Multi-agent architecture for distributed simulation : Teaching application for Industrial Management

Stphane Galland, Frdric Grimaud, Jean-Pierre Campagne
Systmes Industriels Coopratifs laboratory*
Ecole Nationale Suprieure des Mines
158, Cours Fauriel, Saint-Etienne, 42023 Cedex 2, France
e-mail: {galland,grimaud,campagne}@emse.fr

Additional reference: In *14th European Simulation Multiconference*, eds. Rik Van Landeghem, Society for Computer Simulation International (p. 756-762). Ghent, Belgium, May 2000. ISBN 1-56555-204-0.

ABSTRACT

We are located in the context of the simulation of complex industrial systems, which are distributed in geographical, decisional and informational terms.

This paper is divided into three parts. The objective of the first one is to position us within the studying framework of a modeling and specification methodology proposed by (Galland et al. 1999). We expose, by the way of a short state of the art, the considerated approach. The second part is dealing with a multi-agent model corresponding to the first stage of our methodology development. Finally we conclude by the application of this model to a teaching case.

Keywords: Distributed simulation, multi-agent systems, teaching models, industrial management

1 Introduction

Simulation is regarded as one of the rare technics taking into account the dynamism of the industrial systems. But the evolution of the industry towards decentralization imposes the technological development of industrial system modeling. Hence simulation must support the physical, informational and decisional flow distribution. Although the distribution of physical flows can be carried out by the current tools, the two others are still difficult to model. Moreover simulation softwares are seldom associated with precise methodologies.

To answer such problems, we are located within the framework of the methodology suggested by (Galland et al. 1999), which allows the taking into account of the informational and decisional physical distributions in the models of simulation. We propose to develop a first multi-agent model used and checked by the intermediary of a teaching application.

In the section 2, we present a short state of the art in the field of distributed simulation and multi-agent systems. We explain our methodological approach for distributed simulation. In the following section, we define a simple multi-agent model. In section 4, the application on a teaching case is presented. Finally we conclude and expose our perspectives.

2 State of the art

2.1 Distributed simulation

The advent of the simulation technics brought to the industrialists the possiblity to model the behavior of their industrial systems in a mush more realistic way. But simulation models make it difficult to model the physical, informational and decisional distributions.

To partly solve the problems of the decentralization of information and knowledge as those of times in modelling and simulation, the scientific community considere the implementation of distributed simulations according to two major approaches: the data-processing distribution of the model (especially synchronization problems (Filloque 1992)) and distribution of knowledge and decision-making processes (society representation in international context (Burlat 1996)).

Distributed simulation, including these two axes, taking the international characteristics of the companies into account *i.e.*, the problems of technical culture, knowledge and geographical distribution can be supported by a distributed simulation model. However this field has not become fully develop yet.

2.2 Multi-agent systems

In this section we describe the concepts which make up a multi-agent system (Ferber 1995). Within the framework of our works on the multi-agent systems, we consider a particular structuring : the vowel approach (Demazeau 1995), which identify four axes: **A**gents, **E**nvironment, **I**nteractions and **O**rganization.

The first axis defines a model of agent. Two

*SIC : Cooperative Industrial Systems

large schools are opposed: cognitive agents (Boissier and Demazeau 1998) and reactive agents (Ferber 1996).

The axis Environnement is based on the modelling of the space in which are the agents, and actions and observations on this Environment.

The third axis allows to study interactions existing in the modeled system and especially on temporal aspects (Carron et al. 1999).

The last axis is the implementation of an society's organisational model : hierarchy, market, ... (Hannoun et al. 1998).

2.3 Multi-Agent Methodological Approach for Simulation

(Galland et al. 1999) proposes a multi-agent methodological approach that carry out simulations integrating physical, informational and decisional distributions. He notes that problems arising from distributed simulation can completely or be partially solved by the use of the multi-agent concepts. The figure 1 illustrates this approach in the case of the distributed simulation of three models. In order to allow a progressive study of all aspects of the suggested approach, (Galland et al. 1999) recommends an iterative development of the various methodology parts (analyze, specification, design and implementation) along a fifth axis corresponding to the AEIO approach.

