Towards a Unified View of the Environment(s) within Multi-Agent Systems

Abdelkader Gouaïch
Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier
LIRMM, 161 rue Ada, 34392 Montpellier Cedex 5, France
gouaich@lirmm.fr and www.lirmm.fr/∼gouaich

Fabien Michel
Laboratoire d'Etudes et de Recherches Informatiques
LERI, Rue des Crayères, B.P. 1035, 51687 REIMS Cedex 2, France
fmichel@leri.univ-reims.fr and www.univ-reims.fr/Labos/LERI/membre/fmichel

Keywords: MASs, environment, multi-environments

Received: May 6, 2005

Within the Multi-agent systems (MASs) paradigm, the concept of the environment plays a central role. In fact, the autonomous agents do only exist when they are deployed on an environment. Still, there is an implicit hypothesis in current trends of the MASs considering that the agents are related to only one environment that captures all the different aspects of the application domain. In this paper we challenge this implicit hypothesis by enabling multiple occurrences of the agent-environment relationship. This brings clarity and modularity for the design and implementation of complex MASs since each environment targets a specific aspect of the application. Thanks to the proposed characterization of the agent-environment relationship, the agents are still offered a unified view about all the environments.

Povzetek: Analizira povezave MAS z okoljem in pokaže, da jih je bolje imeti več.

1 Introduction

The paradigm of Multi-agent systems (MASs) naturally implies the idea that the agents are embedded in an environment. Indeed, as J. Odell and colleagues have pointed out: "without an environment, an agent is effectively useless" [11]. In fact, even if designers do not always consider the environment as a primary abstraction when engineering MAS applications, the main agency definitions do always mention that an agent is an entity that is operating in an environment using perception and action means, see e.g. [15, 3, 21, 20]. As highlighted in [19], this is even more obvious when considering situated MASs where the agents are placed within an environment that may comprise processes that do modify the state of the world independently from the actions of the agents. Consequently, such processes are modeled as parts of the environment which is thus considered as a first-class entity. Recent works have shown that the environment has clearly a rich potential not only for situated MASs but for the paradigm of MASs as a whole [18].

In this paper, we clearly embrace the idea that the environment has to be considered as a first order abstraction. However, the implicit hypothesis stating that all the agents of a MAS **share a common environment** has to be revised. Indeed, the agency definitions implicitly propose a vision where the agents only belong to one and unique environment that captures all the different aspects of the application domain.

Complex MASs often need to define different environments to capture different aspects of the application domain. However, due to the usual *single environment view*, this is generally done in an ad hoc manner. This has contributed to the confusion related with the notion of environment within MASs. The main goal of this of paper is to promote the idea that several environments can coexist within a single MAS application. To achieve this, we propose to characterize the relationship that maps an agent to an environment by the following features: *(i)* ontology of the environment, *(ii)* perceptions means, *(iii)* action means, *(iv)* interaction functions and *(v)* localization function in case of situated environments.

Once the relationship that links an agent to an environment has been characterized, it is then possible to consider multiple occurrences of this relationship between an agent and several environments. The agents are still offered a unified view of what is an environment, but each specific environment has its own way to implement the features of the agent-environment relationship.

2 Background

In [11], J. Odell and colleagues identify different types of environments that have been used within MASs applications. This work clearly identifies that MASs applications, depending on their application domain, need different kind of environments. However, J. Odell and colleagues have studied the characteristics of the environments rather than studying the characteristics of the relationship that maps agents to environments. Furthermore, the different classes of environments were studied and analyzed separately and the coexistence of several environments within the same MAS has not been considered.

In [19], D. Weyns and colleagues present the 3-Layer model which is a first step towards a better understanding of the different concerns which have to be considered when studying the environments of MASs. Figure 1 presents the 3-Layer model that distinguishes between three different classes of environments that exist at three different layers: *(i)* the environment of the MAS application layer, namely the *application environment*, *(ii)* the environment defined by the *execution platform* (a generic middleware) and *(iii)* the environment defined by the *physical infrastructure*. This decomposition identifies that there are several kinds of environments within a single MAS application. Still it implicitly considers that the agents, at the MAS application layer, are in relation with only one single environment. Furthermore, the relationship between the agents and their different environments is not studied. In this paper, we try to characterize this relationship independently from the layer where the environment is defined. For instance, the agent-environment relationship is the same for the environments that are defined within the application layer and those defined within the execution platform.

J. Ferber and colleagues were facing a domain of application, social simulations, where the autonomous agents are both situated in a spatial environment and in an organizational environment. They have developed a model that includes both the spatial and the organizational aspects in a single environment, namely the AGRE model [5] (cf. figure 2). The AGRE model merges the concepts of two aspects of the application domain: the organizational aspects and the spatial and temporal aspects to represent the agents within a virtual space. But this approach is limited since if another aspect of the application domain is identified, then the entire AGRE model has to be revised in order to include new concepts.

