Programming Open Systems with Agents, Environments and Organizations

Michele Piunti,
Alessandro Ricci
Università di Bologna
Sede di Cesena
{michele.piunti,a.ricci}@unibo.it

Olivier Boissier
Ecole Nationale Supérieure des Mines
St-Etienne, France
boissier@emse.fr

Jomi F. Hübner
Universidade Regional de Blumenau
Blumenau, SC - Brazil
jomi@inf.furb.br

***Abstract*—MAS research pushes the notion of openness related to systems combining heterogeneous computational entities. Typically, those entities answer to different purposes and functions and their integration is a crucial issue. Starting from a comprehensive approach in developing agents, organizations and environments, this paper devises an integrated approach and describes a unifying programming model. It introduces the notion of *embodied organization*, which is described first focusing on the main entities as separate concerns; and, second, establishing different interaction styles aimed to seamlessly integrate the various entities in a coherent system. An integration framework, built on top of Jason, CArtAgO and $\mathcal{M}$oise (as programming platforms for agents, environments and organizations resp.) is described as a suitable technology to build embodied organizations in practice.**

I. Introduction

Agent based approaches consider *agents* as autonomous entities encapsulating their control, characterized (and specified) by epistemic states (beliefs) and motivational states (goals) which result in a goal oriented behavior. Recently, organization oriented computing in Multi Agent Systems (MAS) has been advocated as a suitable computation model coping with the complex requirements of socio-technical applications. As indicated by many authors [8], [2], [6], *organizations* are a powerful tool to build complex systems where computational agents can autonomously pursue their activities exhibiting social attitudes. The organizational dimension is conceived in terms of functionalities to be exploited by agents, while it is assumed to control social activities by monitoring and changing those functionalities at runtime. Being conceived in terms of human organizations, i.e., being structured in terms of norms, roles and global objectives, this perspective assumes an organizational layer aimed at promoting desired coordination, improving control and equilibrium of social dynamics. Besides, the need for openness and interoperability requires to cope with computational *environments* populated by several entities, not modellable as agents or organizations, which are supposed to be concurrently exploited by providing functionalities supporting agents objectives. These aspects are even more recognized in current ICT, characterized by a massive interplay of self-interested entities (humans therein) developed according to different models, technologies and programming styles. Not surprisingly, recent approaches introduced environment as pivotal dimension in MAS development [22], [14]. Such a multifaceted perspective risks to turn systems into a scattered aggregation of heterogenous elements, while their interplay, as well as their interaction, is reduced to a problem of technological interoperability. To prevent this, besides the different mechanisms and abstractions that must be considered, there is a strong need of binding these elements together in a flexible and clear way.

Providing a seamless integration of the above aspects places the challenge to conceive the proper integration pattern between several entities and constructs. A main concern is agent awareness, namely the need for agents to exhibit special abilities and knowledge in order to bring about organizational and environmental notions—which typically are not native constructs of their architectures [21], [15]. Once the environment dimension is introduced as an additional dimension, a second concern is how to connect in a meaningful way the organizational entities and the environmental ones, thereby (*i*) how the organization can ground normative measures as regimentation and obligations in environments, and (*ii*) how certain events occurring in environments may affect the global organizational configuration. These aspects enlighten a series of drawbacks on existing approaches, either on the conceptual model and on the programming constructs to be adopted to build systems in practice.

Taking a programming perspective, this work describes an infrastructural support allowing to seamlessly integrate various aspects characterizing an open MAS. In doing so, the notion of *Embodied Organization* is detailed, aimed at introducing each element in the MAS as an integral part of a structured infrastructure. In order to reconcile organizations, agents and environments, *Embodied organization* allows developers to focus on the main entities as separate concerns, and then to establish different interaction styles aimed to seamlessly integrate the various entities in a coherent system. In particular, the proposed approach defines a series of basic mechanisms related to the interaction model:

- How the agents could profitably interact with both organizational and other environmental entities in order to attain their design objectives;
- How the organizational entities could control agent activities and regiment environmental resources in order to promote desired equilibrium;

- How environmental changes could affect both organizational dynamics and agents activities;

The rest of the paper is organized as follows: Section II provides a survey of situated organization as proposed by existing works. Starting from the description of the basic entities characterizing an integrated perspective, Section III presents a unified programming model including agents, organizations and environments. The notion of Embodied Organization is detailed in Section IV, while Section V discusses a concrete programming model to implement it in practice. Finally, Section VI concludes the paper discussing the proposed approach and future directions.

II. Organizations situated in MAS Environments

Although early approaches in organization programming have not been addressed at modeling environments explicitly, recent trends are investigating the challenge to situate organizations in concrete computational environments. In what follows, a survey on related works is discussed, enlightening strengths and drawbacks of existing proposals.

A. Current Approaches

Several agent based approaches allow to implement situated organizations instrumenting computational environments where social interactions are of concern. A remarkable example of situated organization is due to Okuyama et al. [12], who proposed the use of "normative objects" as reactive entities inspectable by agents working in "normative places". Normative objects can be exploited by the organization to make available information about norms that regulate the behavior of agents within the place where such objects can be perceived by agents. Indeed, they are supposed to indicate obligations, prohibitions, rights and are readable pieces of information that agents can get and exploit in computational environments. The approach envisages a distributed normative infrastructure which is assumed to control emergent dynamics and to allow agents to implicitly interact with a normative institution. The mechanism is based on the intuition that the reification of a particular state in a normative place may constitute the realization of a particular institutional fact (e.g., "being on a car driver seat makes an agent to play the role driver"). This basic idea is borrowed from John Searle's work on speech acts and social reality [16], [17] Searle envisaged an institutional dimension rising out of collective agreements through special kind of rules, that he refers as *constitutive rules*. Those rules constitute (and also regulate) an activity the existence of which is logically dependent on the rules themselves, thus forming a kind of tautology for what a constitutive rule also defines the notion that it regulates. In this view, "being on a car driver seat makes an agent to play the role driver" strongly situate the institutional dimension on the environmental one, both regulating the concept of role adoption and, at the same time, defining it.

