*S-M*oise$^+$: A Middleware for developing Organised Multi-Agent Systems

Jomi Fred Hübner[1], Jaime Simão Sichman[2], and Olivier Boissier[3]

[1] GIA / DSC / FURB
Braz Wanka, 238
89035-160, Blumenau, Brazil
jomi@inf.furb.br
[2] LTI / EP / USP
Av. Prof. Luciano Gualberto, 158, trav. 3
05508-900 São Paulo, SP, Brazil
jaime.sichman@poli.usp.br
[3] SMA / G2I / ENSM.SE
158 Cours Fauriel
42023 Saint-Etienne Cedex, France
Olivier.Boissier@emse.fr

Abstract. The Multi-agent Systems (MAS) area, while concerning heterogeneous and open systems, has evolved towards the specification of global constraints that agents are supposed to follow. A subset of these constraints are known as organisation of the MAS. This paper describes a software implementation, called $\mathcal{S}$-$\mathcal{M}$OISE$^+$, that tries to fill the gap between the organisational constraints and the agents autonomy. This software ensures that all agents will follow the organisation without requiring that they are developed in a specific language or architecture.

Keywords: Multi-agent Systems, MAS organisations, Engineering organisations for MAS.

1 Introduction

The assignment of an organisation to a Multi-Agent System (MAS) is useful to deal with the problems that could arise from the agents' autonomy, specially in *open* MAS [12] where we do not know what kind of agent will enter into the system (this motivation for organised MAS is well described in [20,4]). In this context, the organisation is a set of behavioural constraints that a group of agents adopts in order to control the agent's autonomy and easily achieve their global purposes [5]. This approach is based on human societies that are successfully using organisation (e.g. social roles) to have a global coherent behaviour. The definition of a proper organisation for a MAS is not an easy task, once the organisation could be too flexible (the organisation does not help the achievement of the global purpose) or too stiff (the organisation extinguishes the agent's autonomy). A initial good organisation is normally set up by the MAS designer,

however it may become not suitable in dynamic environments. In these cases the system must support dynamic changes on its organisation [16].

The precise concept of constraint that will be used to describe an organisation is defined by the underlying organisational model. These models may be divided in two points of view: *agent* centered or *organisation* centered [17]. While the former takes the agents as the engine for the organisation formation, the latter sees the opposite direction: the organisation exists *a priori* (defined by the designer or by the agents themselves) and the agents ought to follow it. In addition to this classification, we propose to group these organisational models in (*i*) those that stress the society's *global plans* and their execution coordination (e.g. TÆMS [18], STEAM [21]); (*ii*) those that have their focus on the society's *roles* and groups (e.g. AGR [8], TOVE [9]); and (*iii*) the models based on a deontic approach where *norm*, among others, is the main concept (e.g. ISLANDER [6], OPERA [4]). Thus we should state that organisation models usually take into account the functional (the first group), the structural (second group), and/or the deontic (the third group) dimension of the organisation. The $\mathcal{M}$OISE$^+$ organisational model is an attempt to join these three dimensions into an unified model suitable for the reorganisation process [14,15]. The $\mathcal{M}$OISE$^+$ main property concerning the reorganisation problem is to be an organizational centered (OC) model where the first two dimensions can be specified almost *independently* of each other and after properly linked by the deontic dimension. This linkage allows the MAS to change the structure without changing the functioning, and vice versa, the system only needs to adjust its deontic relation.

In order to implement a system that follows organisational constraints we can also take either an agent centered or an organisational centered point of view (in [22] these points of view are called agent and institutional perspectives). In the former point of view, the focus is on how to develop an agent reasoning mechanism that follows the organisation. The implementation approach is endogenous to the agent. In the latter, the main concern is how to develop an MAS framework that ensure the satisfaction of the organisational constraints. This point of view is more suitable for heterogeneous and open systems, since, as an exogenous approach, the agent implementation, architecture, and programming language does not matter. Of course the agents probably need to have access to an organisational specification that enable them to eventually reason about it. However, the agents will follow the organisation despite their organisational reasoning abilities. As far as we know, the following implementations of such a kind of framework are availble: AMELI [7] (based on ISLANDER), MadKit [11] (based on AGR), and KARMA [19] (based on STEAM). Hence we are concerned with dynamic organisation, the $\mathcal{M}$OISE$^+$ should be used as the underlying organisational model. In this paper we describe an MAS framework called $\mathcal{S}$-$\mathcal{M}$OISE$^+$ (Sec. 3) which ensure that agents running on it will follow the constraints specified using the $\mathcal{M}$OISE$^+$ model (Sec. 2).

