Auton Agent Multi-Agent Syst (2010) 20:369–400
DOI 10.1007/s10458-009-9084-y

Instrumenting multi-agent organisations with organisational artifacts and agents

"Giving the organisational power back to the agents"

Jomi F. Hübner · Olivier Boissier · Rosine Kitio · Alessandro Ricci

Published online: 5 April 2009

Abstract The social and organisational aspects of agency have led to a good amount of theoretical work in terms of formal models and theories. However, the conception and engineering of proper organisational infrastructures embodying such models and theories are still an open issue. The introduction of normative concerns with requirements of openness and adaptation stresses this issue. The corresponding mechanisms for the current infrastructures appear to be not appropriate for managing distributed and open normative organisations. There is still the need of proper abstractions and tools to facilitate application agents taking part in the monitoring of the organisation on one hand, and in the adaptation and definition of the organisation in which they are situated on the other hand. In this paper we present and discuss ORA4MAS (Organisational Artifacts for Multi-Agent Systems), a proposed approach aiming at these issues. Based on the Agents and Artifacts meta-model (A&A), it introduces organisational artifacts as first class entities to instrument the organisation for supporting agents activities within it.

Keywords Multiagent systems · MAS organisations · Artifacts

J. F. Hübner (✉) · O. Boissier · R. Kitio
SMA/G2I/ENSM.SE, 158 Cours Fauriel, 42023 Saint-Etienne Cedex, France
e-mail: hubner@emse.fr; jomi@inf.furb.br

O. Boissier
e-mail: boissier@emse.fr

R. Kitio
e-mail: kitio@emse.fr

A. Ricci
DEIS, Università di Bologna, 47023 Cesena (FC), Italy
e-mail: a.ricci@unibo.it

1 Introduction

Current applications of IT show the interweaving of both human and technological communities (e.g. pervasive computing and ambient intelligence [23]), resulting in the construction of connected communities (ICities [37]) in which software entities act on behalf of users, cooperate and co-evolve with infohabitants, taking into account issues like trust, security, flexibility, adaptation, and openness [23,37]. Current applications have led to an increase in the number of agents, in the duration and repetitiveness of their activities, and in the range of decision possibilities for the agents [24]. Moreover the number of agents designers is also increasing, leading to a huge palette of heterogeneity in these systems.

One relevant topic in such systems is the coordination and control of the autonomous agents so that coherent functioning is promoted at the system level. The social and organisational aspects of agency are one possible answer to this problem. It has become nowadays an important focus of interest in the MAS community (e.g. the COIN workshop series [39]). Recent research in the MAS domain has provided many proposals for organisation modelling languages used to reify organisations into organisation specifications that are interpreted and used by application agents. In some sense, agents have become 'organisation-aware'. In the context of open systems, organisation-oriented middlewares have been proposed to interpret and support the management of the organisation according to its specification aiming mainly at ensuring that agents behave in accordance to the specified organisation.

While no consensual model of organisation emerges from these different works, the design of organisation-oriented middleware has led to a general architecture composed of services and agents responsible for organisational concerns such as monitoring, consistency maintenance, sanction applications and reorganisation. This middleware is generally introduced between the application agents and the agent communication platform. In those cases, the application agents do not have the possibility to take part in the management of the organisation in which they participate. In some sense, the agents are under the 'control' of the organisational middleware with respect to the management and use of *their* organisation. Our motivation is to soften the management of openness promoted by these organisational middlewares. We want not only to conceive an organisational infrastructure that supports and controls the agents in their participation to the organisation, but also consider an infrastructure that the agents can manage and use.

To this end, we consider the organisational infrastructure as part of the *computational environment*, engineered by MAS designers, where agents of the MAS are logically situated and where they can perceive and act upon as a first-class entity of their world. The role the *environment* can play in the MAS engineering has been remarked by recent works in the literature [45], as a first-class abstraction useful to encapsulate and provide functionalities for enabling and mediating agent interactions and supporting agent coordination [44]. Moreover, besides being a first-class abstraction for MAS designers at design time, suitable *environment infrastructures* make it possible to keep the abstraction alive at runtime, making the computational environment a first-class entity that agents can perceive, exploit, and manipulate to achieve their goals [42]. Among the others, the A&A meta-model introduces the notion of *artifact* as the basic abstraction to design and structure complex computational environments [28,35]: an MAS is conceived as an open collections of agents working together in *workspaces* where they dynamically construct, share, and use artifacts modelling *resources* and *tools* populating the agent world. While agents model the goal and task oriented part of the system, i.e. the locus of decision and control, artifacts model *function-oriented* parts, the tools that are used and adapted by agents to achieve their individual and collective goals.

Following the A&A perspective, in this paper we introduce a model called ORA4MAS which conceives organisational infrastructures inside MAS composed by both agents responsible of organisation management, referenced as *organisational agents*, and suitably designed artifacts, referenced as *organisational artifacts*. These artifacts are *intentionally* used on the one side by agents participating in the organisation to make their activities more effective and, on the other side, by organisational agents to control, manage, and adapt the organisation at runtime. This is in analogy with human organisations that are populated by humans (as participants and part of the organisational machinery), and by a rich set of artifacts and tools that humans use to support their activities both inside the organisation and in the management of the organisation itself. By promoting these mechanisms into first class entities as artifacts, open normative multi-agent organisations may be conceived and supported at the same abstraction level as the agents populating them.

In the first part of the paper (Sect. 2), we have a look at the different approaches that have been developed in the field of multi-agent organisation infrastructures, pointing out some drawbacks. Section 3 briefly presents the $\mathcal{M}$OISE$^+$ and A&A models, which our proposal is based on. In Sect. 4 we describe the general approach of ORA4MAS: our contribution towards a solution to some limitations identified in current works. The Sect. 5 and Sect. 6 exemplify the approach for a concrete organisational model and present some implementation solutions. The contribution is then discussed in Sect. 7. We finally provide concluding remarks and perspectives on the work in Sect. 8.

2 Organisational infrastructures for multiagent systems

Recent developments in the MAS domain originated many proposals towards the organisation oriented programming of MAS [3]. In all of them, we distinguish two important components: a declarative *Organisation Modelling Language* (OML) (e.g. $\mathcal{M}$OISE$^+$ [22], ISLANDER [11]) and an organisation implementation architecture (e.g. $\mathcal{S}$-$\mathcal{M}$OISE$^+$ [21], AMELI [12]). The OML defines the language used to write the specification of the organisation for an MAS. It is used to express specific constraints and cooperation patterns imposed on the agents by the designer (or the agents), resulting in an explicit representation that we call an *Organisation Specification* (OS). A collective entity, called an *Organisation Entity* (OE), is created by the agents when they adopt the roles specified in the OS. A set of computational tools related to the OML supports these agents to properly 'play' their roles. The development of these tools normally considers both an agent-centred and a system-centred point of view.[1] In the former, the focus lies on *organisational agent-level deliberative mechanisms* to interpret and reason about the OS and OE to which the agents participate [5,7]. In some sense, the aim is to develop *organisation-aware agents*. In the latter, the main concern is the *organisational infrastructure* (OI). In the sequel, we will focus on this second component.

The OI has two purposes: (i) to help and support the agents in the achievement of the organisation's global purpose by providing them access to organisational services and (ii) to control and enforce the satisfaction of the organisational constraints (e.g. agents playing the right roles, following the specified norms). This second point of view is important in heterogeneous and open systems where the agents that enter into the organisation may have unknown architectures and behaviours.

The implementation of an OI normally follows a similar trend to multi-agent communication platforms [3]. They introduce the organisation support mechanisms into a three layered

[1] In some proposals, these points of view are called agent and institutional perspectives [41].

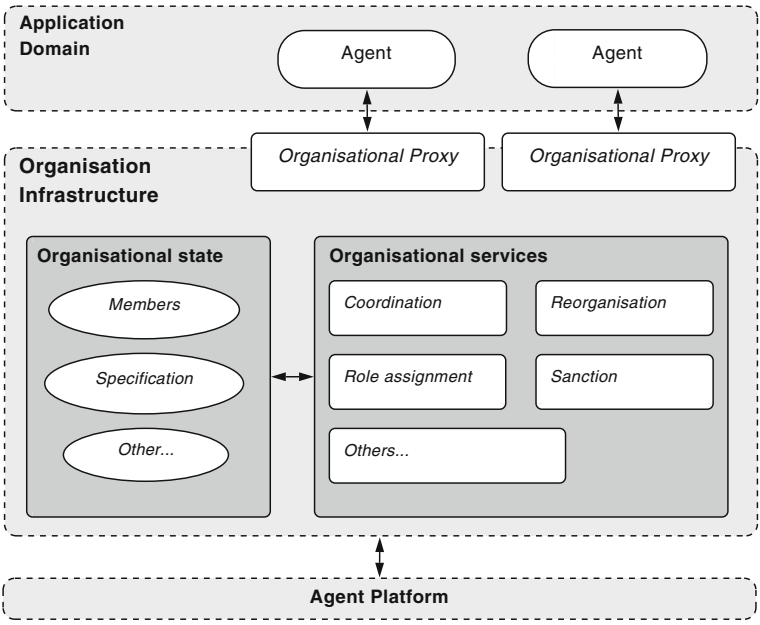


Fig. 1 Common architecture for organisation infrastructures in open MAS

middleware architecture, as depicted in Fig. 1: (i) domain or application agents that use an *organisational proxy* component to interact with the organisation and achieve application goals, (ii) the organisational infrastructure that is responsible for providing the agents with *organisational services* for helping and controlling their actions in the OE according to the OS, and (iii) the agent platform that connects all components in a distributed and heterogeneous context (e.g. JADE [2]). Examples of organisational proxies in some OI implementations are angels in OPERA [10], governors in AMELI [12], wrappers in TEAMCORE [32], and OrgBox in $\mathcal{S}$-$\mathcal{M}$OISE$^+$ [21]. One important point to notice is that all the access to the OI is mediated by these organisational proxies. This layered architecture results in an engineering approach where the MAS development is considered to be addressed by three kinds of designers: domain or application designers (for the agents and the specification of the OS using the OML), MAS or OI designers (for the organisational infrastructure and OE management), and agent platforms designers.

In contrast to this general picture, we will turn for a moment to the way that human organisations are used to facilitate common tasks. They are populated by humans (as participants and part of the organisational machinery), and by rich sets of artifacts and tools that humans use to support their activities inside the organisation. They also are used to support the organisation itself by encapsulating essential infrastructure services. According to psycho-sociological frameworks such as Activity Theory and Distributed Cognition [25], the notion of artifact (and tool, taken here as a synonym) plays a key role for the overall sustainability of an organisation and the effectiveness and efficiency of activities taking place inside it. In particular, some of these artifacts appear to be vital for supporting the coordination of organisational processes and management: for instance, by making more effective the communication among the members of an organisation (e.g. the telephone, instant-messaging services, chat-rooms), by providing information useful for orienting the activities

of organisation participants (e.g. signs inside a building), by coordinating participants (e.g. queue systems at the post-office), and by controlling access to resources and enforcing norms (e.g. the badge used by members in a computer science department to access certain rooms or use some other artifacts, such as copiers). Human societies and organisations continuously improve their experience in designing artifacts more and more effectively to support both organisation participation—helping members to cope with the complexity of social activities and work—and organisation management—helping managers to monitor and control the organisation behaviour as a whole.

From this brief introduction, we can point out some features of the current OI that motivate our proposal. It is important to note that the issues stated here do not concern solely the implementation level, but also the conceptual and theoretical levels.

