

# An Ontology for Defining Environments within Multi-Agent Simulations

Fabio Y. Okuyama<sup>1</sup>, Renata Vieira<sup>2</sup>,  
Rafael H. Bordini<sup>3</sup>, Antônio Carlos da Rocha Costa<sup>4</sup>

<sup>1</sup> PPGC – Universidade Federal do Rio Grande do Sul (UFRGS)

<sup>2</sup> PIPCA – Universidade do Vale do Rio dos Sinos (UNISINOS)

<sup>3</sup> Department of Computer Science – University of Durham, UK

<sup>4</sup> Escola de Informática – Universidade Católica de Pelotas (UCPel)

okuyama@inf.ufrgs.br, renata@exatas.unisinos.br

R.Bordini@durham.ac.uk, rocha@atlas.ucpel.tche.br

**Abstract.** *This paper presents an ontology for defining environments where agents are to be situated in simulations based on multi-agent systems. The ontology has been integrated into a platform for developing cognitive multi-agent simulations using agent-oriented programming technologies. The top-level environment ontology sets the basis for defining ontologies that are specific to the simulation of particular virtual environments. We present the top ontology and expand it for one particular scenario. We also explain how a simulation that uses the expanded ontology is executed in our platform.*

**Resumo.** *Este artigo apresenta uma ontologia para a definição de ambientes a serem compartilhados por agentes em simulações baseadas em sistemas multi-agente. A ontologia foi integrada a uma plataforma para o desenvolvimento de simulações multi-agente usando tecnologias de programação orientada a agentes. A ontologia de topo determina a base para a definição de ontologias que são específicas à simulação de ambientes virtuais específicos. É apresentada a ontologia de topo e uma possível expansão para um cenário particular. Além disso, o artigo explica como uma simulação que usa a ontologia é executada em nossa plataforma.*

## 1. Introduction

Multi-agent systems are often situated in a real-world environment. Therefore, autonomous agents that operate in multi-agent systems are programmed to perceive and act on an existing environment. In agent-based simulation, the environment of relevance needs to be simulated computationally, as much as we need to simulate the agents acting in that environment. Therefore, modelling a multi-agent environment is an essential task in building agent-based simulations. Although the importance of environment modelling is recognised, the specification of environments for such systems is not usually thoroughly addressed in the literature. Environments are often considered as “given”, and sometimes environments are modelled as agents. However, the mentalistic notions often used in cognitive agents are not natural for modelling environments.

In this paper we propose a top level ontology for specifying environments, through which a project-level, complete, and executable definition of a multi-agent environment can be derived. This ontology is part of our platform for creating multi-agent based simulations, called MAS-SOC [Bordini et al. 2002]. The platform incorporates several agent technologies for specifying and running cognitive agents.

The paper is structured as follows. The next section covers the main ideas of our approach to multi-agent based simulation and the role that environment ontologies play in it. Section 3 discusses multi-agent environments and how they can be modelled by environment ontologies. Section 4 presents the definition of a complete example of an environment ontology and how a simulation that uses it is executed in our platform.

## **2. Overview**

### **2.1. Multi-Agent Simulations**

In our approach to agent-based simulation [Bordini et al. 2002], agents' reasoning is specified in AgentSpeak [Rao 1996, Bordini and Moreira 2004, d'Inverno and Luck 1998] using the *Jason* interpreter [Bordini et al. 2006], which in turn uses the SACI toolkit [Hübner 2003, Hübner and Sichman 2000]. SACI supports communication between agents, the interactions of agents with the environment (transmission of requests for action execution to, and receiving perception from, the environment) and interactions between agents (such as information and plan exchange). SACI also provides the infrastructure that makes it possible to run distributed simulations, an important factor in allowing large-scale simulations with cognitive agents.

Our main goal is to provide a framework for the creation of agent-based simulations that does not require much programming expertise and efforts from users and at the same time allows the use of cognitive agents for modelling and simulation. The user specifies multi-agent environments, agents (their beliefs and plans), and simulation procedures, through a graphical interface. From this specification, the system generates the corresponding source code for the associated interpreters. A simulation of social aspects of urban growth based on this infrastructure was presented in [Krafta et al. 2002].