2 The *M* oise$^+$ Organisational Model

The $\mathcal{M}$OISE$^+$ (Model of Organisation for multI-agent SystEms) considers the organisational structure and functioning. However, this model adds an explicit deontic relation among these first two dimensions to better explain how an MAS's organisation collaborates for the global purpose and make the agents able to reason on the fulfilment of their obligations or not [15]. These three dimensions form the Organisational Specification (OS). When a set of agents adopts an OS they form an Organisational Entity (OE) and, once created, its history starts and runs by events like other agents entering and/or leaving the OE, group creation, role adoption, mission commitment, etc.

The $\mathcal{M}$OISE$^+$ Structural Specification (SS) is built in three levels: (*i*) the behaviours that an agent is responsible for when it adopts a role (*individual* level), (*ii*) the acquaintance, communication, and authority links between roles (*social* level), and (*iii*) the aggregation of roles in groups (*collective* level). The $\mathcal{M}$OISE$^+$'s SS also allows us to ascribe the well formed attribute to a group in case the roles of the agents are compatible among them, the minimum and maximum number of role players are satisfied inside a group, etc.

Throughout the text, a soccer team is used as an example to describe the model (a formal definition is found in [14]). A soccer team that we will specify is formed by players with roles like goalkeeper, back player, leader, attacker, coach, etc. These role players are distributed in two groups (defense and attack) which form the main group (the team group). This team structure is specified, using the $\mathcal{M}$OISE$^+$ notation, in the Fig. 1. For instance, in the defense group specification, three roles are allowed and any defense group will be well formed if there is one, and only one, agent playing the role goalkeeper, exactly three agents playing backs, and, optionally, one agent playing the leader role (see the composition relation in Fig. 1). The goalkeeper has authority on the backs. The leader player is also allowed to be a back since these roles are compatible. Due to the role specialisation (see the inheritance relation in Fig. 1), the leader also can play the goalkeeper role. In the same example, a team is well formed if it has one defense sub-group, one attack sub-group, one or two agents playing the coach role, one agent playing the leader role, and the two sub-groups are also well formed. In this structure, the coach has authority on all players by an authority link. The players, in any group, can communicate with each other and are allowed to represent the coach (since they have an acquaintance link). There must be a leader either in the defense or attack group. The leader has authority on all players on all groups, since s/he has an authority link on the player role. For every authority link there is an implicit communication link and for every communication link there is an implicit acquaintance link.

A $\mathcal{M}$OISE$^+$ group can have intra-group and inter-group links. The intra-group links state that an agent playing the link source role in a group *gr* is linked to all agents playing the destination role in the *same* group *gr* or in a *gr* sub-group. The inter-group links state that an agent playing the source role is linked to all agents playing the destination role despite the groups these agents belong to. For example, the coach authority on player is an inter-group link (the coach and

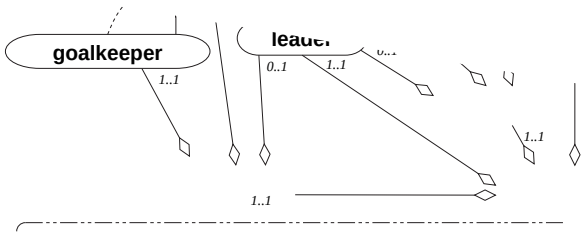

Fig. 1. The structure of the soccer team.

the player agents do not need to belong to the same group), while the goalkeeper authority on backs is an intra-group link (both agents must belong to the same group to "use" this link).

The Functional Specification (FS) describes how an MAS usually achieves its *global* (collective) goals [2] stating how these goals are decomposed (by plans) and distributed to the agents (by missions). The scheme can be seen as a goal decomposition tree where the root is a global goal and the leafs are goals that can be achieved by one agent. Such decompositions may be set either by the MAS designer who specifies its expertise in the scheme or by the agents that store their past (best) solutions. In the soccer example, suppose the team has a rehearsed play as the one specified in the Fig. 2. This scheme has three missions (m_1, m_2, and m_3) — a mission is a set of coherent goal that an agent can commit to. When an agent commits to a mission, it is responsible for all this missions' goals. For example, an agent committed to the mission m_3 has the goals "be placed in the opponent goal area", "shot at the opponent's goal", and, a common goal, "score a goal".

