Available online at www.sciencedirect.com

SCIENCE DIRECT®

ELSEVIER

Int. J. Production Economics 85 (2003) 11–31

international journal of production economics

www.elsevier.com/locate/dsw

# $\mathscr{M}_{\mathscr{A}}\mathscr{MA}$-$\mathscr{S}$: An introduction to a methodological approach for the simulation of distributed industrial systems

S. Galland[a,*], F. Grimaud[c], P. Beaune[b], J.P. Campagne[c]

[a] *Systems and Transportation Laboratory, University of Technology of Belfort-Montbéliard, Belfort 90010, France*
[b] *Multi-Agent Systems Department, Higher National School of Mines of Saint-Étienne, Saint-Étienne 42023, France*
[c] *Scientifical Methods for Industrial Management Department, Higher National School of Mines of Saint- Étienne, Saint-Étienne 42023, France*

**Abstract**

We are located in the context of the industrial system simulation, which is complex and distributed in operational, informational and decisional terms. In this article, we present the problems of the simulation of such systems. We propose a methodological approach based on the systemic and on the multi-agent concepts. It authorizes the data-processing and modeling distributions.

*Keywords:* Industrial system simulation; Multi-agent systems; Methodology; Distribution; Decisional processes

## 1. Introduction

The simulation is a tool adapted to the modern industrial problems. It permits to support and study the dynamic behavior of industrial systems. Even if the simulation is a powerful tool, some problems always exist. We concentrate our work on four kinds of them. First, we assume that the simulation tools are still seldom packaged with a methodology, which allows an easier modeling of an industrial system. The second problem, that we want to tackle, is the poor support of the component modeling or of the modular modeling. It is still difficult to reuse the existing models (or a part of them) without substantial changes or a complete rewriting. The third problem is the strong relationship between the physical, the informational and the decisional aspects of an industrial system. For example, if we want to simulate a system with a pulled-flow management policy, and just after with a pushed-flow policy, we must completely rewrite the simulation model, even if only the decisional part changed. Why must the rest of the model be rewritten? Finally, the last problem is about the difficult to model the new industrial organizations, such as the enterprise consortiums or the virtual enterprises. To solve these different problems, we propose a methodological approach: $\mathscr{M}_{\mathscr{A}}\mathscr{MA}$-$\mathscr{S}$[1] (Galland, 2001). It offers a modeling framework that is independent of any software platform (simulation tool or multi-agent system). In this paper, we present the major

*Corresponding author. Tel.: +33-3-84583418; fax: +33-3-84-583342.
*E-mail address:* stephane.galland@utbm.fr (S. Galland).

[1] Multi-agent methodological approach for the simulation of industrial systems.

concepts attached to our methodology. First, we explain the context in which our works are. In Section 3, we make a brief state of the existing works. In Section 4, we present the life cycle of $\mathscr{M}_{\mathscr{A}}\mathscr{M}\mathscr{A}$-$\mathscr{S}$ and our major propositions and works on it.

## 2. Context

Since 20 years, the concept of system appears from the works of Von Bertalanffy, Wiener, Shannon, Forrester (Durand, 1975) and De Rosnay (1979): a system is a "whole of interacting elements, which are organized to reach a goal". More precisely, we are interested by the class of the manufacturing systems defined as the "centers that transform raw material to final products by using a whole of resources (processing units, operators, raw material, storage areas, etc.)" (Dupont, 1998; Ye, 1994). During their design and their use, some structural, functional and organizational problems appear (sizing, working, productivity, maintenance, resource management, and scheduling) (Grimaud, 1996). To solve these problems, the system managers have some possibilities. One of them is the discrete-event simulation. It allows "to imitate, in the time, the operations of a process or a real system. The simulation implies the generation of an artificial evolution of the system, and the observation of this evolution to carry out deductions on the operational characteristics of the real system represented" (Banks, 1999; Law and Kelton, 1991). Moreover, it supports the stochastic phenomenas and the dynamics of the systems.

In this context, we are especially interested about the support of the new industrial organizations. For example, one of them is the enterprise consortium, which is "a whole of companies related the ones to the others by a cycle of production. The bond is neither legal, nor structural; it has often the form of simple agreements. These companies have in common a powerful system of functional cooperation" (Butera, 1991). Some other kinds of organizations exist: virtual enterprise (Goranson et al., 1997; Ettinghoffer, 1992), network of enterprises (Nunes, 1994), etc.

In the following sections, we present the four major problems that we want to tackle, and our propositions to solve them.

### 2.1. Problems

We consider four main classes of problems, which occur during the design and the simulation process:

*Formalization*: A problem, that occurs during the modeling phase, is the poor formal definition of the modeling elements and rules. The simulation tools have a strong influence to the point of view of the designers. For example, the tools Aren® and Simple++® offer two different modeling views. Then, the change of simulation tools applies the complete rewriting of the simulation models. This formalization problem is partly solved by the existing methodologies or languages, e.g., ASCI[2] (Kellert and Force, 1998; Kellert and Ruch, 1998), CM[3] (Nance, 1981), IDEF (US Air Force, 1993) or UEML[4] (Vernadat, 2001). But, they do not propose any element that permits to support a distributed industrial system. Moreover, the underlying simulation models, e.g., the Petri nets (David and Alla, 1992; Wang, 1998) or the queuing theory (Knuth, 1998), offer tools adapted to simulation (Zimmermann, 1994). But, they must be used by specialists and are not really usable in an industrial context.

*Highlight of flows and subsystems*: An industrial system is composed of physical and informational flows. They are different, even if they are strongly interacted. Currently, the simulation models include these two kinds of flows, but they do not highlight each of them. The understanding of a simulation model is still difficult because a significant effort must be done to distinguish the flow of entities and the flow of information.

