A Look Inside of CORMAS Platform: An Analysis of the Requirements for an Agent-Based Platform

Maria das Graças B. Marietto, Edson Pinheiro Pimentel, André Filipe de M. Batista, Emerson Aguiar Noronha, Guiou Kobayashi
Centre of Mathematics, Computation and Cognition
Federal University of ABC
Campus Universitário Santo André
São Paulo, Brazil
{maria.marietto, edson.pimentel, andre.batista, emerson.noronha, guiou.kobayashi}@ufabc.edu.br

ABSTRACT: *Since its early stages, science has been developing and using models to explain and understand our life, our world and the universe. Among different approaches to develop these models, the Multi-agent Based Systems (MABS) approach has been gaining attention and importance as a third way of doing science. This third way consists of using MABS to develop models capable to reproduce properties of a complex system. Thereby, investigating the foundations to the development of such models, this paper presents the main concepts of complex systems, as well as the computational requirements implemented by the CORMAS, which is a MABS platform.*

Keywords: Agent-Based Platform, CORMAS, Multi-Agent Based Simulation, Complex Systems, Emergent Behavior

Received: 13 July 2009, Revised 16 August 2009, Accepted 21 August 2009

1. Introduction

A simulation model is a kind of model that aims to represent a target system. More specifically, according to [6] a multi-agent based simulation (MABS) model is based on a relationship between an individual and a computer program, being possible to simulate an artificial world composed by computational interactive entities. In this context there is a one-to-one equivalence between observable entities in the target (or an aggregate of observed entities) and agents in the simulated environment.

In the last decade the role of simulation has been progressively acquiring importance in Multi-Agent Systems (MAS) and is undoubtedly growing. The multitude of intended use and application areas have led the MABS area to a rich interdisciplinary cross-fertilization and is enlarging the set of traditional research ends um MAS, beyond the analysis and design of computer applications. We consider that at the present maturing stage of MABS a more precise portray of the area is in need, specially related with a requirement analysis for computer platforms. Such analysis seems to play a fundamental role, since there is not a consensus with respect to specification of services and its standards of behavior. In [8] we presented a detailed requirement analysis with a set of functional and non-functional requirements guiding the specification of MABS platforms.

In this paper we use such specification to analyze in more details the platform CORMAS. This analysis aims to detect the requirements that are more developed at the platform CORMAS to investigate the hypothesis that there are a number of requirements that are more or less appropriate for the kind of simulation model that are developed at the platform CORMAS.

This paper is structured as follows. First, in section 2 we present the foundations of complex systems, which is necessary to a fully understand of MABS based models. Next, in section 3 is presented a set of requirements that characterize both the general structure and global behavior of MABS platforms. Then, in the section 4 there is an analysis of the CORMAS platform, taking into account the requirements presented in the section 3. After that, the section 5 analyzes the kind of simulation models developed in the platform CORMAS, and investigate the relationships between such models and the requirements offered by the platform. And finally, in section 6 we draw some conclusions.

2. Multi-agent Based Systems (MABS)

The contemporary problems need more sophisticated responses, in less time and based on specific methods of solution, and the theories and approaches based on complex systems can handle this kind of demand in a efficient form. One approach for

tackling this challenge lies in the Multi-agent Based Systems (MABS). This area is formed by the intersection of the areas of Distributed Artificial Intelligence (DAI) and Computer Simulation, aiming to use the framework provided by the DAI in the modeling, implementation and analysis of simulations that lead to a better understanding of a variety of complex systems. The agents defined in the DAI area are able to provide a rich metaphor of sets of interactive entities, like live organisms, as well as its patterns of relationships. The MABS approach allows to create models representing complex systems, such as biological, socio-cognitive or many-particle systems.

A MABS model is based on the idea that is possible to simulate an artificial world populated with interactive computational entities. Simulation can be achieved by transposing the population of a target system to its artificial counterpart. In this sense, an agent is equivalent to an entity of the target system, or a group of them. The sources of analogy between agent-based technologies and models of actual complex systems have created an intense interdisciplinary effort, opening new interfaces of research across various disciplines under the umbrella of a new scientific field, the Multi-Agent Based Simulation (MABS). In a multi-agent based simulation, the relationship between agents can be specified in several ways, ranging from reactive approaches to deliberative attitude approaches.