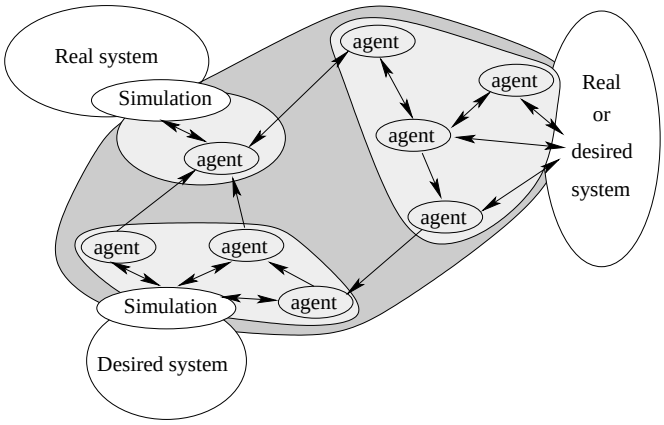

Figure 1: Multi-agent system allowing distributed simulation

3 First model : AEIO specification

In this article, we are located in the extension of the work initiated by (Galland et al. 1999) allowing to answer the problems of the distributed simulation through multi-agent concepts.

We limits ourselves to the definition and the use of a primitive multi-agent model. Its development allows a progressive implementation of the concepts, which will appear in the final multi-agent system. We present this first stage of the model design according to the vowel approach[1]. This primitive model only supports the reuse of existing simulation softwares. In addition, the built multi-agent system should be within a flowshop framework *i.e.*, a production system comprising neither cycles nor demultiplexed flows.

In this sub-section we present the specification of our primitive multi-agent model according to the four AEIO parts.

3.1 Agents

The agent architecture remains simple and close to the reactive concepts. Currently we consider that an agent is only a communication vector. It does not have reason to implement a cognitive mechanism on the messages. Thus decisions are made by entities (simulation softwares) associated with agents *i.e.*, in point of view of them, simulation models have an unspecified complexity and in particular could implement decision-making technics. Indeed, these models will use the communication interface represented by agents. There last consider models as black boxes.

We use the event programming concepts to describe our agent model.

Two technics are used to support communication between simulation softwares and agents :

- when the first wishes to send a message towards another software, it calls upon its agent by the intermediary of a method;
- the agent generates an event in the simulation software in order to notify a new message arriving from other agents.

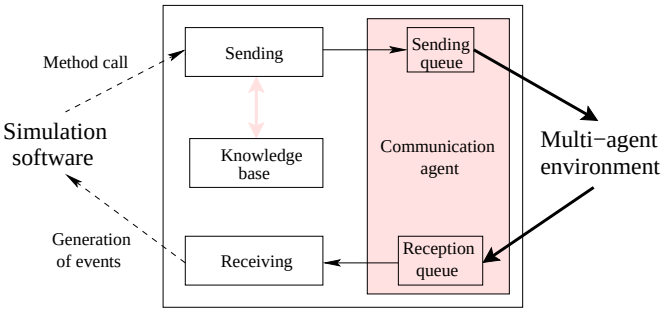

Figure 2: Software architecture of an agent

The figure 2 illustrates the operating mode of an agent. This last is composed of four autonomous entities:

[1] **A**gent, **E**nvironment, **I**nteraction et **O**rganization

- the *Sending module* is launched each time that a call of a sending method is carried out by the simulation software; its role is to correctly format messages (receiver, standard agent message, *etc.*), and to transmit them to the communication module;
- the *receiving module* generates an event in the simulation software; this module uses informations contained in the message stored at the head of the reception queue;
- the *knowledge module* only exists to allow the storage of information as the list of the agents or recipients according to message types;
- the *communication module* allows the sending and the reception of messages between agents; it is composed of two queues respectively containing messages to be sent and those received; its role is to manage communications on a level closer to the operating system than the remainder of the agent components.