Constitutive rules in the form X *counts as* Y *in* C are also at the basis of the formal work proposed by Dastani et al. [5]. Here a normative infrastructure (which is referred as "normative artifact") is conceived as a centralized environment that is explicitly conceived as a container of *institutional facts*, i.e., facts related to the normative/institutional states, and *brute facts*, i.e. related to the concrete/ "physical" workplace where agents work. To shift facts from the brute dimension to the normative one the system is assumed to handle constitutive rules defined on the basis of "count-as" and "sanctioning" constructs, which allows the infrastructure to recast brute facts to institutional ones. The mechanism regulating the application of "count-as" and "sanctioning" rules is then based on a monitoring process which is established as an infrastructural functionality embedded inside the normative system. Thanks to this mechanism, agents behavior can be automatically regulated through enforcing mechanisms, i.e. without the intervention of organizational agents.

A similar approach is proposed in the work by Tinnemeier et al. [20], where a normative programming language based on conditional obligations and prohibitions is proposed. Thanks to the inclusion of the environment dimension in the normative system, this work explicitly grounds norms either on institutional states either on specific environmental states. In this case indeed the normative system is also in charge of monitoring the outcomes of agent activities as performed in the work environment, in so doing providing a twofold support to the organizational dimension and to the environmental one.

With the aim to reconcile physical reality with institutional dimensions, an integral approach has been proposed with the MASQ approach, which introduces a meta-model promoting an analysis and design of a global systems along several conceptual dimensions [19]. The MASQ approach relies on the less recent AGR model, extended with an explicit support to environment as envisaged by the AGRE and AGREEN [1]. Four dimensions are introduced, ranging from endogenous aspects (related to agent's mental attitudes) to exogenous aspects (related to environments, society and cultures where agents are immersed). In this case, the same infrastructure used to deploy organizational entities is also regulated by precise rules for interactions between agents and environment entities. The resulting interaction model relies on the theory of influences and reactions [9], in the context of which several interaction styles can be established among the heterogenous entities dwelling the system.

Besides conceptual and formal integration, few approaches have accounted a programming approach for situated organizations. By relating situated activities in the workplace, the Brahms platform endows human work practices and allows to represent the relations of people, locations, agent systems, communication and information content [18]. Based on existing theories of situated action, activity theory and distributed cognition, the Brahms language promotes the interplay of intelligent software agents with humans their organizations. A similar idea is provided by Situated Electronic Institutions (SEI) [4], recently proposed as an extension of Electronic Institutions (EI) [7]. Besides providing a runtime management of the normative specification of dialogic interactions between agents, the notion of observability of environment states is

at the basis of SEI. They are aimed at interceding between real environments and EI. In this case, special governors, namely modelers, allow to bridge environmental structures to the institution by instrumenting environments with "embodied" devices controlled by the institutional apparatus. Participating agents can, in this case, perform individual actions and interactions (either non message based) while operating upon concrete devices inside the environment. Besides, SEI introduces the notion of staff agents, namely organization aware agents which role is to monitor ongoing activities performed by agents which are not under the direct control of the institution. Staff agents are then assumed to bridge the gap between participating agents and the institutional dimensions: they typically react to norm violations, possibly ascribing sanctioning and enforcements to disobeying agents. Institutional control is also introduced by the mean of feedback mechanisms aimed at comparing observed properties with certain expected values. On the basis of possible not standard properties detected, an autonomic mechanism specifies how reconfigure the institution in order to re-establish equilibrium.

The ORA4MAS approach [11] proposed a programming model for concretely building systems integrating organizational functionalities in instrumented work environment. In ORA4MAS organizational entities are viewed as artifact based infrastructures. Specialized organizational artifacts (OAs) are assumed to encapsulate organizational functions, which can be exploited by agents to fulfill their organizational purposes. Using artifacts as basic building blocks of organizations, allows agents to natively interact with the organizational entity at a proper abstraction level, namely without being constrained to shape external actions as mechanism-level primitives needed to work with middleware objects. The consequence is that the infrastructure does not rely on a sort of hidden components, but the organizational layer is placed beside the agents as a suitable set of services and functionalities to be dynamically exploited (and created) as an integral part of the MAS work environment. On the other side, ORA4MAS does not provide an explicit support to environmental resources which are not included in the organizational specification. Two types of agents are assumed to evolve in ORA4MAS systems: *(i)* participating agents, assumed to join the organization in order to exploit its functions (i.e., adopting roles, committing missions etc.), while *(ii)* organization aware agents, assumed to manage the organization by making changes to its functional and structural aspects (i.e., creating and updating functional schemes or groups) or to make decisions about the deontic events (i.e. norm violations).

B. Open Issues and Challenges

Despite the richness of the models proposed for organizations of agents situated in computational environments, many aspects are still under discussion and have still to converge in a shared perspective between the different research lines. This variety of approaches have been dealt with separately in current programming approaches, each forming a different piece of a global view, with few consideration for how they could fit all together.