The MABS area offers an adequate infra-structure to model, for example, processes related to complex social interactions such as coordination, collaboration, group formation, evolution dynamics of norms and conventions, free will, conflict resolution, among others. The emergent process related to this kind of dynamic and unpredictable systems can be achieved by relating local and global behavior allowing, therefore, the analysis of how agents can affect the environment and other agents, and vice-versa. This leads to explicit chains of cause and effect of how internal agent components affect the agents behavior, how this behavior affects the agency and, dialectically, how the agency affect its agents components [21, 24]. The importance of simulation in general, and agent-based simulation in particular becomes clear as some authors have gone as far as to consider it as a "third way of doing science", in addition to traditional deduction and induction reasoning [23, 15].

2.1 Main Aspects of Multi-Agent Models

Agents in MABS are modeled to behave autonomously, with a degree of initiative, in a self-organized model based on simple local rules of interactions. There is no central authority in charge, because the ultimate goal of such model is to allow the emergence of system-level phenomena resulting from these local interactions between agents themselves and the environment. A more specific definition of agent would be of an discreet entity with its own objectives and behaviors, examples of agents include people, organizations, social insects, robots, and so forth.

2.2 Bottom-Up Modeling

There are two main approaches to addressing the substantive question of how nature works: the Top-down and the Bottom-up approach [15]. The first one works with representations which are symbols and computations as rules (algorithms) that operate on symbols. Whereas the second approach is rooted in many natural sciences, such as biology, neuroscience, and evolutionary theory, drawing on data concerning how the most elementary units work, interact, and process information. Modeling a complex system using a top-down approach would prove to be much complex and not appropriate to model the behavior of complex systems. The MABS based models are created in a Bottom-up approach, which means that is concerned with the fundamental behaviors and rules that drives the entities from a specific domain.

2.3 Complex Systems

The word "complexity" has roots in two Latin words: "complexus" which means "totality" and "completere" which means "to embrace" [17]. A complex system is a system featuring a large number of interacting components whose aggregate activity is nonlinear, not derivable by summing the behavior of individual components, and typically exhibit self-organization [15]. In sum, emergence is what characterizes a complex system and its result is the self-organization of its entities.

The fact is that it's not possible to understand complex behavior by examining its component parts alone. Thereby, agent-based models are often useful under conditions of deep uncertainty where reliable prediction of the future, either in a best estimate or probabilistic sense (such as the ones in traditional simulation models) are not possible [16].

2.4 Emergent Behavior

A phenomenon is emergent if it requires new categories to describe it which are not required to describe the behavior of the underlying components [21, 18]. And, according to [17], the emergent process of a system can be described as the large-scale

effects of locally interacting agents, noticed as non self-evident, stable macroscopic patters arising from individual agent's local rules of interaction. In some models, the emergent properties can be formally deduced, but they can also be unpredictable and unexpected as anticipating the consequences of even simple local interactions sometimes proves to be a hard task [19].

2.5 Open Systems and Self-Organization

Self-organization is a process where the organization of a system is not guided or managed by an outside source. According to [19], a basic feature of self-organized systems is the means by which they acquire their order and structure. In self-organizing systems, pattern formation occurs through interactions internal to the system, without intervention by external directing influences.

Self-organization is considered an effective approach for modeling the complexity in modern systems, allowing the development of systems with complex dynamics and adaptable to environmental perturbations without complete knowledge of future conditions. The self-organization approach promotes the development of simple entities that, by locally interacting with others sharing the same environment, collectively produce the target global patterns and dynamics by emergence [20, 23]. In simulations, thereby, the patterns found in the nature or in the society can be build in the basis of a self-organizing process, or may involve other mechanisms instead: following a leader, building a blueprint, following a recipe or using a template [22].

