Organisational Artifacts and Agents For Open Multi-Agent Organisations: "Giving the power back to the agents"

Rosine Kitio[1], Olivier Boissier[1] *, Jomi Fred Hübner[2], and Alessandro Ricci[3]

[1] SMA/G2I/ENSM.SE, 158 Cours Fauriel
42023 Saint-Etienne Cedex, France
{kitio,boissier}@emse.fr

[2] GIA/DSC/FURB, Braz Wanka, 238
89035-160, Blumenau, Brazil
jomi@inf.furb.br

[3] DEIS, ALMA MATER STUDIORUM Università di Bologna
47023 Cesena (FC), Italy
a.ricci@unibo.it

Abstract. The social and organisational aspects of agency have have become nowadays a major focus of interest in the MAS community, and a good amount of theoretical work is available, in terms of formal models and theories. However, the conception and engineering of proper organisational infrastructures embodying such models and theories is still an open issue, in particular when open MAS are considered. Accordingly, in this paper we discuss a model for an organisational infrastructure called ORA4MAS that aims at addressing these issues. By being based on the A&A (Agents and Artifacts) meta-model, the key and novel aspect introduced with ORA4MAS is that organisations and the organisation infrastructure itself are conceived in terms of agents and artifacts, as first-class basic abstractions giving body to the MAS from design to runtime. This is in analogy with human organisation and organisation infrastructures, that are populated by humans (as participants and part of the organisation machinery), and by rich sets of artifacts and tools that humans use to support their activities inside the organisation and the organisation itself, encapsulating essential infrastructure services.

Keywords: Multi-agent Systems, MAS organisations, Open systems, Artifacts.

1 Introduction

Nowadays, current applications of IT show the interweaving of both human and technological *communities* (e.g. pervasive computing and ambient intelligence [15]), resulting in the construction of *connected* communities (ICities [30])

* Partially supported by USP-COFECUB.

in which software entities act on behalf of users and cooperate with infohabitants, taking into account issues like trust, security, flexibility, adaptation and *openness*. As shown in [20], current applications have led to an increase in number of agents, in the duration and repetitiveness of their activities, with a decision and action perimeter still enlarging. Moreover the number of agents' *designers* is also increasing, leading to a huge palette of heterogeneity in these systems. The complex system engineering's approach needed to build such applications highlights and stresses requirements on *openness* in terms of ability to take into account several kinds of changes and to adapt the system configuration while it keeps running.

In this paper, we are interested in social and organisational aspects of agency. They have always been a topic of study in the multiagent domain since the seminal work of [12, 7] and have become a major focus of interest in the MAS community (e.g. [21, 4]). However, most designers have doubts about how to put these concepts in practice, i.e., how to program them, while both addressing the openness and scalability issues and keeping agent's autonomy and decentralization which are essential features of MAS. Addressing the requirements stated above at the organisation level leads to a shift of design and programming paradigm. We denote it as a shift from closed to *open* organisations introducing the need for agents' systems that: (i) allow agents to arrive/leave dynamically into/from it, (ii) permit the dynamic change of both agents' individual behaviors in response to evolving environmental constraints and agents' *social* or *collective* behaviors due to changes of the systems' goals, and (iii) move from off-line to *on-line* adaptation, in the sense that designers should be replaced by the *agents* themselves to control and to realize the above issues.

Since it is a huge and complex work to develop systems with this kind of openness, in this paper we propose an organisational infrastructure referred as ORA4MAS which is meant to provide a conceptual and architectural step towards the simplification of this problem. Our proposal is based on the A&A approach where instead of a lot of different components and concepts (e.g., agents, services, proxies, objects, ...), only two types of entities are involved: agents and artifacts. Roughly, while agents model the decisions of the system, the artifacts model its functions. We especially demonstrate this approach showing how the organisational aspect of the MAS can be conceived and designed by only *organisational agents* and *organisational artifacts*. This is in analogy with human organisation and organisation infrastructures, that are populated by humans (as participants and part of the organisation machinery), and by rich sets of artifacts and tools that humans use to support their activities inside the organisation and the organisation itself, encapsulating essential infrastructure services.

In the first part of the paper (cf. sec. 2), we will have a look at the different approaches that have been developed in the field of multi-agent organisation, stressing what limitations we consider. This is complemented by a look at what has been done in the other dimensions of a MAS, i.e. environment and interaction. Then, we present the basic concepts underlying ORA4MAS infrastructure (cf. sec. 3), and we briefly describe the shapes of the organisational artifacts

devised in ORA4MAS reifying the $\mathcal{M}$OISE$^+$ organisational model (cf. sec. 4). Finally, we provide concluding remarks and perspectives for the work in (cf. sec. 5)

2 Background

The recent developments in MAS domain, belonging to what we call *Organisation Oriented Programming (OOP)* [3], have provided many proposals of organisation-oriented middleware. In the different approaches related to OOP, we distinguish two important components: an declarative *organisation modeling language* (OML) and an *Organisation Implementation Architecture* (OIA). The OML *specifies* the organisation(s) of a MAS. It is used to collect and express specific constraints and cooperation patterns imposed on the agents by the designer (or the agents), resulting in an explicit representation that we call *organisation specification* (OS). A collective entity, called *Organisation Entity* (OE), instantiates this OS by assigning agents to roles. The OIA will then help these agents to properly "play" their roles as they are specified in the OS.