Typically interactions are based on a sub-agentive level, and are founded on protocols and mechanisms, instead on being based on the effective capabilities and functionalities exhibited by the entities involved in the whole system. Different approaches are provided for the interaction model between environment, agents and their organizations. Besides, there is not a clear vision on how environment and organizational entities should support agents in their native capabilities, as for instance the ones related to action and perception.

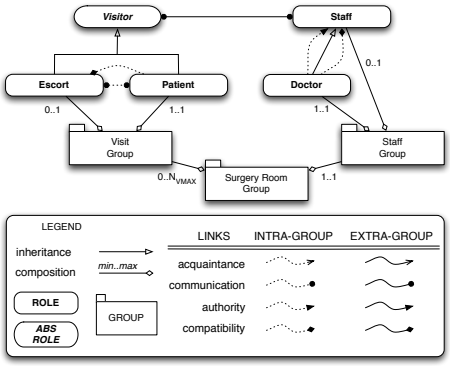
The computational treatments of goals clashes different approaches once they are referred to agents and their subjective goals, and when they are related to organizations and their global goals. For instance, approaches as MASQ, ORA4MAS describe in a rather abstract terms (i) *how* the subjective and global goals should be fulfilled in practice; (ii) *which* brute state has to be reached in order to consider a goal as achieved. By considering environments explicitly, either agents and organizations should be able to ground goals to actual environment configurations, thus recognizing the fulfillment of their objectives once the pursued goals have been reached in practice (this approach is adopted, for instance, in [5]). Other approaches, as for instance ORA4MAS [11], do not assume organizations able to automatically detect the fulfillment of global goals in terms of environment configurations.

As for goals, a weak support is provided for grounding norms in concrete application domains, thus allowing to establish how and when a norm has been fulfilled or violated. Furthermore few approachess manage norm lifecycle with respect to distributed and (highly) dynamic environments. No agreement is then established on which kind of monitoring and sanctioning mechanisms must be adopted. Some approaches envisage the role of organizational/staff agents [4], other approaches propose the sole automatic regulation provided by a programmable infrastructure [5], [20].

Different solutions are provided for defining agent capabilities, namely which grade of awareness is required for agents to exploit the functionalities provided by the organizational and environmental resources. Related to organizations, some approaches propose agents able to automatically internalize organizational specifications (i.e. MASQ, "normative objects"), other approaches, as (ORA4MAS and SEI) assume agents' awareness to be *encoded* at a programming level.

Finally, few approaches account technological integration, for instance with respect to varying agent architectures, protocols and data types. Besides, the described proposals typically focus on a restricted set of interaction styles (i.e. dialogical interactions supported by an institutional infrastructure in SEI, environment mediated interactions in normative objects, an hybrid approach in ORA4MAS).

With the aim to respond the above mentioned challenges, the next sections describe an integrated approach aimed at devising a unified programming model seamlessly integrating agents, organizations and environments.



(a) Structural Specification

id	*condition*	*role*	*type*	*mission*	*TTF*
*n*1		*Escort*	*obligation*	*mVisit*	–
*n*2		*Patient*	*obligation*	*mVisit*	–
*n*3		*Patient*	*obligation*	*mPatient*	–
*n*4		*Escort*	*permission*	*mPay*	5 minutes
*n*5	*unfulfilled*(*n*4)	*Patient*	*obligation*	*mPay*	5 minutes
*n*6		*Staff*	*obligation*	*mStaff*	–
*n*7		*Doctor*	*obligation*	*mDoc*	–
*n*8	*unfulfilled*(*n*5)∧ *unfulfilled*(*n*4)	*Staff*	*obligation*	*mSan*	1 day
*n*9	*fulfilled*(*n*4)∨ *fulfilled*(*n*5)	*Staff*	*obligation*	*mRew*	1 day
*n*10	*unfulfilled*(*n*6)	*Doctor*	*obligation*	*mStaff*	–

(b) Deontic Specification

Fig. 1. Structural (a) and Normative (b) specifications for the hospital scenario, represented using the $\mathcal{M}$oise graphical notation.

III. Unifying Agents, Organizations and Environments Programming

This section figures out the main elements characterizing an Embodied Organization. It envisages an integrated MAS in terms of societies of agents, environmental and organizational entities. In doing this, we refer to the consistent body of work already addressed at specifying existing computational models, while only the aspects which are relevant for the purposes of this work will be detailed. In particular, we refer to Jason [3] as agent development framework, CArtAgO [14] for environments and $\mathcal{M}$oise [10] for organizations.

In order to ease the description, the approach will be sketched in the context of an hospital scenario. It summarizes the dynamics of an ambulatory room, and can be seen as an open system, where heterogenous agents can enter and leave in order to fulfill their purposes. In particular, two types of agents are modeled as organization participants. *Staff agents* (namely physicians and medical nurses) are assumed to cooperate with each other in order to provide medical assistance to visitors. Accordingly, *visitor agents* (namely patients and escorts) are assumed to interact themselves in order to book and exploit the medical examinations provided by the staff.