The various agent's modules can be agents. This allows a recursive approach during their design.

3.2 Environment

The Environment is the space in which agents evolve. This object aggregate takes into account all that is not an agent and which has an influence on the system. For example, within the framework of the ants' nest simulation, the Environment is not only made up by the geographical structure of rooms, but also by objects as food, cocoons, *etc.*.

In our definition of the multi-agent Environment, it only exists to allow agent communications with an higher abstraction than simple message sending. Indeed we consider that each agent is known in the Environment and can be easily localized via this one. Thus any agent can use the "Environmental" base of connected agents to establish its dependency links.

Figure 3 illustrates the position of the Environment in a layer structure representing the multi-agent system. Those communicate between them by the intermediary of the operating system communication module *e.g.*, TCP/IP sockets. The operating system is also used as a vector between agents and the Environment. This one is composed of an autonomous module whole, which correspond to preset behavior agents. They have the role to maintain a list of agents, whish exist in the system.

3.3 Interactions

Agents communicate with a simple interaction protocol based on message sending. Receivers could be the environment, or another agents (table 1). In the first case, valid messages are notifications of arrivals and departures

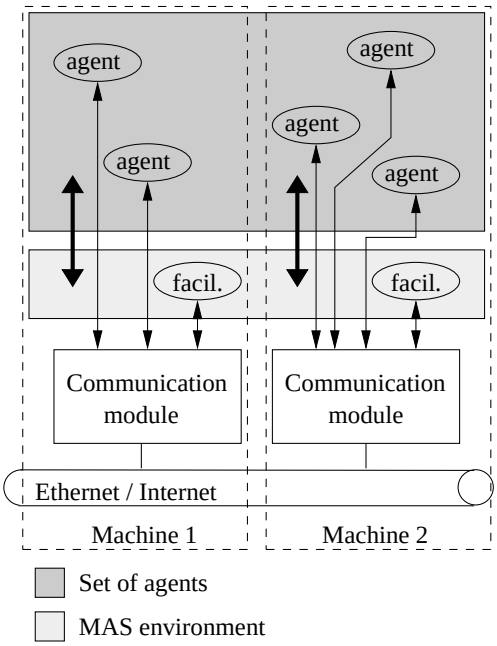

Figure 3: Environment integration in the system

of agents from the multi-agent system, as well as the message sendings by diffusion. In the second case, agents exchange domain-dependent messages *e.g.*, production entity transfer.

Message Id	**Receiver**
	Description
`ADD`	environment
	Arrival of an agent in the MAS
`REMOVE`	environment
	Departure of an agent from the MAS
`LOCAL_BROADCAST`	environment
	Sending a message by diffusion to all agents belonging to the same physical machine
`BROADCAST`	environment
	Sending a message by diffusion to all agents
`domain_dependant`	agent
	Sending a message with a simulation domain-dependant format

Table 1: Interactions into the MAS

3.4 Organization

Our organization is currently static and predefined. This aspect is not a critical problem in our first approach. Thus agents are satisfied to send messages generated by the simulation software. They are not worried a possible organization. It is implicitly deduced from simulated model parts *i.e.*, agents are not organized, they are satisfied to send messages towards the agents specified by simulation models.

We can bring our organization closer to the concepts released by (Burlat 1996). They illustrate firm modelling using multi-agent systems. The elementary model is composed of cognitive agents, which supervise activity centers. Figure 4 illustrates an example of an activitie chain that is made up of three elementary modules. The meta-model concept introduced by (Burlat 1996) allows to consider each activity centers as chain of activities.