3. Requirements of MABS Platforms

Currently, there are a large number of computational systems in MABS. While analyzing such systems it is possible to detect certain groups of requirements that characterize different kinds of technologies. Such groups of requirements will be called facilities. In [8] we identified four facilities that can be found in these computational systems: technological, domain, development and analysis. In addition, we described the requirements that are related with each facility. In this work, computational systems that present at some degree of development these four facilities will be called MABS platforms.

In this section we present a small subset of these requirements. The selection was guided with the aim of identifying those that (i) seem to compose a basic structure of MABS platforms; (ii) are common to, or that better differentiate, current existing platforms. In the rest of this section we will describe requirements clustered around the foresaid facilities. In section 3 we will use these requirements to guide the analysis of the platform CORMAS.

3.1 Technological Facilities

Technological facilities encompass services that (i) intermediate the platform with both the operational system and the network services; (ii) provide services to support controlled simulation worlds.

A. Manage Agents Life Cycle: A platform should provide services to control the life cycle of agents, considering a multitude of agent status such as active, suspended, sleeping and killed.

B. Manage Agent Connection: A platform should provide services to manage information relative to agents' provided services and capabilities, such as yellow pages and broker services.

C. Manage Communication: A platform should provide services for asynchronous and synchronous message passing among the agents.

D. Manage Scheduling Techniques: A platform should support controlled simulations and allow repeatability, providing libraries including at least one scheduling technique, like discrete-time and event-based simulation [see 10].

E. Manage Security: A platform should manage and perform security checks on the execution of the simulation, especially considering distributed execution. Without services like certification of MAS components, the obtained results can be distorted, for instance, with accepting messages proceeding from non-authorized components.

F. Manage Mobility: Mobility describes the ability of an agent to navigate within electronic communication networks [12]. A platform should provide: (i) adequated agents architectures; (ii) services to transmit and to receive agents through the network; (iii) persistence mechanisms to maintain the agents' states when travelling.

3.2 Domain Facilities

A domain can be defined by the environment and causal knowledge of how the objects and agents interact with one another. Domain facilities embrace two types of requirements:

The first type deals with requirements that have a considerable importance in the modeling and implementation of domains. This importance is assessed by considering how the absence of requirements may hamper, or even inhibit, the modeling and implementation of domains.

The second type deals with requirements whose technological and logical functionalities must be modeled in a personalized way according to the relevant domain. They usually demand further implementation work.

A. Launch Agents: The platform should provide agent templates related with different manners to launch agents like threads, applets and objects. This requirement determines, for instance, if the platform allows distributed simulation.

B. Manage Intentional Failures: From a technical-operational point of view, there are two classes of intentional failures that can be manipulated in a simulation. The first class, called operational failures, works with disturbances on the technical-operational infrastructure (corrupted messages, server failures, etc). The second, called logical failures, manipulates patterns of behavior that can be viewed as dysfunctional exceptions in the simulation system. Such failures are strongly domain dependent, and the user may have to engage in further implementation work in order to utilize them. The platform should offer: (i) libraries to manipulate basic operational failures; (ii) mechanisms to store and search templates of logical failures created by the users.

C. Integrate Controlled and Non-Controlled Environments: Typically the simulation environment must be totally controlled, every event in the simulation world must be performed under the control of the simulator. These situations characterize what we call controlled environments. Nevertheless, there are cases where the agents can (or must) perform actions outside the controlled environment, in real environments. To support this functionality a platform should offer agent architectures that separate the agent domain-dependent behavior from the simulator design patterns. Also, in order to keep the simulation consistent and guarantee a good level of repeatability, when agents are running in non-controlled environments some of their events should be notified to the simulator, which should update its local view (see [11]).

D. Define the Domain: A domain independent platform allows the user to define the application area of a simulation. Regarding this functionality, a platform should provide services to assist in: (i) constructing specific agents and objects to compose the simulation environment; (ii) modelling causal roles of how the objects and agents interact with one another. On the other hand, domain dependent platforms define a priori an application area. To work with this functionality a platform should provide: (i) a repository of domain's foundation knowledge; (ii) adequated agents architectures to match with the domain's features.