The OIA considers both an agent centered and a system centered point of view [4]. In the former, the focus lies on how to develop different *agent reasoning mechanisms* to interpret and reason on the OS and OE applying on the agents (e.g. [5, 6]). In the latter, the main concern is how to develop an infrastructure, that we call Organisation Infrastructure (OI), that ensures 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 system may have unknown architectures. Of course, to develop the overall MAS, the former point of view is necessary since the agents probably need to have access to an organisational representation that enable them to reason about it.

The implementation of OI normally follows a common trend in multiagent platforms like JADE [2] and FIPA-OS [1]. These platforms have demonstrated the requirement and utility of the notion of "infrastructure" for MAS development [13]. Not only have they supported the implementation of the agents, but are being noticed as a provider of fundamental global generic services going further of only directory facilitator, agent management system or agent communications by also addressing coordination [24]. Therefore, agents related to the application domain operate on top of a middleware layer.

As shown in [3], many implementations of the OI follow the general layered architecture depicted in Fig. 1: (i) domain (or application) agents, responsible to achieve organisational goals and use an *organisational proxy* component to interact with the organisation, (ii) *organisational layer*, responsible to bind all agents in a coherent OS and OE and provides some services for them, and (iii) communication layer for connecting all components of the infrastructure

[4] We prefer here system-centred to organisation-centred in order to avoid confusion even if, as we have seen, the organisation is reified in OE. Let's notice that in [33] these points of view are called agent and institutional perspectives.

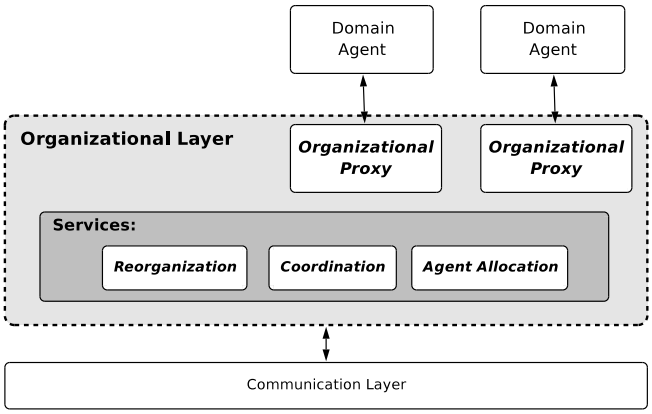

Fig. 1. Common Organisation Infrastructure (OI) for open MAS

in a distributed and heterogeneous applications. This layered structure 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 layer and OE management), and communication designers.

From the study of the different works considering this organisation layer, we can identify a set of specialized services and proxies (e.g., angels [8], governors [10], managers [18]). In order to stress their ability to manage organisational concepts and to develop dedicated reasoning/processing abilities on the organisation, let's call them respectively *organisational services* (OrgServices). One important point to notice is that all the access to the OE by the agents is mediated by these OrgServices in the OI.

This brief general introduction of OI designs allow us to point out some drawbacks:

1. In some proposals, like $\mathcal{S}$-$\mathcal{M}$OISE$^+$ [18], OrgServices are implemented as agents. The problem is that, conceptually, services are not in the same abstraction level as agents.
2. In the proposals where OrgServices are not agents, whenever an application designer needs to customise some decisions of the system in the organisational dimension (e.g., a sanction system, a reorganisation strategy, the allocation of agents to roles), s/he has to develop/change an OrgService. It can be quite confusing to deal with both OrgServices and agents concepts while developing a system. It will be better to always use the same abstraction level when modelling and implementing the decision aspect of system.
3. The designer (and the agents) also have to deal with two kinds of environments: a virtual organisational environment (where the agents adopt roles, send messages) and the real environment (where the agents act). An unified view of the environment simplifies the concept of agent interaction.

4. In the general architecture of Fig. 1, the middleware has too much power. Most of the organisational "decisions" are performed at this layer. It is more suitable if the agents make decisions and not the OrgServices. For example, if some agent wants to perform some action or send a message that its organisation does not allow, it can not do it since the middleware (and its organisational proxy) will detect this violation tentative and deny it. The middleware is thus performing two functions: detection and decision. In some cases agents operating on the application layer should get their control power back in the sense that they could play some of the roles of the OrgServices. As another example, reorganisation requires that *agents* should be able to manage and access the creation of new organisations.
   The problems of existing approaches of organisations are consequence of some properties of the organisational managers design: (i) the enforcement of organisational functions and constraints and (ii) the inclusion of reasoning and decision aspects that can be managed by agents and thus should be in the agent layer.

It's worth noting that the issues stated in this section do not concern solely the implementation level, but also the conceptual and theoretical level: what is the nature of OrgServices in MAS where only agents are considered as first-class entities?