### **2.2. Ontologies**

Ontologies are known to be useful in many communication situations: between humans, between humans and computational agents, and also between agents themselves. In our case, the use of environment ontologies adds three important features to the existing platform/approach, as explained below.

First, ontologies provide a common vocabulary with which simulation developers can specify environments. The development of social simulations usually depends on the work of a team of people from different areas (such as social scientists and computer scientists). It is possible that those team members developing the agents are not the same as those working on the environment, the interactions, and organisational structure. In such type of collaborative work, minor misunderstandings on terms used to refer to environment and agent properties may compromise the whole process. An environment ontology presents a consensual model for environment and agents: the essential properties of the environment and agents' actions and percepts must apply in all situations as defined by the semantic relationships between the elements of the environment, which

are all explicitly represented in the ontology. For this reason, defining environments by means of ontologies facilitates and improves significantly the development of multi-agent simulations.

Second, an environment ontology is useful for agents acting in the environment because it provides a common vocabulary for communication within and about the environment. Such explicit conceptualisation is also essential to allow interoperability of intelligent heterogeneous systems.

Third, environment ontologies can be defined through an ontology editor with a graphical user interface (we use Protégé [Noy et al. 2000], in particular). This makes it easier for those unfamiliar with programming to understand and even design such ontologies and consequently the overall simulation.

### **3. Multi-Agent Environments**

According to [Wooldridge 1999], agents are computational systems situated in some environment, and are capable of autonomous action within it, in order to meet their design objectives. Therefore, the environment has an important role in a multi-agent system, whether that environment is the Internet, the real world, or some virtual environment. Agents perceive and interact with each other via the environment, and they act upon it so that it reaches a certain state where their goals are achieved. In accordance with this view, multi-agent systems developers need to model some aspects of the perceptions of the agents and the actions that they can perform over the environment, as discussed in Section 3.1.

In multi-agent systems where agents do not act directly on a physical or existing environment, environment modelling is an important design issue. Nevertheless, the related literature seldom considers this part of the engineering of agent societies (as environments are simply assumed as given), in particular in association with cognitive agents.

In a reactive multi-agent system, the environment plays a major role: since reactive agents have no memory and no high-level (i.e., speech-act based) direct communication with other agents, it is only through perception of the environment that they can make decisions on how to act. On the other hand, cognitive agents have an internal representation of the environment, yet they make decisions (e.g., to adopt new goals, and to change courses of actions) based on the changes that perception of the environment causes on that representation. Thus, environment modelling is an important issue in both classes of multi-agent systems.

#### **3.1. Modelling Environments in ELMS**

In our approach, environment modelling is done using ELMS (**E**nvironment **D**escription **L**anguage for **M**ulti-**A**gent **S**imulation), a language designed specifically for the specification and execution of multi-agent environments [Okuyama 2003, Okuyama et al. 2004, Bordini et al. 2002]. An environment description is a specification of the properties and behaviour of the environment. In our approach, such specification includes sets of: objects, to which we interchangeably refer as *resources* of the environment; *agents*, or more precisely, their “physical” representation that is visible to other agents in the environment; *actions* that each type of agent can perform in the environment; *reactions* that objects have when agent actions affect them; the *perception classes* available to each type of agent; and

the *observable properties*, that is the information about the simulation to which external observers (e.g., the users) have access.

The resources can be modelled as a set of properties and the actions that they effect in response to stimuli. That is, resources can *react*, only agents are *pro-active*. In our approach agents are considered components of the environment where only certain properties of an agent can be perceived by other agents, and this must be specified by the designers. Thus, in the environment description, agents are defined through a list of properties determining the perceptible aspects of agents, a list of actions that they are able to execute (pro-actively), and a list of perception classes to which they have access. From the point of view of the environment, the deliberation activities of an agent are not relevant, since they are internal to the agent (i.e., not observable to other agents). The internal aspects of agents are described with the use of the AgentSpeak agent-oriented programming language.

For the definition of a perception class, it is necessary to define which properties of the environment, agents, and objects are to be perceived by agents with access to that type of perception. The conditions associated with each perceptible property can be specified as well; that is, users can state that the specified properties will be informed to agents with that perception type only under certain conditions. An action is defined as a sequence of changes in properties of the environment, resources, and agents, as well as the preconditions for the action to be executable.