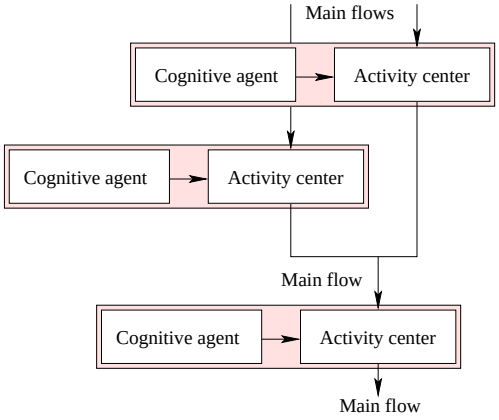

Figure 4: Chain of activities

3.5 Synchronization

The temporal synchronization of agents is a significant problem of distributed simulation. We use a simplified algorithm based on a pessimistic approach (Filloque 1992). The dates of simulation d_m and $d_{m'}$ of two models M and M' are increasing. We suppose that the stamp date d_e of an entity message sent by M to M' is equal to d_m. We consider that the model M' is authorized to simulate until the older-received message stamp is reached. This algorithm permits to synchronize models without loss of entities under two assumptions : the sending order of entities corresponds to the reception order, and only the flowshops are treated (neither cycle nor flow junction).

We illustrate the behavior of this synchronization technic in the section 4.3 by applying our first model to a teaching case.

3.6 Conception and Implementation

In this subsection we discuss the software implementation of our multi-agent system.

We decided to establish a runnable model with VISUAL BASIC® and ARENA® tools. The first enables to create communication modules using ACTIVEX® standard, as well as the software, which represents each decisional system. ARENA® permits to implemente our physical models.

*3.6.1 Communication modules or agents*Figure 5 illustrates the object that instances our agent concept. It is divided into four parts, which represents the modules defined in figure 2. The first part is the object interface *i.e.*, the whole of attributes and methods that can be used by users. On the figure appears only the method *Sending* that corresponds to the forwarding module defined in the agent model. The second part is the knowledge base. It corresponds to the same-named module in our model. These two sections permit to prepare the message forwarding.

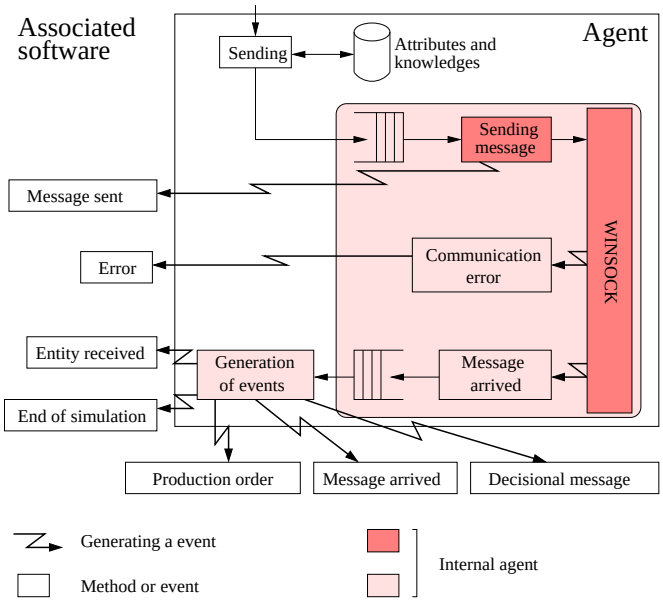

Figure 5: Agent instance

The communication module is made up by two queues, two sending and message-arrival submodules, and by a *WinSock* control. This last corresponds to a preset object that allows to send messages on a *TCP/IP* network. The role of this module is double. Initially it must put messages in the sending queue. Its second role is to move all received messages into a second queue. This one is emptyed by an event generator modeule. It permits to notify the message arrival to the user object.

As we mentioned in section 3, the various modules can be regarded as internal agents. They permit an autonomous and parallel execution of the different implementation parts.