1. *Mismatch of the abstraction level* for organisational support. The components of an OI are considered as services. Even if, in some proposals (e.g. $\mathcal{S}$-$\mathcal{M}$OISE$^+$ [21]), they have an agent status, they are mainly considered as 'services' with well-defined operation-oriented interfaces and contracts, and not really intelligent agents to interact and cooperate with through conversations and articulated ACL-based protocols, possibly providing unexpected and autonomous behaviours. However, those service-agents are taking decisions on the adaptation of the organisation, on the sanctions, and on the enforcement of norms which target the agents in the application layer. They are nonetheless not accessible by an application designer when s/he needs to customise some decisions of the system in the organisational dimension (e.g. a sanction, reorganisation or role assignment strategies).
2. *Organisation and Environment are not integrated*. In the current development process for MAS, the designer (and the agents) have to deal with two kinds of shared medium of interaction: a virtual organisation (where the agents adopt roles, send messages according to particular protocols, execute goals according to collective procedures, and participate in the manipulation, construction and realisation of shared social workflow throughout which they coordinate themselves) and the real physical environment (where the agents act, move, coordinate to access shared resources). Although their concerns are similar, their reification has led to different approaches: dedicated layer in the case of organisation, proper abstractions accessible and usable by the agents in case of environment. Moreover, even if organisation and environment have each their own specifics, they are not independent in the sense that organisations are situated in an environment. As shown by Okuyama et al. [27] some of its constructs (e.g. norms) take a different meaning according to their anchoring in the environment.
3. *Overwhelming power of the OI* in the services. According to the general architecture depicted in Fig. 1, services operating in the OI perform three main types of function: coordination, detection and decision/judgement on norm enforcement. For example, if some agent wants to perform some action or send a message that its organisation does not allow, the infrastructure (by means of the organisational proxy) will detect this violation attempt and may decide whether to apply a sanction or not without any interference of the application agents. Based on the way that human organisations work, we consider that it is more suitable that the agents operating on the application layer should get their organisational power back and take decisions on the sanctions and reorganisation.[2] The middleware approach to design the OI precludes the incorporation of strategies for sanction and reorganisation in the application level.

[2] Although different aspects of the state of an organisation can change (e.g. the current agents and groups), we consider here reorganisation as the process of changing the current OS of the MAS.

> Even if the services are configured to use some application knowledge, the clear separation into two different abstraction layers hinders agents operating at the application level in taking part in the regulation or reorganisation of the application. As stated by Castelfranchi et al. [6], social order is a necessary mechanism, and this mechanism requires that the agents themselves become actors of the mechanism.

These global statements lead us to propose a general model for structuring and architecting organisational support mechanisms by changing their status and promoting them as first class entities accessible and manageable by the agents. In concordance with this approach, we propose to move these mechanisms from the organisational middleware layer to the application layer. A global picture of this proposal is presented in Sect. 4.

3 Foundations

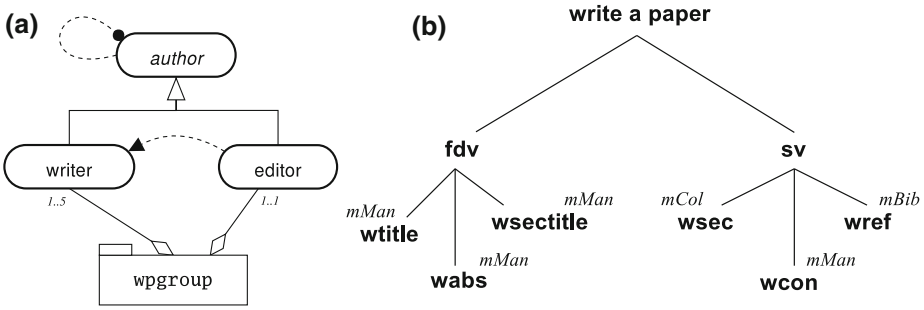
Before presenting ORA4MAS in Sect. 4, we set the context of our work. We briefly describe the $\mathcal{M}$OISE$^+$ organisational model, which is used in the sequel to reify and illustrate ORA4MAS (a fuller account of it can be found in [22]). We also introduce the basic ideas provided by the A&A meta-model that are used as guidelines for our approach. A more detailed description of this meta-model—including aspects not essential for this paper—can be found in [36].

3.1 The $\mathcal{M}$OISE$^+$ model

The $\mathcal{M}$OISE$^+$ model proposes an organisational modelling language that explicitly decomposes the specification of organisation into structural, functional, and deontic dimensions [22]. The structural dimension specifies the *roles*, *groups*, and *links* of the organisation. The definition of roles states that when an agent decides to play some role in a group, it is accepting some behavioural constraints related to this role. The functional dimension specifies how the *global collective goals* should be achieved, i.e. how these goals are decomposed (in global *plans*), grouped in coherent sets (by *missions*) to be distributed to the agents. The decomposition of global goals results in a goal-tree, called *scheme*, where the leaf-goals can be achieved individually by the agents. The deontic dimension is added in order to bind the structural dimension with the functional one by the specification of the roles' *permissions* and *obligations* for missions.

As an illustrative and simple example of an organisation specified using $\mathcal{M}$OISE$^+$, we consider agents that aim at writing a paper and therefore have an organisational specification to help them to collaborate. The structure of this organisation has only one group (`wpgroup`) with two roles (*editor* and *writer*) that inherit all properties defined for the role *author*. The cardinalities and links of this group are specified, using the $\mathcal{M}$OISE$^+$ notation, in Fig. 2a: the group `wpgroup` can have from one to five agents playing *writer* and exactly one playing *editor*; the *editor* has authority over *writer* and every agent playing *author* (and by inheritance everyone playing *writer* or *editor*) has the possibility to communicate with every agent playing *author* (there is a communication link from *author* to *author*). In this example, the *editor* and the *author* roles are not compatible. To be compatible, a compatibility relation must be explicitly added in the specification.

To coordinate the achievement of the goal of writing a paper, a scheme is defined in the functional specification of the organisation (Fig. 2b). In this scheme, a draft version of the paper has to be initially defined (identified by the goal *fdv* in Fig. 2b). This goal is



(c)

mission	cardinality
$mMan$	1..1
$mCol$	1..5
$mBib$	1..1

(d)

role	deontic relation	mission
editor	*permission*	$mMan$
writer	*obligation*	$mCol$
writer	*obligation*	$mBib$

Fig. 2 Graphical representation of the organisational specification for the writing paper example in the $\mathcal{M}$OISE$^+$ OML. **a** Structural specification. **b** Functional specification. **c** Mission cardinalities (part of FS). **d** Deontic specification

decomposed into three sub-goals: write a title, an abstract, and the section titles. The agents then have to 'fill' the paper's sections to get a submission version of the paper (identified by the goal *sv*). The goals of this scheme are distributed in three missions which have specific cardinalities (cf. Fig. 2c): the mission $mMan$ is for the general management of the process (one and only one agent can commit to it), mission $mCol$ is for the collaboration in writing the paper's content (from one to five agents can commit to it), and mission $mBib$ is for getting the references for the paper (one and only one agent can commit to it). A mission defines all goals an agent commits to when participating in the execution of a scheme, for example, a commitment to the mission $mMan$ is indeed a commitment to achieve four goals of the scheme. Goals without an assigned mission are satisfied by the achievement of their subgoals. The deontic relation from roles to missions is specified in Fig. 2d. For example, any agent playing the role *editor* is permitted to commit to the mission $mMan$.

The specification of an organisation is written in a suitable language [19], that the agents and the OI are supposed to interpret. This language is founded on components represented by predicates and functions. We present here only those components that are used in the sequel of the paper. Considering an OS, $\mathcal{G}$ is the set of all group specifications, $\mathcal{R}$ is the set of all roles, $\mathcal{S}$ is the set of all scheme specifications, $\mathcal{M}$ is the set of all missions, and Φ is the set of all goals.

- $compat(g, \rho, C)$: is a predicate that is true when the role ρ ($\rho \in \mathcal{R}$) is compatible with all roles in the set C ($C \subseteq \mathcal{R}$) when played in the group g ($g \in \mathcal{G}$);
- $mission\_scheme(m, s)$ is a predicate that is true when the mission m ($m \in \mathcal{M}$) belongs to the scheme s ($s \in \mathcal{S}$);
- $goal\_mission(\varphi, m)$: is a predicate that is true when the goal φ ($\varphi \in \Phi$) belongs to the mission m ($m \in \mathcal{M}$);
- $obl(\rho, m)$: is a predicate that is true when the role ρ has an obligation relation to the mission m;
- $per(\rho, m)$: is a predicate that is true when the role ρ has a permission relation to the mission m;

– $goal\_role(\varphi, \rho)$: is a predicate that is true when the role ρ is obliged to the goal φ, this predicate is defined as follows

$$goal\_role(\varphi, \rho) \leftrightarrow goal\_mission(\varphi, m) \ \wedge \ obl(\rho, m)$$

– $deadline$: $\mathcal{G} \rightarrow \mathbb{Z}$: is a function that maps each goal to an achievement deadline;
– $maxrp$: $\mathcal{R} \times \mathcal{G} \rightarrow \mathbb{Z}$: is a function that maps pairs of roles and groups to the maximum number of players of that role in the group (upper bound of role cardinality);
– $minrp$: $\mathcal{R} \times \mathcal{G} \rightarrow \mathbb{Z}$: is a function that maps pairs of roles and groups to the minimum number of players of that role necessary for the group to be considered well-formed (lower bound of role cardinality);
– $maxmp$: $\mathcal{M} \times \mathcal{S} \rightarrow \mathbb{Z}$: is a function that maps pairs of missions and schemes to the maximum number of agents that can commit to that mission in the scheme (upper bound of mission cardinality);
– $minmp$: $\mathcal{M} \times \mathcal{S} \rightarrow \mathbb{Z}$: is a function that maps pairs of missions and schemes to the minimum number of agents that have to commit to that mission for the scheme to be considered well-formed regarding to that mission (lower bound of mission cardinality).

3.2 The agents and artifacts (A&A) conceptual framework

The notion of MAS environment, as noted in recent literature, has gained a key role in the recent past, becoming a *mediating* entity, functioning as an enabler but possibly also as a manager and constrainer of agent actions, perceptions, and interactions (a comprehensive survey is presented in the JAAMAS special issue on environments for MAS [45]). According to such a perspective, the environment is not a merely passive source of agent perceptions and target of agent actions, which actually is the dominant perspective on agency and in MAS, but a first-class abstraction that can be suitably designed to *encapsulate* some fundamental functionalities and services, supporting coordination in MAS besides agent mobility, communications, security, etc.

Among the various approaches, the A&A conceptual framework introduces a notion of *work environment*, representing such a part of the MAS explicitly designed on the one hand by MAS designers to provide various kinds of functionality—including, for instance, MAS coordination—and perceived on the other hand as a first-class entity by agents of the MAS, as part of their context that they can reason about, use and possibly adapt to perform their individual and social tasks [28,36]. A&A work environments are made of *artifacts*, representing resources and tools that agents can create and use to perform their individual and social activities. Artifacts then can be conceived as complementary first-class abstractions to agents populating an MAS: while agents are goal-oriented pro-active entities, artifacts are *non-autonomous function*-oriented entities, i.e. designed by MAS designers to provide some kind of functionality or service for agents. Among the several sorts of artifacts, *coordination artifacts* have been introduced as an important class of artifacts [29], as artifacts mediating agent interactions and encapsulating some kind of coordinating functionality—whiteboards, event services, shared task schedulers are examples of such artifacts. Besides the notion of artifact, A&A introduces the notion of *workspace* as a logic container of agents and artifacts. Workspaces can be used to structure the overall sets of entities, defining a topology of the agent environment. In this view an MAS is conceived as a set of (distributed) workspaces, containing dynamic sets of agents working together by both directly communicating and sharing artifacts.