Note that our approach allows quite flexible environment definitions. It is up to the designer to decide what aspects of the environment can be perceived by the agents and observable by the users, as well as to define how actions change the environment. In summary, ELMS can be used to specify environments that are (from the point of view of the agents): inaccessible, non-deterministic, non-episodic, and dynamic; however, they are usually discrete.

### **3.2. The ELMS System**

The actual running of the environment is done by a process that controls the access and changes made on the data structure that represents that environment; this process is called the *environment controller*. Such process is automatically created from the environment specification written in ELMS. In each cycle, the environment controller sends to all agents the percepts to which they have access. Percepts are transmitted in messages as a list of AgentSpeak ground beliefs. After sending the percepts, the process waits for the actions that the agents have chosen to perform in that reasoning cycle.

The main responsibilities of the environment controller process are to: initialise variables; check percepts available at the time (check which agent's perception types satisfy the specified conditions); send the appropriate percepts to agents (those percepts that satisfy the conditions) to the agents; receive execution requests for the actions selected in that cycle; randomise the action queue<sup>1</sup>; check which actions satisfy the conditions; execute the actions; send values of observable properties to the interface; and maintain an internal record of the agents that enter and leave the society.

---

<sup>1</sup>This is done to ensure that agents have similar chances of executing their action first (this is only relevant in the synchronous mode of operation).

The data structure that represents the environment is generated by the interpreter for a specification given as input. The language has constructs that allow the use of a specification as a snapshot of a simulation. This is useful for the visualisation of complex multi-agent simulations as well as for on-the-fly manipulation of the simulation configuration by the users.

For the communication between agents, environment, and the interface, the SACI [Hübner and Sichman 2000] toolkit is used; it also provides an infrastructure for managing distributed agents. Through that toolkit, it is possible for an agent to interact with the environment, so that, for example, developing an interface for human users to interact with a running multi-agent simulation is straightforward. This feature (of open SACI societies) is also very useful for debugging and analysis of simulations. SACI is available open source at <http://www.lti.pcs.usp.br/saci/>. The ELMS interpreter too will be made available open source.

#### 4. An Ontology for Environments in Multi-Agent Systems

We have defined a top-level ontology for environments that limits the possible constructs in an environment definition. Part of the top ontology, with the hierarchy of the main concepts related to multi-agent environments can be seen in Figure 1.

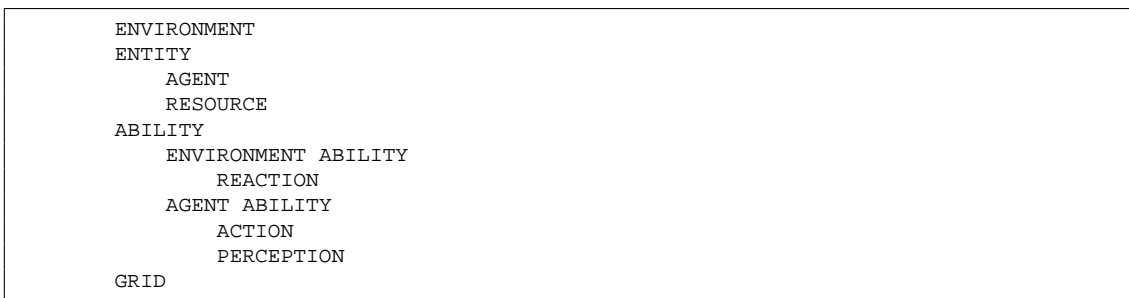


Figure 1. Excerpt of the Top-Ontology

The current version of the whole top ontology can be found at <http://www.inf.ufrgs.br/~okuyama/envOntology>. Table 1 shows the main concepts related to the ontology (explained in detail below). The first column presents the concepts available to describe an environment, the second shows the properties related to each concept, and the third briefly describes each property.

The main concepts in the environment top-ontology are explained next.

