Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 56 (2015) 526 – 531

Procedia Computer Science

International Workshop on Communication for Humans, Agents, Robots, Machines and Sensors (HARMS 2015)

A New Perspective on Multi-Agent Environment with SARL

Sebastian RODRIGUEZ[a,*], Stéphane GALLAND[b], Nicolas GAUD[b]

[a]GITIA Laboratory, Facultad Regional Tucumán, Universidad Tecnológica Nacional, San Miguel de Tucumán, Argentina
[b]Univ. Bourgogne Franche-Comté, UTBM, IRTES, Belfort, France

Abstract

The environment is now considered as a first class abstraction in multiagent systems. However, the boundary between real and simulated environment and the application logic is not so well defined. Depending on applications, the environment as a space shared between agents may integrate physical, communication or social dimensions where agents interact. In this paper, authors introduce an agent environment model supporting the intrasic distributed and hierarchical natures of the environment. It is defined using the fundamental concepts provided by metamodel related to the SARL programming language.

Keywords: Multiagent systems, Environment, Interaction, Programming language

1. Introduction

Current trends in industry and research towards Cyberphysical Systems (CPS) and the Internet of Things (IoT) let us foresee an immersive environment where intelligent systems, humans, robots and a multitude of components interact. Multiagent Systems have emerged as a promising technology to develop software applications in this context. The environment, as a space shared between agents, is a key component of multiagent systems[1]. Depending on systems, this space may integrate physical, communication or social dimensions where agents interact.

Each dimension of the environment has its own processes and rules to support its interaction model. For instance, in the physical dimension, the rules may be based on the location of the agents, i.e. the interaction between agents is allowed according to the distance or any existing obstacle between them. These rules are independent of the agents, i.e. even if it is initiated by the agents, the interaction occurs independently of the decision process of the agents, and is performed by the environment. When the environment has several dimensions, the issue is to model and to manage the relations between them. In most systems, these environmental dimensions are considered either independent and connected only through the decision process of the agents. Interactions resulting from the combinations of input information depend of the agent's decision process. The problem is that an agent should not be the place where interaction rules are triggered because the interaction could not rely on its own responsibility. For instance, a rule

* Corresponding author. Tel.: + 54 381 421 7150.
E-mail address: sebastian.rodriguez@gitia.org

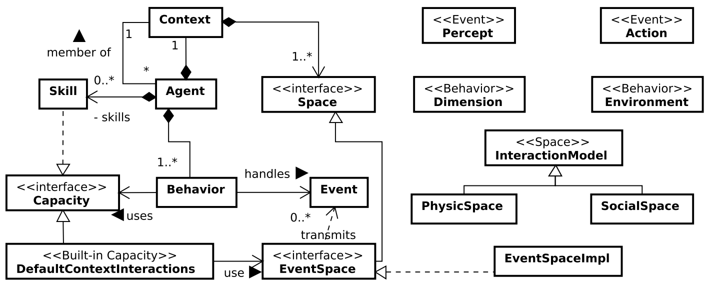

Fig. 1. Major concepts in the SARL metamodel

could be designed to regulate the communication in the social dimension according to information coming from the physical dimension (is the receiver agent physically able to receive the message?). The agent cannot decide by itself. This rule should be triggered by the environment using information coming from the physical and communication dimensions.

A language to implement MAS should enable designers and developers to map these concepts in an intuitive way, providing the appropriate concepts and infrastructure. In this work, we present an overview of how this can be achieved within the SARL programming language. This work proposes a new view on the environment structure and associated model. Further, we introduce an extension of the SARL language that enables the definition of environment level application logic simply, and with a consistent syntax.

This paper is structured as follows : Section 2 presents a brief overview of SARL's main concepts and Section 3 introduces the environment structure and model. Then, Section 4 describes a case study to illustrate the concepts previously discussed. Section 5 discusses our proposal according to related works. Finally, Section 6 concludes and gives some future work perspectives.

