The Governing Environment

Michael Schumacher[1] and Sascha Ossowski[2]

[1] Artificial Intelligence Lab,
Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland
michael.schumacher@epfl.ch
[2] Artificial Intelligence Unit, Universidad del Rey Juan Carlos, Spain
sascha.ossowski@urjc.es

Abstract. Whenever a multiagent system is designed, many dependencies in the system are identified and must be solved in a correct way. Coordination deals with the management of such dependencies. For that, two complementary viewpoints can be distinguished: *subjective coordination* manages intra-agent aspects while *objective coordination* essentially deals with inter-agent aspects. On the basis of this separation of concerns, the paper discusses the need of infrastructures for objective coordination. As in usual agent software platforms, this can be done by offering implicit support for objective coordination, by establishing the conditions necessary for running agent programs and maintaining agent interactions. Other infrastructures such as Electronic Institutions go one step further and shape the governing aspects of objective coordination. However, this is usually done through dedicated middle-agents that belong to the institution. An alternative approach is to transfer the governing or regulating responsibility from institutional agents to the *environment* of a multiagent system. A promising way of doing this is to view the environment as a rule-based infrastructure that defines reactions to events. This has the advantage of allowing for the definition of laws that not only regulate agent interaction (as most work in governed interaction), but *any* action within the environment. We illustrate this approach by several examples in different domains of laws.

1 Introduction

The interest for multi-agent systems (MASs) has grown increasingly in the last years. These systems are used in a great variety of applications such as process control, manufacturing, electronic commerce, patient monitoring, or games. MASs present very attractive means of more naturally understanding, designing and implementing several classes of complex distributed and concurrent software. The main reason resides in their unique paradigm of combining populations of autonomous active entities (agents) within a shared structured entity (the environment).

An autonomous agent is classically seen as *a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future* [13]. This description stresses the importance of the environment as the

D. Weyns, H. Van Dyke Parunak, and F. Michel (Eds.): E4MAS 2005, LNAI 3830, pp. 88–104, 2006.

living medium, the condition for an agent to live, or the first entity an agent interacts with [39,8]. Thus an agent is part of the environment. However, it remains autonomous, so that the environment may not "force" the agent integrity. Franklin and Graesser's definition shows that the environment is strongly related to the notion of *embodiment*, which refers to the fact that an agent has a "body" that delineates it from its environment in which the agent is situated. It is in this environment that an agent senses and acts. The acting of the agent on the environment is done autonomously; it directly influences its future sensing, because the environment is changed by the agent actions.

This view of the importance of the environment originally comes from Behavior Based AI (also named Bottom Up AI), which considers interaction with an environment as an essential feature for intelligent behavior. In this field, the main work has been done in systems that interact with a physical environment such as robots. This has influenced many research efforts within the software agent community, which deal with logical environments. However, most approaches in agent research have viewed the environment as something being modelled in the "minds" of the agents, thus using a minimal and implicit environment that is not a first-order abstraction, but rather the sum of all data structures within agents that represent an environment. This is a typical *subjective view* of the multiagent system inherited from distributed AI, which contrasts with an *objective view* that deals with the system from an external point of view of the agents [33,29].

Whenever a multiagent system is to be implemented and deployed, an underlying *infrastructure* becomes essential [26]. It offers to the MAS basic services to be used by the agents. Example functionalities are agent communication, naming or life-cycle management. The abstractions provided by such infrastructures are essential for agent-oriented software engineering, as they should be as close as possible to the concepts used for analysis and design.

Today's infrastructures primarily offer agent-related abstractions for the programming of agent architectures using for instance libraries for BDI agents [16], thus supporting subjective coordination. But they also offer implicit support for objective coordination (which we consider as *enabling* aspects), as they establish the conditions necessary for running agent programs (e.g. life-cycle management) and for setting the basic interaction means (e.g. message-enabled middleware between agents). However, current infrastructures have a main drawback. The used abstractions are not adapted for open systems, where participating agents may have totally different architectures, goals and interests, thus possibly behaving in a non-benevolent manner.

An appealing way to exert the necessary level of control over an agent in a truly open system is through an adequate MAS infrastructure. The type of services provided by the infrastructure, and the way in which these services are enacted, limit the set of possible actions (or modify their preconditions and/or effects). For that, a MAS designer can use a *governing infrastructure* to structure and shape the space of (inter-)action within a MAS in an open environment [26]. This governing perspective of objective coordination mainly allows to manage agent interactions from an external point-of-view. This has the strong advantage

that agents may be defined independently, and that some control is overtaken externally. In the area of virtual organizations, the Electronic Institutions (EI) approach [25] does this by defining so-called *governors* which are middle agents that mediate all (communicative) actions within a MAS [1]. This solution has, however, important disadvantages. Providing each agent with a governor puts a heavy computational burden on the infrastructure. But, more importantly, middle agents do not capture a natural design for the functionality they are expected to fulfill, i.e. mediation of communication. The governing or regulating responsibility should be transferred from specialized middle agents to the environment of a MAS, calling for the *environment as a governing infrastructure*. This can be done with the idea of a *programmable coordination medium* [9], which essentially defines *reactions to events* happening in an environment. This schema has the strong advantage to allow the definition of laws that not only regulate agent interactions, but also any happening within an environment. The paper proposes different domains of laws that can be identified with this idea. Overall, we expect that viewing the environment as a governing infrastructure dramatically simplifies the design and deployment of open multiagent systems.