3 An Organisational Infrastructures based on Agents and Artifacts

The proposal presented in this paper draws its inspiration from human organisation infrastructures. Human organisation and organisation infrastructures, that are populated by humans (as participants and part of the organisation machinery), and by rich sets of artifacts and tools that humans use to support their activities inside the organisation and the organisation itself, encapsulating essential infrastructure services. According to psycho-sociological theories and studies such as Activity Theory and Distributed Cognition [22]—recently adopted in computer science fields such as CSCW, HCI and MAS [32, 31, 27]—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 the organisation.

In particular, some of these artifacts—that we call here *organisational artifacts*—appear to be vital for supporting the coordination of organisation 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), by controlling access to resources and enforcing norms (e.g. the badge used by members in a computer science department to access certains rooms or use some other artifacts, such as copiers). Human societies and organisations continuously improve

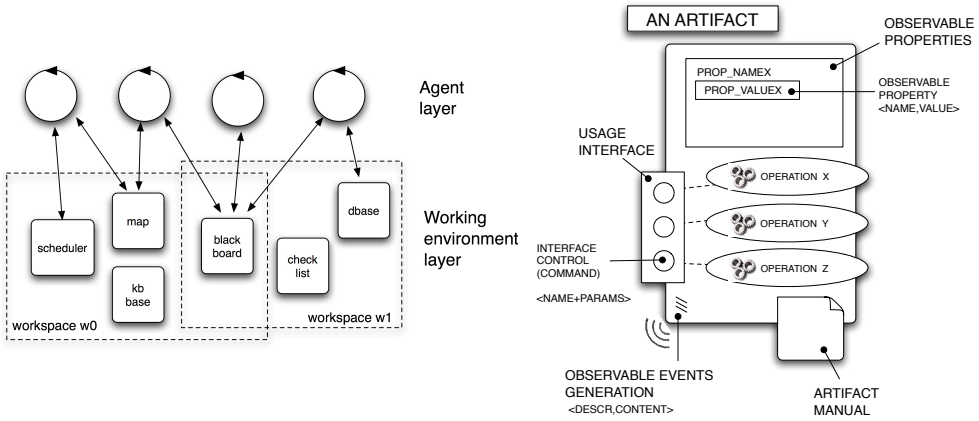

Fig. 2. *(Left)* abstract representation of workspaces, populated by agents—represented by circles—and artifacts—represented by squares. *(Right)* A representation of the main parts and properties of an artifact, with in evidence the usage interface, the observable properties and the manual.

their experience in designing artifacts more and more effective 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.

Analogously, here we propose an organisational infrastructure called ORA4MAS in which both organisations and the organisation infrastructure itself are conceived and engineered in terms of a set of agents and *artifacts*. ORA4MAS in particular exploits the notion of artifact as introduced by the A&A meta-model [28] to encapsulate as first-class abstractions from design to runtime structures, and related functionalities / rules as defined theoretically by the $\mathcal{M}$OISE$^+$ organisational model [17].

In the remainder of the section, first we recall the basic ideas provided by the A&A meta-model, and then describe how such concepts are exploited to shape the ORA4MAS infrastructure.

3.1 The Notion of Artifacts in MAS

The notion of *MAS environment*, as remarked by recent literatures, has gained a key role in the recent past, becoming a *mediating* entity, functioning as enabler but possibly also as a manager and constrainer of agent actions, perceptions, and interactions (see here [34] for comprehensive surveys). According to such a perspective, the environment is not a merely passive source of agent perceptions and target of agent actions—which is, actually, the dominant perspective in agency and in MAS—, but a first-class abstraction that can be suitably designed to *encapsulate* some fundamental functionalities and services, supporting MAS dimensions such as coordination and organisation, besides agent mobility, communications, security, etc.

Among the various approaches, the A&A in particular introduces a notion of *working environment*, representing such a part of the MAS explicitly designed on the one hand by MAS engineers to provide various kinds of functionality—including MAS coordination, organisation—and perceived as first-class entity on the other hand by agents of the MAS [29, 28, 25]. By drawing its inspiration from human society and theories such as Activity Theory and Distributed Cognition, A&A working environment are made of *artifacts*, introduced as first-class abstraction representing function-oriented dynamic entities and tools that agents can create and use to perform their individual and social activities.

Artifacts can be considered as a complimentary abstraction to agent populating a MAS: while agents are goal-oriented pro-active entities, artifacts are a general abstraction to model function-oriented passive entities, designed by MAS designers to encapsulate some kind of functionality, by representing (or wrapping existing) resources or instruments mediating agent activities. Passive here means that—differently from the agent case—they do not encapsulate any thread of control.

As artifacts and tools in human societies play a key role in mediating any not naive social activities, analogously artifacts in MAS are meant to play an important role in supporting the activities performed inside MAS organisations. Among the others, *coordination artifacts* have been introduced as an important class of artifacts for MAS organisations [26], as artifacts mediating agent interactions and encapsulating some kind of coordinating functionality— whiteboards, event services, shared task schedulers are examples.