S. Bandini and colleagues propose the *Multilayered Multi-Agent Situated Systems* (MMASS) model for the definition of structured environments for situated MASs. The MMASS model relies on decomposing the environment in several different layers that represent physical or conceptual abstractions, specifying different aspects of the whole system [1]. Such an approach also highlights the fact that the environment may be defined according to many different aspects. Indeed, MASs may be composed by heterogeneous agents that may need different action and perception means achieving their goals. The MMASS model thus provides an interesting framework that allows to explicitly take into account different aspects of the application. Still the MMASS model focuses on situated MASs aiming to provide an explicit representation of agent environments and interaction mechanisms that are strongly dependant on the position of agents and on the spatial structure of the environment.

3 Revising some Assumptions on Environments

This section presents in an informal manner how some implicit assumptions on environments are revised. The idea is to start from a well established diagram that shows the agent-environment relationship. This diagram is then modified to obtain the vision of the agent-environment proposed in this paper.

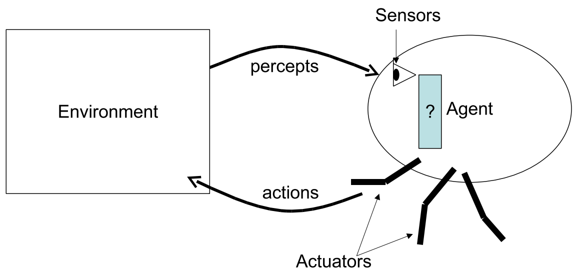


Figure 3: The original agent-environment diagram presented in [15].

Figure 3 shows the original diagram which has been presented in S. Russell and P. Norvig book [15]. This diagram gives an idea of what is the relationship between an agent and an environment. Still, many implicit hypotheses and assumptions are found in this diagram. The first point concerns where the agent's actuators and sensors are defined. Figure 3 situates the actuators and sensors on the side of the agent. This means that the actuators and sensors are defined by the ontology of the agent which makes the environment dependent on the ontology of the agents. This is not suitable since the environment has to be independent from the specific model of an agent. In fact, an environment can hold heterogeneous agents that use different ontologies and reasoning models.

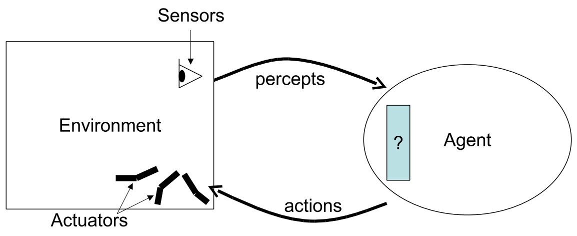


Figure 4: Defining the action and perception means of the agent on the environment side.

By contrast to figure 3, figure 4 places the actuators and sensors of the agent on the environment side. This also introduces the need for the ontology of the environment. In

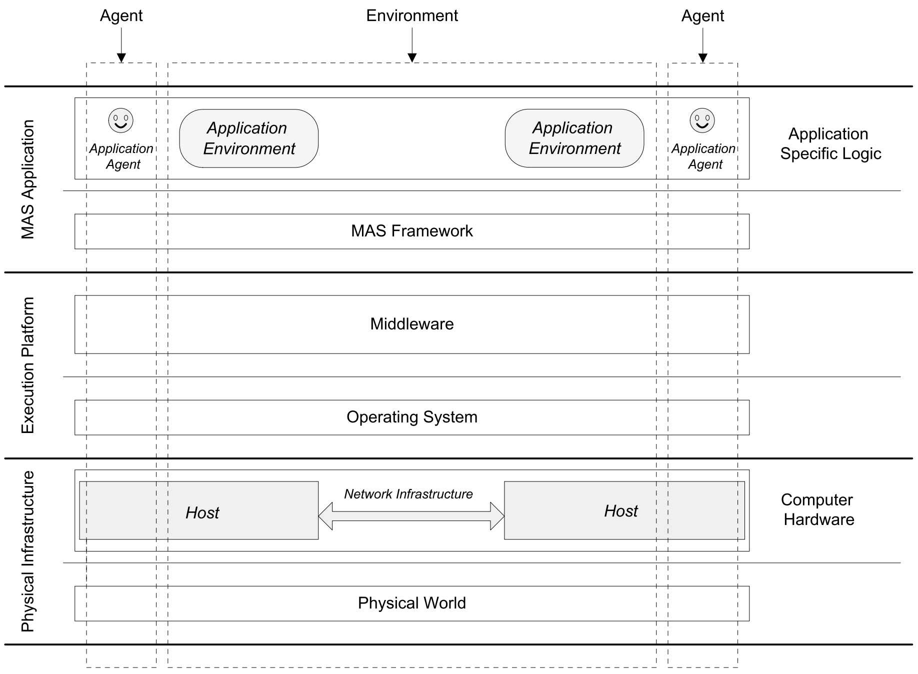


Figure 1: 3-Layer model for MASs [19].

fact, the actuators and sensors have to be explicitly defined by the ontology of the environment as means offered to the agents in order to perceive and act on the environment. The environment also defines the interaction functions that describe the relationship between the action means and perception means. In other words, the environment describes what is the result of the interaction between the actuators and sensors. It is important to notice that the interaction has been shifted from the agents to the action and interaction means. So, the agents do not directly interact, but their action and perception means interact within the environment.