E. Provide Graphical Representation of Domains: A platform should offer user-friendly graphical interfaces to help on building the graphical representation of domain(s), considering the environment, the agents and their interactions. Moreover, it should provide helping systems to guide and direct the interaction between the platform and its users.

3.3 Development Facilities

Development facilities include mechanisms and tools to construct MAS within an agent-centered approach or organization-centered approach [7].

A. Develop Agent Architectures: The platform should provide templates of generic agent architectures, from reactive to intentional agents.

B. Use Organizational Abstractions: organizational abstractions are MAS components that explicitly structure an organization. Roles and groups are the most common ones.

C. Use Ontologism. Ontologies can be powerful tools to assist simulation modelling: The platform should provide: (i) a standard ontology describing its adopted concepts and their causal relationships; (ii) mechanisms to store new ontologies; (iii) a search engine to manipulate ontologies; (iv) mechanisms to relate ontologies with implementation components.

D. Manage Messages: In this requirement a platform should offer: (i) message models with basic attributes and mechanism to extend or add additional attributes; (ii) APIs and parsers to check the correctness of agent communication languages, eventually according to given standards, like KQML.

E. Manage Communication Protocols: To work with this requirement a platform should also provide the "Manage Message" requirement. Additionally, it should provide mechanisms to store and to manipulate new message models, allowing the definition of customized protocols.

F. Provide Translation Mechanisms: Translation mechanisms are necessary when agents use different protocols and/or languages, and need communicate each other. Such functionality is based on different ontologies related to common blocks of

information, shared by components of a simulation. To offer this functionality a platform should provide [13, 14]: (i) knowledge bases containing the ontologies; (ii) tools to translate ontologies into operational data to be used by the platform.

3.4 Analysis Facilities

Analysis facilities emphasize requirements related to observation, simulation, validation and verification of models.

A. Observe Behavior Events: Behavioral events are events that can be observed by an external observer (e.g., message passing, creation/destruction of agents, data base access). The platform should provide mechanisms to select specific windows to observe behavioral events.

B. Observe Cognitive Events: Cognitive events are events in the agents' internal architectures. The platform should offer mechanisms to control the agents' internal mechanisms, in order to trigger specific observation methods.

C. Define Scenarios: A platform should provide services to configure characteristics like the type and quantity of agents, and the graphical topology of the simulation.

D. Control Tracking: A platform should provide services to follow up the execution of a simulation, in both graphical and batch modes. It also should provide services to terminate, initialize, interrupt and temporarily suspend simulations.

E. Provide Data Analysis: In a simulation the amount of output data is huge and graphical analysis at run time cannot capture all aspects of the simulation logic. This requirement should define technical indicators and decision support (e.g., graphical and statistical packages) to work in more depth with generated data.

F. Provide Sensitivity Analysis: To work with sensitivity analysis a platform should offer services that implement controlled variations of initial simulation parameters (e.g. number of agents, value of variables, etc), and provide graphical and statistical packages to observe the relationship between parameter variations and possible changes in simulation's output data.

4. Platform CORMAS: Meta Model and Technical-Operational Framework

The Common-pool Resources and Multi-Agent Systems (CORMAS) platform is being developed by the Gestion des Resources Renouvables, Environment Team at the Centre de Coopération Internationale en Recherche Agronomique pour le Dévelopment (CIRAD), France [4]. According to [1], the main goal of CORMAS is not to make predictions about the behavior of complex systems, but rather to provide a framework to help people develop new ways of thinking through building and testing of new models. It is implemented in Smalltalk using VisualWorks as programming environment. In this section we present the implicit meta model that guide the architecture of the platform CORMAS. After this, we analyze if the facilities are offered, or not, in CORMAS.

4.1 Meta Model of CORMAS

The meta model of CORMAS focuses on the development and analysis of MAS within an agent-centered approach. This meta model works with components that encompass the main concepts of an agent-centered approach. In the platform CORMAS these concepts are: entity, agent and group. Such concepts are interconnected in the modeling process of CORMAS, and this process is composed by three phases. Firstly, the entities of the model are created. Secondly, the designer defines the control and scheduling activities. Finally, it is defined what will be observed in the simulation.