Figure 2 shows an abstract representation of an artifact as defined in the A&A meta-model, exhibiting analogous parts and properties of artifacts as found in human society. The artifact *function*—and related artifact behaviour—is partitioned in a set of *operations*, which agents can trigger by acting on 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: let's consider a coffee machine, for a simple but effective analogy. The set of buttons of the coffee machines 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.

Analogously to the human case, in A&A each artifact type can be equipped by the artifact programmer with a *manual* composed essentially by the *function description*—as the formal description of the purpose intended by the designer—, the *usage interface description*—as the formal description of artifact usage in-

terface and observable states—, and finally the *operating instructions*—as the formal description of how to properly use the artifact so as to exploit its functionalities. Such a manual is meant to be essential for creating open systems 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 they have not pre-programmed by MAS programmers for the purpose.

Finally, artifacts can be composed together by means of *link interface*s, that are sets of input / output ports that can be (dynamically) linked together by means of suitable channels and through which artifacts can exchange data. Link interfaces serve two purposes: on the one side, to explicitly define a principle of composability for artifacts, enabling the ruled construction of articulated and complex artifacts by means of simpler ones; on the other side, to realise distributed (composed) artifacts: channels can connect link interfaces of artifacts possibly belonging to different workspaces.

3.2 ORA4MAS Infrastructure: The Basic Idea

The basic idea in ORA4MAS is to engineer the organisational infrastructure—and the organisations living upon it—in terms of agents and artifacts, following the basic A&A metamodel.

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, and that are responsible of activities and encapsulate functionalities concerning the management and enacting of the organisation. In particular, organisation agents—analogously to managers and administrators in human organisation—are responsible of management activities inside the organisation, concerning observing, monitoring, and reasoning about organisation dynamics, etc. Such activities take place almost by creating and managing organisational artifacts that are then used by member agents of the organisation. Organisation artifacts are those artifacts that agents of an organisation may want or have to use in order to participate in organisation activities and access to organisation resources, encapsulating organisation rules and functionalities, such as enabling and mediating (ruling) agent interaction, tracing and ruling resource access, and so on. 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.

Even from this abstract characterisation, it is possible to identify some general properties that are of some importance to face the drawbacks listed at the end of Section 2:

- *Abstraction & encapsulation*—By using agents and artifacts to reify both the organisation and the organisation infrastructure—from design to runtime—, we raise the level of abstraction with respect to approaches in which organisation mechanisms are hidden at the implementation / middleware level. 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.

- *"Power back to agents"*—Decision functionalities that were embedded in the OrgServices in the OI go back to the agents' layer in organisational agents. Agents are autonomous with respect to decision of using or not a specific artifact—including the organisational artifacts—and keeps its autonomy—in terms of control of its actions—while using 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 co-using the same 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 functioning can take place without compromising the autonomy of the agents, who are fully encapsulating their control.
- *Distributed management*—Distributing the management of the organisation into different organisational artifacts installs a distributed coordination (meaning here more particularily synchronization) 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 decentralized into different loci of decision with a loosely coupled set of agents.
- *Openness*—Organisational artifacts can be created and added dynamically according to the need. They have a proper semantics description of both the functionalities and operating instructions, so conceptually agents can discover at runtime how to use them in the best way. Related to openness, the approach promotes heterogeneity of agent (societies): artifacts can be used by heterogeneous kinds of 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.
- *Re-organisation and autonomic-properties*—The basic properties of organisational agents and artifacts can be effective in devising scenarios in which the MAS supports forms of self-organisation (and configuration, healing, protection). On the one side we have organisation artifacts that are inspectable—in terms of their manual (static) and observable state (dynamic)—and *malleable*, i.e. they can be designed so as to be manipulated and adapted at runtime. On the other side we have organisation agents that can have suitable reasoning capabilities so as to observe, reason and manipulate organisation artifacts according to the needs. This is particularly important when organisational artifacts mediating the interaction of groups of agents are considered: by observing and changing the mediating behaviour of such kinds of artifacts, agents are able to change the collective behaviour of overall groups of agents. In other words, here the re-organisation process

is modelled as an organisation process itself, in the same vein as proposed in [16].

After sketching the basic concepts underlying the ORA4MAS approach, in next section we finally describe how a full-fledged organisational model—$\mathcal{M}$OISE$^+$ in this case—can be abstractly implemented on top of agents and artifacts.

4 Shaping ORA4MAS Artifacts Upon $\mathcal{M}$oise$^+$

4.1 The $\mathcal{M}$oise$^+$ Model

$\mathcal{M}$OISE$^+$ (Model of Organisation for multI-agent SystEms) [17] is an OML that explicitly decomposes the organisation into structural, functional, and deontic dimensions. The structural dimension defines 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 describes 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 where the leafs-goals can by achieved individually by the agents. The deontic dimension is added in order to binds the structural dimension with the functional one by the specification of the roles' *permissions* and *obligations* for missions. Instead of being related to the agents' behavior space (what they can do), the deontic dimension is related to the agents' autonomy (what they should do).