Figure 5 presents a first step towards a general vision of the agent-environment relationship. In fact, an agent can be related to more than one environment. Each environment defines its specific ontology, perception and action means and interaction functions. If the perception and action means were not extracted from the agent and then placed in the environment, as it is the case in figure 3, the multiple instantiation of the agent-environment relationship would have been more difficult since every environment would have to follow the ontology of each agent.

The agent-environment relationship has to be distinguished from the means which are offered to the agent in one of its environment. In fact, any communication medium that enables the communication between the agent and the environment can be used. Particularly, some environments can be used as communication media. Figure 6 shows this case, where the agent is related to two environments 'environment 1' and 'environment 2'. This agent directly accesses 'environment 1'. However, the access to 'environment 2' is done through 'environment 1' which is considered, in this case, as a communication medium. It is important to notice that 'environment 2' is also related to 'environment 1' in order to act and retrieve the perception results. This schema is similar to the 3-Layer model, where the agents use the 'execution platform' environment in order to implement their relation with the 'application environment'.

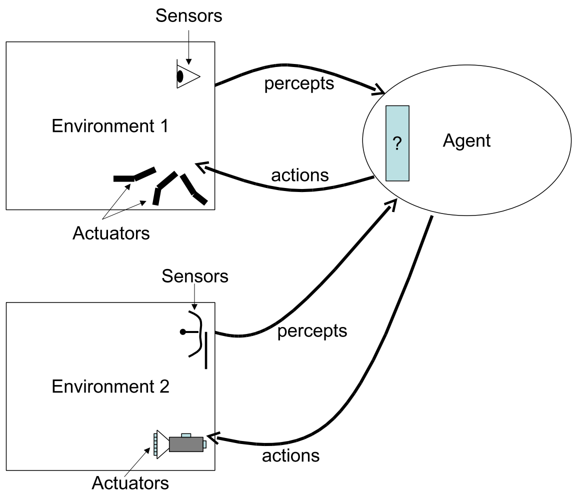


Figure 5: Multiple occurrences of the agent-environment relationship.

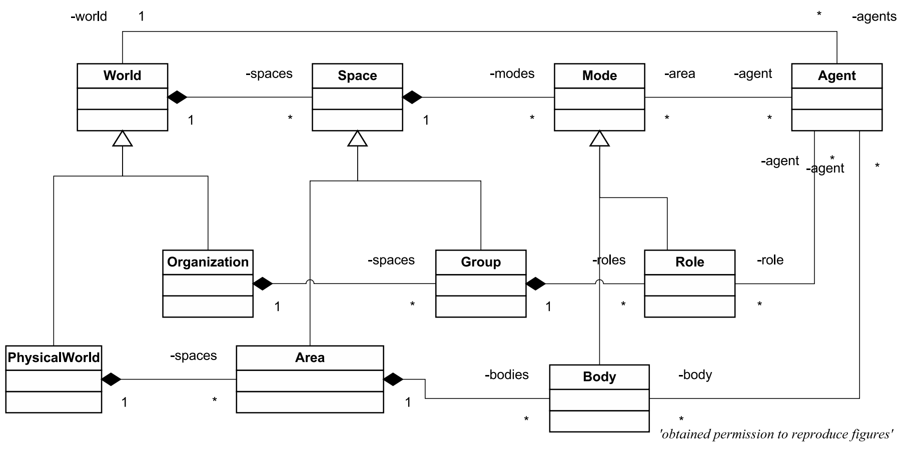


Figure 2: The UML meta-model of the AGRE model [5].

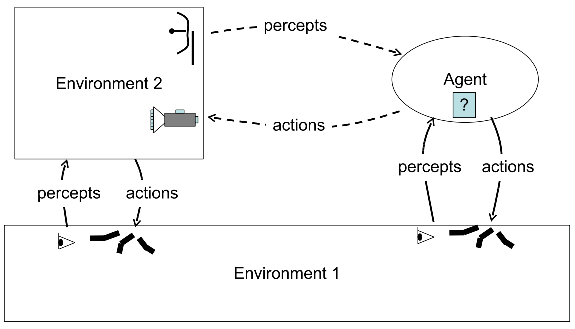


Figure 6: Using an environment as a communication medium to access another environment.