A. Organizations

The first considered dimension concerns the organization. We do adopt the $\mathcal{M}$oise model, which allows to specify an organization based on three different dimensions referred as (*i*) structural, (*ii*) functional, and (*iii*) normative[1]. The Structural Specification (SS) provides the organizational structure in terms of groups of agents, roles and functional relations between roles (links). A role defines the behavioral scope of agents actually playing it, thus providing a standardized pattern of behavior for the autonomous part of the system. An inheritance relation can be specified, indicating roles that extend and inherit properties from parent roles. As showed in Fig. 1 (left), visitor agents can adopt two roles, patient and escort, both inheriting from a visitor abstract role. The doctor role is assumed to be played by a physician. It extends the properties of a more generic staff role, which is assigned in support and administration activities inside the group. Relationships can be specified between roles to define authorities, communication channels and acquaintance links. Groups consist in a set of roles and related properties and links. In the hospital scenario escorts and patients form visit groups, while staff and doctor from staff groups. The specification allows taxonomies of groups (i.e., escorts and patients forming visit group), and intra-group links, stating that an agent playing the source role is linked to all agents playing the target role. Notice that the cardinalities for roles inside a group are specified, indicating the maximum amount of agents allowed to play that role. The constraints imposed by the SS allow to establish global properties on groups, e.g. the well-formedness property means to complain role cardinality, compatibility, and so on.

The Functional Specification (FS) gives a set of functional schemes specifying how, according with the SS, various groups of agents are expected to achieve their global (organizational) goals. The related schemes can be seen as goal decomposition trees, where the root is a goal to be achieved by the overall group and the leafs are goals that can be achieved by the single agents. A mission defines all the goals an agent commits to when participating in the execution of a scheme and, accordingly, groups together coherent goals which are assigned to a role in a group. The FS for the hospital scenario (Fig. 2) presents three rehearsed schemes. The visitor scheme (*visitorSch*) describes the goal tree related to the visitor group. It specifies three missions, namely *mVisit* as the mission to which each agent joining the visit group has to commit, *mPatient* as the mission to be committed by the patient who has to undergo the medical visit, and *mPay* as the mission to be committed by at least one agent in the visit group. Notice that the goals "do the visit" (which is related to the mission *mPatient*) and "pay visit" (which is related to the mission *mPay*) can be fulfilled in parallel. The *monitorSch* describes the activities performed by a staff agent. These plans are aimed at verifying if the activities performed by the visitors follow an expected outcome, namely if the visitors fulfill the

[1]We here provide a synthesis of the $\mathcal{M}$oise approach showing the specification of the hospital scenario. For a more detailed description, see [10].

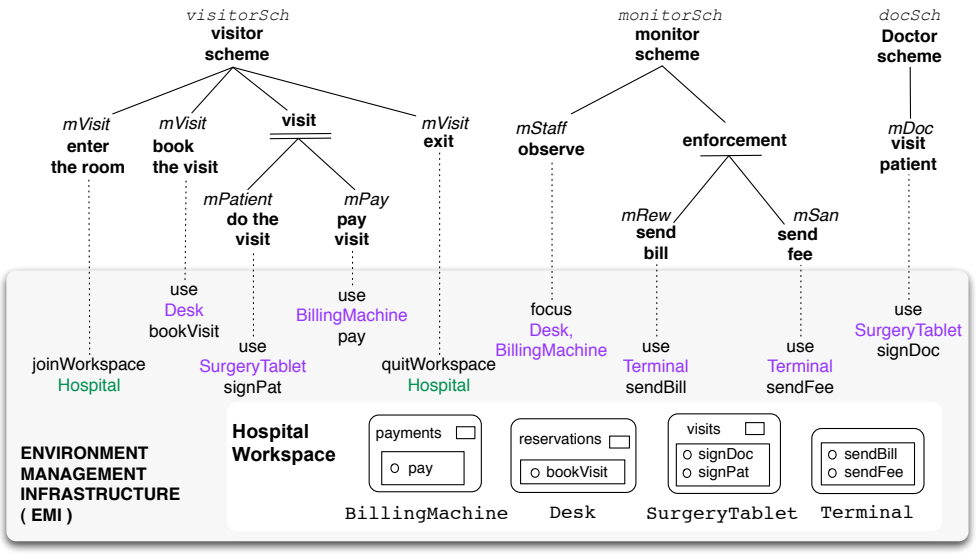


Fig. 2. (Above) $\mathcal{M}$oise Functional Specification (FS) for the hospital scenario. Schemes are used to coordinate the behavior of autonomous agents. (Below) FS is used to find a set of environmental artifacts, and to map their functionalities in the EMI.

payment committing the *mPay* mission (which includes the "pay visit" goal). Finally, the *docSch* specifies the activities to which a doctor has to commit, namely to perform the visit to every patient. Notice that each mission has a further property specifying the maximum amount of time than an agent has to commit to the mission ("time to fulfill", or ttf value). The FS also defines the expected cardinality for every mission in the scheme, namely the number of agents inside the group who may commit a given mission without violating the scheme constraints.

The Normative Specification (NS) relates roles (as they are specified in the SS) to missions (as they are specified in the FS) by specifying a set of norms. $\mathcal{M}$oise norms result in terms of *permissions* or *obligations* to commit to a mission. This allows goals to be indirectly related to roles and groups, i.e. through the policies specified for mission commitment. Fig. 1 (right) shows the declarative specification of the norms regulating the hospital scenario, and refers to the missions described in Fig. 2. "Time to fulfill" (ttf) values refer to the maximum amount of time the organization expects for the agent to fulfill a norm. For instance, norms $n1$ and $n2$ define an obligation for agents playing either patient and escort roles to commit to the *mVisit* mission. A patient is further obliged to commit to *mPatient* mission ($n3$). The norm $n10$ is activated only when the norm $n6$ is not fulfilled: It specifies an obligation for a doctor to commit the *mStaff* mission, if no other staff agent is committing to it inside the group. Based on the constraints specified within the SS and FS, the NS is assumed to include an additional set of norms which are automatically generated in order to control role cardinality, goal compliance, deadline of commitments, etc.