$\mathcal{S}$-$\mathcal{M}$OISE$^+$ is an open source implementation of an OI that supports the $\mathcal{M}$OISE$^+$ OML [18]. The organisational proxy is called *OrgBox* and it consists of an API that agents use to access the OrgServices, provided by a special systen agent called *OrgManager*. The OrgManager receives and manages all the messages from the agents' OrgBox asking for changes in the OE state (e.g. role adoption, group creation, mission commitment). Those changes bring about the OrgManager to modify the OE only if they do not violate any organisational constraint. For instance, when some agent asks for a role adoption in a group g, the OrgManager ensures that: (1) the role belongs to a specified group g; (2) the number of players in g is lesser or equals than the maximum number of players defined in the group's compositional specification; (3) each role ρ_i that the agent already plays is specified as compatible with the new role in g. Besides the organisational compliance, the OrgManager also provides useful information for the agents' organisational reasoning and coordination, for example the missions they are forced to commit to and goals it can pursue.

An important feature of this architecture is that the OS may be interpreted at run-time by the agents because its specification is available to them. Thus agents can be developed as hardwired programmed for some particular OS or they can be programmed to interpret the current available OS. This last feature is not only useful in open systems, but also when one considers a reorganisation process, since the adaptation of the new OS may be dynamic. $\mathcal{S}$-$\mathcal{M}$OISE$^+$ does

not require any specific type of agent architecture. Although agents normally use the OrgBox to interact with the system, an agent could even interact with the OrgManager directly using KQML.

4.2 Organisational Agents and Artifacts based on $\mathcal{M}$oise$^+$

We exploit here the $\mathcal{M}$OISE$^+$ model to identify and shape a basic set of organisational artifacts (kind) and agents that constitute the basic infrastructure building blocks of ORA4MAS, being a sort of "reification" of the SS, FS, DS specifications (see figure 3). This basic set accounts for:

- an OrgBoard artifact—used to keep track of the structure of organisation in the overall;
- a GroupBoard artifact—used to manage the life-cycle of a specific group;
- a SchemeBoard type—used to support and manage the execution of a social scheme.

Here we consider just a core set, skipping most details that would make heavy the overall understanding of the approach: the interested reader is forwarded on this technical report [19] to get further details.

In the following we briefly describe the basic characteristics of these kinds of artifact. In the description, the operations (commands) enlisted in artifact usage interface are abstractly described by a name with input parameters, followed (optionally) by a set of the observable events possibly generated by the operation execution (only events significant for artifact specific functionalities are considered, skipping those generated by default by the artifact). Observable properties are represented just by a name, which corresponds to the name of the property.

OrgBoard **Artifacts.** A simple abstract model for the OrgBoard artifact is depicted in figure 3 (left). The usage interface is composed by operations to:

- enter the organisation: enterOrg;
- leave the organisation: leaveOrg;
- register / de-register a new group: registerGroup(G,GB), removeGroup(G)—where G is an identifier for a group and GB is the identifier of the related group board artifact;
- register / de-register a new scheme: registerScheme(S,SB), removeScheme(S) where S is the identifier for a schema and SB is the identifier of the related scheme board.

Among the observable properties:

- list of current groups: current-groups property;
- list of current schemes: current-schemes property;
- organisation specification (including SS, FS, DS): org-spec property.

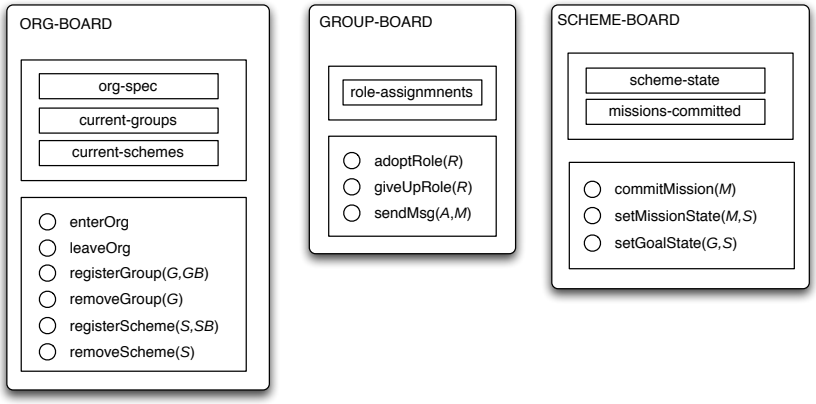

Fig. 3. Basic kinds of artifacts in ORA4MAS, with in evidence their usage interface, including operations and observable properties