The main agent generates a limited number of events to allow user application reacts :

- *Message sent*, this event indicates to the user application that a message was sent;
- *Error* is generated when an error was raised in *WinSock* control;
- *Message arrived* is a general event that is produced when a no-specific message has arrived;

- *Entity received*[2] is generated when the received message was an entity;
- *Production order*[2] is a generated event when a message containing a PO[3] was received;
- *Decisional message*[2] corresponds to the reception of a contextual message of decisional nature (request for information, change of strategy, *etc.*);
- *End of simulation*, this event is generated when the received message is containing the authorization to continue simulation until its term *i.e.*, the envisaged end-of-simulation date.

*3.6.2 Decisional modules*The implementation of decisional modules, which represent decision-making centers of simulation models, is application-dependent. These modules are associated with agents, such as we define in the previous section. They allow a better integration with simulation models.

4 Using first model for the development of a teaching application

In this section, we present a teaching case application of our first model seen in section 3. Indeed, our experiment within the *Ecole Nationale Supérieure des Mines* enables us to have a vision on teaching and on the development of simulation concepts near learning (Girard et al. 1998).

We initially expose the teaching contributions of our model. Then we describe an instance of it. We finish on the synchronization technic behavior on the teaching case example.

4.1 Contribution for pedagogy

Our teaching model highlights three interesting aspects for the decision-making training within an industrial system.

Learner models represent some production lines in interaction, learners must implement a local decision-making policy and thus try to react vis-a-vis unverifiable variations of enter and outgoing flows. Learners have the role to conceive and modify a system to respond to these problems. Moreover they will have to dynamically adapt themselves to interactions existing between there models and other ones.

The second interesting aspect is the possibility to put learners in situations, whish have not the same level of decision-making.

Our teaching model enables to confront learners to various production line risks. Moreover some simulation models can be composed of cognitive mechanisms instead of the learner management. Then we can predefine and control the behavior of specific model parts.

Another interesting aspect is the supervision capacity of our model. Supervisor can place "spying components" in the various simulation models in order to follow the learner-behavior evolution. They can also influence a simulation model to cause an interesting teaching situation. In addition a inlining-helping mechanism allows learners to require the supervisor assistance.

4.2 Presentation of an application

We wish to implement teaching tools, which illustrate the production system behaviour. Taking into account the current state of our model, we limit the teaching cases to systems without cycle and junction point. These last two cases require a more advanced synchronization technic.

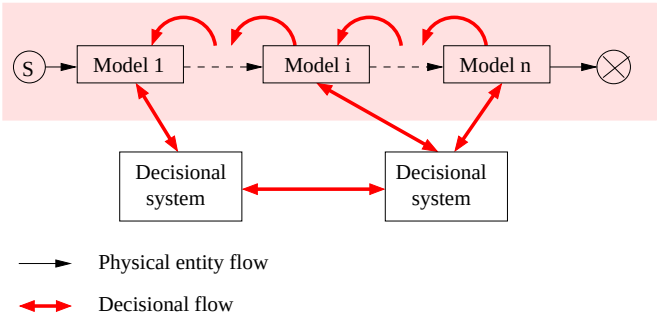

Figure 6: Teaching application of the Industrial Management

Figure 6 illustrates a generalized teaching model made up of physical device models and decisional systems, which can influence the simulation strategy. All these elements are connected by physical and decisional flows. The first correspond to the entity movements between the various physical models. The second ones are composed of production orders (compound of a date and a number of entities to produce), and flow of information contains the running state of the production.

Models of the physical subsystem are elaborated a priori by learners according to constraints given by the supervisor. Indeed the existing simulation tools (ARENA®, *etc.*) do not allow structural changes of models when the simulation is running.

Within the checking and instancing framework of our teaching model, we consider the production system of figure 7.