4 Generalizing the Agent-Environment Relationship

In the previous section, we have seen that there is an implicit hypothesis that considers that the autonomous agents do exist in a single environment. Consequently, this environment is supposed to contain all the different aspects and logics of the MAS application. Still, both from a conceptual point of view and from an engineering point of view, this hypothesis is inappropriate when dealing with complex software systems. Moreover, such an approach does not help to understand the role played by the environment within the MAS framework. In fact, each application domain has its own view of what is an environment and what are the functionalities implemented by an environment. Current approaches that have faced the problem of designing a MAS application where the autonomous agents exist in an environment that captures more than one aspect of the application domain have suggested to merge these aspects in a single environment. These approaches are limited since each time a different aspect of the application domain is identified then this aspect is appended to the environment in an ad hoc manner. As a result, the environment centralizes all the different aspects of the targeted application. Such an environment contradicts the modularity and separation of concerns principles which have been proved to be useful when designing complex software systems.

We suggest an approach that: *(i)* challenges the implicit hypothesis that considers only one environment which is commonly shared by the autonomous agents; *(ii)* generalizes current approaches that have identified that several environments are required to capture all the aspects of the application domain.

Such an approach may seem to contrast with the 3-layer approach proposed in this volume by D. Weyns and T. Holvoet, where only one environment crosscuts the three layers (cf. figure 1) [17]. In fact, such a contrast comes from the fact that the two approaches do not have the same objectives: the 3-layer model aims to highlight that several concerns have to be taken into account when engineering environments for MASs. Rather than that, our goal is to characterize the agent-environment relationship and our claim is that this relation has to be instantiated as many times as required. Further study is necessary to clarify the distinction between the two models.

The problem now is to have a more precise idea of what is meant by the existence of agents in an environment. Initially, the term existence was used in [15] by S. Russell and P. Norvig in order to highlight the central role played by the environment within the MASs paradigm. However, up until now there is not a consensual definition of this existence relationship that links the autonomous agents and their environments.

4.1 Characterizing the Agent-Environment Relationship

In order to offer a general approach, we study the relationship between an agent and an environment rather than defining a specific model of an environment. Once this relationship has been characterized, then different environments implement it differently according to the aspect of the application domain which is captured. Thus, the agent is offered a unified view of the existence relationship that links the agent to the different environments.

Starting from the state of the art proposed by D. Weyns and colleagues in [19] and from previous works on the development of environments for autonomous agents [8], the following elements are suggested as a characterization of the agent-environment relationship:

- the *ontology of the environment* has to be distinguished from the ontology of the agents. In fact, the environment defines its own concepts and their logics. To operate in an environment, the agents need to understand some parts of the ontology of the environment. When the agents use an internal ontology that differs from the ontology of the environment, it is their responsibility to map the concepts of their own ontology to the ontology of the environment. For instance, we can conceive the use of cognitive agents, such as BDI agents, in a spatial grid environment. The ontology of the spatial grid environment defines concepts such as: pheromones, cells, movement, position and so on. This ontology does not have any concept of mental states which are present in the BDI model for instance. So, it is the responsibility of the cognitive agent to translate the concepts of the spatial grid environment into some logical predicates which are included in its mental states. Obviously, it is not always easy to translate the concepts defined by the ontology of the environment into the agent's ontology. For instance, even if it is theoretically possible to imagine BDI agents deployed on a spatial grid environment where they can move and drop pheromones, this is very hard to achieve in practice as the ontology of the BDI agents and the ontology of the spatial grid environment are dissimilar.
- the *perception means* are the concepts defined by the ontology of the environment and that enable the agents to perceive their surroundings.
- the *action means* are the concepts expressed by the ontology of the environment and that enable agents to influence their surroundings.
- the *interaction functions* define the relationship between the action means and perception means. It is important to notice that the interaction is not defined between the agents, but between the action and interaction means that are defined by the ontology of the environment.

Besides, for a situated environment, an additional element characterizes this agent-environment relationship:

- the *localization function* is specifically provided by situated environment. In a situated environment, one can define the location of an agent in terms of coordinates within the environment. The location of an agent is also defined as a concept of the ontology of the environment.

This characterization of the agent-environment relationship offers a unified view of the environments which are considered either as: *(i)* an infrastructure or *(ii)* an application-level environment.

4.2 Illustrating some Agent-Environment Relationships

To illustrate the suggested characterization of the agent-environment relationship, let us consider the review of some existing environments.

4.2.1 Tuple Spaces

Tuple Spaces (TSs) have been introduced by researchers at the Yale University where *Linda* [6] –the first tuple space-based system– has been developed. A TS system is composed by the following elements:

- a *tuple* is basically a list of typed fields. Fields may be *actual* when they hold a value or *formal* otherwise.
- a *tuple space* is an abstract storage location where tuples are deposited and retrieved by the software entities which are called processes.
- the *processes* store and retrieve the tuples using the following primitives:
  - out: this primitive inserts a tuple into a tuple space which becomes visible to all the processes that have access to that tuple space.
  - in: this primitive extracts a tuple from a tuple space, with its argument acting as the template, *anti-tuple*, against which to match. When all the corresponding fields of a tuple match the template the tuple is withdrawn from the tuple space.
  - read: this primitive is equivalent to the 'in', except that a matched tuple is not withdrawn from the tuple space and remains visible to the other processes.
  - eval: this is similar to the 'out', except that it creates an independent process yielding the tuple which is inserted in the tuple space.