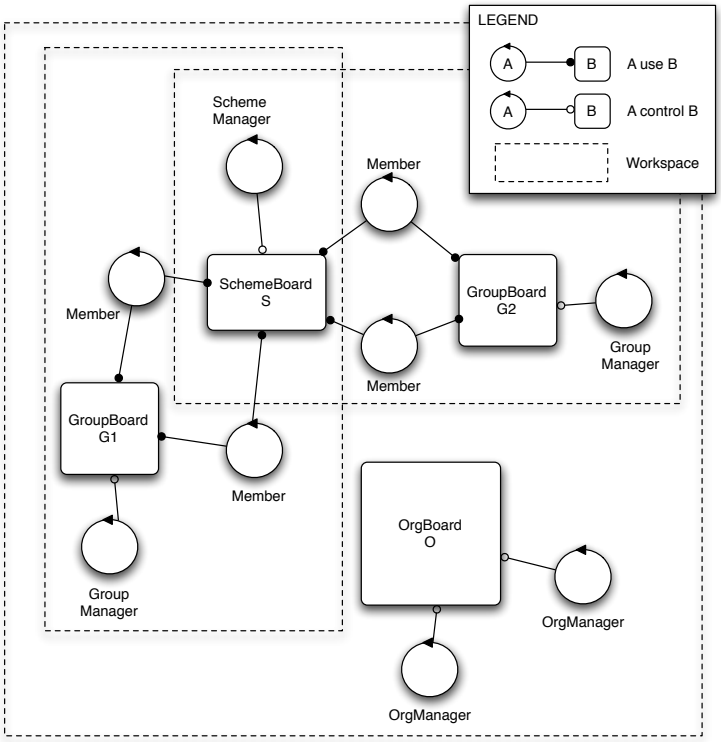

Fig. 4. A simple example of instance of an $\mathcal{M}$OISE$^+$ organisation with 2 groups and a running social scheme, with in evidence the artifacts used

Generally speaking, the observable properties of the artifact make it possible—for agents observing an OrgBoard —to monitor and be aware of which agents are actually participating to the organisation, and which are the schemes and groups instantiated. Also, this artifact can be inspected to know which are the SS, FS, DS currently adopted in the organisation.

GroupBoard **Artifacts.** The GroupBoard artifact type (see figure 3, center) is instantiated upon a specific instance of SS and DS, and provides functionalities to manage a group in terms of set of available roles and agents participation, according to the specific structure and strategy specified in SS and DS. For what concerns the DS, the artifact enforces those deontic rules that entail permissions and obligations in role adoptions and give up for the agents.

The usage interface accounts for the following operations:

- adopt a new role: adoptRole(R):{role_adoption_ok,role_adoption_failed}, where R is the identifier for a role;
- give up a role: giveUpRole(R):{role_giveup_ok,role_giveup_failed};
- sending a message to a specific agent or all the agents part of the group: sendMsg(A,M), sendMsg(M), where A is the identifier for the receiver agent, m is the message content.

Among the observable properties:

- role assignments: role-assignments property.

By observing a GroupBoard artifact, an agent can monitor and be aware of the role-agent assignments inside the group.

SchemeBoard **Artifacts.** The SchemeBoard artifact type (see figure 3, right) is instantiated upon a specific instance of FS and DS, and provides functionalities to manage the execution of a social scheme, coordinating the commitments to missions and goals, and their interaction. It function as a coordination artifact, automating the management of the dependencies between the missions and the goals as described by the social scheme, and embedding such part of the deontic specification concerning permissions and obligations for agents to commit to missions. The usage interface provides commands to:

- commit to a mission: commitMission(M):{commit_ok, commit_failed}, where M is the identifier for a mission;
- set mission state: setMissionState(M,S), where M is the identifier for a mission and S can be either *completed* or *failed*;
- set goal state: setGoalState(G,S), where G is the identifier for a goal and S can be either *satisfied* or *impossible*.

Among the observable properties:

- scheme dynamic state: scheme-state property, that includes all the goals of the scheme and their state;

– list of the current missions committed: missions-committed property.

By observing a SchemeBoard artifact, an agent can monitor then the overall dynamics concerning the scheme execution, and the be aware of which missions are assigned to which agents.

Organisational Agents. The organisational agents introduced are essentially managers responsible to create and manage the organisational artifacts described previously. Such activities typically include observing artifacts dynamics and possibly intervening, by changing / adapting artifacts or interacting directly with agents, so as to improve the overall (or specific) organisation processes or taking some kinds of decisions when detecting violations. As an example, one or multiple *scheme managers* agents can be introduced, responsible of monitoring the dynamics of the execution of a specific running scheme by observing a specific SchemeBoard instance. The SchemeBoard artifact and scheme manager agents can be designed so as 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 in charge of the manager agent.

4.3 Towards a Concrete Architecture

ORA4MAS concrete architecture is realised on top of CARTAGO infrastructure, embedding algorithms used in $\mathcal{S}$-$\mathcal{M}$OISE$^+$. CARTAGO (Common ARtifact Infrastructure for AGent Open environment) is a MAS infrastructure based on the A&A meta-model, providing the capability to define new artifacts types, suitable API for agents to work with artifacts and workspaces, and a runtime supporting the existence and dynamic management of working environments. CARTAGO is meant to be integrated with existing cognitive MAS architectures and models / languages / platforms, so as to extend them to create and work with artifact-based environments. A first example of integration with the *Jason* agent programming platform is briefly described here [28]. CARTAGO technology is based on Java and are available as open-source projects freely downloadable from the project web sites[5].

The engineering of the first prototype of the ORA4MAS infrastructure upon CARTAGO is still a work in progress.