The paper is organized as follows. Section 2 introduces objective and subjective point of views in modelling MASs. On the basis of this separation of concerns, the paper discusses infrastructures for objective coordination in Sect. 3, elaborating on enabling and governing aspects. Following the idea to transfer governing responsibility within the environment using a *programmable coordination medium*, Sect. 4 presents environment entities and events to be considered for defining laws as *reactions to events*. It then lists law domains to be used as a taxonomy. Section 5 concludes the paper.

2 Objectivity Versus Subjectivity

The interaction between agents is absolutely essential in a MAS, because it enables the MAS to exist. If agents are not able to interact with one another, no global behavior in the MAS is possible. One has therefore to model the interaction setup of the multitude of agents participating in the MAS. If this is not done, this typically leads to an *agent-oriented* view of a MAS [7]. This agent-oriented view models a MAS by describing the intra-agent aspects, such as the agent's representation of the world, its beliefs, desires, intentions, and by neglecting the description of the agent interaction means and of the space where these interactions take place.

This necessity for a clear identification of the interaction setup in a MAS naturally calls for a separation between the design of the individual tasks of each agent and the design of their interactions. This can be done at two different levels, according to the types of dependencies.

Indeed the modelling of the setup of multiple agents into a MAS leads to the detection of many dependencies of different nature. On one side, these dependencies rely on the result of the external composition of multiple agents into an

[1] All actions that the EI approach accounts for are communicative by nature.

ensemble; on the other side, they result from the individual or peculiar point of view of each agent interacting with other agents. Two types of dependencies can thus be distinguished and, as coordination is defined as *managing dependencies between activities* [21], two corresponding types or levels of coordination [33]:

- Agents have *subjective dependencies* which refer to intra-agent aspects. The management of these subjective dependencies refers to what we call *subjective coordination*. Thus, subjective coordination treats dependencies in an agent's *model*, its perception of the environment.
- A MAS is built by *objective dependencies* which refer to inter-agent aspects, namely the configuration of the system in terms of the basic interaction means, agent generation/destruction and organization of the environment. We refer to the management of these dependencies as *objective coordination*, because these dependencies are external to the agents. Thus, objective coordination acts directly on the dependencies in an environment.

Subjective coordination is dependent on objective coordination, because the first is based on and supposes the existence of the second. The mechanisms that are engaged to ensure subjective coordination must indeed have access to the mechanisms for objective coordination. If this is not the case, no subjective coordination is possible at all. This does not mean that objective coordination belongs to the intra-agent view, but only that the access mechanisms have a subjective expression in the agent. This is essentially the case for mechanisms like sending or receiving information.

Not differentiating the two levels of coordination leads to MASs that resolve objective coordination with subjective coordination means, i.e. by using intra-agent aspects for describing system configurations. For instance, a MAS intended at modelling the hierarchy in an organization would model this hierarchy internally in each agent by means of knowledge representation of the hierarchy, and would not describe it by establishing the communication flows that represent it.

2.1 Subjective Coordination

We distinguish two types of subjective coordination: *explicit* and *implicit* subjective coordination. They differentiate themselves in the explicit or implicit treatment of the management of subjective dependencies.

The research in distributed artificial intelligence (DAI) has proposed several coordination techniques [20,18,30] that deal with explicit subjective coordination. These techniques typically consider coordination as *the process by which an agent reasons about its local action and the (anticipated) actions of others to try and ensure the community acts in a coherent manner* [20]. Thus, these techniques are qualified as subjective coordination, because they try to resolve subjective dependencies by means of intra-agent structures that often involve high-level mentalistic notions and appropriate protocols. We characterize these techniques as explicit, because they explicitly handle coordination.

Agents may also coordinate themselves implicitly, without having explicit mechanisms of coordination. This may be, for instance, the case in the framework

of collective robotics [3,22,23], in which robots act on the base of the result of the work of other robots. Thus, this result, which is locally perceived through the sensors, allows to resolve a subjective dependency, namely the necessity to sense a specific information in order to act. This kind of coordination, which is also named *stigmergic* coordination, literally means an *incitement to work by the product of the work* [3].

2.2 Objective Coordination

Objective coordination is mainly concerned with the organization of the world of a MAS. This is achieved in two ways: i) by describing how the environment is organized, and ii) by handling the agent interactions. In the following, we address these two points.

The organization of the environment varies according to the space the MAS wants to integrate. We thus propose to distinguish between implicit and explicit environment organizations. An *implicit organization* does not explicitly model the environment, because it is given or imposed by the underlying logical structure or space organization on which a MAS evolves. This is, for instance, the case for network-aware agents roaming on the World Wide Web, where the environment is structured by the nodes of the network. An *explicit organization*, however, establishes a model of an environment that does not necessarily reflect the intended logical structure. This can be done by realizing an approximation of the target. When, for instance, one wants to simulate a continuous physical space, he will not be able to keep the continuity and will have to render the space discrete.

More importantly, the environment allows the arising of the interactions between the agents. Handling agent interactions asks for the description of the interactions between an agent and its environment, and the interactions between the agent themselves. As the environment also has a container function, it can be used to interact. Indeed, an agent has a relation with its environment by means of its perception. Furthermore, it can influence the state of its environment with specific actions. The interaction with the environment can then be understood and used to interact: all information is transmitted within the environment. Interaction thus becomes an action that changes the state of the environment.

