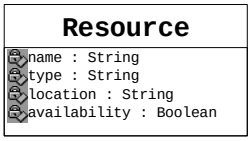

Fig. 4. Class `Resource`

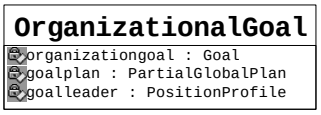

Fig. 5. Class `OrganizationalGoal`

3.3 Organizational services

Human organizations are intended to provide services to both organizations and individuals [14]. In an *agent-based organization*, a service is defined as the set of capacities or resources of a software entity, which can be accessed through the network by external entities such as individual agents, people or other agent-based organizations. Our representation of a service is like the one shown in Figure 6. It includes an identifier, a set of goals, a description (which helps the client to decide wether to contract the service or not), a plan, and a leader which must be a defender agent, because it is the one responsible of guaranteing the service through norm compliance. Due to space constraints the process to verify the fulfilment or the violation of a norm in a NMAS is not explained here, but it can be found elsewhere in [11].

3.4 Contracts

A contract represents *benefits* (rights) and *obligations* for the participants in it. *An obligation is a norm* which unfulfillment is always penalized. To specify a *benefit* we need a *pair of norms*, one specifying what must be done and another specifying a reward to the addressee of the first norm, i. e the *compliance* of the first norm *benefits* its addressee agent. Therefore, we can represent a contract

with sets of norms. In this way norms specify the things that must be done to consider a service as fulfilled. A contract consists of an identifier for the service, the identifiers of the participants, and the set of obligations directed to the service providers (delivered product, deadlines, etc.). The *normative goals of these obligations must match the service goals*, otherwise the service cannot be guaranteed. A contract also includes the set of obligations associated to clients (payment, for instance). The remaining contract details are not discussed in this paper. However, details about the specification of contracts can be found elsewhere [8, 21, 22].

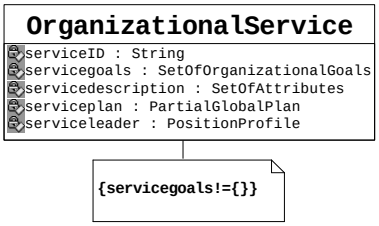

Fig. 6. Class `OrganizationalService`

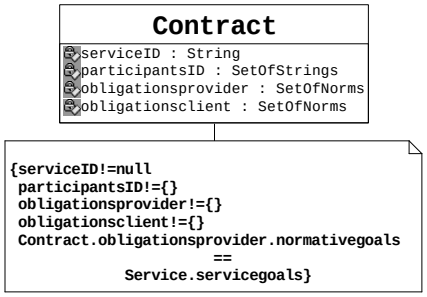

Fig. 7. Class `Contract`

3.5 Position profiles

In order to *represent the hierarchy* of an organization we have used the *functional positions analysis*, as described in [12]. A functional position analysis consists of a *header* which includes the organization's general information (name, address, etc.); an *identification for the position*, this includes all the information required to identify the positions (a key for the position, a hierarchical level, the amount of employees performing that position, the immediate inferior and superior positions, etc.); a *generic description*, here all the position's activities are defined in terms of goals; a *specific description*, this describes in detail all the activities of the profile (i.e activities are ordered chronologically and according to their importance); a *specification for the position*, it is derived from the position's description and emphasizes the minimum requirements to perform the position (abilities, responsibilities, effort and working conditions); and a *position profile*, which contains all the information collected during the analysis. A position profile is intended firstly, to identify a functional position in a hierarchy, secondly, to describe the set of obligations and rights related to a position, and thirdly, to determine the authority and communication paths, i. e. to specify the superior and the set of inferior elements regarding to a given position.

A role is an abstract description of an entity's expected function [23]. Roles have been represented in [3] as a set of forbidden/authorized goals to achieve, plans to follow, actions to execute and resources to use. In [4] a role defines the tasks that must be accomplished in order to achieve the role's targets. In HarmonIA [7] a role is a set of rules which define constraints to ensure that desired states are kept or achieved, and acceptable behaviors are performed, i. e. rules define the actions accepted. In OperA [8] roles are described in terms

of objectives and norms which specify the rights associated with the role, and the type of enactment of the role (institutional or external). Here we use the term *position profile* for describing a role, its main difference and advantage over other models is that our role model include norms in its definition. Thereby, agents can modify a position profile (role) by adopting new norms or updating its current set of norms at runtime.