The concrete computational entities based on the above detailed specification have been developed based on an extended version of ORA4MAS [11]. This programming approach envisages organizational artifacts (OA) are those non-autonomous computational entities adopted to reify organizations at runtime, thereby implementing the *institutional* dimension within the MAS. In particular, ORA4MAS adopts two types of artifacts, referred as *scheme* and *group* artifacts, which manage the organizational aspects as specified in $\mathcal{M}$oise's functional, structural and normative dimensions. The resulting system has been referred as Organizational Management Infrastructure (OMI), where the term infrastructure can be understood from an agent perspective: it embeds those organizational functionalities exploitable by agents to participate the organizational activities and to access organization resources possibly exploiting, creating and modifying OAs on the need. Of course, in order to suitably exploit the OMI functionalities, agents need to be equipped with special capabilities and knowledge about the organizational structures, that is what in Subsection II-B we refer as agent awareness.

B. Environments

As said in Subsection II-A, the ORA4MAS approach does not support environments besides organizational functionalities. To this end, dually to the OMI, an Environment Management Infrastructure (EMI) is introduced to embed the set of environmental entities aimed at supporting pragmatic functionalities. While artifacts are adopted as basic building blocks to implement the EMI, environments also make use of workspaces (e.g., an `Hospital` workspace is assumed to contain the hospital infrastructures). Artifacts are adopted in this case to provide a concrete (*brute*) dimension – at the environment level – to the global system. Workspace are adopted in order to model a notion of locality in terms of an application domain.

As Fig. 2 shows, it is quite straightforward to find a basic set of Environment Artifacts (EA) building the EMI. Taking an

agent perspective, the developer here simply imagines which kind of service may be required for the fulfillment of the various missions/goals, thus mapping artifact functionalities to the functional specification given by the $\mathcal{M}$oise FS.

Designing an EMI is thus not dissimilar to instrumenting a real workplace in the human case: (*i*) to model the hospital room it will be used a specialized *hospital* workspace, (*ii*) to automate bookings it will be provided a *Desk* artifact, (*iii*) to finalize visits it will be provided a (program running on an) *Surgery Tablet* artifact, (*iv*) to automate payments it will be provided a *Billing Machine* artifact, and (*v*) to send fees and bills it will be provided a *Terminal* artifact.

C. Agents

Besides the abstract indication of the different artifacts exploitable at the environment level, the Fig. 2 also shows the actions to be performed by agents for achieving their goals. Thanks to the CArtAgO integration technology, several agent platforms are actually enabled to play in environments: seamless interoperability is provided by implementing a basic set of actions, and related perception mechanisms, allowing agents to interact with artifacts and workspaces [14], [15]. Those actions are directly mapped into artifact operations (functions), or addressed to the workspace: in the case of the EMI, a Jason agent has to perform a `joinWorkspace("Hospital")` action to enter the room (which is related to the *mVisit* mission); to book the visit (related to the *mVisit* mission) the action `bookvisit()[artifact_name("Desk")]` has to be performed on the desk artifact, and so on (see Fig. 2, below).

The same semantic mapping agents' actions into artifact operations is adopted to describe interactions between agents and OMI: e.g., `commitMission` is an operation that can be used by agents upon the *scheme* artifact to notify mission commitments; `adoptRole` (or `leaveRole`) can be used by an agent upon the *group* artifact in order to adopt (leave) a given role inside the group, etc.

Fig. 3 (left) shows a global picture of the resulting system. As showed, agents fulfill their goals and coordinate themselves by interacting with EMI artifacts, while staff agents, which we assume as special agents aware of organizational functionalities, can directly interact with the OMI. Both these dimensions are an integral part of the global infrastructure and, most important, can be dynamically exploited by agents to serve their purposes. From an agent perspective, the whole system can be understood as a set of *facts* and *functions*, which are exploited, from time to time, to the organizational and environmental dimensions. Through artifacts, the global infrastructure provides observable states, namely information readable by agents for improving their knowledge. Artifacts also provide operations, namely process based functionalities, aimed at being exploited by agents for externalizing activities in terms of external actions. Thus, the epistemic nature of observable properties can be addressed to the informational dimension of the whole infrastructure, while the pragmatic nature of artifact operations is assumed to cover the functional dimension.

IV. Embodied Organizations

As far as the global system is conceived, EMI and OMI are situated side by side inside the same work environment, but they are conceived as separated systems. They are assumed to face distinct application domains, the former being related to concrete environment functionalities and the latter dealing specifically with organizational ones. The notion of Embodied Organization provides a more strict integration: it further identifies and implements additional mechanisms and conceives a unified infrastructure enabling functional relationships between EMI and OMI. As some of the approaches discussed in Section II, we theoretically found this relationship on Searle's notion of constitutive rules. Differently from other approaches, we ground the notion of Embodied Organization on a concrete programming model, as the one who lead us to the implementation of EMI and OMI. As explained below, Embodied Organizations rely on a revised management of events in CArtAgO, and can be specified by special programming constructs referred as Emb-Org-Rules.

A. Events

A crucial element characterizing Embodied Organizations is given by the renewed workspace kernel based on events. Events are records of significant changes in the application domain, handled at a platform level inside CArtAgO. They are referred to both state and processes to represent the transitions of configurations inside workspaces. Each event is represented by a type,value pair ($\langle ev_t, ev_v \rangle$): Event *type* indicates the type of the event (i.e., `join_req` indicating agents joining workspace, `op_completed` indicating the completion of an artifact operation, `signal` indicating events signalled within artifact operation execution, and so on); Event *value* gives additional information about the event (i.e., the source of the event, its informational content, and so on). Due to the lack of space, the complete list of events, together with the description of the mechanism underlying event processing, can not be described here. The interested reader can find the complete model, including the formal transition system, in [13]. We here emphasize the relevance of events, which have the twofold role (*i*) to be perceived or triggered by agents (i.e. focusing/using artifacts) and (*ii*) to be collected and ranked within the workspace in order to trace the global dynamic of the system.