Consider, for instance, the case of an insect-like robot dispersing in the environment information similar to pheromone. This information can be sensed by another robot that notices it as a trace of the roaming of the first agent.

In summary, we consider that agent interaction is always dealt through the environment. This is even the case in usual message-passing mechanisms that use agent-communication languages. We thus consider objective coordination as the way to *organize the environment.*

3 Infrastructures for Objective Coordination

Agent software platforms are a key element for the effective implementation and deployment of multiagent systems, as they provide the "interface" between the agents and their environment. They can be seen as an *infrastructure* for MAS,

providing basic resources and critical services to the agents, thus shaping their environment as part of a MAS [2]. Such MAS infrastructures are of foremost importance for Agent-Oriented Software Engineering, as the abstractions provided by them need to be as close as possible to the concepts used for analysis and design.

Any infrastructure for MAS needs to provide some services to allow agents to effectively *interact*, either directly by means of message-passing, or indirectly by leaving "messages" in shared data containers. Therefore, it is obvious that coordination in a multiagent system ultimately relies on the services provided by the infrastructure.

Today's infrastructures essentially provide (different levels of) support for programming a collection of individual *agents* in a multiagent world. So, from this perspective, they primarily render a certain level of support for *subjective* coordination. The more powerful the agent-related abstractions provided by the infrastructure, the higher their level of support for (and potential influence on) the instrumentation of subjective coordination mechanisms. For instance, if an infrastructure offers libraries for the implementation of BDI-type agents [16], the mechanisms for subjectively detecting dependencies, and reacting to them, will most probably be realized in these terms.

Although this is usually not made explicit at a conceptual level, agent platforms such as JADE[4] also provide some implicit support for *objective coordination* as they establish the *conditions* necessary for running agent programs and maintaining agent interactions. Other infrastructures, such as RICA-J [34], increase this level of support, as they provide the abstractions (and their software counterparts) to structure the space of interaction in a MAS. Finally, infrastructures based on the notion of Virtual Organizations or Electronic Institutions shape the *governing* aspects of objective coordination. In the sequel, we will elaborate on these points.

3.1 Enabling Aspects

Most current MAS platforms constitute essentially *enabling infrastructures*, as they provide agents with the means to interoperate. The services provided by FIPA-compliant infrastructures, for instance, refer to services such as agent communication, security, naming, location, etc., which are necessary preconditions that make it possible for agents to "live", "coexist", and interact within a MAS [26]. Other aspects relevant for achieving a basic level of objective coordination, such as the management of concurrency, are usually left to the agent designer. In essence, all but the most basic functionalities are to be dealt with subjectively, from the point of view of the agents (or: agent programmers).

The support for objective coordination by enabling infrastructures is not only determined by the *functionality* of the services that enable the co-existence of agents in a MAS. It is also affected by the conceptual abstractions in terms of

[2] The notion of infrastructure is fundamental for complex systems in general, not only in computer science and engineering, but also in the context of organizational, political, economical and social sciences [26]. Agent software platforms can be conceived as embodiments of this concept in the multiagent domain.

which they are modelled and/or accessed. By shaping these abstractions one exerts influence over the way in which individual agents are designed and deployed. There is a growing awareness that *organizational abstractions* (types of roles, interactions, etc.) are adequate candidates to this respect. In fact, it is tempting to maintain a tight coupling between a MAS, and the relevant features of the (human) organization that it models, during the whole design process. Several authors have put forward conceptual frameworks that conceive MAS in organizational terms [11,19,35].

Software frameworks that directly support such models shall still be conceived as enabling infrastructures, as they provide a basic set of computational abstractions that agent may make use of. Still, they provide an increased level of support for objective coordination. This is not only true because these computational abstractions are coarser grained. Infrastructures such as AGRE [12] or RICA-J [34] also allow for an "extension" of the infrastructure, e.g. by creating new types of interactions as a specialization of others, and thus providing the means for customizing basic services.

Agent programmers may benefit from using the computational abstractions provided by the infrastructure in order to simplify subjective coordination mechanisms, and the agent programs that instrument it. However, notice that agent programmers are not *forced* to do so – they may still decide to directly act upon the outside world instead of or use other lower-level services.

3.2 Governing Aspects

As of today, MAS infrastructures exploit only a minimal part of the potential of organizational abstractions. This is especially true with respect to the *coercive* facets of these abstractions. However, the increasing complexity and articulation of MAS application scenarios call for a more effective engineering support. In particular, designs must account for increased levels of openness (not only with respect to semantic interoperability but also, and maybe even more importantly, respecting the spectrum of different agent interests and goals), and decreased levels of predictability of (and control over) agent behavior. The former part of this requirement is exemplified by the work outlined in [30] where different types of behavioral restrictions (limitations to the set of possible actions of an agent in a given state of the world), operationalized in terms of "prohibitions" and "permissions", are used to bias the (macro-level) outcome of the interaction of autonomous (self-interested) "problem-solving" agents in a desired direction. However, the approach still assumes the possibility to "hand-wire" compliance with these restrictions into the agents' behavior strategies.