As mentioned before, in an *agent-based hierarchical organization*, the position profiles are defined by using norms. It can be done due to the possibility of designing norms to specify authority (by means of benefits) and responsibility (by means of obligations). The class `PositionProfile` shown in Figure 8 depicts the elements needed to define a *profile for a functional position* [12]. A profile must include an authority level, an agent number, a profile key and a profile identifier, i. e the elements needed to identify the profile. The usage of these attributes will be detailed in Section 4. It also specifies the sets of norms representing the obligations and the responsibilities included in a profile, as well as the set of position profiles representing its subordinates and the position profile representing its superior. It is necessary to emphasize that *every organizational goal must be in the set of normative goals of the obligations of at least one position profile* (the leader's profile). There is a relation between profiles and services similar to the one between organizational goals and profiles. *Service goals must be in the set of normative goals of the leader's obligations and/or its subordinates.*

3.6 Organizational agents

This section aims to determine the characteristics and the capabilities of an organizational agent. An *organizational agent* must exhibit abilities to do certain activities or to obtain goals (autonomous agents). It must recognize and fulfill the norms of the organization (normative agents). It must adopt one or several functional positions, which are defined by position profiles; it must recognize the authority lines of an organization and it must have access to some resources (organizational agents). Therefore, we have modelled the class `OrganizationalAgent` as a specialization of the class `NormativeAgent`. In the case of organizational agents which are also authorities, they must be represented by inheriting from either `LegislatorAgent` or `DefenderAgent` classes. The `OrganizationalAgent` class also defines the position profiles that an agent is currently performing as well as the resources to which an agent has access to. Figure 9 shows how an organizational agent is represented.

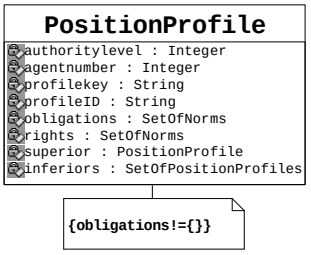

Fig. 8. Class `PositionProfile`

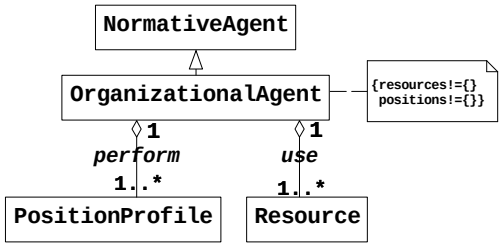

Fig. 9. Class `OrganizationalAgent`

3.7 Administrative process

A *human organization* can offer more than one service. The notion of service needs the knowledge of the administrative process functions [14]. Each service defines an administrative process. The administrative process consists of four interrelated activities as Figure 10 shows.

1. *Planning*. In this phase the objectives of a service or organizational goal are determined.
2. *Organization*. The output of this phase is a plan, which is the result of the decomposition of an organizational goal into subgoals. These subgoals are assigned to specific position profiles. A plan also defines the conditions or states under which it can be applied as well as the resources required to achieve a goal or to provide a service. A plan is depicted as Figure 11 shows. An important restriction related to plans must be highlighted. *All the subgoals of a plan must be in the obligations' set of normative goals of the participants profiles*. This is our model for plans but this is only an option, implementation may use a strip-like planner [24] for instance.

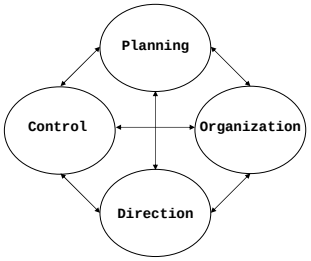

Fig. 10. The administrative process

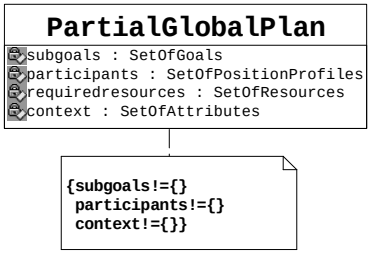

Fig. 11. Class `PartialGlobalPlan`

3. *Direction*. This activity implies the processes through which an administrator tries to influence its subordinates, to make them behave according to the expectations and to reach the organizational goals. In our approach, it is done by establishing performance standards. A performance standard is modelled as an organizational goal.
4. *Control*. The activity of control supervises the organizational activities by comparing the current performance (the goals achieved in a given moment) with the established standard, and executes remedial actions, if needed.