This model is composed of four machines *M1*, *M2*, *M3* and *M4*. Each one is dealt with sub-models that are simulated in "parallel". As requires in our current approach presented in section 3, entity flows cause neither cycle, nor junction point. Decisional systems *D1*, *D2* and *D3* are autonomous software entities, which are manag-

[2]in the context of production system simulation
[3]**P**roduction **O**rder

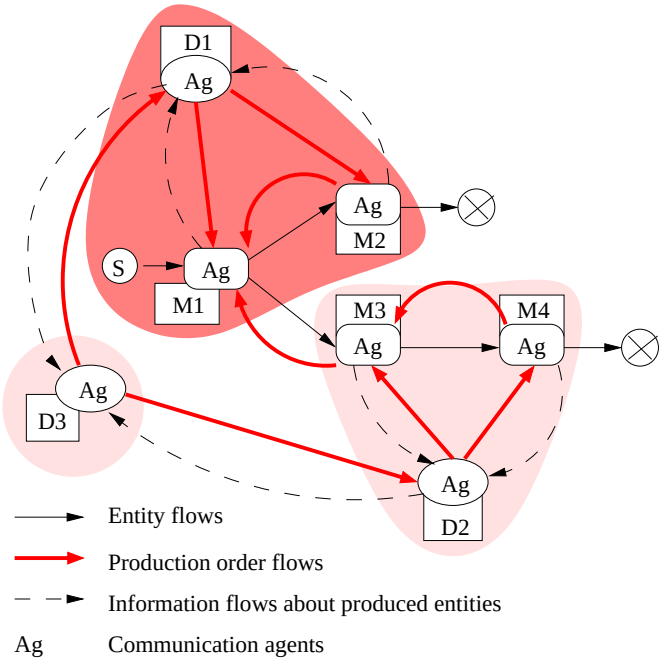

Figure 7: Instance of the teaching model

ing respectively machines *M1* and *M2*, *M3* and *M4*, and decisional systems *D1* and *D2*.

From a conceptual point of view, we consider that *D1*, *M1* and *M2* represents a first workshop; *D2*, *M3* and *M4* a second workshop; and *D3* the production line direction.

One or more learners can deal with each model suggested in this instance. They are able to control the behavioral evolution of the system. Thus they can study, via case practice, the influence of each parameter on production line performances. Moreover the teaching model distribution illustrates competency management problems. Indeed the totality of them seldom belongs to only one-person *e.g.*, the strategic choices are taken on the direction level whereas the production lines only apply them.

4.3 Synchronization in the teaching application

To illustrate the synchronization technic presented in section 3.5, we use the line of production illustrated by figure 7.

Table 2 contains the arrival dates of six batches *P1* to *P6*, as well as the processing duration of those on machine *M1*.

M1	P1	P2	P3	P4	P5	P6
Arrival	1	2	6	6	7	9
Duration	3	2	1	2	1	3

Table 2: Arrival and proceeding duration for each batches

If we concentrate on machine *M1*, we can highlight his behavior. Figure 8 illustrates the message arrivals *e.g.*, *P1(4)* meaning that entity *P1* is arrived at the simulation date *4*. It shows simulation date ts(M_1), the state of *M1* and his queue.

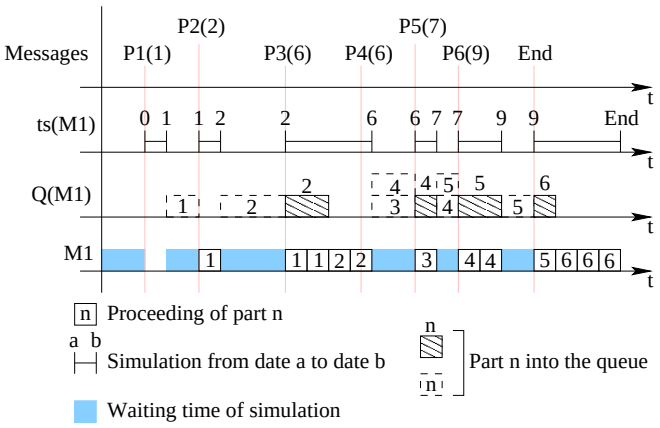

Figure 8: Simulation on *M1*

Thus we can note that *M1* simulation model is stopped where the more sure date, which is the stamp of the oldest received message, is reached. The stamp t_m of a entity message allows to restart simulation until his date is equal to t_m. Lastly, we introduce a particular message call *End*, which allows model to continue the simulation until the completion date of simulation was reached.