An appealing way to exert the necessary level of control over an agent in a truly open system (without relying on unrealistic assumptions on agent behaviors that limit their autonomy) is through an adequate definition of the MAS infrastructure. The type of services provided by the infrastructure (and/or the infrastructure agents that offer them), and the way in which these services are enacted, limit the set of possible actions (or just modify their outcome). Thus, such *governing infrastructures* can be used by the MAS designer to

structure and shape the space of (inter-)action within a MAS in an open environment [26].

Again, from the standpoint of an engineer concerned with designing mechanisms for objective coordination, infrastructures need to incorporate suitable abstractions for the *governance* of interaction. Coercive organizational abstractions, especially the different conceptualizations of the notion of *norms* or , are of foremost importance here. Upon the normative background of a MAS, the agents' social actions create (social) facts and, in turn, these facts constrain (or, at least, modify the outcome of) future behavior options. From this perspective, governing MAS infrastructures refer to the software that embodies and enacts the instititional aspects of Virtual Organizations.

Electronic Institutions (EI) [25], conceived as a particular instantiation of a Virtual Organizations, are of particular interest to this respect. EIs focus on the dialogical aspects of open MAS, and provide abstractions to shape the interactions of the agents participating in it. Agents play different roles in the (sub-)protocols that, together with additional rules of behavior, determine the legal sequences of illocutions that may arise within a particular instance of a *scene*. Scenes, in turn, are interconnected and synchronized by means of transitions within a *performative structure*. Instances of EI abstractions are meant to be of both descriptive and coercive nature. Specific institutional agents, called *governors*, assure that the latter is effectively implemented: in EIs, all relevant (communicative) action of agents is mediated through the corresponding governors. In fact, software tools that support the implementation of EIs [10] are (among the few existing) examples of governing MAS infrastructures.

To provide each agent with a governor puts a heavy computational burden on the infrastructure. An alternative approach is to incorporate the governance functionality into the infrastructure. This approach allows for a much more natural modelling of what the governors are expected to fulfill. It is then the proper infrastructure services that make sure that agent (inter-)actions are compliant with institutional norms. For instance, a governing infrastructure's basic communication service may filter out certain messages or automatically add context information to others. From this perspective, norms or laws are nothing but means to configure and customize the behavior of infrastructure services. As such, they can be conceived as the key component to instil objective coordination in open MAS.

4 The Environment as a Governing Infrastructure

To capture the idea of norms and laws in MASs, we propose to consider the environment as a regulating system. This is best shown by important elements of a definition of environment that were identified at the plenary session of E4MAS 2005[3]:

– The environment is a *first-class entity* [39]. It should be taken into account since the very first phase of modelling.

[3] http://www.cs.kuleuven.ac.be/∼distrinet/events/e4mas/

- The environment provides the *situation* for the existence of the agents and the interactions within the MAS. The situation mainly significates the conditions of existence, i.e. creation/destruction mechanisms, supplier and organization of the living space, and time management.
- The environment *regulates*. This regulation can be weak or strong, in the sense that the environment may control weakly or strongly what happens in and through it. This may be done by control mechanisms that cannot change at runtime, or by some laws that may be adapted at runtime.

Those descriptive elements of a MAS environment belong to the design of a MAS application. Therefore, the environment should be directly supported as an infrastructure. And for a MAS to provide law definitions, its supporting infrastructure should include rule-based mechanisms. Among research efforts that have tackled governed interaction [24,9,17,37,2], a programmable coordination medium [9] is a good candidate to realize an environment as a governing infrastructure.

4.1 Programmable Coordination Medium

Reactive coordination models [6,31] allow reactions associated to specific communication events to be programmed, leading to the notion of *programmable coordination medium* [9]. For that, this family of coordination models integrates an event mechanism into shared data space models, which show to be very useful to capture the idea of laws or norms in environments. This can be done by considering a tuple space as being reactive in the sense that the space can react to communication events rather than to the communication state changes only. Thus, from the point of view of the coordination medium, the observability is shifted from the tuples to the communication operations over the tuples. When a communication operation is executed, a reaction catching the event produced atomically executes a sequence of operations which usually have access both to the space and the information associated with the event. The idea of a programmable medium can be found for instance in TuCSoN (*Tuple Centers Spread Over Networks*) [27] [28].

Law-Governed Linda [24] is also a model that follows the notion of programmable media. The basic motivation of Law-Governed Linda is the inherent security problems of Linda [15,5]. In order to control each exchange of tuples through the tuple space, the model forces every process to adopt specific laws that control each exchange of tuples through the tuple space. For achieving this goal, Law-Governed Linda attaches a *controller* to every process. Each controller, which is in charge of regulating the exchange of tuples, has a copy of the law, and only allows a communication if it conforms to the law. This law regulates the occurrence of so-called *controlled events* that occur at the boundary between a process and the medium. It determines the effect of an event using a prescription, which is the *ruling* of the law, realized with a sequence of primitive operations. An example for the ruling of the law controlling an `out` could be to transform the corresponding tuple by concatenating some useful information.