5 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

[5] `http://www.alice.unibo.it/cartago`

the organisational agents, which encapsulated the decision and reasoning side of the management of organisations. We have thus designed the ORA4MAS model. With this proposal, we provide a decentralized management of an organisational entity.

Extensions to this work includes the instantiation of OrgArts to different OMLs such as ISLANDER [9] or $\mathcal{M}$OISE^{Inst} [14], or those like AGR [11]. Other extensions aim at taking benefit 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 bind both concepts together in order to situate organisations in environment or to install the access to the environment into organisational models (in the same direction as proposed in [23]). Another point of investigation is 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.

References

1. FIPA-OS. Technical report, Nortel Networks, 2000. (`http://www.nortelnetworks.com/products/announcements/fipa/`).
2. F. Bellifemine, F. Bergenti, G. Caire, and A. Poggi. JADE – a java agent development framework. In R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors, *Multi-Agent Programming: Languages, Platforms, and Applications*, number 15 in Multiagent Systems, Artificial Societies, and Simulated Organizations, chapter 5. Springer, 2005.
3. O. Boissier, J. F. Hübner, and J. S. Sichman. Organization oriented programming from closed to open organizations. In G. O'Hare, M. O'Grady, O. Dikenelli, and A. Ricci, editors, *Engineering Societies in the Agents World VII*, volume 4457 of *LNCS*. Springer-Verlag, 2007.
4. O. Boissier, J. Padget, V. Dignum, G. Lindemann, E. Matson, S. Ossowski, J. Sichman, and J. Vázquez-Salceda, editors. *Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems*, volume 3913 of *Lecture Notes in Artificial Intelligence*. Springer Verlag, 2006. AAMAS 2005 International Workshops on Agents, Norms, and Institutions for Regulated Multiagent Systems, ANIREM 2005 and on Organizations in Multi-Agent Systems, OOOP 2005, Utrecht, The Netherlands, July 25-26, 2005, Revised Selected Papers.
5. J. Broersen, M. Dastani, J. Hulstijn, Z. Huang, and L. der van Torre. The BOID architecture: conflicts between beliefs, obligations, intentions and desires. In J. P. Müller, E. Andre, S. Sen, and C. Frasson, editors, *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 9–16, Montreal, Canada, 2001. ACM Press.
6. C. Castelfranchi, F. Dignum, C. M. Jonker, and J. Treur. Deliberate normative agents: Principles and architecture. In *Proceedings of The Sixth International Workshop on Agent Theories, Architectures, and Languages (ATAL-99)*, 1999.
7. D. D. Corkill. *A Frameworkfor Organizational Self-Design in Distributed Problem Solving Networks*. PhD thesis, University of Massachusetts, Amherst, 1983.
8. V. Dignum, J. Vazquez-Salceda, and F. Dignum. OMNI: Introducing social structure, norms and ontologies into agent organizations. In R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah-Seghrouchni, editors, *Proceeding of the Programming Multi-Agent Systems (ProMAS 2004)*, LNAI 3346, Berlin, 2004. Springer.

9. M. Esteva, J. A. Rodriguez-Aguiar, C. Sierra, P. Garcia, and J. L. Arcos. On the formal specification of electronic institutions. In F. Dignum and C. Sierra, editors, *Proceedings of the Agent-mediated Electronic Commerce*, LNAI 1191, pages 126–147, Berlin, 2001. Springer.
10. M. Esteva, J. A. Rodríguez-Aguilar, B. Rosell, and J. L. AMELI: An agent-based middleware for electronic institutions. In N. R. Jennings, C. Sierra, L. Sonenberg, and M. Tambe, editors, *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'2004)*, pages 236–243, New York, 2004. ACM.
11. J. Ferber and O. Gutknecht. A meta-model for the analysis and design of organizations in multi-agents systems. In Y. Demazeau, editor, *Proceedings of the 3rd International Conference on Multi-Agent Systems (ICMAS'98)*, pages 128–135. IEEE Press, 1998.
12. M. S. Fox. An organizational view of distributed systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(1):70–80, Jan 1981.
13. L. Gasser. Mas infrastructure: Definitions, needs and prospects. In *Revised Papers from the International Workshop on Infrastructure for Multi-Agent Systems*, pages 1–11, London, UK, 2001. Springer-Verlag.
14. B. Gâteau, O. Boissier, D. Khadraoui, and E. Dubois. Moiseinst: An organizational model for specifying rights and duties of autonomous agents. In *Third European Workshop on Multi-Agent Systems (EUMAS 2005)*, pages 484–485, Brussels Belgium, December 7-8 2005.
15. I. A. Group. Ambient intelligence: from vision to reality. Technical report, IST, 2003. ftp://ftp.cordis.europa.eu/pub/ist/docs/istag-ist2003_consolidated_report.pdf.
16. J. F. Hübner, O. Boissier, and J. S. Sichman. Programming MAS reorganisation with MOISE+. In J. Meyer, M. Dastani, and R. Bordini, editors, *Dagstuhl Seminar on Foundations and Practice of Programming Multi-Agent Systems*, volume 06261, 2006.
17. J. F. Hübner, J. S. Sichman, and O. Boissier. A model for the structural, functional, and deontic specification of organizations in multiagent systems. In G. Bittencourt and G. L. Ramalho, editors, *Proceedings of the 16th Brazilian Symposium on Artificial Intelligence (SBIA'02)*, volume 2507 of *LNAI*, pages 118–128, Berlin, 2002. Springer.
18. J. F. Hübner, J. S. Sichman, and O. Boissier. $\mathcal{S}$-$\mathcal{M}$OISE$^+$: A middleware for developing organised multi-agent systems. In O. Boissier, V. Dignum, E. Matson, and J. S. Sichman, editors, *Proceedings of the International Workshop on Organizations in Multi-Agent Systems, from Organizations to Organization Oriented Programming in MAS (OOOP'2005)*, volume 3913 of *LNCS*. Springer, 2006.
19. R. Kitio. Organizational artifacts and agents for open multi-agent systems, Jun 2007. Master Thesis report, Available at http://www.emse.fr/~boissier/kitio.
20. M. Luck, P. McBurney, O. Shehory, and S. Willmott. *Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing)*. AgentLink, 2005.
21. Modeling Autonomous Agents in a Multi-Agent World (MAAMAW'2001). *Pre-Proceeding of the 10th European Workshop on Modeling Autonomous Agents in a Multi-Agent World (MAAMAW'2001)*, 2001.
22. B. A. Nardi. *Context and Consciousness: Activity Theory and Human-Computer Interaction*. MIT Press, 1996.
23. F. Y. Okuyama, R. H. Bordini, and A. C. da Rocha Costa. Spatially distributed normative objects. In G. Boella, O. Boissier, E. Matson, and J. Vázquez-Salceda,