**Agent:** The specification of an agent contains its name, a list of attributes, a list of actions, and a list of perception types. The list of attributes characterises the observable properties, from the point of view of the environment and other agents. It is then necessary to specify a list of the actions that agents are able to perform in the environment. Here, the actions that agents are able to perform are defined. This is different from what agents are allowed or forbidden to do: this should be defined at a another level, as a norm or organisational schema. The set of perceptions is a list of perception classes that are available to that agent. Note that defined perceptions and actions can be reused in different agents descriptions (and the same applies to reactions for resources).

Concept	Concept Properties	Property Description
Agent	Does_Action	a list of actions
	Can_Perceive	a list of perceptions
Resource	Has_Reaction	a list of reactions
Action	Has_Condition	a list of preconditions
	Has_Parameter	a list of parameters
	Has_Command	a sequence of commands
Reaction	Has_Condition	a list of preconditions
	Has_Command	a sequence of commands
Perception	Has_Condition	a list of preconditions
	Has_Property	a list of properties
Grid	Has_Property	a list of cell properties
	Has_Reaction	a list of cell reactions
	Size_X, Size_Y, Size_Z	grid dimensions

**Table 1. Properties of Environment Concepts**

**Resource:** In a resource definition, a classe of resources (or objects) is defined; later, during simulation, instances of such classes may be allocated within an environment. A definition of a resource class includes the class name, a list of attributes and a set of reactions. The reactions that a class of resources can have is given by a list of names identifying those reactions.

**Action:** An action definition includes its label, an optional list of parameters, an optional list of preconditions, and a sequence of commands which determine what changes in the environment the action causes. The possible commands defining an action are assignments of values to attributes, and allocations or repositioning of instances of agents or resources within the grid. Resources can be instantiated or removed by commands in an action. If the preconditions are satisfied, then the commands will be executed, changing the environment accordingly.

**Reaction:** For each reaction, its name, a list of preconditions, and a sequence of commands is given. The commands are exactly as for actions. All expressions in the list of preconditions must be satisfied for the reaction to take place. Differently from actions, where only one action (per agent) is performed, all fired reactions will be executed once in each simulation cycle.

**Perception:** A perception definition is formed by a name, an optional list of preconditions, and a list of properties that are perceptible. The listed properties can be any of those associated with the definitions of resources, agents, cells of the grid, or simulation control variables. If all preconditions are satisfied, then the values of listed attributes will be sent to the (relevant) agents as the result of its perception. Note that perception can be based on the spatial position of the agent, but this is not mandatory; any type of perception can be defined by the designer of the environment.

**Grid:** The grid definition is optional. A grid can be two or three-dimensional, and a list of cell attributes can be given. The attribute definition comprises its name, data type (integer, float, string, or boolean), and an optional initial value. Also as a part of the cell definition, a list of the reactions can be specified.

Other concepts related to environment execution and further details can be found in [Okuyama 2003, Okuyama et al. 2004, Bordini et al. 2002].

Using Protégé [Noy et al. 2000] as editing tool, the programmer defines a project environment on the basis of the predefined top ontology. Generating the project ontology in this way can avoid various types of errors in all development stages, from problem analysis to the specification that will be executed. For the agent programmers, the approach also helps make clearer how the environment works, which actions each agent is able to perform, and what the agents will receive as perception of the environment. The programmer does not have to handle code directly, which normally leads to subjective and obscure semantics. Consequently, the ontology also facilitates the understanding and validation of the simulation. The Protégé editor generates an ontology representation in various formats; in our work we use OWL (Ontology Web Language), a W3C standard (<http://www.w3.org/TR/owl-features/>).