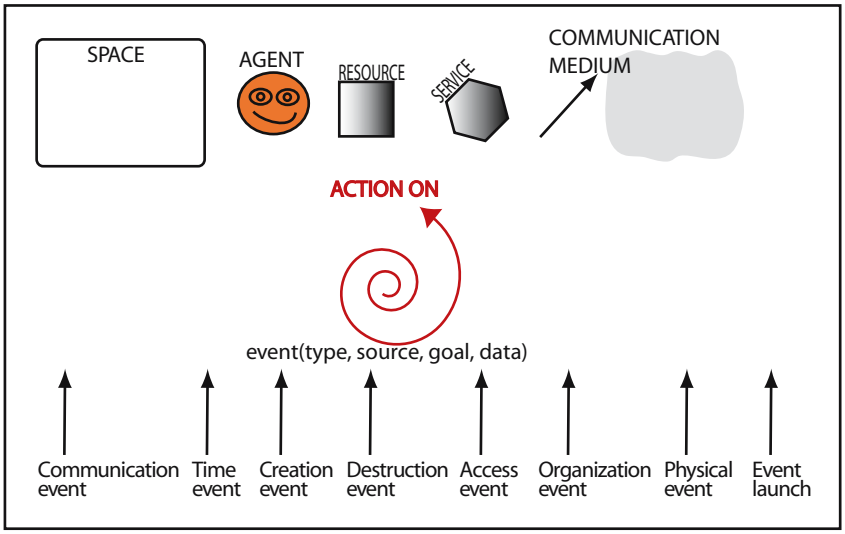

Fig. 1. Law definitions on events

4.2 Mechanisms for Environment Laws

Programmable coordination media can elegantly support the programming of laws regulating the environment in a MAS. The main reason is that their underlying mechanism can be used to support regulation not only of interaction, but also of *any* action in the environment. We therefore propose to extend their basic scheme to support ruled environment systems, i.e. which define laws as reactions to events. These reactions apply actions on environment entities (Fig. 1). In order to support such a definition of laws, the MAS infrastructure should thus be able to define as first-class entities all environment entities, all events happening in an environment, and law mechanisms.

We consider as being part of the environment the following entities:

- *Resources* are passive data being shared within the environment. Conditions of access and modifications are defined by the environment.
- *Services* offer functionalities within the environment, which again control their acting and the access of other entities to them. Services are different from resources, because they are activated and are not purely passive as data resources.
- *Spaces* are a logical organization of agent groups, resources and services. Furthermore, they make locality and localization possible, for instance offering encapsulation of events (e.g. for catching them) or allowing agents to move from one place to the other. Spaces are the most straightforward entity that make situatedness[4] of agents possible.

[4] We refer with this notion to the function of the environment to provide the situation.

– *Communication medium* are abstractions that allow interaction between agents. They can be mainly classified as point-to-point connections, using some message-based middleware (as in JADE [4] or explicit communication canals (as in IWIM [1]); or shared-data interaction where a blackboard is used to publish or retrieve information.

When considering agents from external point of view (i.e. as embodied entities), laws may also be applied on them. Consider for instance laws that have influence on agent creation/destruction.

As Fig. 1 shows, an event is a specific data that is launched at a specific moment within the environment and that may be catched to create a reaction to it. An event may include information such as the type (of event), the source (entity), the goal (entity) and a specific data.

Table 1. Events

Communication	Access to communication means (get, put, ...), configuration changes, communication breaks, ...
Time	Time period change, time exceeded, ...
Creation	Creation, re-launch or arrival of entities
Destruction	Destruction, pause or departure of entities
Access	Access to the entities
Organization	Density (population within a space), location change
Event Launch	At each event

Different types of events can be differentiated in a MAS (see also Tab. 1):

– *Communication events* belong to the most important one, because they give information on all access to communication means. They provide therefore the main source for laws. Examples are access to communication means (get, put, ...), configuration changes, or communication breaks.
– *Time events* are straightforward. Examples are time period change or time excess.
– *Creation or destruction events* of all kind of entities. These events may be general or specific to some types of agents or resources. Examples are creation, re-launch or arrival of entities in a specific space.
– *Access events* are launched whenever a specified entity is accessed.
– *Organization events* are mainly related to the space. For instance, they can be fired whenever a specific agent population density is reached or there is a location change of an agent.
– *Event launch* capture the idea that every event (whatever its type) is itself an event, which may be useful for logs.

The set of events tries to be as general as possible. We estimate that it should be part of an infrastructure offering environment laws for MASs. At a higher abstraction, application specific events are needed in each MAS application.

4.3 Law Domains

On the basis of the regulation capacity of a MAS infrastructure, a catalogue of possible laws helps the definition of the environment responsibilities in each case. We therefore propose a categorization that is based on events. Actually, each kind of event defines different types of laws. We will explain one by one the law domains. This list should be further completed.

- *Security laws* (Fig. 2a) are mainly related to interaction between agents and accesses to resources or services. They can change communicated data for encryption; or they can add some identification information. Access to resources/services may be restricted by some UNIX-like rights. Communication may be forbidden to specific agents.
- *Communication laws* (Fig. 2b) regulate interaction between agents at two levels. They can change transmitted data (such as cast data if it is of a