In a scheme, each goal $g_i \in \mathcal{G}$ (where $\mathcal{G}$ is the set of global goals) may be decomposed in sub-goals through plans using three operators:

– sequence ",": the plan "$g_1 = g_2, g_3$" means that the goal g_1 will be achieved if the goal g_2 is achieved and after that also the goal g_2 is achieved;

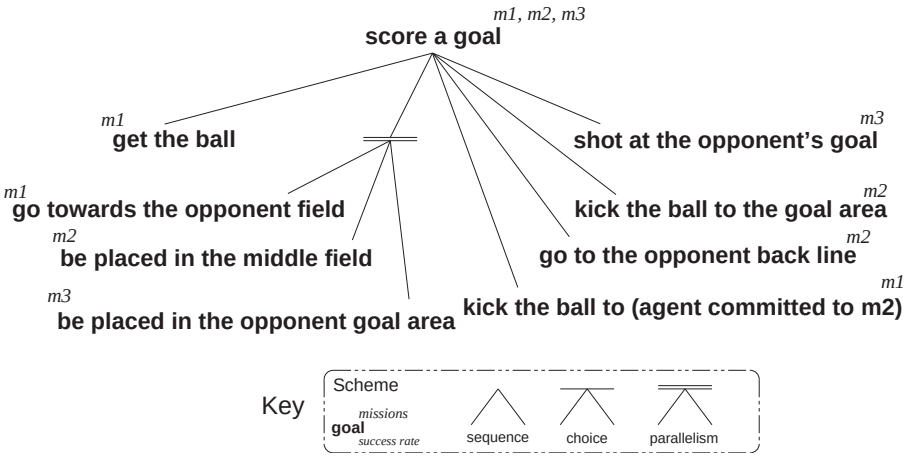

Fig. 2. A "side attack" scheme of the soccer team.

- choice "|": the plan "$g_1 = g_2 \mid g_3$" means that the goal g_1 will be achieved if one, and only one, of the goals g_2 or g_3 is achieved; and
- parallelism "‖": the plan "$g_1 = g_2 \parallel g_3$" means that the goal g_1 will be achieved if both g_2 and g_3 are achieved, but they can be achieved in parallel.

The Deontic Specification (DS) describes the roles' permissions and obligations for missions. A permission $permission(\rho, m)$ states that an agent playing the role ρ is allowed to commit to the mission m. Furthermore, an obligation $obligation(\rho, m)$ states that an agent playing ρ ought to commit to m. For example, in the soccer team DS (Tab. 1), three roles have the right to start the scheme of the Fig. 2 because they have the permission for this scheme's root missions. Once the scheme is created, the other agents (playing back, middle, ...) are obligated by their roles' deontic relations to participate in this scheme. These other agents ought to pursue their mission's goals just in the order allowed by this scheme. For instance, when a middle agent accepts the mission m_2, it will try to achieve its goal "be placed in the middle field" only after the goal "get the ball" is already satisfied by a back agent committed to the mission m_1.

role	deontic relation	mission
back	*permission*	m_1
middle	*obligation*	m_2
attacker	*obligation*	m_3

Table 1. Partial view of the soccer team deontic relations.

3 $\mathcal{S}$-$\mathcal{M}$oise$^+$ Organisational Middleware

$\mathcal{S}$-$\mathcal{M}$oise$^+$ is an open source implementation of an *organisational middleware* that follows the $\mathcal{M}$oise$^+$ model. This middleware is the interface between the agents and the overall system, providing access to the communication layer (see Fig. 3), information about the current state of the organisation (created groups, schemes, roles assignments, etc.), and allowing the agents to change the organisation entity and specification. Of course these changes are constrained to ensure that the agents respect the organisational specification.

$\mathcal{S}$-$\mathcal{M}$oise$^+$ has two main components: an OrgBox API that agents use to access the organisational layer (this component is detailed in Sec. 3.2) and a special agent called OrgManager. This agent has the current state of the OE and maintains it consistent. The OrgManager receives messages from the agents' OrgBox asking for changes in the OE state (e.g. role adoption, group creation, mission commitment). This OrgManager changes the OE only if it does not violate an organisational constraint. For example, if an agent wants to adopt a role ρ_1 but it already has a role ρ_2 and these two roles are not compatible, the adoption of ρ_1 must be denied.