### **Deriving an Environment Ontology**

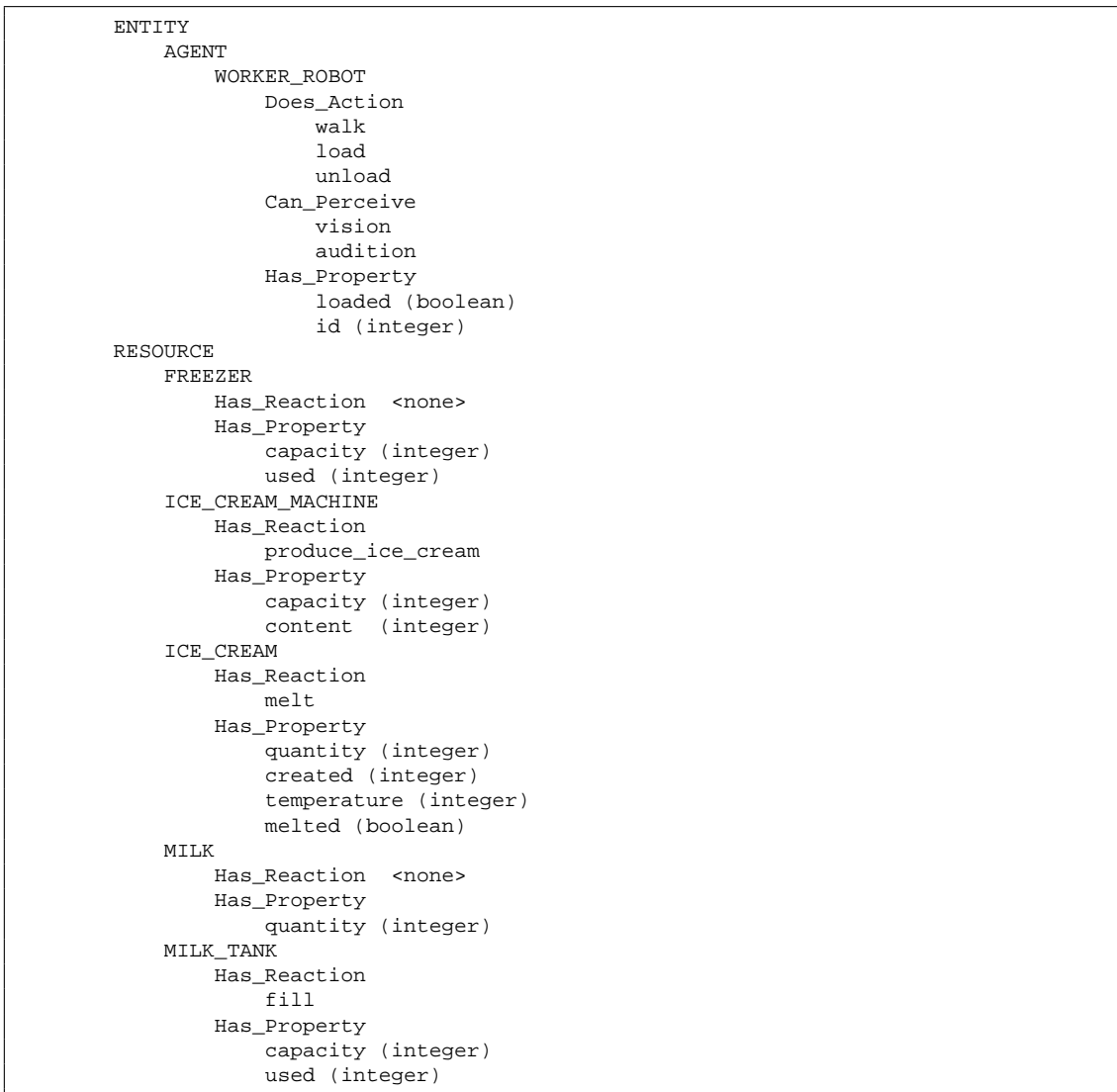
To illustrate our approach, we present the environment ontology used in a simulation of an ice-cream factory. The factory has a milk tank, a machine that produces ice-cream tubs, a freezer to store them, and robots that carry milk from the tank to the ice-cream machine and ice-cream tubs from the machine to the freezer. The milk tank has the capacity to store 160l, and a pump that can fill 10l per simulation step. The ice cream machine can produce 3 tubs of ice cream at each cycle if it has been loaded with at least 90l of milk. The ice cream will melt after 7 cycles, if not stored in the freezer. If the ice cream in a tub has melted, the robot will have to carry it back to the ice-cream machine. The robots can carry 30l of milk or 30Kg of ice cream tubs. There is only one class of agents, which can do all the work, but they are insufficiently numbered so they have to reorganise themselves continuously, according to the service flow. Since it is not relevant for this paper, the agents' reasoning (i.e., the AgentSpeak code for the robots) is not shown here.