The meta model of platform CORMAS is based on the metaphor of agents moving and harvesting resources in a spatial environment [3].

Entity: CORMAS works with three types of entities: passive, spatial and social. Passive entities are subdivided into passive objects and messages [3]. A passive object does not need to be situated on the spatial grid, and in this case there is not a method for locomotion. In other hand, for situated passive object there is a locomotion method (isMoveTo), but these objects are not able to find by themselves their destination, neither to perceive their local environment. A passive object related with a message is used to personalize communication protocols to be used by the agents. Spatial entities define a topological support for simulations. They hold natural resources and arbitrate the allocation of them according to pre-defined protocols based on a metaphor of physical harvest. Social entities are also called agents.

Agent: Taking into account the modeling of the agents, two functionalities are considered: locomotion in the topological grid and communication. Four types of agents can be defined: simple, situated, communicating, situated and communicating. The simple agent is very similar to the passive situated object. The situated agent is used to generate agents that move in the

spatial grid. The communication agent allows the communication between the agents. Finally, the situated and communication agent allows both the locomotion and the communication.

Group: In the platform CORMAS a group is a set of agents [1, 3]. Groups are created through the class Group. The attribute component is the main attribute of this class and contains the list of agents allocated in this group. Agents can be inserted or removed at the execution time.

4.2 Technical-Operational Framework

In this section we have an analysis of platform CORMAS, under the light of the requirements presented in section 2.

4.2.1 Technological Facilities

Technological requirements related with a more cognitive approach (for instance, Manage Agent Connection), or related with distributed simulations (for instance, Manage Security) are not well structured in platform CORMAS.

A. Manage Agents Life Cycle: The kernel of platform controls the entities' life cycle.

B. Manage Agent Connection: Requirement not available in the current version.

C. Manage Communication: CORMAS works with synchronous and asynchronous messages. The control of the messages is based on specific classes. For instance, the class AgentComm allows the communication between agents. A communication agent is linked to a communication channel (using the attribute channel), and works with the messages through the mailbox attribute and messagesFromChanel(), processMessage(), nextMessage() methods.

D. Manage Scheduling Techniques: CORMAS works with discrete time simulation, because this technique is simpler, according to [1].

E. Manage Security: Requirement not available in the current version.

F. Manage Mobility: Requirement not available in the current version.

4.2.2 Domain Facilities

The current version of platform CORMAS offers a good level of services related with the graphical interface. However, regarding the requirements related with distributed simulations (for instance, Launch Agents) it is necessary more improvements to attend the user's needs.

A. Launch Agents: CORMAS uses a single thread of control in a simulation and its agents are SmallTalk objects. They are not instantiated in their own threads, and such feature hampers the distribution of the agents.

B. Manage Intentional Failures: Requirement not available in the current version.

C. Integrate Controlled and Non-Controlled Environments: Requirement not available in the current version.

D. Define the Domain: CORMAS was initially developed to management of renewable natural resources (domain dependent). For this specific area, CORMAS is very complete. However, it was developed into multi-agents conception, and others domains can be implemented using generic methods.

E. Provide Graphical Representation of Domain: The platform CORMAS has a graphical interface to represent spatial entities, situated objects and agents. The spatial entities are responsible for the graphical representation of the domain. Agents and situated objects are allocated in spatial entities, and the leave() and moveTo() methods control the agents locomotion.

4.2.3 Development Facilities

The requirements related to development facilities attend the needs of agent-centered approaches.

A. Develop Agent Architectures: CORMAS does not impose restrictions on the agents' internal architecture. Such decision is delegated to the developer. However, there are some functionalities to integrate the agents in the platform.

B. Use Organizational Abstractions: CORMAS does not adopt the concept of role. This is understandable since its domain-dependent character does not emphasize the work with cognitive agents and organizational-centered approaches. The concept of group represents a composite social entity (agents). Entities in different groups can interact with entities in other groups,

so as to simulate interactions between different scales and organizational levels [3].

C. Manage Ontologies: Requirement not available in the current version.

D. Manage Messages: CORMAS works with message passing and the user can define their own communication language. The class Msg controls the structure of the messages, defining attributes such as sender, receiver and symbol. In addition, the user can add any kind of attributes to be used in more complex conversations [1, 3]. However, there is not a specific communication language, like ACL or KQML.