In an *agent-based hierarchical organization* we can achieve coordination by using the administrative process functions and the position profiles.

3.8 Administrative agents

As mentioned before, we have used the administrative process as a coordination mechanism. Then, we must recognize and describe the types of agents capable of performing the functions of the administrative process (see Figure 12).

- *Administrator*. It is an agent with certain authority over other agents. Its main capabilities are focused on planing and direction activities.
- *Supervisor*. This agent verifies the fulfilment of other agents' obligations, this activity is intended to achieve the organizational goals. This type of agent is capable of performing activities of supervision and control.

An *administrator* must be able to establish the objectives of a goal or service as stated by the first administrative activity: planning. This is done by the methods `setOrganizationalGoals` and `setServiceGoals`. An administrator can also generate *plans* (i. e. it can perform the second administrative activity: organization) by using the methods `createGoalPlan` to satisfy a goal and `createServicePlan` to provide a service. This type of agent is a specialization of the class `LegislatorAgent` because it is responsible of applying changes in legislation in order to coordinate an organizational goal or service as mentioned in Section 3.2. The activity of direction is carried out by a *supervisor* by means of the methods `directGoal` and `directService`. These methods take the plan of a service or goal to establish the expected performance standards. A supervisor can reward or enforce the fulfillment or violation of norms since it is the one in charge of monitoring the agents' behavior in order to assure the achievement of the organizational objectives, thus a supervisor must inherit from class `DefenderAgent`. The activity of control is implemented by the methods `controlGoal` and `controlService`. Both of them verify if performance standards have been reached and execute a remedial action if needed.

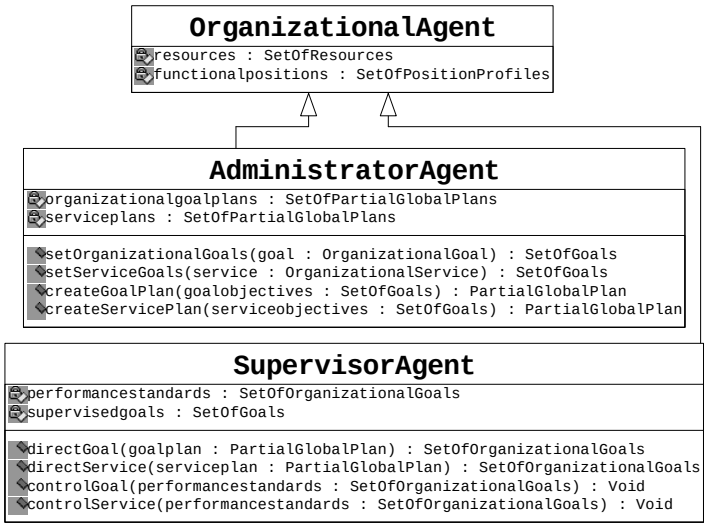

Fig. 12. Administrator and supervisor agents

Previous organizational models [4, 3, 2],do not define a supervisor agent. This agent is necessary because the agents in an open organization can act in a *self-interested* way, and therefore, as the workers of a human organization, they will try to satisfy their *individual goals* [15]. The individual goals and the motivations (preferences) of agents allow them to choose which norms to fulfill. Thus, we need a means to control the unexpected and undesirable behavior of agents, with the purpose of preserving the good performance of the organization. This control is exerted by the supervisor agent. The idea of this kind of agent appears in the HarmonIA framework [7]. There, an institutional role is defined, the *police agent*, which is an agent that checks if the behavior of the other agents follows

the norms. The agent playing this role knows all the roles, and consequently the complete set of rules that define them. Once again, our model of supervisor is better in the sense that if a change in legislation may occur it would not affect the effectiveness of supervision, because the supervisor can update its set of monitored norms at runtime.