The state of an OE is represented by the following tuple:

$$\langle os, \mathcal{A}, \mathcal{GI}, grType, subGr, agRole, \mathcal{SI}, scType, agMis, gState \rangle \quad (1)$$

where:

- os is the initial organisational specification (in $\mathcal{S}$-$\mathcal{M}$oise$^+$, OrgManager reads this OS from an XML file);
- $\mathcal{A}$ is the set of agents in the MAS;
- $\mathcal{GI}$ is the set of created groups;
- $grType : \mathcal{GI} \to \mathcal{GT}$ maps the group specification for each group in $\mathcal{GI}$ ($\mathcal{GT}$ is the set of group specifications defined in os);
- $subGr : \mathcal{GI} \to \mathbb{P}(\mathcal{GI})$ maps the sub-groups of each group;
- $agRole : \mathcal{A} \nrightarrow \mathbb{P}(\mathcal{R} \times \mathcal{GI})$ maps the roles of the agents ($\mathcal{R}$ is the set of roles defined in os);
- $\mathcal{SI}$ is the set of scheme instances;
- $scType : \mathcal{SI} \to \mathcal{ST} \times \mathbb{P}(\mathcal{GI})$ maps the specification and the responsible groups for each scheme instance ($\mathcal{ST}$ is the set of scheme specifications defined in os)[1];
- $agMis : \mathcal{A} \nrightarrow \mathbb{P}(\mathcal{M} \times \mathcal{SI})$ maps the missions of the agents ($\mathcal{M}$ is the set of missions defined in os);
- $gState : \mathcal{SI} \times \mathcal{G} \nrightarrow \{unsatisfied, satisfied, impossible\}$ maps the state of each goal ($\mathcal{G}$ is the set of global goals defined in os).

[1] The current version of $\mathcal{M}$oise$^+$ does not constraint the type of the groups that are allowed to be responsible for a scheme instance.

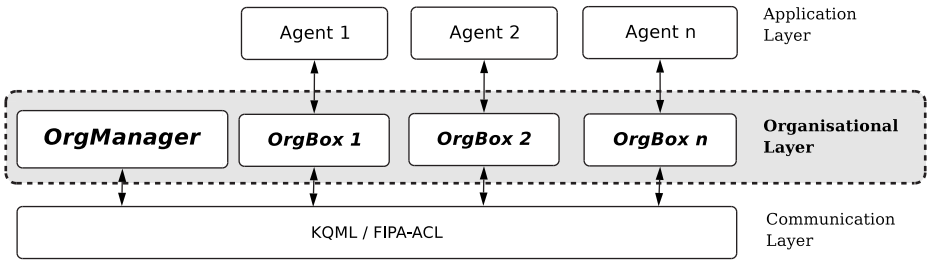

Fig. 3. $\mathcal{S}$-$\mathcal{M}$OISE$^+$ Components.

3.1 Organisational Entity Dynamics

The OE is changed by *organisational events* created by messages that OrgManager receives from the agents. Each event has arguments, preconditions and effects (Tab. 2 summarises these events). In this paper we describe only some of the events using our soccer example, a full formalisation can be found in [13] and `http://www.lti.usp.br/moise`.

As an example, suppose we have an OE where the following events happened:

- `createGroup('team')`: a group, identified hereafter by gr_t, was created from the team group specification defined in Fig. 1;
- `createSubGroup('defense',` gr_t`)`: a group, identified hereafter by gr_d, was created from the defense group specification as gr_t sub-group;
- `createSubGroup('attack',` gr_t`)`: a group, identified hereafter by gr_a, was created from the attack group specification as a gr_t sub-group;
- `createScheme('side_attack',` $\{gr_t\}$`)`: an instance of the side attack scheme specification (Fig. 2), identified by sch_{sa}, was created, the agents of the group gr_t are responsible for this scheme missions.

After these events, the groups are not well formed, since there is no agents engaged with their roles (see Fig. 4). The defense group, for instance, needs one agent playing goalkeeper. If an agent α wants to adopt the role ρ in the group gr, it must create the event `roleAdoption(`α`,` ρ`,` gr`)`. Notice that a role is always adopted inside a group of agents, since role is a relational concept [1]. The reasons for an agent to adopt a role is not covered by the $\mathcal{M}$OISE$^+$ model, for more details regarding motivations for role adoption, the reader is referred to [10,8,3]. The role adoption event in $\mathcal{S}$-$\mathcal{M}$OISE$^+$ has the following preconditions:

1. the role ρ must belong to gr's group specification;
2. the number of ρ players in gr must be lesser or equals than the maximum number of ρ players defined in the gr's compositional specification;
3. for all roles ρ_i that α already plays, the roles ρ and ρ_i must be intra-group compatible in the gr's group specification;
4. for all roles ρ_i that α already plays in groups other than gr, the roles ρ and ρ_i must be inter-group compatible.