2. SARL Overview

SARL is a general-purpose agent-oriented programming language[2]. This language aims at providing the fundamental abstractions for dealing with concurrency, distribution, interaction, decentralization, reactivity, autonomy and dynamic reconfiguration. SARL provides a reduced set of key concepts that are considered as essential for implementing multi-agent systems: Agent, Space, Capacity and Skill (see figure 1).

Space is the abstraction to define *an interaction space between agents or between agents and their environment*. In the SARL toolkit, a concrete default space, which propagates events, called `EventSpace` (and its implementation `EventSpaceImpl`), is proposed. *An **Agent** is an autonomous entity having a set of skills to realize the capacities it exhibits.* An agent has a set of built-in capacities considered essential to respect the commonly accepted competences of agents, like the autonomy, reactivity, pro-activity and social capacities. The agent has the capacity to incorporate behaviors that will determine its global conduct. *Behavior maps a collection of perceptions represented by Events to a sequence of Actions.* By default, the behaviors of an agent communicate using an event-driven approach. *An Event is the specification of some occurrence in a Space that may potentially trigger effects by a listener. A **Capacity** is the specification of a collection of actions.* This specification makes no assumptions about its implementation. It could be used to specify what an agent can do, what a behavior requires for its execution. Indeed, an action is a specification of a transformation of a part of the designed system or its environment. This transformation guarantees resulting properties if the system before the transformation satisfies a set of constraints. *A **Skill** is a possible implementation of a capacity fulfilling all the constraints of this specification.* Each of these generic concepts of the SARL's metamodel is associated to language statements. These statements are used for implementing the multidimensional environment model presented in this paper.

3. SARL Environment Model and Structure

In this work, we advocate that a MAS application is composed of three layers from the environment's standpoint. They are produced by the overlapping of the Application Logic (dotted line) and what is usually called "Environment" (dashed line) as depicted in Figure 2.

Most MAS applications concentrate mainly in "application agents". We describe their behavior, beliefs, interactions, social structures, etc. Agents interact with their "Environment" and it is sometime neglected that this "environment" has two distinct layers: External Environment and Soft Environment.

The External Environment contains all resources / objects that are beyond the boundaries of the system to be developed that the agents may interact with (e.g. physical world objects, robot bodies, etc). While we usually associate the external environment with physical objects, it may also represent software systems (e.g. Databases, webservices, legacy systems, etc). This environment can be "real" (e.g. the physical robot itself) or simulated. Ideally, once the application is implemented the simulated world can be replaced with the real world without any changes in the implemented system.

If this scenario is to be possible, we need mechanisms to implement the environment rules in a way that is independent of the external environment. The Soft Environment contains the set of rules that the environment is supposed to enforce and a set of virtual objects that agents use to achieve their tasks and goals.

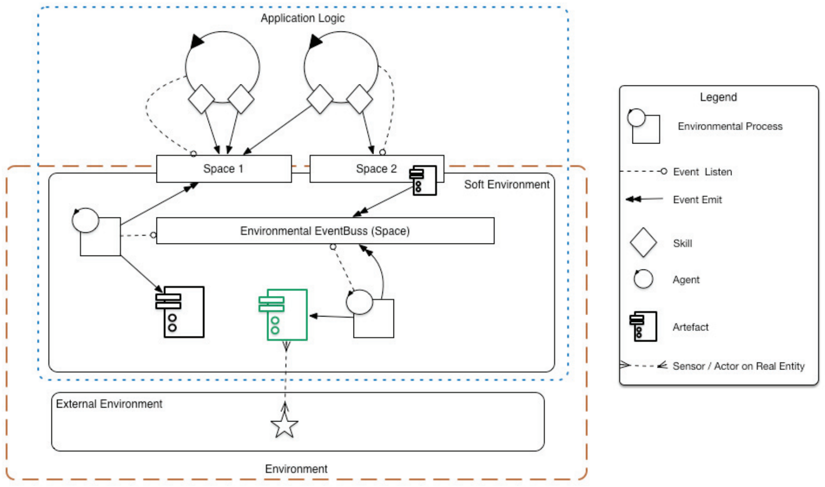

Fig. 2. Overview of the Environment model

The Laws of the Universe (LoU) dictate the set of unbreakable rules to which agents, artifacts and any other elements of the system are attached to. For instance, in a physical world, the laws of the universe will contain the laws of physics, but it may contain any other rules that the agents are bound to and are not attached to their decision processes and autonomy. The degree of precision in these laws definitions is related to the application at hand.