3.9 Global view

The relations between the elements of our model can be summarized as follows. An *agent-based hierarchical organization* consists of a set of *organizational agents* able to perform the administrative process activities (the agents *supervisor* and *administrator*). An organizational agent is the specialization of a *normative agent*. A normative agent can be a *defender* or a *legislator*. A normative agent is the specialization of an *autonomous agent*, which is defined as a BDI agent capable of reaching *goals* and perceiving an *environment*. An agent-based organization aims to offer *services* and to achieve *organizational goals*. These services and goals need to use various *resources* and require the elaboration of a *plan*. A service associates a *contract* for each client of an organization. An agent-based organization defines *position profiles* which are performed by organizational agents. The organization establishes *norms* to regulate the behavior of agents. This description is depicted in the UML class diagram shown in Figure 13.

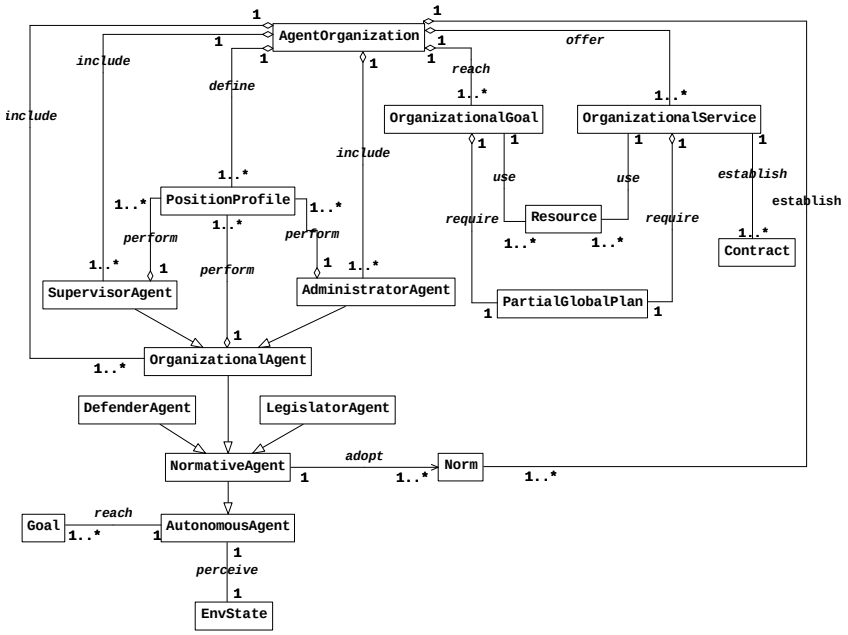

Fig. 13. Global view

4 The Applicability of the Model

In order to show the applicability of our model, we develop the example of a conference organization, previously done in [8] and [5]. Due to the lack of space,

we will only compare our model with OperA [8]. The UML use case diagram for our organization is shown in Figure 14. The functional positions in this example are: the organizer, the PC chairs, the PC members, the local chairs, and the session chairs. The organizational service provided by this society is to organize a conference. To organize a conference, it is necessary to review the papers submitted. This service is described in the instance of the Figure 15. It includes the identifier for the service, it also describes the goals of the service. This goals have associated an integer value which defines the *importance* of each service goal and their *interdependency*. It means that in order to review a paper it must be assigned first, then it must be read and a report must be written, finally, this report must be sent to the leader of the the service. Additional information related to the participation of the client in the conference is also given (the date and the place where the conference will take place). It also establish a plan associated and the profile that acts as the service leader. Once the author has decided to contract the service, an instance like the one shown in Figure 16 is created. It includes the identifier for the service and an identifier for both the client and the provider, it also describes the obligations that the author, as a client, must accept. In this case its obligation is to accept the reviewing results (negative or positive), and to attend to the conference in the case the paper is accepted. On the other hand, the organization, as a service provider, is obliged to assign the paper to a reviewer, who in turn, must review the paper.