B. Embodied Organization Rules

While the former role played by events refers to the interaction between agents and artifacts, the second role is exploited to identify, and possibly govern, intra-workspace dynamics. On such a basis, the notion of *Embodied Organization* refers to the particular class of situated organization structured in terms of artifact based infrastructures and governed by constitutive rules based on workspace events. Events are originated within the infrastructure, being produced by environmental

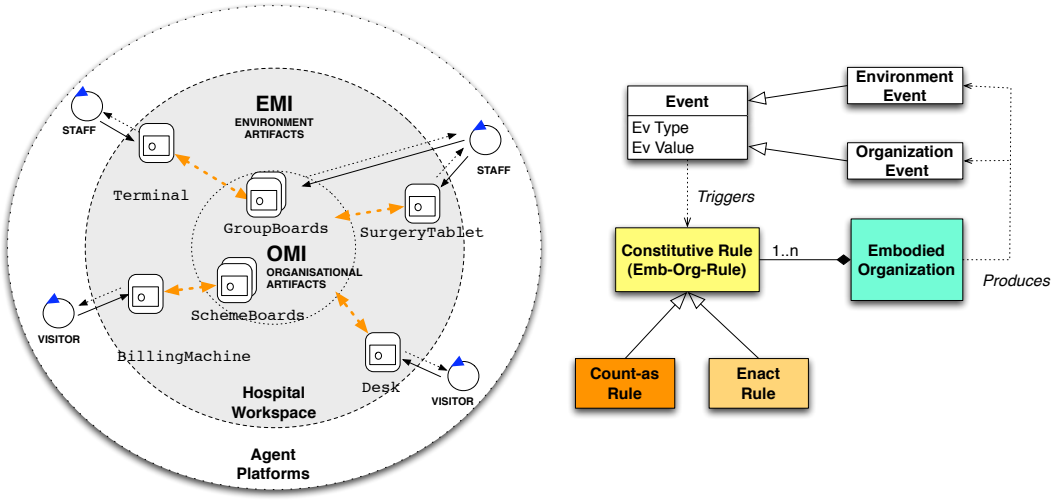


Fig. 3. (Left) Global view of the system presents an open set of agents at work with infrastructures managing Environment and Organization. Functional relationships between EMI and OMI are established by count-as and enact rules. (Right) Meta-model for Organizational Embodied Rules, used to implement count-as and enact rules.

and organizational entities. Computing constitutive rules is realized by Emb-Org-Rule, which consist of a programmable constructs "gluing" together organizational and environmental dimensions. An abstract model of this process is shown by the dotted arrows between EMI and OMI in Fig. 3 (right). Structures defining Emb-Org-Rule refer to *count-as* and *enact* relations.

Count-as rules state which are the consequences, at the organizational level, for an event generated inside the overall infrastructure. They indicate how, since the actions performed by the agents, the system automatically detects relevant events, thus transforming them to the application of a set of operators aimed at changing the configuration of the Embodied Organization. In so doing, either relevant events occurring inside the EMI (possibly triggered by agents actions), either events occurring in the context of the organization itself (OMI) can be vehicled to the institutional dimension: these events can be further translated in the opportune institutional changes inside the OMI, that is assumed to update accordingly.

Enact rules state, for each institutional event, which is the control feedback at the environmental level. Hence, *enact* rules express how the organizational entities automatically control the environmental ones. The use of enact rules allows to exploit organizational events (i.e. role adoption, mission commitment) in order to elicit changes in the environment.

V. Programming Embodied Organizations

Embodied Organizations enable a unified perspective on agents, organizations and environments by conceiving an interaction space based on a twofold infrastructure governed by events and constitutive rules (Emb-Org-Rules). In this section examples of programming such rules are discussed.

Programming Count-as Rules According to the $\mathcal{M}$oise FS previously defined, the organization expects that an agent va_{id} joining the hospital workspace is assumed to play the role visitor, which purpose is to book a medical visit and possibly achieve it. Thus, an event join_req, $\langle va_{id}, t\rangle$, dispatched once an agent va_{id} tries to enter the workspace, from the point of view of the organization "count-as" creating a new position related to the visit group. Making the event join_req to "count as" va_{id} adopting the role visitor, is specified by the first rule in TABLE I (left): it states that since an event signalling that an agent Ag is joining the workspace, an Emb-Org-Rule must be applied to the system. The body of the rule specifies that two new instances of organizational artifacts related to the visit group will be created using the make operator. In this case the new artifacts will be identified by `visitorGroupBoard` and `visitorSchBoard`. The following operator constitutes the new role inside the group: apply acts on the `visitorGroupBoard` artifact just created by automatically making the agent Ag to adopt the role patient. Finally, once the adopt role operator succeeds, the last operator includes the agent Ag in the workspace.