In our approach, Environment Processes are responsible to enforce the Laws of the Universe in a particular MAS definition. Multiple processes can be defined to ensure a single law in order to improve modularity and reusability. For example, one process may update 3D positions of physical objects according to the forces applied to them, while a second process generates endogenous forces like friction, gravity, etc, to be used by the former. While processes may look similar to agents in certain scenarios, processes have no goal-oriented behavior. They are strictly attached to one or more LoUs, and their purpose in the system is to ensure their application.

The environment of a MAS contains a set of resources and objects which future states are influenced by agents and modified by the environment according to the LoUs. These "Environmental Elements" are called "Artifacts". Artifacts are characterized by a set of "observable properties", "operations" used to modify their internal states, and "Events" they emit. In our approach, artifacts are manipulated by a single environmental process and their events are not emitted into the "Environmental Space" directly but by its controlling process. It is the process' responsibility to ensure coordination between concurrent requests of other processes or agents according to the LoU. Artifacts may also also be used to have extended representations for physical objects. For instance, in a robotic application, an artifact can be used to control the robot's body and provide a simplified API.

4. Environment programming in SARL and Case Study

To illustrate these ideas, let us consider a classic example of foraging robots using a pheromone approach for coordination purposes within a shared environment. In this application, robots are not able to deposit chemicals in the physical environment and, therefore, the application needs to implement this feature. We define agents as the robot's minds. Pheromone approaches usually rely in a number of rules that are enforced by the environment such as dissipation and evaporation. These rules should not be enforced by application agents but rather by processes inside the environment.

```
1  /* Package and Import sections omitted*/
2
3  /* Environment level Events */
4  event WorldState  { /*State of the World implementation omitted */ }
5  event PheromoneDelta { var delta: float; var edge: int }
6
7  /*Agent level events */
8  event NodeReached  { var edges: List<Edge> }
9  event TravelEdge { var edge: int }
10
11 behavior Evaporation {
12   uses DefaultContextInteractions
13
14   on WorldState {computeDelta(occurrence).emit}
15   def computeDelta(state: WorldState): PheromoneDelta { /* Generate Evaporation */ }
16 }
17
18 behavior Dissipation {
19   uses DefaultContextInteractions
20
21   on WorldState { computeDelta(occurrence).emit }
22   def computeDelta(state: WorldState): PheromoneDelta { /* Generate Dissipation */ }
23 }
24
25 behavior WorldUpdate  {
26   uses DefaultContextInteractions,Schedules
27
28   var world = World.createRandomWorld /* World is an Artifact */
29
30   on PheromoneDelta{world.edgeById(occurrence.edge).update(occurrence.delta)}
31
32   on TravelEdge {
33     val antPhero = world.getAntBody(occurrence.source).phermoneLevel
34     emit(new PheromoneDelta => [edge = occurrence.edge; delta = antPhero])
35     in(estimateArrivalTime(occurrence.source,occurrence.edge)) [
36       emit(new NodeReached()=>[edges=world.outgoingEdgesOfDestination(occurrence.edge)])
37     ]
38   }
39
40   def estimateArrivalTime(ant: Address, edge: int): long { /* Implementation omitted */ }
41 }
42
43 /* Application Agent - Ant like robot's mind */
44 agent Ant {
45   uses DefaultContextInteractions
46
47   on NodeReached {
48     val e = occurrence.edges.selectEdge
49     emit(new TravelEdge => [edge = e])
50   }
51
52   def selectEdge(edges: List<Edge>): int { /* Agent decision process */ }
53 }
```
Listing 1. Environment Implementation Snippet

Listing 1 presents the fundamental components to implement the proposed example in SARL. LoUs of Evaporation and Dissipation are implemented as separate behaviors (lines 11 and 18). When they receive a `WorldState` event, they compute the appropriate `PheromoneDelta` to be applied to the considered edge of the graph, and send the event within the Environmental Space (e.g. line 14 in the Evaporation behavior). A third behavior is defined (line 25) to ensure the coherence of the virtual representation that will contain pheromone levels, namely `WorldUpdate`. This process controls the world's representation (i.e `World` artifact declared in line 28). Additionally, this process will periodically, according to the desired time management model, send `WorldState` updates and trigger the `Evaporation` and `Dissipation` behaviors.