editors, *Proceedings of the Workshop on Coordination, Organization, Institutions and Norms in Agent Systems (COIN), held with ECAI 2006, 28th August, Riva del Garda, Italy.*, 2006.
24. A. Omicini, S. Ossowski, and A. Ricci. Coordination infrastructures in the engineering of multiagent systems. In F. Bergenti, M.-P. Gleizes, and F. Zambonelli, editors, *Methodologies and Software Engineering for Agent Systems: The Agent-Oriented Software Engineering Handbook*, volume 11 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*, chapter 14, pages 273–296. Kluwer Academic Publishers, June 2004.
25. A. Omicini, A. Ricci, and M. Viroli. *Agens Faber*: Toward a theory of artefacts for MAS. *Electronic Notes in Theoretical Computer Sciences*, 150(3):21–36, 29 May 2006.
26. A. Omicini, A. Ricci, M. Viroli, C. Castelfranchi, and L. Tummolini. Coordination artifacts: Environment-based coordination for intelligent agents. In *AAMAS'04*, volume 1, pages 286–293, New York, USA, 19–23July 2004. ACM.
27. A. Ricci, A. Omicini, and E. Denti. Activity Theory as a framework for MAS coordination. In P. Petta, R. Tolksdorf, and F. Zambonelli, editors, *Engineering Societies in the Agents World III*, volume 2577 of *LNCS*, pages 96–110. Springer-Verlag, Apr. 2003.
28. A. Ricci, M. Viroli, and A. Omicini. A general purpose programming model & technology for developing working environments in MAS. In M. Dastani, A. El Fallah Seghrouchni, A. Ricci, and M. Winikoff, editors, *5th International Workshop "Programming Multi-Agent Systems" (PROMAS 2007)*, pages 54–69, AAMAS 2007, Honolulu, Hawaii, USA, 15 May 2007.
29. A. Ricci, M. Viroli, and A. Omicini. "Give agents their artifacts": The A&A approach for engineering working environments in MAS. In E. Durfee, M. Yokoo, M. Huhns, and O. Shehory, editors, *6th International Joint Conference "Autonomous Agents & Multi-Agent Systems" (AAMAS 2007)*, pages 601–603, Honolulu, Hawai'i, USA, 14–18 May 2007. IFAAMAS.
30. J. Sairamesh, A. Lee, and L. Anania. Introduction. *Commun. ACM*, 47(2):28–31, 2004.
31. K. Schmidt and C. Simone. Coordination mechanisms: Towards a conceptual foundation of CSCW systems design. *International Journal of Computer Supported Cooperative Work (CSCW)*, 5(2–3):155–200, 1996.
32. T. Susi and T. Ziemke. Social cognition, artefacts, and stigmergy: A comparative analysis of theoretical frameworks for the understanding of artefact-mediated collaborative activity. *Cognitive Systems Research*, 2(4):273–290, Dec. 2001.
33. J. Vázquez-Salceda, H. Aldewereld, and F. Dignum. Norms in multiagent systems: some implementation guidelines. In *Proceedings of the Second European Workshop on Multi-Agent Systems (EUMAS 2004)*, 2004.
34. D. Weyns and H. V. D. Parunak, editors. *Journal of Autonomous Agents and Multi-Agent Systems. Special Issue: Environment for Multi-Agent Systems*, volume 14(1). Springer Netherlands, 2007.