In the above described scenario, the effect of the application of the rule provides an institutional outcome to the joinWorkspace actions. Besides joining the workspace, a sequence of operators is applied establishing what this event means in organizational terms. When the effects of the role-adoption are committed, as previously described, a new event is generated by the group board: $\langle$op_completed, $\langle$`"visitorGroupBoard"`, va_{id}, `adoptRole, patient` $\rangle\rangle$. For the organization, such an event may "count-as" committing to mission $mPat$ on the `visitorSchBoard`. This relation is specified by the second rule in TABLE I, where a `commitMission` is applied to the `visitorSchBoard` for the mission *mPat*. Similarly, an event $\langle$ws_leaved, $\langle va_{id}, t\rangle\rangle$, signalling that the visitor agent has left the workspace, from an organizational perspective "count-as" leaving the role patient. This relation is specified by the first rule in TABLE I (right), where

```
+join_req(Ag)
->  make("visitorGroupBoard",
"OMI.GroupBoard",
["moise/hospital.xml","visitGroup"]);
     make("visitorSchBoard",
"OMI.SchemeBoard",
["moise/hospital.xml","visitorSch"]);
     apply("visitorGroupBoard",
adoptRole(Ag, "patient"));
     include(Ag).

+op_completed("visitorGroupBoard", _,
        adoptRole(Ag, "patient"))
-> apply("visitorSchBoard",
commitMission(Ag, "mPat")).
```

```
+ws_leaved(Ag)
-> apply("visitorGroupBoard",
        leaveRole(Ag, "patient")).

+op_completed("BillingMachine",
    Ag, pay)
-> apply("visitorSchBoard",
    setGoalAchieved(Ag, pay_visit)).


+op_completed("Terminal",
    Ag, sendFee)
-> apply("monitorSchBoard",
    setGoalAchieved(Ag, send_fee)).
```

TABLE I
EXAMPLE OF EMB-ORG-RULE (COUNT-AS) IN THE HOSPITAL SCENARIO.

```
+signal("visitorGroupBoard",
  role_cardinality, visitor)
-> disable("Desk", bookVisit).
```

```
+signal("monitorSchBoard",
  goal_non_compliance,
  obligation(Ag,
   ngoa(monitorSch,mRew,send_bill),
   achieved(monitorSch,send_bill,Ag), TTF)
-> exclude(Ag).
```

TABLE II
EXAMPLE OF EMB-ORG-RULE (ENACT) IN THE HOSPITAL SCENARIO.

a `leaveRole` is applied to the `visitorGroupBoard` for the role patient. At the same time, an event like $\langle$op_completed, $\langle$`BillingMachine`, va_{id}, `pay`, $t\rangle\rangle$ signals that a visitor agent has successfully finalized the pay operation upon the billing machine. Such an event "count-as" having achieved the goal pay visit on the `visitorSchBoard` (second rule in TABLE I, right). Finally, an event $\langle$op_completed, $\langle$`Terminal`, sa_{id}, `sendFee` , $t\rangle\rangle$, signalling that a staff agent has successfully used the terminal to send the fee to a given patient, "count-as" having achieved the goal send fee (third rule in TABLE I, right).

Programming Enact Rules *Enact* effects are defined to indicate how, from the events occurring at the institutional level, some control feedback can be applied to the environmental infrastructure. As far as the execution of the operations is conceived in CArtAgO, the OMI automatically dispatches events signalling ongoing violations. Violations are thus organizational events which may suddenly elicit the application of some *enact* rule used to regiment the environment.

In TABLE II, a regimentation is installed by the organization thanks to the enact rule stating that an event $\langle$signal, $\langle$`visitorGroupBoard`, `role_cardinality`, $\emptyset, t\rangle\rangle$ signalled by the `visitorGroupBoard` indicates the violation for the norm `role_ cardinality`. The related enact rule is given in TABLE II (left), where the reaction to this event is specified in order to disable the book operation on the desk artifact, for all the agents inside the workspace. The absence of any parameter related to agent identifier in the `disable("Desk", bookVisit)` operator makes the disabling to affect the overall set of agents inside the workspace. Similarly, violating the obligation imposed to the staff agent to fulfill sanctioning and rewarding missions elicits the scheme board assigned to the *monitorSch* to signal the event $\langle$signal, $\langle$`monitorSchBoard`, `goal_non_compliance`, `obligation(Ag,ngoa(monitorSch,mRew,send_bill), achieved(monitorSch,send_bill,Ag),TTF)`, $t\rangle\rangle$. This event is generated thanks to a special norm (called `goal_non_compliance`) which is automatically generated since the $\mathcal{M}$oise specification and stored inside the OMI. Due to the enact rule specified in TABLE II (right), this causes the exclusion for the `Ag` agent from the hospital workspace.

VI. CONCLUSION AND PERSPECTIVES

The notion of Embodied Organization has been introduced as a unified programming model for a seamless integration of environmental and organizational dimensions of MAS.

In Embodied Organizations, either environmental and organizational entities are implemented in concrete infrastructures instrumenting workspaces, decentralized in specialized artifacts which serve informational and operational functions. The approach establishes a coherent semantic for agent - infrastructure interactions, Embodied Organizations define functional relationships between the heterogenous entities at the basis of organizations and environments. These are placed in terms of programmable constructs (Emb-Org-Rules), governed by workspace events and inspired by Searle's notion of constitutive rules. Implementing organizations in concrete environments allows to deal explicitly with goals and norms, which fulfillment can be structurally monitored and promoted at the organizational level through the use of artifacts. Embodied Organizations are aimed to fit the work of agents and accordingly to allow them to externalize pragmatic and organizational activities. The use of Emb-Org-Rule automates and promotes specific organizational patterns, to which agents may

effortlessly participate simply by exploiting environmental resources. Artifacts can be used in goal oriented activities, and, most important, without the need to be aware of organizational notions like roles, norms, etc. Technological interoperability is ensured at a system level, by providing mechanisms for agent-artifact interactions which are based on a coherent semantic defined in CArtAgO. Besides, several interaction styles can be established at an application level, being agents mediated by infrastructures which can be modified, replaced and created on the need.