The Tuple Space as an Environment

- the ontology of the tuple space defines the concepts that follow: tuple, anti-tuple, tuple space, out, in, read and eval. So, in order to exist in a tuple space, an agent has to understand these concepts and to be able to translate them into the concepts of its own ontology.
- the action means offered by a tuple space are represented by the out primitive. In fact, an agent can influence other agents by putting a tuple within the tuple space using the out primitive.

- the perception means offered by a tuple space are represented by the concept of anti-tuple and by the in and read primitives. So, an agent can perceive within a tuple space by using a template represented by the anti-tuple as argument for the in or read primitives.

- the interaction functions defined by the tuple space are the rules that make an anti-tuple matching a tuple. These rules make the link between the action means and perception means of the agents. Some tuple space architectures such as TuCSoN [12] change the interaction functions of the tuple spaces and add some *reaction rules* in order to dynamically change the interactions that take place between the tuples and anti-tuples.

4.2.2 MIC*

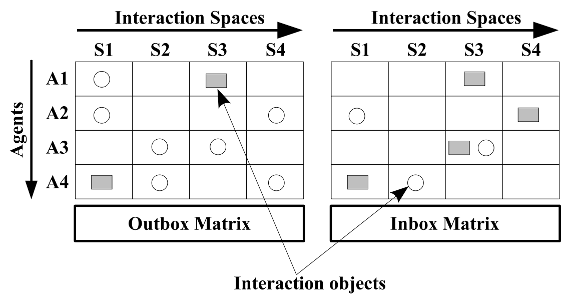


Figure 7: The MIC* structure.

MIC* is an algebraic model of a MAS infrastructure holding autonomous and interacting agents [8]. The MIC* structure (cf. figure 7) is composed of two matrices where rows represent the agents $i \in \mathcal{A}$ and the columns represent interaction spaces $j \in \mathcal{S}$. The elements of both matrices are called *interaction objects*. Interaction objects represent information carriers but also model the interaction means (i.e. actuators and sensors) of the agents within the system. Moreover, interaction objects are formally defined so that their structure is a commutative group $(\mathcal{O}, +)$: a composition of interaction objects is also an interaction object (see [7] for a complete mathematical description of the MIC* model). The outbox matrix and the inbox matrix can be described as follows:

1. The outbox matrix: each element of the outbox matrix $o_{(i,j)} \in \mathcal{O}$ is an interaction object that models the actuators and sensors of the agent i in the interaction space j. In other words, the elements of the outbox matrix model the means that enable an agent to perceive and influence the universe in a particular interaction space. So, having an interaction object in the outbox matrix is the only way for an agent to exist and operate in the MAS. Particularly, when $o_{(i,j)} = 0$, the agent i neither influences nor perceives the universe in the interaction space j: agent i does not exist in j. Moreover, the means used to perceive the universe are distinguished from the result of the perceptions. The perception results are placed in the inbox matrix.

2. The inbox matrix: each element of the inbox matrix $o_{(i,j)} \in \mathcal{O}$ represents the result of the perceptions of the agent i in the interaction space j.

MIC* as an Environment

- the ontology of the MIC* environment defines the following concepts: interaction object, interaction space, inbox matrix, outbox matrix, and the operators of movement, interaction and computation.

- the actions means offered by MIC* are represented by the interaction objects, interaction space, outbox matrix and the computation operator.

- the perception means offered by MIC* are represented by the interaction objects, interaction space, inbox matrix and the interaction operator.

- interaction functions: MIC* defines interaction operators within the scope of an interaction space to calculate the result of the interaction of an actuator on a sensor. Both the actuator and the sensor are uniformly represented as interaction objects. The actuator and the sensor are located in the outbox matrix and the result of the interaction is stored in the row corresponding to the sensor in the inbox matrix. Hence, an agent can retrieve its perception whenever he wants in order to deliberate and emit other interaction objects.

- localization function: MIC* is a situated environment since each agent owns an identifiable location. This location is defined by the rows which are occupied by the agent within the inbox and outbox matrices of MIC*.

4.2.3 Spatial Grid Environment

Spatial grid environments have always been considered as a very useful tool for the modeling of physical environments within MAS applications, see e.g. [13]. A spatial grid environment defines a two-dimensional world which is spatially discretized into *cells* (or *patches*) which define the agents' location and that contain local environmental properties such as a pheromone concentration for instance. MAS platforms such as STARLOGO [14], NETLOGO [16] or TURTLEKIT [10] are some examples which are explicitly based on this kind of environment. Within such environments, the agents have the ability to perceive and act on the environment according to the perception/action means which are afforded by the cell on which they are: the agents can move to another cell, perceive the pheromone concentration of the cell and drop some pheromones on the cell. Additionally, the environment owns some processes that define the pheromone propagation and evaporation phenomena. It is these processes that define the interaction functions since they enable the agents to coordinate their behaviours through the environment.