The robot's mind is implemented as an `Ant` agent (line 44). When the agent reaches a node (notified via `NodeReached` event), it will select an outgoing edge from and inform its decision to the environment using the `TravelEdge` event. On the environment's side, when the `WorldUpdate` behavior perceives this event it will update the pheromone of the selected edge (using a `PheromoneDelta` event as other processes) and schedule a `NodeReached` event according to the estimated arrival time of the ant. A reference overview of these interactions is presented in Figure 3.

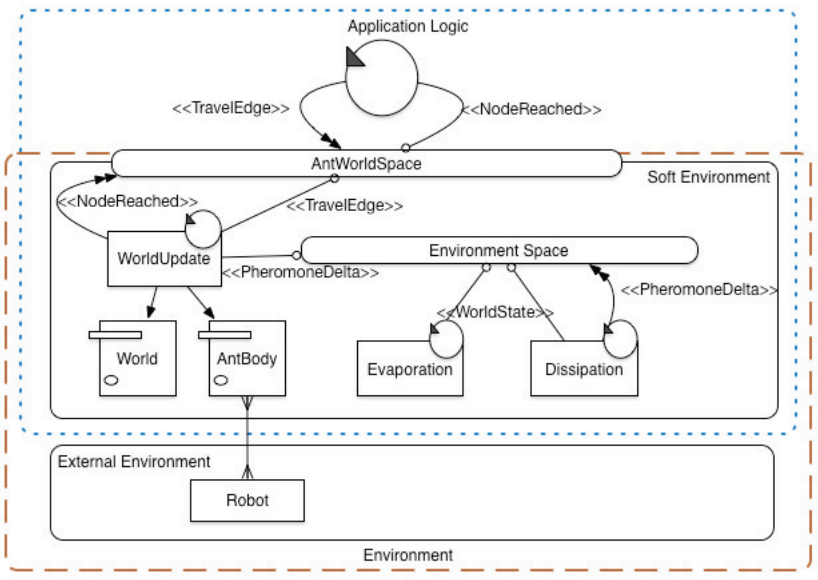

Fig. 3. Foraging Robots Environment Overview

Finally, when the environment is created, it spawns its processes, as shown in Listing 2.

```
agent Environment{
  uses DefaultContextInteractions
  on Initialize { Process.spawn(Evaporation, Dissipation, WorldUpdate) }
}
```
Listing 2. Basic Environmental Process Implementation

5. Related Works

Our inspirations for the physical environment are the models for the simulation of crowds and traffic into virtual environments[3]. The CArtAgO[4], supporting the artifacts, and the Smart Object[5] models are also inspirations. They propose similar interaction models between agents and objects in the environment, and the definition of the latter. Galland et al.[6] have proposed a multidimensional agent environment model. The internal behavior and structures of the environment are not detailed in this proposition. Our model that is proposed in this paper refines the agent environment for supporting the intrinsic distributed and hierarchical natures of the environment.

The problems related to the interaction between an agent, and the physical environment have been treated with different perspectives. One of the models used in our approach is the Influence-Reaction model[7]. It supports the simultaneity of the actions in an environment by considering the interactions initiated by agents as uncertain, and detecting and resolving the conflicts between the interactions. This approach can be compared to the concept of artifact[8], which proposes to model the objects in the environment. They provide a set of actions that can be applied on each of them. A similar model named smart object is proposed for virtual environments[5]. The IODA model and its extension PADAWAN[9] allow modeling the interactions between the agents and the various dimensions of the environment by assuming that every entity is an agent. Our model is partially incompatible with this vision in the context of the physical dimension modeling. Indeed, the bodies of the agents are not agents.