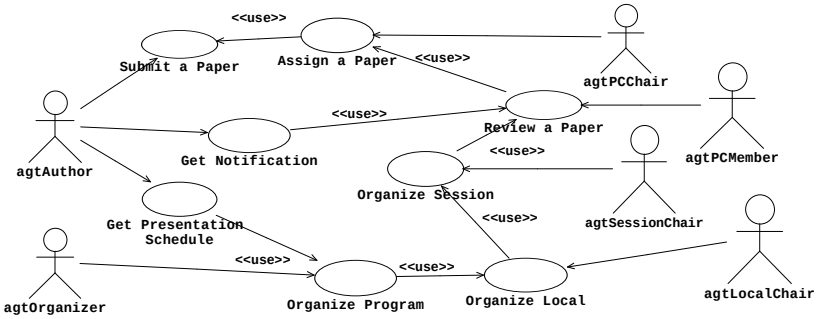

Fig. 14. Use case diagram

```
srvReviewPaper : OrganizationalService
serviceID = "RVW"
servicegoals = {
 {paperAssigned(paper,reviewer,deadline),1},
 {read(paper),2},
 {reportWritten(paper,report),3},
 {reviewReceived(serviceleader,paper,report),4},
 {paperReviewed(paper,report),5}}
servicedescription = {hostCity(cityname),
        presentationDate(month,year)}
serviceplan = plnReviewPaper
serviceleader = prfPCChair
```

Fig. 15. Service `srvReviewPaper`

```
ctrAuthorContract : Contract
serviceID = "srvReviewPaper"
participantsID = {authorID,conferencename}
obligationsprovider = {nrmAssignPaper,
                       nrmReviewPaper}
obligationsclient = {nrmAssistConference,
              nrmAcceptNotificationResult}
```

Fig. 16. Contract `ctrAuthorContract`

In [8] a role is defined as a set of objectives, a set of sub-objectives, a set of rights, a set of norms and a type. In our model, that role corresponds to

the class `PositionProfile`. Figure 17 shows the instance for the PC member profile, which states the authority level for the agent that performs the profile, which in turn has an integer value of 2 assigned (the highest level is given to the organizer profile, in this example the value is set to 0). The agent number indicates the quantity of agents performing the profile, the value 1 means that this agent is the only one performing the profile. The profile key is PCM, which identifies the profile in general. The profile identifier gives a unique identifier to the agent performing that profile. Following the example developed in [8], the obligations given to `pfrPCMember`, are defined by the norms `nrmReviewPaper` and `nrmRefuseColleague`. Figure 18 shows the instance diagram for the norm `nrmReviewPaper`. The normative goals for the norm are: to review the paper, to make a written report of the paper, and to send the report to the PC member in charge. Those actions will lead to the agent to the state `paperReviewed(paper,report)`. These goals have a motivation assigned, such values tells the agent which goals have to be reached first, starting from the most important, which has a 1 assigned, to the least, which is valued to 4. In other words, *the goals' importance of the profile defines individual goal dependency*. Observe that these goals are a subset of the goals of the service. This implies a dependency between the profiles `prfPCMember` and `prfPCChair`. Observe that the context of the norm `nrmReviewPaper` is the goal `paperAssigned(paper,reviewer,deadline)` which is the goal of the norm `nrmAssignPaper` which is directed to the profile `prfPCChair`. This means that a paper cannot be reviewed until that paper is assigned to a PC member. Due to space constraints neither the profile nor the norm mentioned above are depicted here.