Event	*Description* (some preconditions)
`createGroup(`gt`)`	Creates a new group from specification gt ($gt \in \mathcal{GT}$).
`createSubGroup(`gt`, `gi`)`	Creates a new gi sub-group based on specification gt (gi identifies an instance group).
`removeGroup(`gi`)`	Removes the group identified by gi (the group must be empty — no player, no sub-groups, and no schemes).
`createScheme(`st`, `gis`)`	Creates a new scheme instance from specification st ($st \in \mathcal{ST}$), gis ($gis \preceq \mathcal{GI}$) is a set of groups that are responsible for the new scheme execution.
`finishScheme(`si`)`	The scheme si is finished.
`setSatified(`α`, `si`, `g`)`	The goal g of the scheme si is satisfied by the agent α (α must be committed to a mission that includes g).
`setImpossible(`α`, `si`, `g`)`	The goal g of the scheme si is impossible (α must be committed to a mission that includes g).
`enterOrg(`α`)`	The agent α enters in the system.
`leaveOrg(`α`)`	the agent α leaves in the system (it must have neither roles nor missions).
`roleAdoption(`α`, `ρ`, `gr`)`	The agent α adopts the role ρ in the group gr.
`giveRoleUp(`α`, `ρ`, `gr`)`	The agent α gives up the role ρ in the group gr (this role missions must be finished).
`commitMission(`α`, `m`, `si`)`	The agent α commits to the mission m in the scheme si.
`finishMission(`α`, `m`, `si`)`	The agent α finishes its mission m in the scheme si (all the mission's goal must be satisfied or declared impossible).

Table 2. $\mathcal{S}$-$\mathcal{M}$OISE$^+$ main Organisational Events.

In our example, suppose that eleven agents have adopted roles such as the three groups are well formed and the goal "get the boal" of the scheme sch_{sa} is already satisfied. Among these agents, 'Lucio' has adopted the role middle in the gr_d group (once gr_d is a sub-group of gr_t, Lucio also belongs to gr_t). Is this agent following its organisational obligations? No, because he plays a middle role, there is a side attack scheme created by his group, and his role is obligated to commit to mission m_2 (the Fig. 5 describes the algorithm that gets all missions an agent is obligated to). To be organisationally well behaved, Lucio commits to the m_2 mission through the event `commitMission('Lucio',` m_2`,` sch_{sa}`)`. From the OrgManager point of view, this event also has some preconditions:

1. the scheme must not be finished yet;
2. the agent must play a role in the scheme's responsible groups;
3. this role must be permitted or obligated to the mission, as defined in the DS.

After his commitment, Lucio will likely pose the question: which are the global goal I have to achieve? In the case of his m_2 goals, only the goal "be

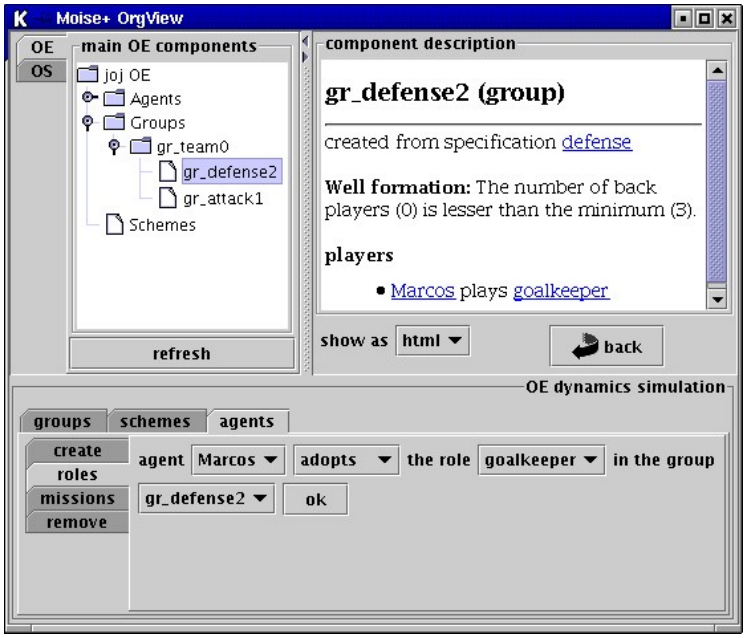

Fig. 4. Example of organisational entity not well formed.