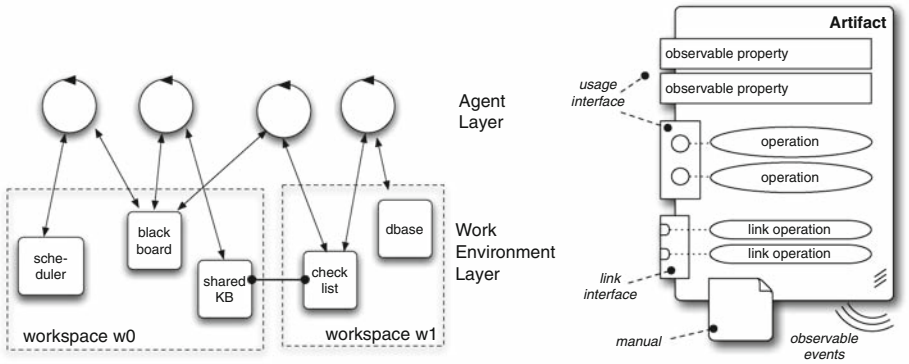


Fig. 3 (*Left*) Abstract representation of workspaces, populated by agents (represented by *circles*) and artifacts (represented by *squares*). (*Right*) A representation of the main parts and properties of an artifact, with the usage interface, the observable properties and the manual

Figure 3 shows an abstract representation of an artifact as defined in the A&A meta-model, exhibiting analogous parts and properties of artifacts to those found in human society. The artifact *function*—and related artifact behaviour—is partitioned into a set of *operations*, which agents can trigger by acting on an artifact *usage interface*. The usage interface provides all the controls that make it possible for an agent to interact with an artifact, that is to *use* and *observe* it. Agents can use an artifact by triggering the execution of operations through the usage interface and by perceiving *observable events* generated by the artifact itself, as a result of operation execution and evolution of its state. Besides the controls for triggering the execution of operation, an artifact can have some *observable properties*, i.e. properties whose value is made observable to agents, without necessarily executing operations on it. The interaction between agents and artifacts strictly mimics the way in which humans use their artifacts. For a simple but effective analogy, consider a coffee machine: the set of buttons of the coffee machine represents the usage interface, while the displays that are typically used to show the state of the machine represent artifact observable properties. The signals emitted by the coffee machine during its usage represent observable events generated by the artifact.

Table 1 shows the basic set of actions the agents can use to work with artifacts in workspaces, as supported by computational and programming models implementing A&A (in particular CArtAgO, described in Sect. 6.2). Among these actions, we remark use and focus. The former makes it possible to trigger and control the execution of an operation inside the artifact—by acting on controls of the usage interface. The latter makes it possible for an agent to be continuously (and automatically) aware of the dynamic value of observable properties and events generated by the artifact, both modelled as percepts from the environment.

Analogously to the human case, in A&A each artifact type can be equipped by the artifact designer with a *manual*, containing a machine-readable description of the artifact's function (i.e. its intended purpose), the artifact's usage interface (i.e. the observable 'shape' of the artifact), and the artifact's *operating instructions* (i.e. usage protocols or simply how to correctly use the artifact so as to take advantage of all its functionalities) [43]. The manual is a key feature when truly open systems are of concern, with intelligent agents that dynamically discover and select which kind of artifacts could be useful for their work, and then can use them effectively even if the artifacts have not been pre-established by MAS designers for the purpose.

Finally, artifacts can be composed together by means of *link interfaces*, which are sets of input/output ports that make it possible to enable inter-artifact interaction and finally link

Table 1 Abstract description of the core action set provided in CArtAgO to work with artifacts, with action parameters and action feedback

Action	Description
joinWsp(w_{id})	Joins an existing workspace, identified by w_{id}
quitWsp(w_{id})	Quits a workspace
use($art_{id}, oname, opars, sid, timeout$)	Triggers the execution of an operation on the artifact identified by art_{id}, by acting on the usage interface control $oname$ and passing parameters $opars$. A sensor sid can optionally be specified to collect events generated by the operation execution. If no sensor is specified, events are made observable to the agent directly as percepts. A timeout can be specified, to generate an internal action failure event if the action is not completed within the specified time
sense($sid, filter, timeout$) : evt	Retrieves (and returns as an action feedback) from the specified sensor an event matching the filter, suspending the action until the event is found or a timeout occurs (and the action fails)
focus(art_{id}, sid)	Starts observing the artifact identified by art_{id}, using the sensor sid. The values of artifact observable properties are directly mapped into percepts perceived by agents
stopFocus(art_{id})	Stops observing an artifact
observeProperty($art_{id}, pname$) : $pvalue$	Read from artifact art_{id} the current value of its $pname$ observable property
makeArtifact($w_{id}, aname, \tau, apars$) : art_{id}	Creates in workspace w_{id} a new artifact named $aneme$ from template τ and parameters $apars$. The artifact id art_{id} will be returned as action feedback
disposeArtifact(art_{id})	Disposes an existing artifact
lookupArtifact($w_{id}, aname$) : art_{id}	Gets the identifier of artifact $aname$ in workspace w_{id}

together artifact functionalities. Actually, link interfaces serve two purposes: on the one side, to explicitly define a principle of composability for artifacts, enabling the construction of articulated and complex artifacts by means of simpler ones; on the other side, to realise distributed (composed) artifacts by linking together artifacts belonging to different workspaces.

4 ORA4MAS model for organisational infrastructure

In this section we propose an organisational infrastructure called ORA4MAS where the basic idea is to instrument the multi-agent environment and the organisations living upon it in terms of agents and artifacts, based on the basic A&A meta-model presented in the previous section. The overall picture accounts for organisational agents that dynamically articulate, manage and adapt the organisation by creating, linking and manipulating the organisational artifacts, which are discovered and used by the member agents to work inside the organisation.

4.1 From organisational model to ORA4MAS

As stated in Sect. 2 and by Dignum and Dignum [9], the organisation of an MAS also serves as an instrument to control the autonomy of the agents. The success of an organisation depends on how the behavioural constraints stated in the organisational specification are ensured by

the organisational infrastructure. These behavioural constraints are entitled by the norms that are established by the specification of the organisation. In the context of this work, a norm is an obligation, permission, or interdiction to perform some action or achieve some goal. A norm also has a condition that states when it is active and a deadline to be fulfilled.[3] For example, whenever an agent plays the role *writer* in a group and there is a writing paper scheme with not enough agents committed to the mission $mCol$ (that is the condition part of the norm), the agent is obliged to commit to the mission $mCol$ (that is the action part of the norm).

As stressed by Grossi et al. [18], norms without enforcement mechanisms are worthless. Therefore, these norms, which are normally abstract concepts, should be implemented by means of mechanisms that instrument them in the organisational entity that is the context in which they have to be considered. In the ORA4MAS approach, we consider two types of such mechanisms:[4]

– *Regimentation* is a mechanism that simply prevents the agents from performing actions that are forbidden by a norm. More precisely, we regiment some actions in order to preserve important features (e.g. wellformedness) of the organisation. For example, the writing paper group can have at most one agent playing editor; to ensure that this constraint is respected, the organisational action 'adopt editor role' in this group is regimented accordingly.
  Since this mechanism has to work in an open system for any kind of agent, it has to be implemented 'outside' the agents in the OI. Therefore, a very special set of actions (those that are under the control of the OI, such as 'send a message' or 'role adoption') can be regimented.
– *Enforcement* is a mechanism which is applied after the detection of the violation of some norm. While regimentation is a preventive mechanism, enforcement is a reactive one. From the point of view of the agents, they may decide to obey the norm or not according to their local view of the organisation. From a system point of view, the fulfilment of the norms should be detected, evaluated as a violation or not, and then judged as worthy of sanction/reward or not.
  While detection can be implemented as an automatic process that does not require decision, the evaluation and the judgement need deliberation and reasoning. Given the difference among these tasks, the implementation of the enforcement mechanism should be assigned to different entities of the system. For instance, the detection could be implemented in the OI, whereas evaluation and judgement are more suitably implemented in agents.

These two mechanisms allow ORA4MAS to tackle the problem of balancing (i) the ensuring of very important properties of the OE by means of regimentation and, by means of enforcement, (ii) the agents' autonomy required to keep the possibility to adapt and evolve the organisation. The norms of the MAS can be instrumented either as regimentations or enforcement mechanisms depending on which side the designer wants to give more weight. Briefly, regimentation should be used to fully constrain the actions of the agents and enforcement should be used when some violation is allowed (or even desired).

[3] We are aware that the concept of norm is broader and more complex than that used in this paper (e.g. the work of Tuomela and Bonnevier-Tuomela [40] and the Deontic Logic in Computer Science workshop series [17]). For the present paper however this simple and informal definition is enough to discuss the proposal.

[4] This classification is based on the proposal described in [14,18]. However, we present them in a more specific context: regimentation is applied only to the interdiction of organisational actions and enforcement is applied to the other cases.

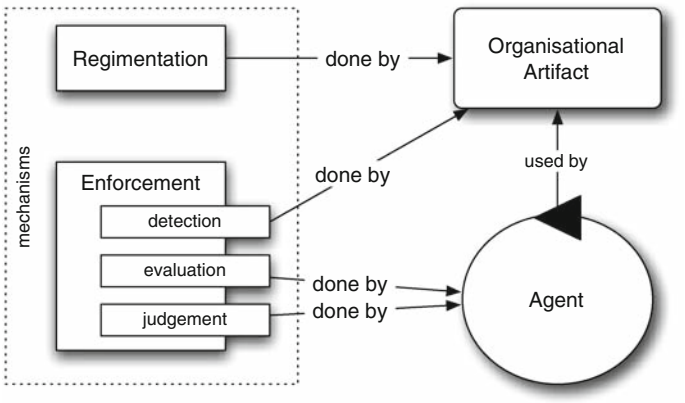


Fig. 4 General relation between the mechanisms that implement the norms and the organisational agents and artifacts of ORA4MAS

As illustrated in Fig. 4, two classes of handling mechanisms can be distinguished: management in terms of interpretation and checking of their satisfaction before executing changes in the OE; and management in terms of detection of non-fulfilment, statement of violation or not and sanction execution. From the requirements stated in Sect. 3, these two mechanisms of the organisation management must be taken into consideration by different entities. In our proposal, organisational artifacts and agents are those entities.

4.2 Organisational artifacts and agents

Here we use the terms *organisational agents* and *organisational artifacts* to identify those agents and artifacts of the MAS which are part of the organisational infrastructure. They are responsible for activities and encapsulate functionalities concerning the management and enactment of the organisation. Different OMLs require a different set of suitable artifacts and agents. For example, in the AGR organisational model [13], we can conceive artifacts to manage groups; for ISLANDER [11], the artifacts can be used to manage the scenes. In Sect. 5 a detailed set of artifacts are presented for the $\mathcal{M}$OISE$^+$ model.

In our approach (see Fig. 4), regimentation mechanisms are implemented in the artifacts that the application agents have to use to access the organisation. For instance, to adopt a role the agent has to use the appropriate artifact. The operation will be successfully executed only in the case the agent is allowed to adopt the role, otherwise the adoption fails. The agent autonomy for some organisational actions is therefore limited by the artifact. Organisational artifacts are therefore those artifacts that agents of an organisation may want or have to use in order to participate in organisational activities and to access organisation resources. They encapsulate organisational norms and functionalities, such as enabling, mediating and ruling on agent interaction, and tracing resource access. Analogously to the human case, artifacts are used here to reify and modularise the functional part of the organisation's machinery. It defines a distributed set of organisational resources and tools that can be perceived and used by agents as first-class entities and that can be dynamically adapted and possibly replaced (by agents themselves) during the organisation lifetime. Moreover, these artifacts do not create a sort of hidden organisational layer where the application is deployed, but rather they are placed beside the agents as available tools in the environment.

Some norms, however, cannot be implemented in the same way since they may be violated. In this case, the functionality provided by the artifacts is to *detect* and show (by means of observable properties) the non-fulfilment of a norm. We consider that agents should be informed of the current status of a norm and can evaluate the existence of a violation or not and take the best decision regarding the application objectives. To stress the embedding of dedicated reasoning and strategies related to the management of the organisation we call these agents *organisational agents*. They can be dedicated agents or agents of the application embedding special knowledge. They are essentially managers responsible for the creation and management of the organisational artifacts. Such activities typically include observing artifacts' dynamics and possibly intervening, by changing and adapting artifacts or interacting directly with other agents, so as to improve the overall (or specific) organisational processes or taking some kinds of decisions when detecting violations.[5] As an example, in the context of the $\mathcal{M}$OISE$^+$ model, one or multiple *scheme manager* agents can be introduced, responsible for monitoring the dynamics of the execution of a scheme by observing a specific artifact. The scheme artifact and scheme manager agents are designed so that the artifact allows for violation of the deontic rules concerning the commitment of missions by agents playing some specific roles, and then the decision about what action to take—after detecting the violation—can be the responsibility of the manager agent.

After sketching the basic concepts underlying the ORA4MAS approach, in the next section we describe how a fully-fledged organisational model can be implemented with organisational agents and artifacts.