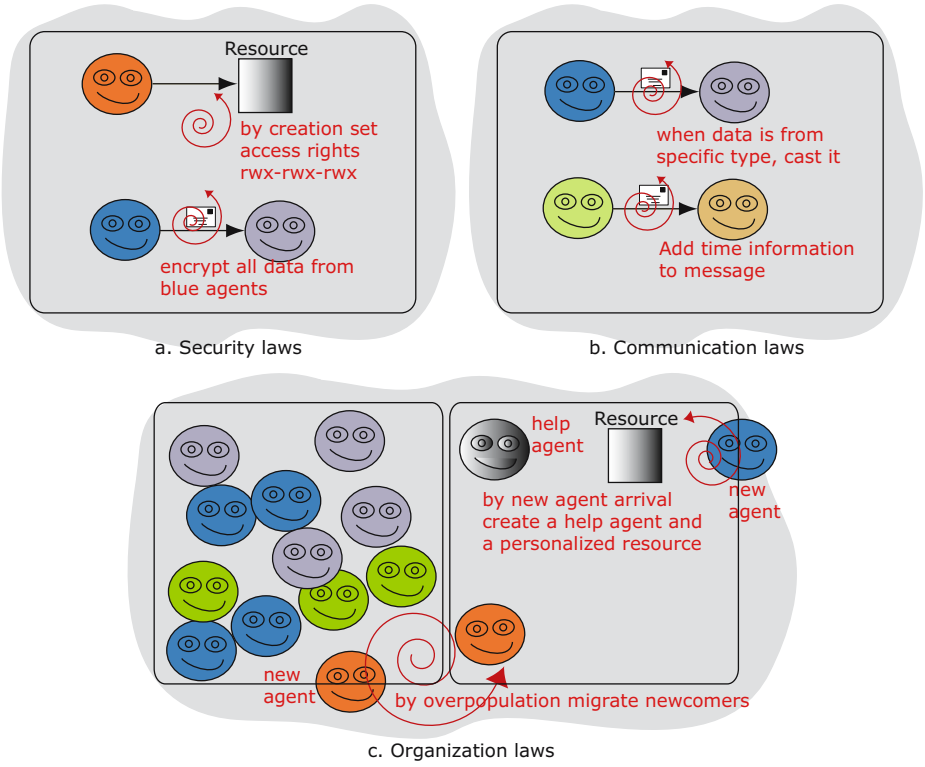

Fig. 2. Example of law domains: a. security laws: i) change resource access rights; ii) or, for all agents of a special type, encrypt communicated data; b. communication laws: i) When the data is from a specific type, cast it to another type; ii) or add time information to all messages in the space; c. organization laws: i) if population exceeds a specific number of agents, migrate newcomers to an alternative space; ii) or, at a new agent arrival, create a help agent and corresponding personalized resource information

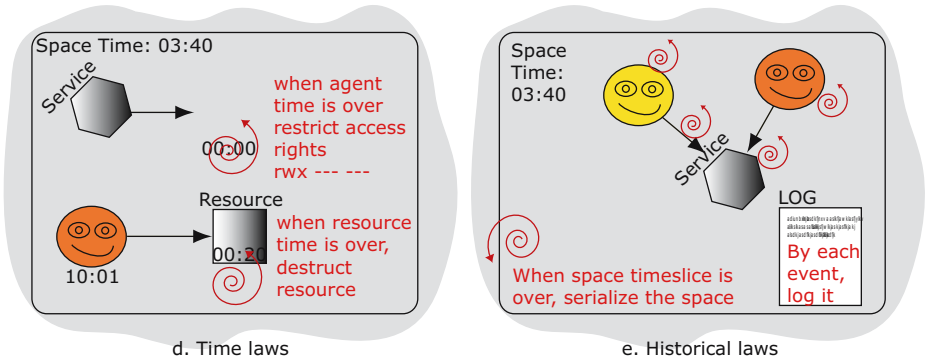

Fig. 3. Example of law domains: a. time laws: i) when agent time is over, restrict access rights to a specific service; ii) or when resource time is over, destruct the resource; b. historical laws: i) By each event, log it (creation, destruction, arrival, departure, etc.); ii) When space timeslice is over, serialize the space

specific type, or add time information), or filter it out. They can also create, destruct or change new communication means, for instance new channels between agents.

- *Organization laws* (Fig. 2c) regulate space populations. They may be applied on agents or on resources/services. Examples are: i) if population exceeds a specific number of agents, migrate newcomers to an alternative space. ii) at a new agent arrival, create a help agent and corresponding personalized resource information.
- *Time laws* (Fig. 3d) influence entities at specific time events. Examples are: i) When an agent's time is over, restrict its access to resources and services; ii) When a resource time is over, destruct it.
- *Historical laws* (Fig. 3e) provide functionalities for loging and serializing spaces. Examples are: i) by each event, log it (creation, destruction, arrival, departure, etc.); ii) When space timeslice is over, serialize the space.

5 Conclusion

Whenever a MAS is to be designed and implemented, dependencies are identified and must be handled in a coherent manner. Subjective coordination solves those raised from intra-agent aspects. Objective coordination, however, manages those that are related to inter-agent aspects. The environment, as a first-class entity, is the main place for objective coordination.

A MAS should both tackle subjective and objective coordination. For that, the use of an adequate infrastructure is essential. Indeed, in order to support implementation and deployment, MAS infrastructures play a central role, because they enable agent existence and interoperability. Usual agent software platforms provide a basic infrastructure for MAS by shaping the enabling aspects for objective coordination (e.g. middleware services for agent communication). Other infrastructures such as Electronic Institutions go one step beyond by governing

agent interactions, which is particularly relevant in open systems where the designer has limited or no control of the internal dynamics of agents. Electronic Institutions make use of institutional middle agents that assure compliance with *communication* laws. Still, if the infrastructure become the agents' only means to access the environment, then *all* types of relevant actions can be regulated *directly* by the infrastructure. In a nutshell: the environment is conceived as a governing infrastructure.