placed in the middle field" is permitted (see Fig. 2). His second goal "go to the opponent back line" is not permitted by the current state of sch_{sa}. This second goal should be pursued only after another global goal is satisfied, since it depends on "kick the ball to" achievement. The algorithm of Fig. 6 is used in the OrgManager implementation to identify permitted global goals. Thus, while some goals are becoming satisfied (event `setSatified`), others become permitted. When a goal becomes permitted, the agents committed to it are informed by the OrgManager. This mechanism is very usefull to *coordinate* the agents in the scheme execution. The agent developer do not need to program messages that synchronise the agents in the schema execution.

The OrgManager ensures that every organisational events generated by the agents will not violate the following organisational constraints specified in $\mathcal{M}$OISE$^+$:

- the maximum number of role players in a group;
- the roles compatibility;
- an agent will commit only to missions it is permitted or obligated by its roles;
- only groups, schemes, and roles specified can be created.

Moreover, OrgManager provides useful information for the agents' organisational reasoning and coordination, for example: missions they are forced to commit

```
function getObligatedMissions(agent α)

all ← empty list {list of obligated missions}
for all role ρ the agent α plays do
  gr ← the group where ρ is being played
  for all scheme si that gr is responsible to do
    if si is not finished then
      for all mission m in the scheme si do
        if obligated(ρ, m) is in the deontic specificaion then
          all ← append(all, m)
        end if
      end for
    end if
  end for
end for
return all
```

Fig. 5. Algorithm to compute the missions an agent is obligated to.

to and goals it can pursue. The agents can get this information through their OrgBox API.

Among the $\mathcal{M}$OISE$^+$ specification elements, only the the authority link is not ensured in the current implementation. We probably need to change the agent reasoning mechanism to ensure authority, and it is out of this paper focus.

3.2 Agents' OrgBox

The OrgBox is the interface the agents use to access the organisational layer and thus the communication layer. When an agent desires to (*i*) change the organisational entity (adopt a role, for instance), (*ii*) send a message to another agent, or (*iii*) get the organisational entity state it has to ask this service for its OrgBox. The OrgBox will therefore interact with the OrgManager or another agent using the communication layer. In the $\mathcal{S}$-$\mathcal{M}$OISE$^+$ current implementation, the communication layer is implemented by SACI (`http://www.lti.pcs.usp.br/saci`) — a KQML compliant multi-agent communication infrastructure. We have developed a protocol in the communication layer that OrgManager and OrgBox follow to exchange information and organisational events. We can see the OrgBox as a component that encapsulates this protocol.

When an agent asks OrgManager for a "copy" of the current state of the OE, it will not receive exactly what is in the OrgManager's memory. In the $\mathcal{M}$OISE$^+$, an agent is allowed to know another agent α only in case it plays a role ρ_1, α plays ρ_2 and these roles are linked by an acquaintance relation. For example the player role of the Fig. 1 has an acquaintance link to the coach role, thus an agent playing this role is allowed to know the agents playing coach. Indeed, since player is an abstract role, no agent will adopt it, however other roles (like back, leader, etc.) will inherit this acquaintance link from the player role. OrgBox also

```
function isPermitted(scheme sch, goal g)

if g is the sch root then
  return true
else
  g is in a plan that match "g0 = ··· g ···"
  if g is in a plan that match "g0 = ··· gi , g ···" then
    if gi is already satisfied then
      return true
    else
      return false
    end if
  else
    return isPermitted(sch, g0)
  end if
end if
```

Fig. 6. Algorithm to verify permitted goals.

ensures that an agent will send messages only to agents it has a communication link.

While the OrgBox is invoked by the agent (to send messages, ask for information, change the organisation), it is also invoked by the OrgManager. When the state of a scheme that some agent is committed to changes, OrgManager informs this agent's OrgBox about its new obligations and goals it can pursue. The OrgBox then notifies the agent about its event. Of course the OrgBox only informs the agent about its permitted goals, it is a matter of the agent to achieve them (by plans, behaviours, etc.). What is stated in the organisational model is that the agent is responsible for such a goal. An important feature of our proposal is that it does not requite any specific type of agent architecture, since we are concerned with open system. The only requirement is that agents use the OrgBox API to interact with the system. An agent could even interact with the OrgManager directly using KQML or FIPA-ACL. However, in this case the communication link contraint will not be guaranteed, since in this case agents are getting direct access to the communication layer.