To implement this example, the designer would derive the `Agent` class, creating a new class of agents (`Worker_Robot`); derive the `Resource` class creating classes such as `Freezer`, `IceCream`, etc.; instantiate the necessary resources; and derive the `Action`, `Perception` and `Reaction` classes likewise, as seen in Figure 2. These are general steps to define an environment; for a functional definition, one should instantiate other elements such as `Commands` and `Properties`.

The figure shows the classes (in uppercase) and some of the properties of the classes derived from the top ontology to define the environment for this simulation. The complete example can be found at <http://www.inf.ufrgs.br/~okuyama/envOntology>. The OWL ontology with the specific (derived) classes contains the information that is required for the ELMS interpreter to generate automatically the environment controller process to be used in simulations, as described in Section 3.2.

## **5. Related Work**

There are various approaches, frameworks, and tools for the development of multi-agent systems; JADE [Bellifemine et al. 2000], JADEX [Braubach et al. 2004], 3APL [Dastani et al. 2003], and JACK [Howden et al. 2001] are examples. Most of them reduce the environment to a communication tool or broker, focusing on agent modeling as the main part of multi-agent systems development.



**Figure 2. Sample Ontology**

There are also approaches that consider agent organisations as the main part of multi-agent systems development; agents are then developed from the organisational model. Among such systems and methodologies, one of the most popular is Gaia [Wooldridge et al. 2000], which has many features for describing organisations and agents' general behaviour in detail, but does not consider environment modelling.

In [Chang et al. 2005], a framework that makes use of OWL ontologies is presented. The environment model, called "reality model", has environment features, such as physical laws. Another module, the "cognitive middle layer", transforms and filters the complex data that comes from the environment model into information understandable to the agents. Although the environment model is a central issue in that work, because they present a framework rather than a platform, much programming is necessary to develop simulations using that approach.

In [Mili et al. 2005] an architecture is proposed where the environment is modeled from different viewpoints. The topological viewpoint defines the underlying structure,

whereas the behavioural viewpoint represents the environment in terms of its states and transitions. The data viewpoint allows the visualisation of the model, and the functional viewpoint describes the functions made available by the environment. Although that work presents a well-structured view environments, not much information is given on how the modelling is actually done (e.g., regarding the user interface or programming tools).

## 6. Conclusion

We have presented a top-ontology to specify multi-agent environments and, as an illustration, an ontological specification of the environment for a particular simulation project was derived from it. We explained how the simulation is executed by ELMS based on an OWL ontology. Our key motivation is to devise an integrated approach in which both (cognitive) agents and environments can be adequately described in the design and specification of simulations.

The use of ontologies is part of our framework for defining and executing cognitive multi-agent simulations, specially aimed for users unfamiliar with programming. Also, the environment ontology helps to bridge the gap between the design conceived by social scientists and the simulation implemented by computer scientists, where large teams are involved in the process. For similar reasons, we believe the approach can be useful for agent-oriented software engineering in general.

As future work, we plan to concentrate on higher-level aspects of environments, namely, aspects of the *social environment* of the agents, such as the specification of social norms and social organisations in agent societies. We also aim, in the long term, to provide conditions for the study of an important problem in the social sciences: the micro-macro link problem [Conte and Castelfranchi 1995]. In particular, we would like to consider approaches that attempt to reconcile cognition and emergence. The latter objective is inspired by Castelfranchi's claim [Castelfranchi 2001] that only social simulation with cognitive agents ("mind-based social simulations" as he calls it) will allow the study of agents' minds individually and the emerging collective actions, which co-evolve determining each other.

## Acknowledgements

This work was partially supported by CNPq and FAPERGS.

## References

- Bellifemine, F., Poggi, A., and Rimassa, G. (2000). Developing multi-agent systems with JADE. In *7th Workshop on Agent Theories, Architectures, and Languages (ATAL'2000)*, Boston, MA.
- Bordini, R. H., Hübner, J. F., et al. (2005). *Jason: A Java-based interpreter for an extended version of AgentSpeak*, manual, release version 0.8 edition.
- Bordini, R. H. and Moreira, Á. F. (2004). Proving BDI properties of agent-oriented programming languages: The asymmetry thesis principles in AgentSpeak(L). *Annals of Mathematics and Artificial Intelligence*, 42(1–3):197–226.
- Bordini, R. H., Okuyama, F. Y., de Oliveira, D., Drehmer, G., and Krafta, R. C. (2002). The MAS-SOC approach to multi-agent based simulation. In Lindemann, G., Moldt, D., and Paolucci, M., editors, *RASTA*, volume 2934 of *LNCS*, pages 70–91. Springer.