Among research efforts that have tackled governed interaction [24,9,17,37,2], a programmable coordination medium [9] is a good candidate to realize an environment as a governing infrastructure. The reason is its basic functionality to define reactions to environment events. This has the advantage to cope with other aspects as "only" ruling agent interactions. Laws become a general tool to describe in high-level terms any *reactions to events*. Thus, more complex inner dynamicity of the environment can be expressed and controlled by environment laws. This is shown by the different proposed domains of laws.

We expect future infrastructures to integrate means for supporting environment laws. For that, several issues can be considered. The first one is related to the expression of laws. It is an advantage to have a language that allows a higher-level law expression [36,14]. In some cases business rules such as DROOLS[5] may offer the sufficient functionality. But additionally, some applications in open systems would benefit to let their agents inspect the rules in order to adapt their own behavior. An agent with special rights may even change a law at runtime [32]. This may be achieved with an interpreter for the law language or through specific law objects that apply on environment entities.

Acknowledgments

This paper reports on work that has been partially supported by the Spanish Ministry of Education and Science (MEC) under grant TIC2003-08763-C02-02.

References

1. F. Arbab. Coordination of Massively Concurrent Activities. Technical report, CWI, Computer Science Department, Amsterdam, The Netherlands, 1995. CS-R9565.
2. S. Bandini, S. Manzoni, and G. Vizzari. A spatially dependent communication model for ubiquitous systems. In Weyns et al. [38], pages 74–90.
3. R. Beckers, O.E. Holland, and J.L. Deneubourg. From local actions to global tasks: stigmergy and collective robotics. In R.A. Brooks and P. Maes, editors, *Fourth Workshop on Artificial Life*, Boston, MA, USA, 1994. MIT Press.
4. F. Bellifemine, A. Poggi, and G. Rimassa. JADE – a FIPA-compliant agent framework. In *4th International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'99)*, pages 97–108, April 1999.

[5] http://drools.org

5. N. Carriero and D. Gelernter. Linda in Context. *Communications of the ACM*, 32(4):444–458, 1989.
6. N. Carriero and D. Gelernter. Coordination Languages and Their Significance. *Communications of the ACM*, 35(2):97–107, February 1992.
7. P. Ciancarini, A. Omicini, and F. Zambonelli. Multiagent engineering: the coordination viewpoint. In J. R. Nicholas and Y. Lespérance, editors, *Proceedings of the 6th International Workshop on Agent Theories, Architectures, and Languages (ATAL'99)*, pages 327–333, Orlando (FL), July 15–17 1999.
8. D. Weyns and H. Van Dyke Parunak and F. Michel and T. Holvoet and J. Ferber. Environments for Multiagent Systems, State-of-the-art and Research Challenges. In Danny Weyns, H. Van Dyke Parunak, and Fabien Michel, editors, *Environment for Multi-Agent Systems - Post-proceedings of the First International Workshop on Environments for Multiagent Systems*, volume 3374 of *Lecture Notes in Computer Science*. Springer Verlag, 2005.
9. E. Denti, A. Natali, and A. Omicini. Programmable Coordination Media. In D. Garlan and D. Le Metayer, editors, *Proceedings of the Second International Conference on Coordination Models, Languages and Applications (Coordination'97)*, number 1282 in LNCS, pages 274–288. Springer Verlag, September 1997.
10. Marc Esteva, Bruno Rosell, Juan A. Rodríguez-Aguilar, and Josep Ll. Arcos. AMELI: An agent-based middleware for electronic institutions. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, volume 1, pages 236–243, 2004.
11. J. Ferber and O. Gutknecht. A meta-model for the analysis of organizations in multi-agent systems. In Y. Demazeau, editor, *ICMAS'98*, pages 128–135. IEEE Press, 1998.
12. J. Ferber, F. Michel, and J.-A. Báez-Barranco. Agre: Integrating environments with organizations. In Weyns et al. [38], pages 48–56.
13. S. Franklin and A. Graesser. Is it an Agent or just a Program? A Taxonomy for Autonomous Agents. In J.P. Muller, M.J. Wooldridge, and N.R. Jennings, editors, *Proceedings of ECAI'96 Workshop (ATAL). Intelligent Agents III. Agent Theories, Architectures, and Languages*, number 1193 in LNAI, pages 21–35, August 1996.
14. A. Garcia-Camino, P. Noriega, and J. A. Rodriguez-Aguilar. Implementing norms in electronic institutions. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 667–673, New York, NY, USA, 2005. ACM Press.
15. D. Gelernter. Generative Communication in Linda. *ACM Transactions on Programming Languages and Systems*, 7(1):80–112, 1985.
16. M. P. Georgeff and A. S. Rao. The Semantics of Intention Maintenance for Rational Agents. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, number 1202 in LNAI, pages 704–710, 1995.
17. A. Gouaich, F. Michel, and Y. Guiraud. Mic: A deployment environment for autonomous agents. In Weyns et al. [38], pages 109–126.
18. L. Lee H.S. Nwana and N.R. Jennings. Co-ordination in Multi-Agent Systems. In H. S. Nwana and N. Azarmi, editors, *Software Agents and Soft Computing*, number 1198 in LNAI. Springer Verlag, 1997.
19. J. Fred Hübner, J. Simão Sichman, and O. Boissier. Using the moise+ for a cooperative framework of mas reorganisation. In *Advances in Artificial Intelligence - SBIA 2004*, pages 506–515, 2004.
20. N.R. Jennings. Coordination Techniques for Distributed Artificial Intelligence. In G.M.P. O'Hare and N.R. Jennings, editors, *Foundations of Distributed Artificial*. John Wiley and Sons, 1996.