Spatial Grid as an Environment

- the ontology of a spatial grid environment defines the following concepts: cell, pheromone, propagation function, evaporation function and movement.
- the action means offered by a spatial grid environment are represented by the movement actions and the drop of pheromones.
- the perception means offered by a spatial grid environment are afforded by the cell that represents the current location of an agent.
- the interaction functions defined by a spatial grid environment maps the intensity of the pheromones to the locations of the agents. The spatial grid environment computes the intensity of the pheromones according to the evaporation and propagation functions.
- localization function: within the spatial grid environment, each agent is located by the coordinates of the current inhabited cell.

4.2.4 AGR Organizational Environment

In [4], J. Ferber and O. Gutknecht have proposed the Agent/Group/Role (AGR) model. The main organizational concepts of AGR are described as follows:

- agent: an agent represents the active entity that tries to achieve its design goals by interacting with other agents.
- group: the organization of the MAS is structured by groups. A group is defined as a set of agents that play specific roles. It is important to notice that the agents can interact by sending messages only when they belong to the same group.
- role: the role represents an abstraction of the function of an agent within a group.

The MadKit platform [9] was then developed as an infrastructure that implements the AGR concepts.

AGR as an Environment

- the ontology of the AGR environment defines the concepts of group, role and message.
- the actions means offered by the AGR environment are represented by the action of acquiring/leaving a role within a group and by the action of sending a message.
- the AGR environment considers the roles played by the agent as its perception means.
- the AGR environment defines the interaction functions as message routing and delivery. Hence, messages are delivered to the agents by using the social position of an agent, namely the role of an agent within a group.
- localization function: within the AGR environment, each agent is located by the roles played within groups.

5 Experiment

This section presents an experiment of a MAS that has been entirely built using the multi-environments approach. This MAS is concerned with a social simulation derived from the *SugarScape* system [2]. The simulated agents are located in a spatial 2D grid that contains some resources. The resources are consumed by the agents and regenerated after a period of time. The agents can either move between the cells of the grid or consume the available resources to augment their energy. When the energy of the agents reaches zero these agents die.

Obviously, the simulation of such a system requires the intervention of other meta-agents that manage the simulation process and dynamically collect the information from the simulated system at the runtime. Even if these agents are considered at the meta-level for the *SugarScape* system, they have to be considered as being part of the MAS since they can change the dynamics of the whole system. Among the meta-agents that are required for the simulation, we have identified the following:

- the *scheduler* implements the dynamics of the simulation process. For that, this entity delivers events to the agents to steer the simulation process.
- the *observer* collects some information from the spatial grid environment and displays it to the end user.

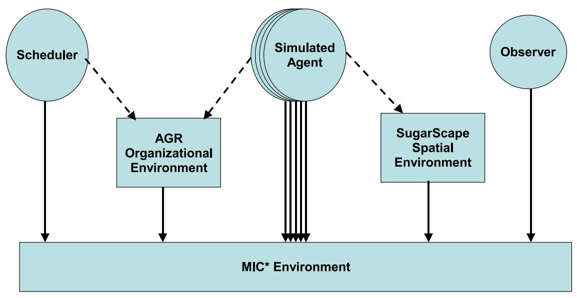

Scheduler
Simulated Agent
Observer
AGR Organizational Environment
SugarScape Spatial Environment
MIC* Environment

Figure 8: Agents and their environments in the application.

As presented in figure 8, several environments have been identified in order to capture the different aspects of the system. These environments are: *(i)* the AGR organizational environment,*(ii)* the MIC* environment, *(iii)* and the SugarScape grid environment.

The AGR organizational environment has been presented in section 4.2.4. This environment holds the simulated agents and the scheduler. In fact, a special group named *sugarscape* is created: the scheduler plays the role of *scheduler* within this group and the agents play the role of *simulation_agent*.

The MIC* environment has been presented in section 4.2.2. This environment is used as an infrastructure and holds all the agents and other environments. It is important to notice that a specific interaction space is associated to each aspect of the application domain. For instance, a first interaction space is created to allow the observer to communicate with the *SugarScape* spatial environment, a second interaction space is created between the simulated agents and the AGR organizational environment and a third interaction space is created between the scheduler and the AGR organizational environment.

Finally, the characterization of the *SugarScape* grid environment is given as follows:

- the ontology of this environment defines the following concepts: location, cell, resource, consumption of resource, and movement.
- the action means offered by this environment are represented by the movement actions and the consumption of resources.
- the perception means offered by this environment are represented by the location of the agents. In fact, according to their location the agents can perceive the available resources within their vicinity.
- the interaction functions defined by this environment maps the resources to the locations of the agents. The *SugarScape* grid environment calculates for each agent what resources are available within its vicinity.
- localization function: each agent is located within this environment by the cell that it occupies.