- Braubach, L., Pokahr, A., and Lamersdorf, W. (2004). Jadex: A short overview. In *5th Annual International Conference on Object-Oriented and Internet-base Technologies, Concepts, and Applications for a Networked World (Net.ObjectDays 2004)*, Germany.
- Castelfranchi, C. (2001). The theory of social functions: Challenges for computational social science and multi-agent learning. *Cognitive Systems Research*, 2(1):5–38.
- Chang, P. H.-M., Chen, K.-T., Chien, Y.-H., Kao, E., and Soo, V.-W. (2005). From reality to mind: A cognitive middle layer of environments concepts to believable agents. In *Proc. of the First International Workshop on Environments for Multiagent Systems (E4MAS), held with AAMAS-04*, number 3374 in LNAI, Berlin. Springer-Verlag.
- Conte, R. and Castelfranchi, C. (1995). *Cognitive and Social Action*. UCL Press, London.
- Dastani, M., van Riemsdijk, B., Dignum, F., and Meyer, J. J. (2003). A programming language for cognitive agents: Goal directed 3APL. In *Programming Multiagent Systems: Languages, frameworks, techniques, and tools (ProMAS03), held with AAMAS-03, 2003, Australia*.
- d’Inverno, M. and Luck, M. (1998). Engineering AgentSpeak(L): A formal computational model. *Journal of Logic and Computation*, 8(3):1–27.
- Howden, N., Ronnquist, R., Hodgson, A., and Lucas, A. (2001). JACK – summary of an agent infrastructure. In *5th Intl. Conference on Autonomous Agents(Agents’2001)*, Canada.
- Hübner, J. F. (2003). *Um Modelo de Reorganização de Sistemas Multiagentes*. PhD thesis, Universidade de São Paulo, Escola Politécnica.
- Hübner, J. F. and Sichman, J. S. (2000). SACI: Uma ferramenta para implementação e monitoração da comunicação entre agentes. In *Proc. of the Intl. Joint Conference, 7th Ibero-American Conference on AI, 15th Brazilian Symposium on AI*, São Carlos. ICMC/USP.
- Krafta, R., Oliveira, D. d., and Bordini, R. H. (2002). The city as object of human agency. In *4th International Space Syntax Symposium*, London.
- Mili, R., Leask, G., Shakya, U., Steiner, R., and Oladimeji, E. (2005). Architectural design of the divas environment. In *Proc. of the 1st Intl. Workshop on Environments for Multiagent Systems (E4MAS), held with AAMAS-04, 19th of July*, number 3374 in LNAI, Berlin. Springer-Verlag.
- Noy, N. F., Ferguson, R. W., and Musen, M. A. (2000). The knowledge model of protégé-2000: Combining interoperability and flexibility. In *2th International Conference on Knowledge Engineering and Knowledge Management(EKAW’2000)*, France.
- Okuyama, F. Y. (2003). Descrição e geração de ambientes para simulação com sistemas multiagentes. Dissertação de mestrado, PPGC/UFRGS, Porto Alegre, RS. In Portuguese.
- Okuyama, F. Y., Bordini, R. H., and da Rocha Costa, A. C. (2004). ELMS: An environment description language for multi-agent simulation. In Weyns, D., Parunak, H. V. D., and Michel, F., editors, *E4MAS*, volume 3374 of *LNCS*, pages 91–108. Springer.
- Rao, A. S. (1996). AgentSpeak(L): BDI agents speak out in a logical computable language. In *Proc. of the 7th Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAA-MAW’96), The Netherlands*, number 1038 in LNAI, London. Springer-Verlag.
- Wooldridge, M. (1999). Intelligent agents. In Weiß, G., editor, *Multiagent Systems—A Modern Approach to Distributed Artificial Intelligence*, chapter 1. MIT Press, Cambridge, MA.
- Wooldridge, M., Jennings, N. R., and Kinny, D. (2000). The GAIA methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3(3):285–312.