E. Manage Communication Protocols: To start the simulation, is necessary the init protocol. All entities (spatial, passive or social) have the id and control protocols. The step() method, of the control protocol, is used in each step of the simulation to determine the behavior of the agent. The init and step() must be defined by the user.

F. Provide Translation Mechanisms: Requirement not available in the current version.

4.2.4 Analysis Facilities

In CORMAS some of the requirements related with analysis are well structured. For instance, Control Tracking and Provide Data Analysis. However, there are requirements not formally structured, and it is necessary further implementation work to obtain some functionalities (for instance, Observe Behavior Events and Observe Cognitive Events).

A. Observe Behavior Events: In CORMAS the designer needs to implement methods in Smalltalk to operate the observation of behavioral events. After such definition, the platform provides sophisticated tools to observe such events (step by step compilation).

B. Observe Cognitive Events: This requirement is not structured in the platform CORMAS, since the designer needs to implement methods in Smalltalk (called point of view) in order to observe the entities' internal attributes [1, 3].

C. Define Scenarios: The Smalltalk methods, developed to gathered information, define the scenarios of a simulation.

D. Control Tracking: CORMAS allows a graphical observation of the topological space dynamic. Also, it allows changing, in the execution time, elements such as instance variables [1].

E. Provide Data Analysis: In CORMAS there are a reasonable number of tools that fulfil this requirement. For instance, CORMAS allows the design of graphical charts with time-steps as the x-values and values returned by methods as the y-values (called probes). Additionally, two levels of data-recording are available. The first level tracks information from individual agents and the second from the simulation.

F. Provide Sensitivity Analysis: CORMAS has variation of variables values. It is possible to store all variation of variables, to analyse and to create graphical charts.

5. CORMAS Analysis: Requirements versus the Socio-Concrete Model

In [9] we propose a classification of paradigmatic models in ABSS. The hypothesis that we put forward is that different research and development objectives interfere in the modeling process, since they call for different types of models specified at different levels of abstraction and based on different kinds of assumptions. We have detected four types of models that vary in relation to different ontological assumptions and formalisms to model one or more targets. We designate these classes of models in the following way: Artificial Societies, Socio-Cognitive, Socio-Concrete and Prototyping for Resolution.

This classification is a good analytical tool to characterize or compare models according to various criteria, such as methodological, philosophical or simply pragmatic and usability criteria. According this classification, simulations developed in the platform CORMAS are based in the socio-concrete model. This type of model should desirably represent direct observations of social and institutional concrete processes. The intention is to establish substantive relationship between the simulation and the target, which typically calls for data-driven empirical research in the target. This socio-concrete characteristic is present in platform CORMAS, since its main focus is to model and to implement simulations in land and water management.

In principle, the empirical data gathered in the target should guide the specification of the model, and should be compatible with the outcomes of the simulation as well. Unfortunately, even though this confrontation is desirable, it is not always possible to close the complete circle. Most modeled systems are usually complex, difficult to specify and/or produce outcomes that are very sensitive to initial conditions. As a result most models simplify the validation process and concentrate their validation efforts either in the specification or exclusively in the outcomes.

Other alternative have proposed a weaker form of empirical validation, suggesting the use of participatory-based simulation, whereby a set of stakeholders, such as domain experts or the system end-users, contribute to discuss and negotiate the validity of the specification and the simulation results. This kind of empirical validation is adopted in the context of the CORMAS project through the use of the Rope-Playing Game (RPG) approach [2, 5].

The RPG approach has its constitution in social sciences and means to reveal some aspects of social relationships, allowing the direct observation of interactions among the players. Combining together both MABS and RPG, one can join the dynamic capacity of MABS to the capacity of generating discussions by RPG techniques. In fact, RPG is a way to validate the simulations and to compare new solutions with the players.