21. T.W. Malone and K. Crowston. The Interdisciplinary Study of Coordination. *ACM Computing Surveys*, 26(1):87–119, March 1994.
22. A. Martinoli and F. Mondada. Collective and Cooperative Group Behaviours: Biologically Inspired Experiments in Robotics. In *Proceedings of the Fourth Symposium on Experimental Robotics ISER-95*, Stanford, USA, June 30- July 2 1995.
23. M.J. Mataric. From Local Interactions to Collective Intelligence. In L. Steels, editor, *The Biology and Technology of Intelligent Autonomous Agents*, pages 275–295. NATO ASI series, Hillsdale, NJ, USA, 1995.
24. N.H. Minsky and J. Leichter. Law-Governed Linda as a Coordination Model. In *Proceedings of the ECCOP Workshop on Models and Languages for Coordination of Parallelism and Distribution*, LNCS, Berlin, 1994. Springer Verlag.
25. P. Noriega and C. Sierra. Electronic institutions: Future trends and challenges. In Matthias Klusch, Sascha Ossowski, and Onn Shehory, editors, *Cooperative Information Agents VI*, volume 2446 of *Lecture Notes in Computer Science*. Springer Verlag, 2002. 6th International Workshop (CIA 2002), Madrid, Spain, September 18-20, 2002. Proceedings.
26. A. Omicini, S. Ossowski, and A. Ricci. Coordination infrastructures in the engineering of multiagent systems. In Federico Bergenti, Marie-Pierre Gleizes, and Franco Zambonelli, editors, *Methodologies and Software Engineering for Agent Systems: The Agent-Oriented Software Engineering Handbook*, chapter 14, pages 273–296. Kluwer Academic Publishers, June 2004.
27. A. Omicini and F. Zambonelli. TuCSoN: a Coordination Model for Mobile Agents. In *Proceedings of the First Workshop on Innovative Internet Information Systems*, pages 183–190, Pisa, Italy, June 1998.
28. A. Omicini and F. Zambonelli. Tuple Centres for the Coordination of Internet Agents. In *Proceedings of Fourteen ACM Symposium on Applied Computing (SAC'99). Special Track on Coordination, Languages and Applications*, pages 183–190, San Antonio, Texas, USA, February 28 - March 2 1999. ACM Press.
29. Andrea Omicini and Sascha Ossowski. Objective versus subjective coordination in the engineering of agent systems. In Matthias Klusch, Sonia Bergamaschi, Peter Edwards, and Paolo Petta, editors, *Intelligent Information Agents: An AgentLink Perspective*, volume 2586 of *LNAI: State-of-the-Art Survey*, pages 179–202. Springer-Verlag, March 2003.
30. S. Ossowski. *Co-ordination in Artificial Agent Societies – Social Structure and Its Implications for Autonomous Problem-Solving Agents*. Number 1535 in LNAI. Springer Verlag, 1999.
31. G.A. Papadopoulos and F. Arbab. Coordination Models and Languages. In M. Zelkowitz, editor, *Advances in Computers, The Engineering of Large Systems*, volume 46. Academic Press, August 1998.
32. A. Ricci, A. Omicini, and E. Denti. Activity Theory as a framework for MAS coordination. In Paolo Petta, Robert Tolksdorf, and Franco Zambonelli, editors, *Engineering Societies in the Agents World III*, volume 2577 of *LNCS*, pages 96–110. Springer-Verlag, April 2003. 3rd International Workshop (ESAW 2002), Madrid, Spain, 16–17 September 2002. Revised Papers.
33. M. Schumacher. *Objective Coordination in Multi-Agent System Engineering - Design and Implementation*. Number 2039 in LNAI. Springer Verlag, 2001.
34. J. M. Serrano, S. Ossowski, and S. Saugar. Reusability issues in the instrumentation of agent interactions. In *Third International Workshop on Programming Multi-Agent Systems: Languages and Tools (ProMAS'05)*, Utrecht, Jul 2005.

35. J.M. Serrano, S. Ossowski, and A. Fernández. The pragmatics of software agents – analysis and design of agent communication languages. In Klusch et al., editor, *Intelligent Information Agents – The AgentLink Perspective*. Springer-Verlag, 2003.
36. J. Vazquez-Salceda. *The Role of Norms and Electronic Institutions in Multi-Agent Systems*. Whitestein Series in Software Agent Technologies. Springer-Verlag, Berlin, 2004.
37. D. Weyns and T. Holvoet. A formal model for situated multi-agent systems. *Fundam. Inform.*, 63(2-3):125–158, 2004.
38. D. Weyns, H. V. D. Parunak, and F. Michel, editors. *Environments for Multi-Agent Systems, First International Workshop, E4MAS 2004, New York, NY, USA, July 19, 2004, Revised Selected Papers*, volume 3374 of *Lecture Notes in Computer Science*. Springer, 2005.
39. D. Weyns, M. Schumacher, A. Ricci, M. Viroli, and T. Holvoet. Environments for Multiagent Systems. *Knowledge Engineering Review*, 2005. to appear.