5 ORA4MAS artifacts for $\mathcal{M}$OISE$^+$

We use here the $\mathcal{M}$OISE$^+$ model presented in Sect. 3.1 to identify and shape a basic set of organisational artifacts that constitute the building blocks of ORA4MAS. They are a sort of 'reification' of the structural specification (SS), functional specification (FS), and deontic specification (DS). This basic set accounts for: OrgBoard artifacts, used to keep track of the current state of deployment of the organisational entity in the overall; GroupBoard artifacts, used to manage the life-cycle of a specific instance of a group of agents; SchemeBoard artifacts, used to support and manage the execution of a social scheme; and NormativeBoard artifacts, used to maintain information concerning the agents compliance or not according to norms.

An organisational entity is then reified by a set of artifacts: exactly one OrgBoard; one GroupBoard for each instance of a group of agents; one SchemeBoard for each scheme being executed by the agents; and one NormativeBoard for each SchemeBoard. These organisational artifacts are linked together to allow the synchronisation of some operations and to share information required to maintain a coherent and consistent state of the OE. For example, the state of the NormativeBoard artifact needs to be updated with the information about the agents committed to missions so that it can evaluate the fulfilment of norms. The links between the artifacts are the following: the OrgBoard is linked to all other artifacts of the organisation; the GroupBoard is linked to all schemes its agents are responsible for; the SchemeBoard is linked to exactly one NormativeBoard that verifies the status of the norms related to the execution of the scheme; and finally the NormativeBoard is linked to its SchemeBoard and all GroupBoards responsible for the corresponding scheme.

[5] In this paper we focus on the regimentation management and detection of non-fulfilled norms done by the organisational artifacts. The use of artifacts as an instrument to help the entire enforcement process is better discussed in Hübner et al. [20].

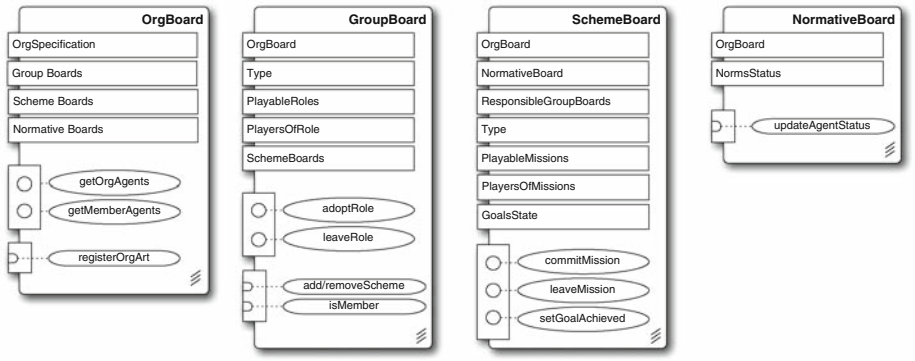


Fig. 5 Basic kinds of artifacts in ORA4MAS, with their usage interface, including operations (represented by *circles*) and observable properties (represented by *rectangles*), and the link interface (represented by *half eclipses* in the bottom of the artifacts)

In the following we briefly describe these artifacts. Here we consider just a core set of the characteristics and functioning of the artifacts, skipping most details that would make heavy the overall understanding of the approach.

5.1 OrgBoard artifact

An abstract representation of the OrgBoard is depicted in Fig. 5. The observable properties of this artifact are:

OrgSpecification: contains the specification of the OE written in the $\mathcal{M}$OISE$^+$ OML. Agents may use this observable property to get the organisational specification. They can then reason about it and decide whether they want to enter in the organisation.

GroupBoards, SchemeBoards, and NormativeBoards: contain identifiers of all instances of GroupBoard, SchemeBoard, NormativeBoard, respectively, within the OE. Generally speaking, these observable properties make it possible for agents observing an OrgBoard to know the current set of SchemeBoard, GroupBoard, NormativeBoard instrumenting the OE.

The usage interface of the OrgBoard has the following operations:

getOrgAgents(): used to get the set of agents having the status of organisational agent in the OE (this status is set during the deployment of the system).

getMemberAgents(): used to get the set of all agents playing at least one role within a group of the organisation entity.[6]

The main link operation of the OrgBoard is registerOrgArt. It is used in the initialisation process of each new instance of GroupBoard, SchemeBoard, and NormativeBoard to be registered as an artifact linked to the OE represented by the OrgBoard.

5.2 GroupBoard artifact

A GroupBoard is an artifact instantiated upon a specific group specification and provides functionalities to manage the corresponding group instance in the OE. It maintains a consistent

[6] This operation would be implemented as an observable property; however due to distributed characteristic of the information (its is managed by all group boards), to maintain the observable property always up to date may consume a lot of time. Using an operation, the list of member agents can be computed only when demanded and then cached.

state of that group by regimenting some important norms bearing on structural properties. For instance, whenever some agent asks for a role adoption in the group managed by the GroupBoard, the GroupBoard regiments a set of norms that state when a role can be adopted: (1) the role belongs to its group specification; (2) each role that the agent already plays is specified as compatible with the new role; and (3) the number of players is less or equal to the maximum number of players defined in the group's compositional specification.

As an artifact, the GroupBoard has some observable properties (Fig. 5). Among them, the most relevant are:

OrgBoard: is the reference to the OrgBoard that represents the OE to which the group belongs.
Type: is the identification of the group specification in the structural specification (an element of $\mathcal{G}$).
PlayableRoles: contains all roles that can still be played in this group, i.e. those for which the number of players is not the maximum yet. This property changes whenever a new agent enters into the group by adopting a role.
PlayersOfRole: contains the names of all agents belonging to the group and their corresponding roles.
SchemeBoards: contains a set of all schemes the group is responsible for.

Generally speaking, the above defined observable properties of the GroupBoard enable agents to know which are the available roles and their constraints, which are the participant agents, and which are the other organisational artifacts linked to the GroupBoard. To perceive the observable properties of a GroupBoard, agents can use either *active perception* (polling the artifact) or the *focus* mechanism (which continuously send signals to the agent whenever some observable property change, cf. Table 1).

The usage interface accounts for the following operations:

adoptRole(ρ): used by an agent to adopt a new role in the group, where $\rho \in \mathcal{R}$ is the identifier for a role in the SS. Note that an agent enters an organisation entity by adopting a role within a GroupBoard.
leaveRole(ρ): used by an agent to give up the role ρ that it had adopted previously.

The operations that are regimented, like adoptRole, can fail. For example, when the agent that triggered it is not allowed to do that. In such cases, instead of receiving a 'successful' signal as the result of the operation, the agent receives a 'failure' signal which includes the reasons for the failure.

The link operations of a GroupBoard manage the coordinations with its linked organisational artifacts. Among them, we have:

addSchemeBoard(sb): used by a SchemeBoard initialisation process to notify the GroupBoard that it is responsible for the scheme sb. The GroupBoard updates accordingly its SchemeBoards observable property.
removeSchemeBoard(sb): used by a SchemeBoard linked to the GroupBoard, to notify that the GroupBoard is no longer responsible for the execution of the scheme sb. The GroupBoard updates accordingly its SchemeBoards observable property.
isMember(α): used by a SchemeBoard to inquire whether an agent α is a member (i.e. is playing at least one role) of the group managed by the GroupBoard.

Before presenting the norms that a GroupBoard regiments, we introduce some predicates and functions related to the internal state of this artifact. Let $\mathcal{GB}$, $\mathcal{R}_g$, and $\mathcal{A}$ be, respectively, the set of current group boards, the set all roles that can be played in a group board created from specification g, and the set of all agents participating in the OE.

– $group\_type(gb, g)$: this predicate is true when the group board $gb \in \mathcal{GB}$ is created based on the group specification $g \in \mathcal{G}$ (cf. Sect. 3.1);
– $plays(\alpha, \rho, gb)$: this predicate is true when the agent $\alpha \in \mathcal{A}$ plays the role ρ in the group board $gb \in \mathcal{GB}$;
– $rplayers : \mathcal{R} \times \mathcal{GB} \rightarrow \mathbb{Z}$ with

$$rplayers(\rho, gb) \stackrel{\text{def}}{=} |\{\alpha | plays(\alpha, \rho, gb)\}| \tag{1}$$

this function gives the number of current players of the role ρ in the group gb.

Given the above definitions and the functions $maxrp$ and $minrp$ (cf. Sect. 3.1), we are able to define the wellformedness property of a group

$$\begin{aligned} well\_formed(gb) \leftarrow\ & group\_type(gb, g)\ \wedge \\ & \forall_{\rho \in \mathcal{R}_g} rplayers(\rho, gb) \geq minrp(\rho, g)\ \wedge \\ & rplayers(\rho, gb) \leq maxrp(\rho, g) \end{aligned} \tag{2}$$

Since role adoption is the very action that may bring a group into an inconsistent state, two norms bearing on this organisational action are regimented by a GroupBoard: the role compatibility norm and the role cardinality norm. In the following, these norms are represented as a pair where the first argument is the condition part stating when the norm is active and the second argument is the action part stating an obligation, permission, or interdiction.

Role compatibility norm: roles are incompatible unless the contrary is explicitly stated in the OS. Thus, if it is stated that two roles ρ_1 and ρ_2 are compatible inside a group g ($compat(g, \rho_1, \{\rho_2\})$), it implies that an agent that plays ρ_1 in a group board gb created from the specification g cannot perform the operation $\mathsf{adoptRole}(\rho_i)$ for any $i \neq 2$ on the corresponding artifact. This constraint on role adoption is formalised by the following norm:

$$\begin{aligned} & (plays(\alpha, \rho, gb)\ \wedge\ group\_type(gb, g)\ \wedge\ compat(g, \rho, C), \\ & \forall_{\rho_i \in \mathcal{R} \setminus C}\ FORBIDDEN_\alpha\ \mathsf{adoptRole}(\rho_i)) \end{aligned} \tag{3}$$

The condition of the norm (the first line) is a conjunction of predicates. Its evaluation is given by the particular circumstance of the group board (that defines whether $plays(\alpha, \rho, gb)$ and $group\_type(gb, g)$ hold or not) and the structural specification being used by the artifact (that defines whether $compat(g, \rho, C)$ holds or not). The action part of the norm (the last line) states that it is forbidden for agent α to execute the action adoptRole for any role that does not belong to the set of compatible roles C. Based on this norm, as soon as an agent adopts a role (activating the norm), the adoption of any other incompatible role is forbidden for it.

Role cardinality norm: the number of players of a role in a group is limited by the function $maxrp(\rho, g)$ defined from the SS. The following norm constrains the role adoption based on the cardinality of the role:

$$\begin{aligned} & (group\_type(gb, g)\ \wedge\ rplayers(\rho, gb) \geq maxrp(\rho, g), \\ & \forall_{\alpha \in \mathcal{A}} \forall_{\rho \in \mathcal{R}}\ FORBIDDEN_\alpha\ \mathsf{adoptRole}(\rho)) \end{aligned} \tag{4}$$

Since these two norms are of the type 'action interdiction', they can be easily implemented in the artifact: whenever the adoptRole operation is requested by the agent α, the conditions of all norms have to be verified using the structural specification and the current state of the group artifact. If the condition of some of these norms holds, the execution of the corresponding operation is denied.

5.3 SchemeBoard artifact

The SchemeBoard is an artifact instantiated upon a specific social scheme specification of the FS. It provides functionalities to manage the execution of a social scheme, coordinating the commitments to missions and the achievement of goals. It manages the dependencies between the missions and the goals as described by the social scheme.

The execution of a scheme has three phases: formation, goal achievement, and finishing. In the formation phase, agents commit to the missions of the scheme. As in the GroupBoard, a property of wellformedness is defined: roughly a scheme is well-formed if the mission cardinalities are satisfied, i.e. there are enough agents committed to the missions (the formal definition is presented below). To be well-formed is the condition to pass the execution to the second phase. In the second phase, the goals should be fulfilled by the agents. Each agent has to achieve the goals of the missions they are committed to. When the root goal of the scheme is satisfied, the third phase starts and the scheme can be finished and removed from the organisation (i.e. the corresponding artifact is destroyed).