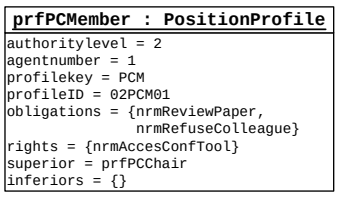

Fig. 17. Position profile `prfPCMember`

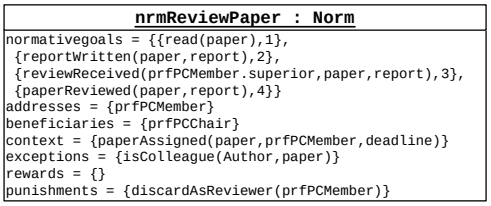

Fig. 18. Norm `nrmReviewPaper`

To model the processes done in organizations, OperA uses scene scripts. A scene script serves as a blueprint for the actual interactions between actors [8]. A *scene script is equivalent* to the activities of *planning and direction* of the administrative process. In what follows we describe the administrative process done when a paper is reviewed, by using UML sequence diagrams. The use case begins when an author uploads a paper to the ConfMaster Tool, this action changes the environment in the organization by adding the predicate `newPaper(author,paper)`. This induces the deliberation process of the agent enacting the position profile `prfOrganizer` who is the leader of the organizational service (`agtOrganizer`). The current environmental state matches the context of the norm `nrmCoordinateReviewProcess` which is defined in the profile `prfOrganizer`. Service's goals are the goals of the norm activated. The

agent `agtOrganizer` creates a new organizational goal `oglReviewPaper`, which aim is to review the paper recently sent. To do so, the organizer uses method `setOrganizationalGoal` which returns the set of subgoals for the organizational goal (see Figure 12), these goals are the same defined in the goals of the service (assign and read the paper, write and deliver the report as Figure 15 shows). This method corresponds to the first activity in the administrative process. Once the subgoals are established, the organizer uses method `createGoalPlan`, which creates the partial global plan to achieve the organizational goal. Thus, the agent `agtOrganizer` get its inferiors' position profiles, to choose the profile that best fits the subgoals. This decision is taken based on the *obligations' normative goals* of the profile. Then, the agent creates a plan establishing the set of subgoals, the set of position profiles, the set of resources required, and the context for the plan, in order to add that plan to the organization's set of plans. Once a resource is assigned to a profile, it is no longer available to other until the goal plan has been achieved or the plan had failed. These actions correspond to the second administrative process activity (See Figure 19).

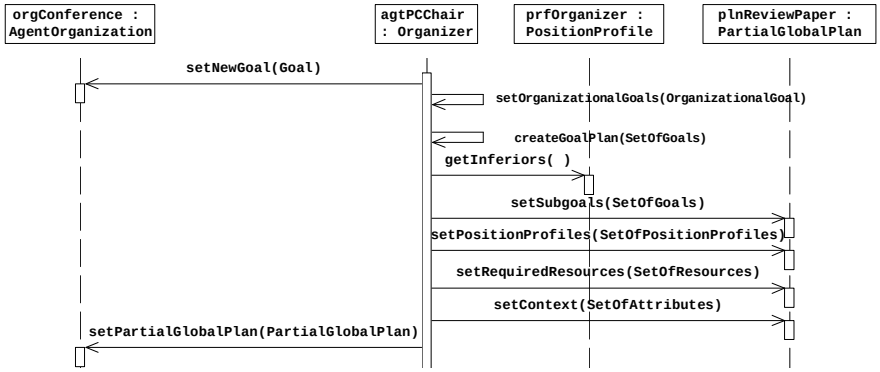

Fig. 19. Planning and organizing a global goal

The following example is intended to show, the actions taken when agents do not comply with their obligations. In OperA [8] this can be done by monitoring the contracts of the role enactors, but not specific activities or mechanism are defined. Verification of norm compliance is an optional clause in the contract that specifies by whom and how the norm will be verified and which are the actions to be executed if norms are ignored. In HarmoniA norm compliance is verified by the police agents as mentioned in Section 3.8. In the approach presented in [25] the verification of norm compliance is done by checking de *safety* and *liveness* properties of protocols. The former states that if there is no steps in a protocol that violate any norm, the protocol will not violate any of the norms as a whole [25], the latter checks if the protocol achieves a specific goal at its end.

Here, the actions to verify norm compliance are implemented using the activities of control and direction. If the agent PC member decides to unfulfill the norm `nrmRefuseColleague`, the following process is initiated. The PC chair

detects that a PC member did not comply with norm `nrmRefuseColleague`, and uses its method `controlGoal`, which takes the set of organizational goals that cannot be reached, and decides whether to apply a corrective action or not. A corrective action can be either, the issue or the enforcement of a norm, the modification of either a global plan or a global goal, or the creation of a contingency plan among other actions. This undesirable behavior, activates method `enforceNorm`, by applying the punishment associated. In our example, it produces the activation of method `directGoal`, which creates a contingency plan, by adding a new global goal directed to a new revisor, i. e. other agent performing the PC member position profile, this in turn gets both the new plan and the new goal directed to it. Figure 20 illustrates this activities.