This system has been implemented and the outputs of the observer agent are presented in figure 9. The process of building such a MAS that merges several aspects such as: the management of the dynamics of the simulated system, the management of the simulation process and the visualization and interpretation of simulation outputs, is made clearer at both the design and implementation levels. This is due to the separation of concerns and the modularity brought by the multi-environments approach. Moreover, each environment is concerned only with a specific aspect and can be developed independently from other environments. To include an additional environment that models another aspect of the application, one has only to describe how this environment implements the agent-environment relationship and to define the set of the deployed agents. Existing environments have not to be redefined or modified.

6 Conclusion

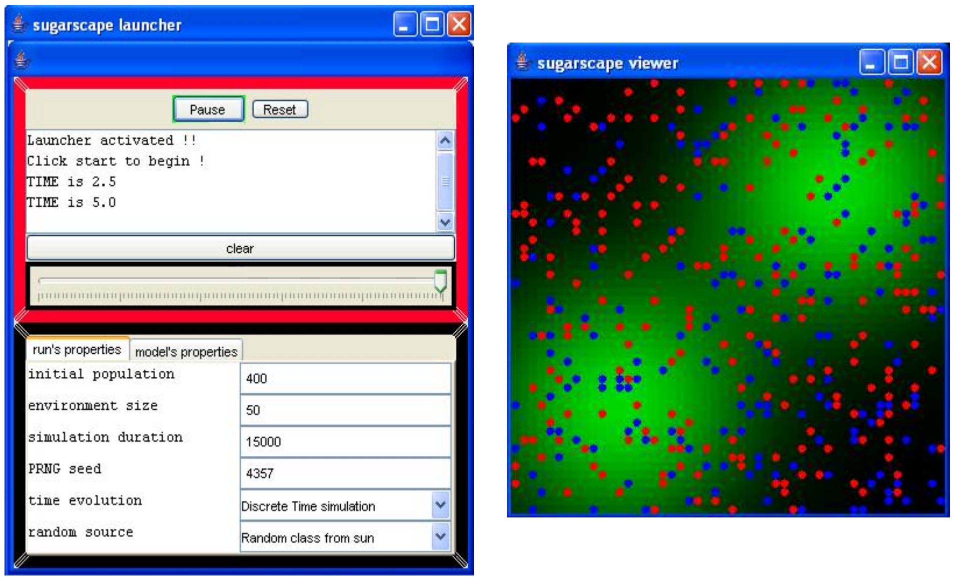
In this paper we have challenged an implicit hypothesis of MASs stating that the agents exist in a single, common and shared environment.

In fact, an agent can be associated with several environments. Each environment captures a specific aspect of the application domain. To reach this point, we have characterized the agent-environment relationship by the following: *(i)* ontology of the environment, *(ii)* perception means, *(iii)* action means and *(iv)* interaction functions. Besides, situated environments define another feature which is the localization function.

Once, the agent-environment relationship has been characterized, it becomes conceivable to allow its multiple instantiation. So, the agents can exist in several and independent environments. We have also seen that the agent-environment relationship has to be distinguished from the means which are used by an agent to access its environments. In fact, an agent can exist in an environment A and use an environment B as a communication medium to access A. This is typically the schema that is generally used to access an application level environment using an infrastructure environment. Still, all these types of environments are captured uniformly using the proposed characterization.

From an engineering point of view, the multi-environments approach brings the necessary modularity and separation of concerns to build MASs that address multi-aspects problems and domains. This has been shown for instance by the *SugarScape* simulation where several aspects of the application have to be considered. If the design models and the implementation of such a system do not explicitly reflect *all* these aspects then some parts of the system would have been implemented in an *ad hoc* manner. Consequently, it would be impossible to capture the dynamics of the whole MAS. For instance, if the infrastructure was ignored in the presented system, then some behaviors of the global MAS would be neither explained nor understood. For instance, the infrastructure, as an environment, acts actively and influences the dynamics of the entire MAS. As illustrated by the example, the multi-environment approach to build MASs can bring an appreciable flexibility within MASs to address complex domains of applications that are not reducible to only one aspect.

As highlighted by D. Weyns and T. Holvoet in this volume [17], the engineering of environments for MASs is still in its infancy and further investigations have to be done considering the environment as a first order abstraction. Concerning the multi-environments approach proposed in this paper, an important research track will be to establish a classification of agent-environment relationships with respect to the five points which have been identified, enabling reusability of agent-environment relationship classes both at the conceptual level and implementation level. For instance, environments that exploit pheromone infrastructures may be considered as a particular instance of the physical environment class. So, one of the primary objec-



```
Launcher activated !!
Click start to begin !
TIME is 2.5
TIME is 5.0
```

initial population	400
environment size	50
simulation duration	15000
PRNG seed	4357
time evolution	Discrete Time simulation
random source	Random class from sun

Figure 9: The outputs of the observer during the simulation process.

tives of our future work is to establish a hierarchy of classes for each part of the agent-environment relationship. Such a taxonomy is necessary developing a clearer understanding of what is really beyond the notion of "environments for MASs".