During the execution of a scheme, its goals can be in three different states: $waiting$, $possible$, or $achieved$. The initial state is $waiting$. It indicates that the goal cannot be pursued yet because it depends on the achievement of other goals (called pre-conditions for a goal) or the scheme is not yet well-formed. The set of pre-condition goals are deduced from the goal decomposition tree of the scheme (as presented in Fig. 2b). For example, the goal 'to write the conclusion of the paper' ($wcon$) can be achieved only after the goal of 'writing sections' ($wsec$) was achieved. When all pre-condition goals are verified and the scheme is well-formed, the state of a goal switches to $possible$. Then the agent(s) committed to a mission containing the goal can pursue it. Note that the change from the state $waiting$ to $possible$ is performed by the SchemeBoard, whereas the change from the state $possible$ to $achieved$ is performed by the agents.

The observable properties of a SchemeBoard are the following (Fig. 5):

OrgBoard: is the reference to the OrgBoard that represents the OE in which the scheme is being executed.
NormativeBoard: is the reference to the NormativeBoard of the scheme.[7]
ResponsibleGroupBoards: contains the references to the groups that are responsible for the scheme.
Type: is the identification of the scheme specification in the functional specification (an element of $\mathcal{S}$).
PlayableMissions: contains all missions that can still be committed to in the scheme.
PlayersOfMission: contains all the agents committed to the scheme and their corresponding mission.
GoalsState: contains the current state of the goals of the scheme.

By observing a SchemeBoard, an agent can monitor the overall dynamics of the scheme execution. It can be aware of which missions are assigned to which agents, which goals are achieved and which can be pursued.

The usage interface provides the following operations:

commitMission(m): used by an agent to commit to a mission $m \in \mathcal{M}$;
leaveMission(m): used by an agent to give up a mission m it is committed to;
setGoalAchieved(φ): this operation is used to set the state of a goal to $achieved$.

[7] This observable property indirectly links all groups responsible for the scheme to the normative board of the same scheme.

Based on the current state of a SchemeBoard, we can define the following predicates and functions. Let $\mathcal{SB}$ and $\mathcal{M}_s$ be, respectively, the set of current scheme boards and the set of all missions that can be played in a scheme board created from specification s.

- $scheme\_type(sb, s)$: this predicate is true when the scheme board $sb \in \mathcal{SB}$ is created based on the scheme specification $s \in \mathcal{S}$ (the type of a scheme board is defined in its creation);
- $resp\_group(gb, sb)$: this predicate is true when the group $gb \in \mathcal{GB}$ is responsible for the execution of the scheme $sb \in \mathcal{SB}$;
- $committed(\alpha, m, sb)$ this predicate is true when the agent $\alpha \in \mathcal{A}$ is committed to the mission $m \in \mathcal{M}$ in the scheme $sb \in \mathcal{SB}$;
- $achieved(\varphi, sb)$: this predicate is true when the goal φ is already achieved in the scheme sb;
- $possible(\varphi, sb)$: this predicate is true when the state of the goal φ is possible in the scheme sb; considering Φ' the set of all goals that are pre-condition of φ, this predicate can be deduced by

$$possible(\varphi, sb) \leftarrow \bigwedge_{\varphi' \in \Phi'} achieved(\varphi', sb) \ \wedge \ well\_formed(sb) \tag{5}$$

- $mplayers : \mathcal{M} \times \mathcal{SB} \rightarrow \mathbb{Z}$ with

$$mplayers(m, sb) \stackrel{\text{def}}{=} |\{\alpha | committed(\alpha, m, sb)\}| \tag{6}$$

  this function gives the number of current players of the mission m in the scheme of sb.

Given the above definitions and the functions $maxmp$ and $minmp$ (cf. Sect. 3.1), we are able to define the wellformedness property of a scheme:

$$\begin{aligned} well\_formed(sb) \leftarrow\ & scheme\_type(sb, s) \ \wedge \\ & \forall_{m \in \mathcal{M}_s} mplayers(m, sb) \geq minmp(m, s) \ \wedge \\ & mplayers(m, sb) \leq maxmp(m, s) \end{aligned} \tag{7}$$

Analogously to the role cardinality norm, we define a norm to forbid an agent to commit to missions in a scheme.

Mission commitment norm: the number of agents already committed to a mission constrains the action commitMission (mission cardinality). Another constraint is that only agents that play some role in a responsible group of the scheme can commit to a mission in the scheme. This norm is defined as follows:

$$\begin{aligned} & ((scheme\_type(sb, s) \ \wedge \ mplayers(m, sb) \geq maxmp(m, s)) \ \vee \\ & (resp\_group(gb, sb) \ \wedge \ \neg plays(\alpha, \rho, gb)), \\ & FORBIDDEN_{\alpha} \ \text{commitMission}(m)) \end{aligned} \tag{8}$$

The implementation of this norm follows the same algorithm used by the GroupBoard: whenever an agent attempts to commit to a mission, if the condition of the norm holds, the operation is denied.

Note that in the current version of ORA4MAS, there is no regimentation on the leaving of a mission or a role. We consider that these organisational actions should give rise to enforcement and violation. For instance, we could imagine as a violation the fact of leaving a mission while still having goals of this mission to be achieved. Following the detection of a violation, sanctions have to be decided by organisational agents.

5.4 NormativeBoard artifact

The NormativeBoard artifact (Fig. 5) embeds the functionalities to manage the specification concerning permissions and obligations defined between roles and missions. This artifact has no operation available to the agents since its function is to detect and show as observable properties the current status of the norms according to the agents' behaviour related to the groups and scheme the normative artifact is linked to. There is one link operation (updateAgentStatus) used by the assigned scheme and groups to trigger an update of the current status of a particular agent, normally because the agent has performed some operation in the scheme or group.

The set of norms are defined from the deontic specification and the current state of related artifacts. As examples, in the sequel some of these norms are presented.

Obligation to commit to a mission: based on the deontic relations $obl(\rho, m)$ included in the organisation specification (as defined in Sect. 3.1), the roles played by the agents (as defined in the Sect. 5.2), and the current number of agents committed to a mission (an agent is not obliged to commit to a mission if the minimum number of players is already achieved), we can define the following norm:

$$\begin{aligned}&(obl(\rho, m) \ \wedge \ plays(\alpha, \rho, gb) \ \wedge \ resp\_group(gb, sb) \ \wedge \\ &scheme\_type(sb, s) \ \wedge \ mission\_scheme(m, s) \ \wedge \\ &mplayers(m, sb) < minmp(s), \\ &OBLIGED^{t}_{\alpha} \ \mathsf{commitMission}(m))\end{aligned} \tag{9}$$

The three first lines of this norm are the condition that states when the norm is active and the last line represents the obligation for the target agent α to commit to the mission before the deadline t (where t is a value defined by the application designer).

Permission to commit to a mission: based on deontic relations $per(\rho, m)$ included in the organisation specification, and the roles played by the agents, we can define the following a norm:

$$\begin{aligned}&(per(\rho, m) \ \wedge \ plays(\alpha, \rho, gb) \ \wedge \ resp\_group(gb, sb) \wedge \\ &scheme\_type(sb, s) \ \wedge \ mission\_scheme(m, s), \\ &PERMITTED_{\alpha} \ \mathsf{commitMission}(m))\end{aligned}$$

which can be rewritten as an interdiction as follows

$$\begin{aligned}&(\neg(per(\rho, m) \ \wedge \ plays(\alpha, \rho, gb) \ \wedge \ resp\_group(gb, sb) \ \wedge \\ &scheme\_type(sb, s) \ \wedge \ mission\_scheme(m, s)), \\ &FORBIDDEN_{\alpha} \ \mathsf{commitMission}(m))\end{aligned} \tag{10}$$

Obligation to achieve a goal: once an agent α is committed to a mission m, it is obliged to fulfil the possible goals of the mission before the deadline of the goals (given by the function $deadline$, cf. Sect. 3.1). The norm below specifies that rule.

$$\begin{aligned}&(committed(\alpha, m, sb) \ \wedge \ goal\_mission(\varphi, m) \ \wedge \ possible(\varphi, sb), \\ &OBLIGED^{deadline(\varphi)}_{\alpha} \ \varphi)\end{aligned} \tag{11}$$

These norms are not implemented by regimentation, since we would like to allow the agents to violate them. Their implementation is thus not as simple as the implementation of the norms of group and scheme artifacts (where only interdictions are considered and regimentation is used as the mechanism).

The NormativeBoard manages the state of the norms as follows (more details are described in algorithm Algorithm 1).[8] When an agent starts playing a role in a group and this group is responsible for some scheme, instances of norms are created for the agent in all related normative boards. This instantiation process consists of copying the norm definition and grounding their variables (e.g. replacing ρ of norm (9) by the identification of the role the agent is playing). The state of a norm instance is initially *inactive*. It becomes *active* when its condition holds. When the agent executes the action as it is stated in the action part of the norm, the status of the norm becomes *fulfilled*. In the other case, i.e. the agents does not behave in time accordingly to the action part of the norm, the status of the norm becomes *unfulfilled*.

6 ORA4MAS in action

In this section we illustrate how the above defined artifacts can be used in the deployment of an example. We first describe how a set of agents can *use* those artifacts and after how those agents and artifacts can be implemented. The chosen example is very simple and is used just to illustrate the functioning of the proposal. Its reduced complexity also makes it entirely comprehensible.

6.1 The writing paper example

We consider that four agents (Tom, Eva, Bob, and Alice) want to write a paper together using the proposed architecture described above. Among these agents, Tom is also an organisational agent and thus it may create the OrgBoard for the system. The creation of the OrgBoard is based on an organisational specification where the roles, groups, schemes, etc. are defined as presented in Sect. 3.1. The following steps show one possible sequence of evolution of this system until their goal of writing a paper is achieved (these steps are also illustrated in Fig. 6).[9]

1. Tom creates the GroupBoard based on the specification of Fig. 2a. Once the GroupBoard is created, the artifact registers itself in the OrgBoard by means of the OrgBoard's link interface. To succeed in the creation, the new group should satisfy all constraints defined in the OS (the group cardinality, for instance).
2. Tom sends then a message to the other agents telling them about the new artifact.[10] While Tom decides to adopt the role *editor*, Eva, Alice, and Bob decide to adopt the role *writer* in this group. To adopt the role, they use the adoptRole operation of the GroupBoard. As this operation is regimented it may fail in case the agents do not fit the requirements for the role (cardinality of the role in the group, compatibility of roles, etc., as defined in norms (3) and (4)). The reasons for the adoption of roles is not covered here, but, for example, they may decide to become a writer because Tom has invited them to enter into the group.

[8] We are aware that this algorithm is not a general norm interpreter—which would be the ideal solution. Indeed it works for only the three norms defined for the normative board. Nevertheless, the use of norms in the overall process of developing this artifact is very useful both as a conceptual abstraction and a development tool. Although we need to change the algorithm if new norms are included, the changes are quite straightforward. The development of a more general algorithm with good performance is a future work in the project.

[9] All the code used to run this example is available at http://moise.sourceforge.net/demos/jaamas09.

[10] Agents can be aware of the current artifacts in the system either by communication or by inspecting a registry artifact that contains a list of all artifacts in the system.

// Considering $\mathcal{N}$ as the set of instance norms maintained by the normative board, this algorithm uses two functions: one that gives the state of the norms for each agent and another that gives their activation time. These functions are updated by the operator $\oplus$, where the expression $f \oplus m$ adds (or changes) the mapping m in the function f.