4 Contributions and Future Work

In this paper we describe a proposal towards declarative organisation programming. In our proposal, a middleware called $\mathcal{S}$-$\mathcal{M}$OISE$^+$ ensures that the agents will follow the organisational constraints. These constraints are declared by the developer (or even by the agents themselves) according to an organisational model. The organisational model used in our proposal enable the declaration of MAS organisational structure (role, groups, links), functioning (global goals, global plans, missions), obligations, and permission. The main features of $\mathcal{S}$-$\mathcal{M}$OISE$^+$ are:

– $\mathcal{S}$-$\mathcal{M}$OISE$^+$ follows an organisational centred point of view where the organisational specification is interpreted at runtime, it is not hardwired in the agents' code.
– It provides a synchronisation mechanism for scheme execution.
– It is suitable for heterogeneous and open system, since $\mathcal{S}$-$\mathcal{M}$OISE$^+$ is an exogenous approach and therefore does not require a special agent architecture or programming language.
– It is suitable for reorganisation where the declaration of the organisation can dynamically change. We have successfully used this framework in a soccer team that change its $\mathcal{M}$OISE$^+$ organisational at runtime [16]. Like the organisational events described in Sec. 3.1, the $\mathcal{S}$-$\mathcal{M}$OISE$^+$ also has reorganisational events that changes the current specification. However, these events are controlled by a special group of agents responsible for the reorganisation process.

Regarding related frameworks, $\mathcal{S}$-$\mathcal{M}$OISE$^+$ is quite complementary to AMELI, MadKit, and KARMA. Many implementation solutions proposed by these frameworks was adopted in $\mathcal{S}$-$\mathcal{M}$OISE$^+$ (like the OrgBox which is very similar to Teamcore proxy from KARMA and governor from AMELI). AMELI has a good support for communication and protocols that $\mathcal{S}$-$\mathcal{M}$OISE$^+$ does not have. However, it does not stress the structural and deontic dimensions like $\mathcal{S}$-$\mathcal{M}$OISE$^+$. MadKit is focused on the structural dimension and does not include functional and deontic dimension. KARMA is concerned with both the structure and the functioning and has an excellent support for coordination of global plan execution, however it lacks an explicit deontic dimension.

As a future development, we intend to extends $\mathcal{S}$-$\mathcal{M}$OISE$^+$ with new features like communication dimension, detection of violation of an agent obligation, and a sanction system. We also plan to define an organisational meta level, independently of the organisational model adopted, to create a (i) generic ontology of organisational terms and (ii) to provide translation to and from a particular organisational model to other.

References

1. Cristiano Castelfranchi. Commitments: From individual intentions to groups and organizations. In Toru Ishida, editor, *Proceedings of the 2nd International Conference on Multi-Agent Systems (ICMAS'96)*, pages 41–48. AAAI Press, 1996.
2. Cristiano Castelfranchi. Modeling social action for AI agents. *Artificial Intelligence*, (103):157–182, 1998.
3. Mehdi Dastani, Virginia Dignum, and Frank Dignum. Role-assignment in open agent societies. In Je□rey S. Rosenschein, Tuomas Sandholm, Wooldridge Michael, and Makoto Yokoo, editors, *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'2003)*, pages 489–496. ACM Press, 2003.
4. Maria Virgínia Ferreira de Almeida Júdice Gamito Dignum. *A model for organizational interaction: based on agents, founded in logic*. PhD thesis, Universiteit Utrecht, 2003.