5 Conclusion and perspectives

Simulation is a recognized tool by the industrial world for its qualities to highlight the behavior of industrial systems. (Galland et al. 1999) presents distributed simulation problems and proposes a methodological approach based on multi-agent concepts.

In this context, we decided to define a simulation model incorporating physical, informational and decisional distributions. We create a simple multi-agent model, which allows to carry out flowshop simulations.

The application on a teaching case permits to instance the suggested model and thus to study and check our propositions. Moreover it permits to place learners into interesting teaching positions. Indeed, they are not only confronted with design problems and production line changes. They must also dynamically adapt themelves to the interactions that there system have with other production lines. The distribution of decisions and informations enables to highlight local decision-making difficulties.

Our final goal is to conceive a simulation methodology integrating physical, decisional and informational distributions; and an simulation environment based on multi-agent systems. We plan to develop, in an iterative way, the various aspects to reach these two objectives. Finally, within the particular framework of the *Ecole Nationale*

Supérieure des Mines, we will apply our theories on real industrial and teaching cases.

REFERENCES

O. Boissier and Y. Demazeau. 1998. "Une architecture multi-agent pour des systèmes de vision ouverts et décentralisés". In *Techniques et Sciences Informatiques* (Oct.), 1039-1062.

P. Burlat. 1996. "Contribution à l'Évaluation Économique des Organisations Productives : vers une modélisation de l'entreprise-compétences". Phd thesis Université Lyon 2 (Jan.).

T. Carron, H. Proton, and O. Boissier. 1999. "A Temporal Agent communication language for dynamic Multi-Agent Systems". In *Modelling Autonomous Agents in a Multi-Agent World* (Spain, July), 115-127.

Y. Demazeau. 1995. "From Interactions to Collective Behaviour in Agent-Based Systems". In *European conference on cognitive science* (Saint-Malo, France, Apr.).

J. Ferber. 1995. *Les Systèmes Multi-Agents - Vers Une Intelligence Collective*. InterEditions (Sep.).

J. Ferber. 1996. "Reactive Distributed Artificial Intelligence : Principles and Applications". In *Foundations of Distributed Artificial Intelligence*, G. O'Hare and N. Jennings eds, 287-314.

J.-M. Filloque. 1992. "Synchronisation Répartie sur une Machine Parallèle à Couche Logique Reconfigurable". Phd thesis Institut de Formation Supérieure en informatique et Communication - Université de Rennes 1 (Nov.).

S. Galland, F. Grimaud, P. Beaune, and J.-P. Campagne. 1999. "Multi-Agent Methodological Approach for Distributed Simulation". In *Simulation in Industry - 11th European Simulation Symposium* (Erlangen - Germany, Oct.), G. Horton, D. Möller, and U. Rüde eds, 104-108.

M.-A. Girard, Y. Ouzrout, P. Burlat, and F. Grimaud. 1998. "Teaching "Production Management" using Software based on Simulation Models". In *ArenaSphere'98* (Pittsburg, May).

M. Hannoun, J. S. a. Sichman, O. Boissier, and C. Sayettat. 1998. "Dependence Relations Between Roles in a Multi-Agent System". In *Multi-agent Systems and Agent-based Simulation, Lecture Notes in artificial Intelligence* (Berlin-Alemanha), J. Sichman, R. Conte, and N. Gilbert eds, 169-182.