$state : \mathcal{A} \times \mathcal{N} \rightarrow \{inactive, active, fulfilled, unfulfilled\}$;
$activation : \mathcal{A} \times \mathcal{N} \rightarrow Time$;

$\mathcal{N} \leftarrow \{\}$
when agent i adopts the role ρ in a related group board **do**
  forall norms maintained by the NormativeBoard **do**
    $n_i \leftarrow$ an instance of the norm applied to agent i playing role ρ;
    add n_i into $\mathcal{N}$;
    $state \oplus (i, n_i) \mapsto inactive$; /* initial state of n_i */

when the state of a linked artifact has changed **or** some time has passed since last update **do**
  forall norm $n_i \in \mathcal{N}$ **do**
    if $state(i, n_i) = inactive$ **then**
      if the condition of n_i holds **then**
        $state \oplus (i, n_i) \mapsto active$;
        $activation \oplus (i, n_i) \mapsto now$; /* activation time of n_i */
    else if $state(i, n_i) = active$ **and** type of n_i is OBLIGED **then**
      if compliant(i, n_i) **then**
        $state \oplus (i, n_i) \mapsto fulfilled$;
      else if $(now - activation(i, n_i)) > deadline(n_i)$ **then**
        $state \oplus (i, n_i) \mapsto unfulfilled$;
    else if $state(i, n_i) = active$ **and** type of n_i is FORBIDDEN **then**
      if $\neg$compliant(i, n_i) **then**
        $state \oplus (i, n_i) \mapsto unfulfilled$;

function compliant(Agent i, Norm n) **begin**
  $sb \leftarrow$ the scheme board linked to this NormativeBoard;
  if n is FORBIDDEN commitMission(m) **and** $committed(i, m, sb)$ **then return** $false$;
  if n is OBLIGED commitMission(m) **and** $\neg committed(i, m, sb)$ **then return** $false$;
  if n is OBLIGED φ **and** $\neg achieved(\varphi, sb)$ **then return** $false$;
  return $true$;
end

Algorithm 1 Dynamics of the norms' state in a normative board

3. Tom creates the SchemeBoard to start the process of writing the paper. This artifact embeds the specification of Fig. 2b. As for the GroupBoard, scheme artifacts, once created, register themselves in the OrgBoard. As every SchemeBoard has one NormativeBoard and at least one group responsible for it, the initialisation process of SchemeBoard creates its corresponding normative board and then properly links the artifacts.
4. Once the scheme is created, obligations and interdictions are computed and verified by the NormativeBoard based on information about players and commitments in the group and scheme artifacts. The norms defined in ORA4MAS (as those exemplified in norms (9)–(10)) and algorithm Algorithm 1 are used to perform such computation. The deadline to fulfil the norms is considered as infinity in this example. The observable properties of this artifact just after the creation of the scheme are roughly as follows (the value 'active' in the fifth column means that the condition of the norm holds and the value 'fulfilled' means the agent is respecting the norm):

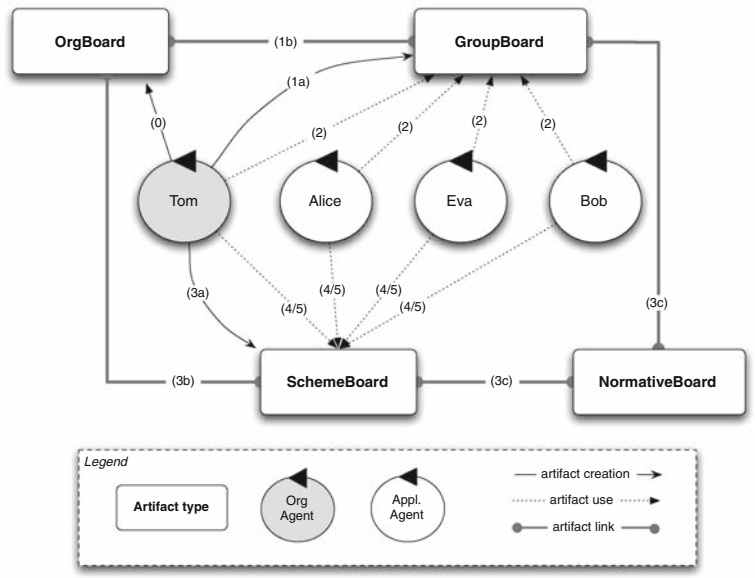


Fig. 6 Example of the construction and use of organisational artifacts by the agents

NormativeBoard—missions					
agent	role	missions	is committed	state of	
				norm (9)	norm (10)
Tom	editor	$mMan$	no	inactive	inactive
		$mCol$	no	inactive	active
		$mBib$	no	inactive	active
Alice	writer	$mMan$	no	inactive	active
		$mCol$	no	active	inactive
		$mBib$	no	active	inactive
Eva	writer	$mMan$	no	inactive	active
		$mCol$	no	active	inactive
		$mBib$	no	active	inactive
Bob	writer	$mMan$	no	inactive	active
		$mCol$	no	active	inactive
		$mBib$	no	active	inactive

By observing the normative board, Alice, that is playing the role *writer*, realises that she is obliged to commit to the missions $mCol$ and $mBib$ (cf. Fig. 2d and norm (9)). Let us suppose that she is obedient and decided to commit to both. Since the mission $mBib$ has a cardinality constraint that sets the minimum and maximum number of commitments to one, the other agents are not obliged to commit to this mission anymore. After Alice's commitment to $mBib$ and $mCol$, the observable properties of the normative board include:

NormativeBoard—missions					
agent	role	missions	is committed	state of	
				norm (9)	norm (10)
Tom	editor	*mMan*	no	inactive	inactive
		mCol	no	inactive	active
		mBib	no	inactive	active
Alice	writer	*mMan*	no	inactive	active
		mCol	yes	fulfilled	inactive
		mBib	yes	fulfilled	inactive
Eva	writer	*mMan*	no	inactive	active
		mCol	no	active	inactive
		mBib	no	inactive	inactive
Bob	writer	*mMan*	no	inactive	active
		mCol	no	active	inactive
		mBib	no	inactive	inactive

The other writers in the system commit thus only to the mission *mCol*. Tom, that plays editor, commits to *mMan*. We are assuming here that the agents are obedient and always commit to their obligations and pursue their organisational goals.

5. Once the scheme is well formed, according to the norm (7), the goals of the scheme can be pursued. The observable properties related to goals in the SchemeBoard are initially like the following:

SchemeBoard		
mission	goal	state
mMan	*wtitle*	*possible*
	wabs	*waiting*
	wsectitle	*waiting*
	wcon	*waiting*
mCol	*wsec*	*waiting*
mBib	*wref*	*waiting*

Initially only the goal *wtitle* is possible and thus can be pursued (i.e. *possible*(*wtitle*) holds), all the other goals are in the waiting state. The goal *wtitle* belongs to the *mMan* mission, so only Tom has something to do, the others will wait for him to achieve the *fdv* sub-goals. To know which goals can be pursued and to set them as achieved, the agents perceive the SchemeBoard and act on it using the setGoalAchieved operation. The above observable properties after setGoalAchieved(wtitle), performed by Tom, are:

SchemeBoard		
mission	goal	state
mMan	*wtitle*	*achieved*
	wabs	*possible*
	wsectitle	*waiting*
	wcon	*waiting*
mCol	*wsec*	*waiting*
mBib	*wref*	*waiting*

By showing the possible goals to the agents, the SchemeBoard works like a coordination artifact.

In this example, the NormativeBoard simply *shows* the state of the norms for each agent, i.e. whether the norm is inactive, active, fulfilled, or unfulfilled. As an artifact, it is not taking decisions from this status. An organisational manager agent, like Tom, must perceive this artifact and decide what to do when some norm violation occurs.

Here we have taken a simple example with only one group board, one scheme board, one normative board, and one organisational agent just to illustrate the ORA4MAS approach. We can however suitably design and manage an organisation with several artifacts which are independent or structured in an hierarchy (in case of an organisation with a group and subgroups as can be stated in $\mathcal{M}$OISE$^+$); where the goals of a scheme board can be distributed to agents from several responsible group boards; etc. These artifacts can then be distributed and supervised by several organisational agents.

6.2 A concrete architecture

ORA4MAS is realised on top of the CArtAgO infrastructure [34], embedding algorithms used in $\mathcal{S}$-$\mathcal{M}$OISE$^+$. CArtAgO (Common Artifact Infrastructure for Agent Open environment) is a infrastructure based on the A&A meta-model for developing and running artifact-based environments for MAS, integrated with (existing) agent platforms. CArtAgO technology[11] includes (i) a Java-based basic programming model for programming artifacts, (ii) a suitable API for agents to work with artifacts and workspaces, and (iii) a run-time platform supporting the existence and dynamic management of artifact-based environments.

A foremost important aspect of CArtAgO is its orthogonality with respect to the specific agent model and architecture adopted: this promotes the integration between CArtAgO and (existing) heterogeneous agent platforms, making it possible in principle to develop distributed, heterogeneous and open MAS, where agents dwelling in different agent platforms can autonomously join and work together in distributed and shared workspaces. Among those agent platforms, CArtAgO is integrated with ***Jason*** [4], 2APL [8], and JADEX [31]—these integrations are presented in [30,33]. ***Jason*** and JADEX are used here to exemplify how the agents can be programmed to use organisational artifacts. The code presented in Fig. 7 contains an excerpt of the program of agent Tom as programmed in ***Jason*** and Fig. 8 the code of agent Eva programmed in JADEX. To avoid a more detailed explanation of these two languages, we added comments in the figures explaining the code.

The final system is therefore formed by an organisation specification written in $\mathcal{M}$OISE$^+$ OML, distributed organisational artifacts, and agents programmed in different languages that are able to cooperate by means of these artifacts.

7 Related works and discussion

Several proposals for organisational infrastructures have been proposed in the literature: MADKIT, based on AGR organisational model [13]; AMELI [12] and AMELI$^+$ [15], based on ISLANDER [11]; KARMA, based on TEAMCORE [32]; OPERA [10]; $\mathcal{S}$-$\mathcal{M}$OISE$^+$ [21], based on $\mathcal{M}$OISE$^+$ [22]. In the sequel, these works and our proposal are discussed considering important topics and features of OI as identified in Sect. 2.

Abstraction & encapsulation The current OIs' components are either agents (AMELI, $\mathcal{S}$-$\mathcal{M}$OISE$^+$, OPERA) or services (MADKIT). However, as argued in this paper, the organisation should be implemented by both agents and services. The approaches that use only services

[11] CArtAgO is open-source and available at: http://cartago.sourceforge.net.

```
!create_org. // initial goal

+!create_org // plan to achieve the goal 'create_org'
   <- // join the ORA4MAS workspace
      cartago.joinWorkspace("ORA4MAS-Workspace","localhost:3047");

      // create org board artifact, the OS is stored in the file 'wp-spec.xml'
      cartago.makeArtifact("paper1","ora4mas.moise.OrgBoard",["wp-spec.xml"], OId);

      // crate group board artifact based on specification 'wpgroup'
      cartago.makeArtifact("grp1", "ora4mas.GrpBoard",[wpgroup], GId);

      // trigger the operation used to configure the group board,
      // informing the organisational board and the group type
      cartago.use(GId, configure(OId, "wpgroup", ""));

      // create the scheme
      cartago.makeArtifact("sch1", "ora4mas.moise.SchemeBoard", SId);
      // conf. the scheme board with the org. art., the scheme type and the resp. group
      cartago.use(SId, configure(OId, "writePaperSch", [GId]));
      // note that the normative board is created automatically

      // focus on the scheme and normative boards the receive events from them
      cartago.focus(SId);
      cartago.focus(NId);

      // adopt the role editor in the group
      cartago.use(Gid, adoptRole(editor));

      // ask bob to enter in the organisation
      // (other agents are waiting for the creation of the artifacts)
      .send(bob, achieve, enterOrg).

// react to changes in the group board
//    if I play editor, then commit to the manager mission
+role_adopted(GrpId,Role,Agent)
   <- cartago.lookupArtifact("sch1",SId);
      cartago.use(SId, commitMission(mMan)).

// react to changes in the scheme board
//    if a goal becomes possible, then start pursuing that goal
+new_possible_goal(G)[source(SId)]
   <- !G[source(SId)].

// plan to achieve the goal 'wtitle'
!wtitle[source(SId)]
   <- // perform all the actions required to achieve the goal
      // ...
      // inform the scheme board that the goal was satisfied
      cartago.use(SId, setGoalAchieved(wtitle)).
```