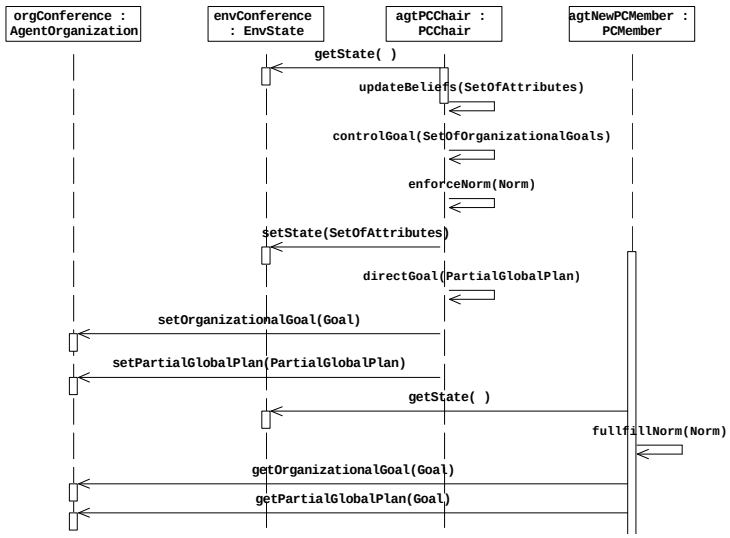

Fig. 20. Directing and controlling a global goal

Here we have proposed a norm-based model to represent hierarchical organizations which main difference with other models, and specifically with OperA is its flexibility. Autonomous normative agents are designed in such a way that changes in norms and the issuance of new ones does not produce any harm to the effectiveness of the whole system, since agents can adopt new norms and update its current set of norms. Therefore, the implementation of dynamical position profiles (or roles) is possible, this can also lead us to represent flexible organizations. This flexibility is extended to the mechanisms of supervision and verification of norm compliance, since changes in norms are updated by supervisor and defender agents at runtime.

5 Conclusions

We have presented a model for hierarchical organizations based on a normative framework for agent-based systems [10]. The model comprises elements from the administrative area [13, 14] like, the administrative process, which defines the set of internal actions performed when an organization needs to provide a

service or to reach some organizational goals. The model defines suitable agents to support and to perform the administrative process functions. It also provides a representation for a position profile, which is used to make a functional position analysis in human organizations [12]. Just as in human organizations [15] our model also represents the norms that control the behavior of their members [11]. The usage of norms in the definition of the position profiles allow to adapt agent's behavior at runtime. The model can be used to represent and to automatize the procedures that are present in human organizations. So, we can make possible the implementation of human-like organizations open and heterogeneous, based on normative multi-agent systems. This implementation can be done by using the middleware available until now and the services can be provided through the Internet. In this paper we have assumed that the organization is already formed, details concerning to the creation of the system such as, how are the norms created, how the agents enter or leave the organization or how are they selected to perform a position profile, etc., as well as more specific details related to the implementation of the model are beyond the scope of this paper, however these issues conform our proposal for future research.