In-between games and theatre, RPG are group settings that determine the roles or behavioral patterns of players as well as an imaginary context. A RPG is the performance of a roughly defined situation that involves people with given roles. Players genuinely use RPG as a "social laboratory". It is a way to experiment with a variety of ways of positioning themselves in a group with presumably few consequences in the real world [2].

Traditional tools for teaching and training purpose (RPG, for example) have been successfully used in certain phases of negotiation over water and land-use planning and they have proved to be efficient in facilitating communication on a complex real scenario [2]. According to D'Aquino et al. [5], RPG has already been used to support land use management. The games, which depend upon the prior diagnosis of the situation by experts, help players to share this analysis and to draw upon some improvements based on them.

6. Conclusions

The description of platform CORMAS presented in this paper, based on the requirements analysis of MABS platforms allowed us to characterize both the general structure and global behavior of the platform CORMAS.

7. Acknowledgments

This study is part of the project entitled "Nucleus of Research in Optical Networks - NuPRO" supported by FAPESP, Brazil (Grant number 2006/04650-6).

References

[1] Bousquet, F., Bakam, I., Proton, H., Le Page, C. (1998). *Cormas: Common-Pool Resources and Multi-Agent Systems.* Lecture Notes in Artificial Intelligence, V. 1416, p. 826-838.

[2] Barreteau, O., Bousquet, F., Attonaty, J. (2001). Role-playing games for opening the black box of multi-agent systems: method and lessons of its application to Senegal River Valley irrigated systems. *JASSS* 4 (2) available at http://www.soc.surrey.ac.uk/JASSS/4/2/5.html.

[3] CORMAS (2004). Cormas User's Guides, http://cormas.cirad.fr/en/outil/outguid.htm.

[4] CORMAS (2004). *Common-Pool Resources and Multi-Agent System*, http://cormas.cirad.fr.

[5] D'Aquino, P., Le Page, C., Bousquet, F., Bah, A. (2003). Using Self-Designed Role-Playing Games and a Multi-Agent Systems to Empower a Local Decision-Making Process for Land Use Management: *The SelfCormas Experiment in Senegal*. *JASSS* 6 (3) available at http://jasss.soc.surrey.ac.uk/6/3/5.html.

[6] Ferber, J. (1996). Reactive Distributed Artificial Intelligence: *Principles and Applications.* In O'Hare and Jennings. Foundations of Distributed Artificial Intelligence, New York: John Wiley & Sons.

[7] Lemaitre, C., Excelent, B. (1998). Multiagent Organization Approach. *In: Proceedings of II Iberoamerican Workshop on DAI and MAS,* Toledo, Spain.

[8] Marietto, M. B., David, N., Sichman, J. S., Coelho, H. (2002). Requirements Analysis of Agent-Based Simulation Platforms: *State of the Art and New Prospects, MABS02,* Springer-Verlag, LNAI series.

[9] Marietto, M. B., David, N. C., Sichman, J. S., Coelho, H. (2003). A Classification of Paradigmatic Models for Agent-Based Social Simulation. *In: Proc. 4th. International Workshop on Multi-Agent Based Simulation (MABS'03),* Melbourne, Australia.

[10] Michael, F., Ferber, J., Gutknecht. (2003). Generic simulation tools based on MAS orgnizations. *In 10th European Workshop on Multi-Agent Systems, Multi-Agent Systems Organisations*.

[11] Vincent, R., Horling, B., Lesser, V. (2000). An Agent Infrastructure to Build and Evaluate MAS: The Java Agent Framework and Multi-Agent System Simulator. In Infrastructure for Agents, MAS and Scalable MAS, Lecture Notes in Artificial Intelligence, V. 1887, Berlin: Springer-Verlag, p. 102-127.