Fig. 7 Excerpt of the code of agent Tom written in ***Jason***

are not flexible enough to allow the management of the OI by the agents. The approaches that use only agents are using them for reactive and task oriented services; some of those agents are not really pro-active and autonomous entities. By using agents and artifacts to reify both the organisation and the organisation infrastructure, we raise the level of abstraction with respect to approaches in which organisation mechanisms are hidden at the implementation level by means of artifacts. Such mechanisms become parts of the agent world, suitably encapsulated in proper entities that agents then can inspect, reason and manipulate, by adopting a uniform approach.

```
<agent name="eva" package="ora4mas.jadex">

<beliefs>
    <!-- the role is empty until it will be adopted within the group board -->
    <belief name="myRole" class="String"/>
    ....
</beliefs>

<goals>                                                    <!-- properties of goals -->
  <performgoal name="adopt_role">       <!--  Goal to adopt a role in the organisation -->
    <parameter name="role_name" class="String"/>
  </performgoal>
  <performgoal name="write_section" >                    <!--  Initiate a new section -->
    <creationcondition> ($beliefbase.myRole != null ) </creationcondition>
  </performgoal>
  ...
</goals>

<plans>
  <plan name="adopt_role_action">    <!-- plan to adopt a role on GroupBoard Artifact  -->
    <body>new AdoptRolePlan()</body>
    <trigger> <goal ref="adopt_role"/> </trigger>
  </plan>

  <plan name="write_section_plan">         <!-- plan to achieve the goal write_section -->
     <body>new WriteSectionPlan()</body>
     <trigger> <goal ref="write_section"/> </trigger>
  </plan>

  <plan name="deliberate">   <!-- react to new possible goals event: create a new goal -->
     <body>new InitiateGoalPlan()</body>
     <trigger> <internalevent ref="artifact_event"/> </trigger>
     <precondition>
       $event.getParameter("label").getValue().startsWith("new_possible_goal" )
     </precondition>
  </plan>
  ...
</plans>

<configurations>
  <configuration name="default">
    <goals>
      <initialgoal ref="adopt_role">
        <parameter ref="role_name"> <value>"writer"</value> </parameter>
      </initialgoal>
      ...
    </goals>
  </configuration>
</configurations>
```

Fig. 8 Excerpt of the code of agent Eva written in JADEX

We will consider the following illustrative example to show one advantage of the proposed approach. In $\mathcal{S}$-$\mathcal{M}$OISE$^+$ or in MADKIT, the middleware managing the organisation is deployed and running with a specific strategy which is not defined nor decided by the agents. In our approach, agents may decide the creation of a particular group type, either $\mathcal{M}$OISE$^+$ or AGR. This creation is simply a matter of creating the corresponding artifact. This operation is realised and decided by agents. We could imagine, and our approach would facilitate such a possibility, that agents embed some domain knowledge leading them to reason and decide on the best machine on which to deploy this artifact, on the best number of artifacts to manage groups, etc.

Environment It is interesting to compare ORA4MAS and middlewares for e-Institutions, such as AMELI, for what concerns the notion of *environment* that both approaches refer to. The

abstract architecture of e-Institutions (AMELI in particular), places a middleware composed of governors and staff agents between participating agents and an agent communication infrastructure (e.g. JADE) [1]. The notion of environment is the one of *dialogical environment*: it is not a concrete environment that agents sense and act upon, but a conceptual one that agents playing within the institution can interact with by means of norms and laws, based on specific ontologies, social structures, and language conventions. In fact, governors and staff agents are not meant to be necessarily perceived by agents playing inside the institution as first-class entities of their world: agents communicate with each other by means of speech acts and behind the scene the middleware mediate such communication by means of institutional agents. The extension proposed for situated e-Institutions [1], introduces the notion of World of Interest (WoI) to model such environment *external* to the MAS and which is relevant to the MAS application. Participating agents in this case are able to interact with the WoI (the *entities*) not directly but always through the e-Institution, i.e. governors and staff agents.

In ORA4MAS we take a similar approach, where instead of governors and staff agents we are using artifacts to enable and mediate the access to the organisation. However, instead of a dialogical environment we provide a *work environment* that agents can perceive, act upon, and adapt as a first-class entity of their world. As first-class entities, organisational artifacts can be intentionally created and used by the agents—they are not "behind the scenes". Along with the other kinds of artifacts, in the A&A perspective, they are concrete bricks used to structure the agents' world: part of this world is represented by the organisational infrastructure, part by artifacts introduced by specific MAS applications, including some representing (wrapping) entities/services belonging to the external environment, i.e. the WoI of situated e-Institutions. In this case, participating agents are able to sense and act over the WoI using such artifacts.

Agent autonomy Although one of the goals of an OI is to manage in a more or less flexible way the constraints imposed by the organisation on the autonomy of the agents, it should not extinguish their autonomy. This issue is considered by all the above mentioned OIs. In AMELI, for instance, the agents are autonomous in the achievement of goals but the communication is constrained (or regimented) by the OI; in $\mathcal{S}$-$\mathcal{M}$OISE$^+$ the agents are autonomous concerning the communication protocols but constrained (or regimented) in the achievement and coordination of collective goals. In our proposal, agents are still autonomous with respect to the decision of using a specific artifact or not—including the organisational artifacts—and keeping their autonomy—in terms of control of their actions—while using organisational artifacts. Agents however can depend on the functionalities provided (encapsulated) by artifacts, which can concern, for instance, some kind of mediation with respect to the other agents sharing the same organisational artifact. Then, by enforcing some kind of mediation policy an artifact can be both an enabler and a constrainer of agent interactions. However, such a constraining function can take place without compromising the autonomy of the agents regarding their decisions.

Another important issue of our proposal is to clearly consider two kinds of mechanisms to implement the norms that are entitled in the OS: regimentation that is implemented in the artifacts and cannot be violated; and enforcement that is implemented both in the artifacts (the detection) and in the organisational agents (evaluation and judgement).

Distributed management Some OIs, such as $\mathcal{S}$-$\mathcal{M}$OISE$^+$ and MADKIT , centralise all the management of the organisation in one agent or service bringing scalability problems. Distributing the management of the organisation into different organisational artifacts realises a distributed coordination (meaning here more particularly synchronisation) of the different functions related to the management of the organisation. Completing this distribution

of the coordination, the reasoning and decision processes which are encapsulated in the organisational agents may be also distributed among the different agents. Thanks to their respective autonomy, all the reasoning related to the management of the organisation (monitoring, reorganisation, control) may be decentralised into different loci of decision with a loosely coupled set of agents.

Openness To be open to the entrance of heterogeneous agents is an important feature for MAS in general and a reason to establish an organisation for the system. This is thus also an issue considered by all the above OIs. In most cases (e.g. $\mathcal{S}$-$\mathcal{M}$OISE$^+$, AMELI), the agents have access to the organisational infrastructure by means of an agent communication language (KQML, FIPA-ACL) or other open protocols. ORA4MAS does not use a protocol or communication language; operations are used instead. The interaction between the agents and the organisation is no longer expressed with a ACL semantics. Besides that, organisational artifacts, as with any other kind of artifact, can be created and added dynamically as needed. They have a proper semantic description of both the functionalities and operating instructions, so conceptually agents can discover at runtime how to use them in the best way.

Still related to openness, the approach promotes heterogeneity of agent societies: artifacts can be used by heterogeneous agents, with different kinds of reasoning capabilities. Extending the idea to multiple organisations, we can have the same agents playing different roles in different organisations, and then interacting with organisational artifacts belonging to different organisations.

As shown in the example of Sect. 6.2, the use of artifacts, and particularly the CArtAgO implementation, allows agents implemented in different languages to use the artifacts and cooperate using them. Most of the OIs listed above give tools and support only for agents implemented in a particular language, normally Java—which is not the most appropriate language to code some types of agents.

'Organisational power back to agents' As stressed in Sect. 2, the current implementations of OIs conceive the organisation as a layer that the application agents rely on to participate in the organisation's activities. The agents are not *managers* of this layer, they are simply passive users. This conception of OI is captured by the notion of regimentation and organisation artifacts in our proposal. However, our contribution in this context is to allow some decisions that were embedded in the services to go back to the agents' layer by means of organisational agents. In ORA4MAS, artifacts encapsulate the coordination and synchronisation which were implemented in services. Control and judgement procedures are separated from these aspects and are embedded in organisational agents. Organisational agents can then use organisational artifacts to help them in deciding and eventually applying sanctions to other agents.

Besides the above mentioned advantages of the ORA4MAS approach, there are still open questions to be investigated. First, although artifacts promote an open environment for the agents, we need to define some mechanics that allows the designer (or the organisational agents) to specify some sort of control on the access to the artifacts. For example, in the current version of ORA4MAS every agent can see all the observable properties of the organisational artifacts. Some privacy problems could therefore arrive from that. All the operations are also available to all agents. Of course the operations may fail in case the agents are either not allowed to use them or are using them incorrectly. But it could be interesting to have different 'interfaces' for different agents, particularly, different interfaces for member and organisational agents. An initial work is being developed in the context of CArtAgO platform to add a kind of RBAC (Role Base Access Control) system so that the roles being played by the agents constrain their access to the artifacts. Second, an important element for the interoperability and openness of the approach is the artifact's manual. The manual is to be

used by an arriving agent to understand how the artifact can be used and, in our case, how the organisation can be 'used'. A good language to write the manual should (1) allow the developer to specify the organisational actions, (2) be easily readable and processed by the agents, and (3) have a clear semantics. Other research directions are mentioned as future work in the next section.

8 Conclusion and perspectives

In this paper, we have followed the A&A approach to give back the power to agents in an organisational approach. From this perspective, we have defined on the one hand the organisational artifacts which encapsulate the functional aspects of an organisation and organisation management, and on the other hand the organisational agents, which encapsulate the decision and reasoning side of the management of organisations.

Although we already have some initial results of the ORA4MAS project, as those presented in this paper, we have concretely evaluated the proposal for only one OML (the $\mathcal{M}$OISE$^+$). The first future work of the project will therefore be an evaluation of its application for different OMLs such as ISLANDER [11], OPERA [10], $\mathcal{M}$OISE^{Inst} [16], and AGR [13]. Following this broader application we can then better compare our approach with related works (e.g. [15]) and even others such as those managing the organisation with communication acts (e.g. RICA-J [38]) or exploiting the environment to coordinate and constrain the agents' behaviour [46].

Other extensions aim at taking advantage of the uniform concepts used to implement the environment and the organisation abstractions through the concept of artifacts. Such an homogeneous conceptual point of view will certainly help us to find a suitable solution for the second problem identified in Sect. 2 binding both concepts together in order to situate organisations in an environment or to install the access to the environment into organisational models (in the same direction as proposed by [26]). Other points of investigation are (1) the study of the reorganisation process of an MAS using the ORA4MAS approach, (2) the impact of the reorganisation on the organisational artifacts, and (3) the definition of a meta-organisation for the ORA4MAS, so that we have special roles for organisational agents that give them access to the organisational artifacts.

Acknowledgements We are grateful to the anonymous reviewers of this paper and the earlier versions of it published in COIN and EUMAS for their valuable suggestions, contributions, and motivation. We would like to thank Michele Piunti for the development of the JADEX agents and useful remarks to improve ORA4MAS. This work is strongly based on previous works developed with Jaime S. Sichman in the context of the project USP-COFECUB 98-04. Jomi F. Hübner is financed by the project ForTrust (Social Trust Analysis and Formalization) funded by the French Agence nationale de la Recherche (ANR-06-SETI-006).

References

1. Arcos, J. L., Noriega, P., Rodríguez-Aguilar, J. A., & Sierra, C. (2007). E4mas through electronic institutions. In D. Weyns, H. V. D. Parunak, & F. Michel (Eds.), *Environments for multi-agent systems III, third international workshop, E4MAS 2006, Hakodate, Japan, May 8, 2006, Selected revised and invited papers*, Vol. 4389 of *Lecture Notes in Computer Science* (pp. 184–202). Springer.
2. Bellifemine, F. L., Caire, G., & Greenwood, D. (2007). *Developing multi-agent systems with JADE*. Wiley Series in Agent Technology. Wiley.
3. Boissier, O., Hübner, J. F., & Sichman, J. S. (2007). Organization oriented programming from closed to open organizations. In G. O'Hare, M. O'Grady, O. Dikenelli, & A. Ricci (Eds.), *Engineering Societies in the Agents World VII (ESAW 06)*, Vol. 4457 of *LNCS* (pp. 86–105). Springer-Verlag.