References

1. AgentLink: Agent technology roadmap: Overview and consultation report. Technical report, AgentLink (2004)
2. Ferber, J., Gutknecht, O.: A meta-model for the analysis and design of organizations in multi-agent systems. In: Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS98). (1998) 128–135
3. Hannoun, M., Boissier, O., Sichman, J.S., Sayettat, C.: MOISE: An organizational model for multi-agent systems. In Monard, M.C., Sichman, J.S., eds.: Proceedings of the International Joint Conference IBERAMIA-SBIA, Springer (2000) 156–165
4. DeLoach, S.A., Wood, M.F., Sparkman, C.H.: Multiagent systems engineering. International Journal of Software Engineering and Knowledge Engineering **11** (2001) 231–258
5. Zambonelli, F., Jennings, N.R., Wooldridge, M.: Developing multiagent systems: The gaia methodology. ACM Transaction on Software Engineering Methodology **12** (2003) 317–370
6. Gonzalez-Palacios, J., Luck, M.: A framework for patterns in gaia: A case-study with organisations. In Odell, J., Giorgini, P., Müller, J.P., eds.: AOSE. Volume 3382 of Lecture Notes in Computer Science., Springer (2004) 174–188
7. Vázquez-Salceda, J.: Thesis: The role of norms and electronic institutions in multiagent systems applied to complex domains. the harmonia framework. AI Community **16** (2003) 209–212
8. Dignum, V.: A model for organizational interaction: based on agents, founded in logic. PhD thesis, Utrecht University (2003)
9. Carley, K.M., Gasser, L.: Computational organization theory. In Weiss, G., ed.: Multiagent systems: A modern Approach to Distributed Artificial Intelligence. MIT Press (1999) 299–330
10. López, F., Luck, M., d' Inverno, M.: A normative framework for agent-based systems. In: NorMAS '05 : Proceedings of the Symposium on Normative Multiagent Systems, The Society for the Study of Artificial Intelligence and the Simulation of Behaviour (2005) 24–35

11. López, F., Luck, M.: A model of normative multi-agent systems and dynamic relationships. In Lindemann, G., Moldt, D., Paolucci, M., eds.: RASTA. Volume 2934 of Lecture Notes in Computer Science., Springer-Verlag (2002) 259–280
12. Gama, E.: 4. In: Bases para el Análisis de los Puestos. Manual Moderno (1992) 59–89
13. Chiavenato, I.: Teoría Clásica de la Administración. In: Introducción a la Teoría General de la Administración. McGraw-Hill (2000) 88–112
14. Chiavenato, I.: Teoría Neoclásica de la Administración. In: Introducción a la Teoría General de la Administración. McGraw-Hill (2000) 201–250
15. Chiavenato, I.: Implicaciones de la Teoría de las Relaciones Humanas. In: Introducción a la Teoría General de la Administración. McGraw-Hill (2000) 141–196
16. Rumbaugh, J., Jacobson, I., Booch, G.: Unified Modeling Language Reference Manual. Addison Wesley (1998)
17. López, F., Luck, M., d' Inverno, M.: Constraining autonomy through norms. In: AAMAS '02: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems, New York, NY, USA, ACM Press (2002) 674–681
18. López, F., Luck, M.: Modelling norms for autonomous agents. In Chavez, E., Favela, J., Mejia, M., Oliart, A., eds.: ENC '03: Proceedings of the Fourth Mexican International Conference on Computer Science, Washington, DC, USA, IEEE Computer Society (2003) 238–245
19. López, F., Arenas, A.: An architecture for autonomous normative agents. In: ENC '04: Proceedings of the Fifth Mexican International Conference in Computer Science (ENC'04), Washington, DC, USA, IEEE Computer Society (2004) 96–103
20. Montealegre, L.: Modelado de organizaciones jerárquicas usando sistemas multiagentes normativos. Master's thesis, Benemérita Universidad Autónoma de Puebla (2005)
21. Boella, G., van der Torre, L.: Contracts as legal institutions in organizations of autonomous agents. In: Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems. (2004) 948–955
22. Sandholm, T., Lesser, V.: Leveled-commitment contracting: A backtracking instrument for multiagent systems. AI Magazine **23** (2002) 89–100
23. Kendall, E.: Agent roles and role models: New abstractions for intelligent agent system analysis and design. In: Proceedings of the International Workshop on Intelligent Agents in Information and Process Management, Bremen, Germany (1998)
24. Russell, S., Norving, P.: Artificial Intelligence. A Modern Approach. Prentice Hall, Englewood Cliffs, NJ. (1995)
25. Aldewereld, H., Vázquez-Salceda, J., Dignum, F., Meyer, J.: Verifying norm compliancy of protocols. In Lindemann, G., Ossowski, S., Padget, J., Vázquez-Salceda, J., eds.: Proceedings of AAMAS'05 International Workshop on Agents, Norms and Institutions for Regulated Multi Agent Systems (ANI@REM), Utrecht (2005) 47–60