Future work will be addressed at covering missing aspects, such as the dialogical dimension of interactions, and the inclusion of real embodied entities in the system (i.e., humans, robots, etc.). An important objective is the definition of a general purpose approach, towards the full adoption of the proposed model in the context of concrete application domains and mainstream agent oriented programming.

REFERENCES

[1] José-Antonio Báez-Barranco, Tiberiu Stratulat, and Jacques Ferber. A unified model for physical and social environments. In *Environments for Multi-Agent Systems III, Third International Workshop (E4MAS 2006)*, volume 4389 of *Lecture Notes in Computer Science*, pages 41–50. Springer, 2006.

[2] Olivier Boissier, Jomi Fred Hübner, and Jaime Simão Sichman. Organization Oriented Programming: From Closed to Open Organizations. In *Engineering Societies for Agent Worlds (ESAW-2006). Extended and Revised version in Lecture Notes in Computer Science LNCS series, Springer*, pages 86–105, 2006.

[3] Rafael H. Bordini, Jomi Fred Hübner, and Michael Wooldrige. *Programming Multi-Agent Systems in AgentSpeak using Jason*. Wiley Series in Agent Technology. John Wiley & Sons, 2007.

[4] Jordi Campos, Maite Lòopez-Sànchez, Juan A. Rodrìguez-Aguilar, and Marc Esteva. Formalising Situatedness and Adaptation in Electronic Institutions. In *COIN-08, Proc.*, 2008.

[5] Mehdi Dastani, Nick Tinnemeier, and John-Jules CH. Meyer. A programming language for normative multi-agent systems. In *Multi-Agent Systems: Semantics and Dynamics of Organizational Models*. IGI-Global, 2009.

[6] Virginia Dignum, editor. *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*. IGI-Global, 2009.

[7] Marc Esteva, Juan A. Rodríguez-Aguilar, Bruno Rosell, and Josep L. AMELI: An agent-based middleware for electronic institutions. In *Proceedings of International conference on Autonomous Agents and Multi Agent Systems (AAMAS'04)*, pages 236–243, New York, 2004. ACM.

[8] Jacques Ferber, Olivier Gutknecht, and Fabien Michel. From Agents to Organizations: An Organizational View of Multi-agent Systems. In *Proceedings of (AOSE-03)*, volume 2935 of *Lecture Notes Computer Science (LNCS)*. Springer, 2003.

[9] Jacques Ferber and Jean-Pierre Müller. Influences and Reaction: a Model of Situated Multi-Agent Systems. In *Proc. of the 2nd Int. Conf. on Multi-Agent Systems (ICMAS'96)*. AAAI, 1996.

[10] Jomi F. Hübner, , Jaime S. Sichman, and Olivier Boissier. 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, 2007.

[11] Jomi F. Hübner, Olivier Boissier, Rosine Kitio, and Alessandro Ricci. Instrumenting Multi-Agent Organisations with Organisational Artifacts and Agents. *Journal of Autonomous Agents and Multi-Agent Systems*, April 2009.

[12] Fabio Y. Okuyama, Rafael H. Bordini, and Antônio Carlos da Rocha Costa. A Distributed Normative Infrastructure for Situated Multi-Agent Organisations. In *Decl. Agent Lang. & Techn. (DALT-VI)*, volume 5397 of *LNCS*. Springer, 2009.

[13] Michele Piunti. *Designing and Programming Organizational Infrastructures for Agents situated in Artifact-based Environments*. PhD thesis, ALMA MATER STUDIORUM Universitá di Bologna, April 2010.

[14] Alessandro Ricci, Michele Piunti, and Mirko Viroli. Environment programming in multi-agent systems: An artifact-based perspective. *Autonomous Agents and Multi-Agent Systems*, 2010. Springer, ISSN 1387-2532 (Print) 1573-7454 (Online).

[15] Alessandro Ricci, Andrea Santi, and Michele Piunti. Action and Perception in Multi-Agent Programming Languages: From Exogenous to Endogenous Environments. In *Proceedings Programming Multiagent Systems (PROMAS-10)*, 2010.

[16] John R. Searle. *Speech Acts*, chapter What is a Speech Act? Cambridge University Press, 1964.

[17] John R. Searle. *The Construction of Social Reality*. Free Press, 1997.

[18] M. Sierhuis. *Modeling and Simulating Work Practice; Brahms: A multiagent modeling and simulation language for work system analysis and design*. PhD thesis, University of Amsterdam, SIKS Dissertation Series, 2001.

[19] Tiberiu Stratulat, Jacques Ferber, and John Tranier. MASQ: Towards an Integral Approach of Agent-Based Interaction. In *Proc. of 8th Conf. on Agents and Multi Agent Systems (AAMAS-09)*, 2009.

[20] Nick Tinnemeier, Mehdi Dastani, J.-J.Ch. Meyer, and L. van der Torre. Programming normative artifacts with declarative obligations and prohibitions. In *IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT 2009)*, 2009.

[21] M. Birna van Riemsdijk, Koen Hindriks, and Catholijin Jonker. Programming organisation-aware agents: a research agenda. In *In 10th Engineering Societies for Agents Worlds (ESAW 09)*, 2009.

[22] Danny Weyns, Andrea Omicini, and James J. Odell. Environment as a first-class abstraction in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 14(1):5–30, February 2007. Special Issue on Environments for Multi-agent Systems.