4. Bordini, R. H., Hübner, J. F., & Wooldrige, M. (2007). *Programming multi-agent systems in agentSpeak using Jason*. Wiley Series in Agent Technology. Wiley.
5. Broersen, J., Dastani, M., Hulstijn, J., Huang, Z., & van der Torre, L. (2001). The BOID architecture: Conflicts between beliefs, obligations, intentions and desires. In J. P. Müller, E. Andre, S. Sen, & C. Frasson (Eds.), *Proceedings of the fifth international conference on autonomous agents*, Montreal, Canada (pp. 9–16). ACM Press.
6. Castelfranchi, C. (2000). Engineering social order. In A. Omicini, R. Tolksdorf, & F. Zambonelli (Eds.), *Engineering societies in the agent world, first international workshop, ESAW 2000, Berlin, Germany, August 21, 2000, revised papers*, Vol. 1972 of *Lecture Notes in Computer Science* (pp. 1–18). Springer.
7. Castelfranchi, C., Dignum, F., Jonker, C. M., & Treur, J. (2000). Deliberate normative agents: Principles and architecture. In N. R. Jennings & Y. Lespérance (Eds.), *Intelligent agents VI, Agent Theories, Architectures, and Languages (ATAL), 6th international workshop, ATAL '99, Orlando, Florida, USA, July 15–17, 1999, Proceedings*, Vol. 1757 of *LNCS* (pp. 364–378). Springer.
8. Dastani, M. (2008). 2APL: A practical agent programming language. *Autonomous Agent and Multi-Agent Systems, 16*, 241–248.
9. Dignum, V., & Dignum, F. (2001). Modeling agent societies: Co-ordination frameworks and institutions. In P. Brazdil & A. Jorge (Eds.), *Proceedings of the 10th Portuguese conference on artificial intelligence (EPIA'01)*, Berlin, LNAI 2258 (pp. 191–204). Springer.
10. Dignum, V., Vazquez-Salceda, J., & Dignum, F. (2004). OMNI: Introducing social structure, norms and ontologies into agent organizations. In R. H. Bordini, M. Dastani, J. Dix, & A. El Fallah-Seghrouchni (Eds.), *Proceedings of the programming multi-agent systems (ProMAS 2004)*, LNAI 3346, Berlin. Springer.
11. Esteva, M., de la Cruz, D., & Sierra, C. (2002). ISLANDER: An electronic institutions editor. In C. Castelfranchi & W. L. Johnson (Eds.), *Proceedings of the first international joint conference on autonomous agents and multiAgent systems (AAMAS 2002)*, LNAI 1191 (pp. 1045–1052). Springer.
12. Esteva, M., Rodríguez-Aguilar, J. A., Rosell, B., & Arcos, J. L. (2004). AMELI: An agent-based middleware for electronic institutions. In N. R. Jennings, C. Sierra, L. Sonenberg, & M. Tambe (Eds.), *Proceedings of the third international joint conference on autonomous agents and multi-agent systems (AAMAS'2004)*, New York (pp. 236–243). ACM.
13. Ferber, J., & Gutknecht, O. (1998). A meta-model for the analysis and design of organizations in multi-agents systems. In Y. Demazeau (Ed.), *Proceedings of the 3rd international conference on multi-agent systems (ICMAS'98)* (pp. 128–135). IEEE Press.
14. Fornara, N., & Colombetti, M. (2006). Specifying and enforcing norms in artificial institutions. In A. Omicini, B. Dunin-Keplicz, & J. Padget (Eds.), *Proceedings of the 4th European workshop on multi-agent systems (EUMAS 06)*.
15. García-Camino, A., Rodríguez-Aguilar, J., & Vasconcelos, W. W. (2007). A distributed architecture for norm management in multi-agent systems. In J. Sichman, P. Noriega, J. Padget, & S. Ossowski (Eds.), *Coordination, organizations, institutions, and norms in agent systems III*, Vol. 4870 of *LNAI* (pp. 275–286). Springer, Revised Selected Papers.
16. Gâteau, B., Boissier, O., Khadraoui, D., & Dubois, E. (2005). MOISEinst: An organizational model for specifying rights and duties of autonomous agents. In *Third European workshop on multi-agent systems (EUMAS 2005)*, Brussels, Belgium, December 7–8, pp. 484–485.
17. Goble, L., & Meyer, J.-J. C. (Eds.). (2006). *Proceedings of the 8th international workshop on deontic logic in computer science, DEON 2006, Utrecht, The Netherlands, July 12–14, 2006*, Vol. 4048 of *Lecture Notes in Computer Science*. Springer.
18. Grossi, D., Aldewered, H., & Dignum, F. (2007). *Ubi Lex, Ibi Poena*: Designing norm enforcement in e-institutions. In P. Noriega, J. Vázquez-Salceda, G. Boella, O. Boissier, V. Dignum, N. Fornara, & E. Matson (Eds.), *Coordination, organizations, institutions, and norms in agent systems II*, Vol. 4386 of *LNAI* (pp. 101–114). Springer, Revised Selected Papers.
19. Hübner, J. F. (2003). *Um Modelo de Reorganização de Sistemas Multiagentes*. PhD thesis, Universidade de São Paulo, Escola Politécnica.
20. Hübner, J. F., Boissier, O., & Vercouter, L. (2008). Instrumenting multi-agent organisations with reputation artifacts. In V. Dignum & E. Matson (Eds.), *Proceedings of coordination, organizations, institutions and norms (COIN@AAAI), held with AAAI 2008, Chicago, EUA* (pp. 17–24). AAAI Press.
21. Hübner, J. F., Sichman, J. S., & Boissier, O. (2006). S-MOISE+: A middleware for developing organised multi-agent systems. In O. Boissier, V. Dignum, E. Matson, & J. S. Sichman (Eds.), *Coordination, organizations, institutions, and norms in multi-agent systems*, Vol. 3913 of *LNCS* (pp. 64–78). Springer.
22. Hübner, J. F., Sichman, J. S., & Boissier, O. (2007). Developing organised multi-agent systems using the MOISE+ model: Programming issues at the system and agent levels. *International Journal of Agent-Oriented Software Engineering, 1*(3/4), 370–395.

23. IST Advisory Group. (2003). Ambient intelligence: From vision to reality. Technical report, Information Societies Technologies. ftp://ftp.cordis.europa.eu/pub/ist/docs/istag-ist2003_consolidated_report.pdf.
24. Luck, M., McBurney, P., Shehory, O., & Willmott, S. (2005). *Agent technology: Computing as interaction (A roadmap for agent based computing)*. AgentLink. http://www.agentlink.org/roadmap
25. Nardi, B. A. (1996). *Context and consciousness: Activity theory and human–computer interaction*. MIT Press.
26. Okuyama, F. Y., Bordini, R. H., & da Rocha Costa, A. C. (2007). Spatially distributed normative objects. In P. Noriega, J. Vázquez-Salceda, G. Boella, O. Boissier, V. Dignum, N. Fornara, & E. Matson (Eds.), *Coordination, organizations, institutions, and norms in agent systems II*, Vol. 4386 of *LNAI*, pp. 133–146.
27. Okuyama, F. Y., Bordini, R. H., & da Rocha Costa, A. C. (2008). A distributed normative infrastructure for situated multi-agent organisations. In L. Padgham, D. C. Parkes, J. Müller, & S. Parsons (Eds.), *Proceedings of 7th international conference on autonomous agents and multiagent systems (AAMAS 2008)*, May, 12–16, 2008, Estoril, Portugal, pp. 1501–1504.
28. Omicini, A., Ricci, A., & Viroli, M. (2008). Artifacts in the A&A meta-model for multi-agent systems. *Autonomous Agents and Multi-Agent Systems, 17*(3), 432–456.
29. Omicini, A., Ricci, A., Viroli, M., Castelfranchi, C., & Tummolini, L. (2004). Coordination artifacts: Environment-based coordination for intelligent agents. In *AAMAS'04*, Vol. 1, New York, USA, 19–23 July 2004 (pp. 286–293). ACM.
30. Piunti, M., Ricci, A., Braubach, L., & Pokahr, A. (2008). Goal-directed interactions in artifact-based mas: Jadex agents playing in cartago environments. In *IEEE/WIC/ACM conferences on web intelligence and intelligent agent technology (IAT-2008)*. IEEE/WIC/ACM.
31. Pokahr, A., Braubach, L., & Lamersdorf, W. (2005). Jadex: A BDI reasoning engine. In R. H. Bordini, M. Dastani, J. Dix, & A. El Fallah Seghrouchni (Eds.), *Multi-agent programming: Languages, platforms, and applications*, number 15 in Multiagent Systems, Artificial Societies, and Simulated Organizations, Chap. 6 (pp. 149–174). Springer.
32. Pynadath, D. V., & Tambe, M. (2003). An automated teamwork infrastructure for heterogeneous software agents and humans. *Autonomous Agents and Multi-Agent Systems, 7*(1–2), 71–100.
33. Ricci, A., Piunti, M., Acay, L. D., Bordini, R. H., Hübner, J. F., & Dastani, M. (2008). Integrating heterogeneous agent programming platforms within artifact-based environments. In L. Padgham, D. C. Parkes, J. Müller, & S. Parsons (Eds.), *7th International joint conference on autonomous agents and multiagent systems (AAMAS 2008)*, Estoril, Portugal, May 12–16, 2008 (pp. 225–232). IFAAMAS.
34. Ricci, A., Viroli, M., & Omicini, A. (2006). CArtAgO: A framework for prototyping artifact-based environments in MAS. In D. Weyns, H. V. D. Parunak, & F. Michel (Eds.), *Environments for multiAgent systems III*, Vol. 4389 of *LNAI* (pp. 67–86). Springer. *3rd International workshop (E4MAS 2006)*, Hakodate, Japan. Selected Revised and Invited Papers.
35. Ricci, A., Viroli, M., & Omicini, A. (2007). A general purpose programming model & technology for developing working environments in MAS. In M. Dastani, A. El Fallah Seghrouchni, A. Ricci, & M. Winikoff (Eds.), *5th International workshop "Programming multi-agent systems" (PROMAS 2007)*, AAMAS 2007, Honolulu, Hawaii, USA, pp. 54–69.
36. Ricci, A., Viroli, M., & Omicini, A. (2008). The A&A programming model & technology for developing agent environments in MAS. In M. Dastani, A. E. Fallah-Seghrouchni, A. Ricci, & M. Winikoff (Eds.), *Programming multi-agent systems, 5th international workshop, proMAS 2007, Honolulu, HI, USA, May 15, 2007, Revised and Invited Papers*, Vol. 4908 of *LNCS*, pp. 89–106. Springer.
37. Sairamesh, J., Lee, A., & Anania, L. (2004). Introduction of the special issue on information cities. *Communications of the ACM, 47*(2), 28–31.
38. Serrano, J. M., & Ossowski, S. (2007). A compositional framework for the specification of interaction protocols in multiagent organizations. *Web Intelligence and Agent Systems, 5*(2), 197–214.
39. Sichman, J., Noriega, P., Padget, J., & Ossowski, S. (Eds.). (2008). *Coordination, organizations, institutions, and norms in agent systems III*, Vol. 4870 of *LNCS*. Springer.
40. Tuomela, R., & Bonnevier-Tuomela, M. (1995). Norms and agreement. *European Journal of Law, Philosophy and Computer Science, 5*, 41–46.
41. Vázquez-Salceda, J., Aldewereld, H., & Dignum, F. (2004). Norms in multiagent systems: Some implementation guidelines. In *Proceedings of the second European workshop on multi-agent systems (EUMAS 2004)*. http://people.cs.uu.nl/dignum/papers/eumas04.PDF
42. Viroli, M., Holvoet, T., Ricci, A., Schelfthout, K., & Zambonelli, F. (2007). Infrastructures for the environment of multiagent systems. *Autonomous Agents and Multi-Agent Systems, 14*(1), 49–60.
43. Viroli, M., Ricci, A., & Omicini, A. (2006). Operating instructions for intelligent agent coordination. *The Knowledge Engineering Review, 21*(1), 49–69.

44. Weyns, D., Omicini, A., & Odell, J. J. (2007). Environment as a first-class abstraction in multi-agent systems. *Autonomous Agents and Multi-Agent Systems, 14*(1), 5–30. Special Issue on Environments for Multi-agent Systems.
45. Weyns, D., & Parunak, H. V. D. (Eds.). (2007). *Journal of Autonomous Agents and Multi-Agent Systems. Special Issue on Environments for Multi-Agent Systems, 14*(1). Netherlands: Springer.
46. Weyns, D., Parunak, H. V. D., & Michel, F. (Eds.). (2006). *Environments for multi-agent systems II, second international workshop, E4MAS 2005*, Utrecht, The Netherlands, July 25, 2005, Selected Revised and Invited Papers, Vol. 3830 of *Lecture Notes in Computer Science*. Springer.