[12] Brenner, W., Zarnekow, R., Wittig, H. (1998). Intelligent Software Agents: *Foundations and Applications.* Springer-Verlag, Berlin.
[13] Decker, K. (1996). Distributed Artificial Intelligence Testbeds. In O'Hare, G., Jennings, N. (editors), *Foundation of Distributed Artifical Intelligence,* p. 119-138, New York, John Wiley & Sons.
[14] Uschold, M., Jasper, R. A. (1999). Framework for Understanding and Classifyng Ontology Applications. In Benjamins, V. R., Chandrasekaran, B., Gomez-Perez, A., Guarino, N., Uschold, N. (editors). *In: Proceedings of the IJCAI-99 – Workshop on Ontologies and Problem-Solving Methods.*
[15] Castro, L. N., (2006). Fundamentals of Natural Computing: *Basic Concepts, Algorithms, and Applications,* Florida: Chapman & Hall/CRC.
[16] Lempert, R. (2002). Agent-based modeling as organizational and public policy simulators. *In: Proceedings of the National Academy of Sciences of the United States of America (PNAS),* V.99, p. 7195-7196.
[17] Dimitrov., Vladimir, D., Eriksen., Harald, M. (2006). How to Teach Oral Ecology Using Complexity Approach? *In: Proceedings of the 12th ANZSYS conference - Sustaining our social and natural capital,* V. 1.
[18] Gilbert, N., Terna, P. (2000). How to build and use agent-based models in social science, *Journal of Mind & Society,* V.1, p. 57-72.
[19] Axelrod., Robert., Tesfatsion., Leigh, S. (2006). A Guide for Newcomers to Agent-Based Modeling in the Social Sciences, Iowa State University, Department of Economics (Staff General Research Papers).
[20] Gardelli, L., Viroli, M., Casadei, M., Omicini, A. (2008). Designing self-organising environments with agents and artefacts: *a simulation-driven approach,* Int. J. Agent-Oriented Softw. Eng., V.2, p. 171-195.
[21] Gilbert, N., Troitzsch, K. G. (2005). Simulation for the Social Scientist, Berkshire: Open University Press.
[22] Camazine, S., Deneubourg, J. -L., Franks, N. R., Sneyd, J., Theraulaz, G., Bonabeau, E. (2001). Self-Organization in Biological Systems, New Jersey: Princeton University Press.
[23] Parker, L. E., (1962). Local versus Global Control Laws for Cooperative Agent Teams, Massachusetts Institute of Technology: Artificial Intelligence Laboratory.
[24] David, N., Marietto, M. G. B., Sichman, J., Coelho, H. (2004). The Structure and Logic of Interdisciplinary Research in Agent-Based Social Simulation, *Journal of Artificial Societies and Social Simulation* 7 (3) available at: http://jasss.soc.surrey.ac.uk/7/3/4.html.

Author's biographies

Maria das Graças Bruno Marietto. Associate Professor at the Federal University of ABC (UFABC) Degree in mathematics from Catholic University Dom Bosco with a Masters in Systems Engineering and Computer Science Federal University of Rio de Janeiro (UFRJ / COPPE) and PhD from the Technological Institute of Aeronautics (ITA, Sao Jose dos Campos, SP). Acts in Distributed Artificial Intelligence, Multiagent Simulation, Collective Behavior in Crowds and the Semantic Web.

Edson Pinheiro Pimentel. Ph. D. in Electrical Engineering and Computer Science at ITA - Aeronautical Institute of Technology and Master in Computer Science from UPM - Mackenzie University. Conducts research in the database, Artificial Intelligence and Computer Science and Education. He is an Associate Professor at UFABC – Federal University of ABC.

André Filipe de Moraes Batista. Bachelor of Science and Technology at Federal University of ABC (UFABC). Has experience in computer science, with emphasis on Artificial Intelligence, acts in Artificial Intelligence, Multiagent Simulation and Complex Systems. He works in the Kyatera (FAPESP) project and in the Working Groups of the Network National Education and Research Network (RNP).

Emerson Aguiar Noronha. Mastering degree at Federal University of ABC (UFABC) and Bachelor of Science Computer at Cruzeiro do Sul University. Has experience in computer science, with emphasis on Artificial Intelligence. Acts in Artificial Intelligence, Multiagent Simulation, Complex Systems, Collective Behavior in Crowd and Social Systems.

Guiou Kobayashi. PhD in Electrical Engineering and Computer Science at ITA - Aeronautical Institute of Technology and Master in Computer Science from UPM - Mackenzie University. Conducts research in the Database, Artificial Intelligence and Computer Science and Education.