References

[1] S. Bandini, S. Manzoni, and G. Vizzari. A Spatially Dependant Communication Model for Ubiquitous Systems. In Weyns et al. [18], pages 74–90.

[2] J. M. Epstein and R. L. Axtell. *Growing Artificial Societies*. Brookings Institution Press, Washington D.C., 1996.

[3] J. Ferber. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison-Wesley Longman Publishing Co., Inc., 1999.

[4] J. Ferber and O. Gutknecht. A meta-model for the analysis and design of organizations in multi-agent systems. In Y. Demazeau, editor, *Proceedings of the 1998 International Conference on Multi-Agent Systems ICMAS98, Cité des Sciences - La Villette, Paris, France, July 4-7*, pages 128–135. IEEE Computer Society Press, Los Alamitos, CA, 1998.

[5] J. Ferber, F. Michel, and J.-A. Báez-Barranco. AGRE: Integrating Environments with Organizations. In Weyns et al. [18], pages 48–56.

[6] D. Gelernter. Generative communication in Linda. *ACM Transactions on Programming Languages and Systems*, 7(1):80–112, 1985.

[7] A. Gouaïch. *Movement, Interaction, Calculation as Primitives for Everywhere and Anytime Computing*. PhD thesis, Université Montpellier II, Montpellier, France, 2005.

[8] A. Gouaïch, F. Michel, and Y. Guiraud. MIC*: A deployment environment for autonomous agents. In Weyns et al. [18], pages 109–126.

[9] O. Gutknecht, J. Ferber, and F. Michel. Integrating tools and infrastructures for generic multi-agent systems. In E. André, S. Sen, C. Frasson, and J. P. Müller, editors, *Proceedings of the fifth international conference on Autonomous agents, AA 2001, Montreal, Quebec, Canada, May 28-June 1*, pages 441–448. ACM Press, New York, NY, USA, 2001.

[10] F. Michel. An Introduction to TurtleKit : a Platform for Building Logo Based Multi-Agent Simulations with MadKit. Technical Report RR LIRMM 002215, Laboratoire d'Informatique de Robotique et de Microélectronique de Montpellier, LIRMM, CNRS, Montpellier, 2002.

[11] J. Odell, H. V. D. Parunak, M. Fleischer, and S. Brueckner. Modeling agents and their environment. In F. Giunchiglia, J. Odell, and G. Weiss, editors, *Agent-Oriented Software Engineering III: Third International Workshop, AOSE 2002, Bologna, Italy, July 15, 2002. Revised Papers and Invited Contributions*, volume 2585 of *Lecture Notes in Computer Science LNCS*, pages 16–31. Springer, Berlin, 2003.

[12] A. Omicini and F. Zambonelli. TuCSoN: a coordination model for mobile information agents. In D. G. Schwartz, M. Divitini, and T. Brasethvik, editors, *1st*

International Workshop on Innovative Internet Information Systems (IIIS'98), Pisa, Italy, June 8–9, pages 177–187. IDI – NTNU, Trondheim (Norway), 1998.

[13] M. Pollack and M. Ringuette. Introducing the Tileworld: experimentally evaluating agent architectures. In T. Dietterich and W. Swartout, editors, *Proceedings of the Eighth National Conference on Artificial Intelligence, Boston, MA, July 29–August 3, 1990*, pages 183–189. AAAI Press, Menlo Park, CA, 1990.

[14] M. Resnick. *Turtles, termites, and traffic jams: explorations in massively parallel microworlds*. MIT Press, Cambridge, MA, 1994.

[15] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition, 2003.

[16] The NetLogo system.
http://ccl.northwestern.edu/netlogo/ (date of last access: 2005/08/25).

[17] D. Weyns and T. Holvoet. On the Role of Environments in Multiagent Systems. *In this volume*.

[18] D. Weyns, H. V. D. Parunak, and F. Michel, editors. *Environments for Multi-Agent Systems, First International Workshop, E4MAS 2004, July 19, 2004, New York, NY, USA, Revised Selected Papers*, volume 3374 of *Lecture Notes in Computer Science LNCS*. Springer, Berlin, 2005.

[19] D. Weyns, H. V. D. Parunak, F. Michel, T. Holvoet, and J. Ferber. Environments for Multiagent Systems: State-of-the-Art and Research Challenges. In Weyns et al. [18], pages 1–47.

[20] M. Wooldridge and P. Ciancarini. Agent-Oriented Software Engineering: The State of the Art. In P. Ciancarini and M. Wooldridge, editors, *Agent-Oriented Software Engineering: First International Workshop, AOSE 2000, Limerick, Ireland, June 10, 2000. Revised Papers*, volume 1957 of *Lecture Notes in Computer Science LNCS*, pages 1–28. Springer, Berlin, 2001.

[21] M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.