5. Virginia Dignum and Frank Dignum. Modelling agent societies: Co-ordination frameworks and institutions. In Pavel Brazdil and Alípio Jorge, editors, *Proceedings of the 10th Portuguese Conference on Artificial Intelligence (EPIA'01)*, LNAI 2258, pages 191–204, Berlin, 2001. Springer.
6. Marc Esteva, Juan A. Rodriguez-Aguiar, Carles Sierra, Pere Garcia, and Josep L. Arcos. On the formal specification of electronic institutions. In Frank Dignum and Carles Sierra, editors, *Proceedings of the Agent-mediated Electronic Commerce*, LNAI 1191, pages 126–147, Berlin, 2001. Springer.
7. Marc Esteva, Juan A. Rodríguez-Aguilar, Bruno Rosell, and Josep L. AMELI: An agent-based middleware for electronic institutions. In Nicholas R. Jennings, Carles Sierra, Liz Sonenberg, and Milind Tambe, editors, *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'2004)*, pages 236–243, New York, 2004. ACM.
8. Jacques Ferber and Olivier Gutknecht. A meta-model for the analysis and design of organizations in multi-agents systems. In Yves Demazeau, editor, *Proceedings of the 3rd International Conference on Multi-Agent Systems (ICMAS'98)*, pages 128–135. IEEE Press, 1998.
9. Mark S. Fox, Mihai Barbuceanu, Michael Gruninger, and Jinxin Lon. An organizational ontology for enterprise modeling. In Michael J. Prietula, Kathleen M. Carley, and Les Gasser, editors, *Simulating Organizations: Computational Models of Institutions and Groups*, chapter 7, pages 131–152. AAAI Press / MIT Press, Menlo Park, 1998.
10. Norbert Glaser and Philippe Morignot. The reorganization of societies of autonomous agents. In Magnus Boman and Walter Van de Velde, editors, *Multi-Agent Rationality*, LNAI 1237, pages 98–111, Berlin, 1997. Springer.
11. Olivier Gutknecht and Jacques Ferber. The MadKit agent platform architecture. In *Agents Workshop on Infrastructure for Multi-Agent Systems*, pages 48–55, 2000.
12. Carl Hewitt. Open information system semantics for distributed artificial intelligence. *Artificial Intelligence*, (47):79–106, 1991.
13. Jomi Fred Hübner. *Um Modelo de Reorganização de Sistemas Multiagentes*. PhD thesis, Universidade de São Paulo, Escola Politécnica, 2003. `http://www.inf.furb.br/~jomi/pubs/2003/Hubner-tese.pdf`.
14. Jomi Fred Hübner, Jaime Simão Sichman, and Olivier Boissier. $\mathcal{M}$OISE$^+$: Towards a structural, functional, and deontic model for MAS organization. In Cristiano Castelfranchi and W. Lewis Johnson, editors, *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'2002)*, pages 501–502. ACM Press, 2002. `http://www.inf.furb.br/~jomi/pubs/2002/Hubner-aamas2002.pdf`.
15. Jomi Fred Hübner, Jaime Simão Sichman, and Olivier Boissier. A model for the structural, functional, and deontic specification of organizations in multiagent systems. In Guilherme Bittencourt and Geber L. Ramalho, editors, *Proceedings of the 16th Brazilian Symposium on Artificial Intelligence (SBIA'02)*, LNAI 2507, pages 118–128, Berlin, 2002. Springer. `http://www.inf.furb.br/~jomi/pubs/2002/Hubner-sbia2002.pdf`.
16. Jomi Fred Hübner, Jaime Simão Sichman, and Olivier Boissier. Using the $\mathcal{M}$OISE$^+$ for a cooperative framework of MAS reorganisation. In Ana L. C. Bazzan and Sofiane Labidi, editors, *Proceedings of the 17th Brazilian Symposium on Artificial Intelligence (SBIA'04)*, LNAI 3171, pages 506–515, Berlin, 2004. Springer. `http://www.inf.furb.br/~jomi/pubs/2004/Hubner-sbia2004.pdf`.

17. Christian Lemaître and Cora B. Excelente. Multi-agent organization approach. In Francisco J. Garijo and Christian Lemaître, editors, *Proceedings of II Iberoamerican Workshop on DAI and MAS*, 1998.
18. M.V. Nagendra Prasad, Keith Decker, Alan Garvey, and Victor Lesser. Exploring organizational design with TÆMS: A case study of distributed data processing. In Toru Ishida, editor, *Proceedings of the 2nd International Conference on Multi-Agent Systems (ICMAS'96)*, pages 283–290. AAAI Press, 1996.
19. David V. Pynadath and Milind Tambe. An automated teamwork infrastructure for heterogeneous software agents and humans. *Autonomous Agents and Multi-Agent Systems*, 7(1–2):71–100, 2003.
20. Carles Sierra, Juan Antonio Rodríguez-Aguilar, Pablo Noriega, Marc Esteva, and Josep Lluís Arcos. Engineering multi-agent systems as electronic institutions. *European Journal for the Informatics Professional*, V(4), August 2004.
21. Milind Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Reseearch*, 7:83–124, 1997.
22. J. Vázquez-Salceda, H. Aldewereld, and F. Dignum. Norms in multiagent systems: some implementation guidelines. In *Proceedings of the Second European Workshop on Multi-Agent Systems (EUMAS 2004)*, 2004.