In the communication environment models, the environment is a shared space in which agents drop off or withdraw filters that describe the context of their interactions, in order to manage their multiparty communications[10,11]. These filters are managed by the environment. The physical environment is not separated from the communication environment, and filters always involve an agent.

Several organizational approaches consider the environment[12,13].In the context of this paper, the key element is the introduction of the concept of space as an abstraction for organizational groups and spatial areas. However, these models do not explicitly propose to consider the physical and communication dimensions jointly, as well as their direct interactions.

6. Conclusion and Future Work

The environment plays an important role in MAS design and development. We need to provide a platform for MAS that seamlessly integrate agents and the physical and virtual world they involve in. SARL aims at providing such a platform by taking advantage of the concept of Space as an interface between the Environment and Agents. Even more, different types of spaces can be defined with their own interaction mechanisms. In this paper we present the foundations of the environment definition for the SARL Language. In our approach a virtual environment, called Soft Environment, is defined as a abstraction layer and support to program application level logic delegated to the environment. Future works will provide means to support multiple environment definitions considering the holonic capabilities of SARL.

References

1. Weyns, D., Omicini, A., Odell, J.. Environment as a first-class abstraction in multi-agent systems. *Autonomous Agents and Multi-Agent Systems* 2007;**14**(1):5–30.
2. Rodriguez, S., Gaud, N., Galland, S.. SARL: a general-purpose agent-oriented programming language. Warsaw, Poland: IEEE Computer Society Press; 2014, .
3. Galland, S., Gaud, N.. Holonic Model of a Virtual 3D Indoor Environment for Crowd Simulation. In: *International Workshop on Environments for Multiagent Systems (E4MAS14)*. Springer; 2014, .
4. Ricci, A., Viroli, M., Omicini, A.. CArtAgO: A Framework for Prototyping Artifact-Based Environments in MAS. In: *E4MAS*. Springer Verlag; 2007, .
5. Kallmann, M., Thalmann, D.. Modeling Objects for Interaction Tasks. In: *Proc. Eurographics Workshop on Animation and Simulation, 1998*. 1998, p. 73–86.
6. Galland, S., Balbo, F., Gaud, N., Rodriguez, S., Picard, G., Boissier, O.. Contextualize agent interactions by combining social and physical dimensions in the environment. In: Demazeau, Y., Decker, K., editors. *13th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS)*. 2015, .
7. Michel, F.. The IRM4S model: the influence/reaction principle for multiagent based simulation. In: *Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS07)*. Honolulu, Hawaii, USA: ACM. ISBN 978-81-904262-7-5; 2007, doi:10.1145/1329125.1329289.
8. Ricci, A., Viroli, M., Omicini, A.. Programming MAS with Artifacts. In: *International Workshop on Programming Multi-Agent Systems*. Springer Verlag; 2005, .
9. Picault, S., Mathieu, P., Kubera, Y.. PADAWAN, un modèle multi-échelles pour la simulation orientée interactions. In: *JFSMA*. Cépaduѐs; 2010, p. 193–202.
10. Badeig, F., Balbo, F., Pinson, S.. A contextual environment approach for multi-agent-based simulation. In: *ICAART 2010 - Proceedings of the International Conference on Agents and Artificial Intelligence, Volume 2 - Agents, Valencia, Spain, January 22-24, 2010*. 2010, p. 212–217.
11. Saunier, J., Balbo, F., Pinson, S.. A formal model of communication and context awareness in multiagent systems. *Journal of Logic, Language and Information* 2014;:1–29.
12. Ferber, J., Michel, F., Baez, J.. AGRE: Integrating Environments with Organizations. In: *Environments for Multi-Agent Systems*; vol. 3374 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. ISBN 978-3-540-24575-9; 2005, p. 48–56.
13. Piunti, M., Ricci, A., Boissier, O., Hübner, J.. Embodying organisations in multi-agent work environments. In: *IEEE/WIC/ACM Int. Conf. on Web Intelligence and Intelligent Agent Technology (WI-IAT 2009)*. Milan, Italy.; 2009, .