Another problem is highlighted when a designer wants to change one of the flows. For example, when the management policy passes from a pulled

[2] Analyse Spécification Conception Implantation.

[3] Conical Methodology.

[4] Unified Enterprise Modeling Language.

one to a pushed one, the designer must completely rewrite the simulation model, even if only the decisional part of the system changed. In an ideal world, only the decisional part must be updated.

*Distribution*: A problem, which appears in the methodologies and the simulation tools, is the distribution of the models and of the simulation process. In the first case, it is difficult, with the simulation tools and the existing models, to realize the simulation of a strongly distributed manufacturing system (such as an enterprise consortium). For example, the confidentiality needed by this kind of system does not permit to realize a centralized simulation. The second case of distribution corresponds to the realization of the simulation process on a computer network. The main problem is the solving of a whole of technical constraints such as the synchronization of the models.

*Reuse*: The forth problem is the poor modularity of the simulation models. In fact, even if the concepts of model and submodel exist, it is still difficult to build a modular simulation model. For example, the reuse of an existing submodel applies to copy and paste it into an another model. This copy does not support the automatic update of all the instances of the copied submodel when it changed.

Our goal is to allow the simulation of a manufacturing system, which is complex and distributed (enterprise network, strongly distributed enterprise, etc.). We are interested by the systems which always include a distribution aspect. But, we are not limited to the distribution on a computer network. In fact, our interest is also on the distribution of the information and of the decision-making processes. But, the existing tools do not support these two last aspects. Moreover, they are domain dependent, e.g., ARéVi for virtual representation of systems (Duval et al., 1997; Chevaillier et al., 1997), SWARM for simulation of artificial way of life (Burkhart, 1994); or they only include one aspect of the distribution, e.g., HLA[5] (US Department of Defense, 1996) is a architecture that supports the interoperability of models. In another point of view, the underlying models for simulation (Petri nets, queuing theory, etc.) and the distributed simulation techniques are partly adapted to the simulation of distributed industrial systems. In fact, even if they permit to realize the simulation process, they do not really support the specific constraints of the distributed systems (confidentiality, evolution's dynamic, etc.).

[5] High Level Architecture.

### 2.2. Propositions

The previous problems drive us to propose a methodological approach named $\mathcal{M}_{\mathcal{A}}\mathcal{MA}$-$\mathcal{S}$ (Galland, 2001). This approach proposes a methodological framework for simulation and a software architecture. This last is based on the concepts from the multi-agent systems (Ferber, 1995). In fact, the autonomy and the interactional capabilities of agents allow the distribution of the physical, the informational and the decisional subsystems. In the case of a cognitive agent (in opposition to a reactive agent), the cognitive mechanisms permit to support the decisional processes. Moreover, the modularity, which is borned from the use of a multi-agent system, allows to respond to an another crucial point for an industrial point of view: the reuse of knowledge and of the tools that are already used.

Fig. 1 is a schematic illustration of the multi-agent architecture that we propose. It is composed of simulation models that represent industrial systems. The models are interconnected with an agent's society.

If we consider the two distribution aspects (on the simulation process and on the method), we can describe our contribution as the following:

- *Centralized simulation, centralized method*: Our methodology allows a more powerful use of submodels. It also permits a better understanding of the simulation model in order to distinguish the different kinds of flows in the manufacturing system.
- *Centralized simulation, distributed method*: $\mathcal{M}_{\mathcal{A}}\mathcal{MA}$-$\mathcal{S}$ can contribute to support and facilitate the merging of simulation models. Moreover, it will propose a formal framework

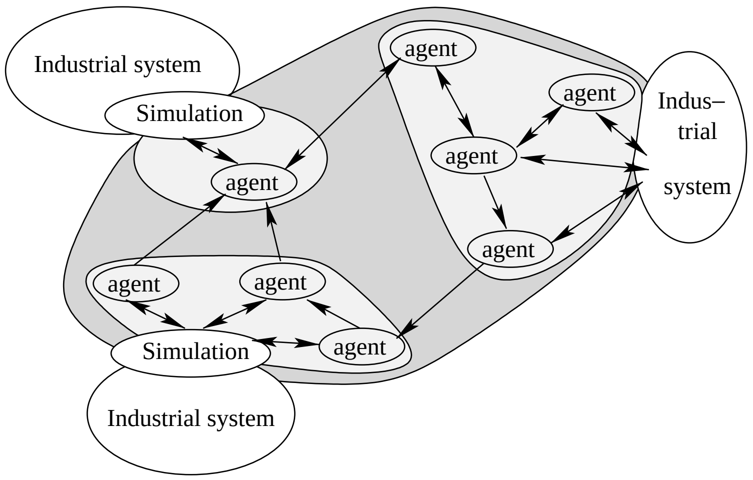


Fig. 1. Example of a multi-agent system for distributed simulations.

in which communication capabilities between designers could be defined.

- *Distributed simulation, centralized method*: This point of view is used by SWARM (Burkhart, 1994) and ARÉVI (Duval et al., 1997). Our methodology proposes a formal framework and an architecture to support the communication between the simulation tools which cannot interact naturally. This approach is also adopted by HLA (US Department of Defense, 1996).
- *Distributed simulation, distributed method*: This is the natural point of view of $\mathcal{M}_{\mathcal{A}}\mathcal{MA}$-$\mathcal{S}$. In this case, our methodology permits to respond to all the problems from the three previous points. Moreover, it supports the new industrial organizations.

## 3. State of the art

In this section, we present a brief state of the works on the three domains that influence our works: the modeling of enterprise, the simulation and the multi-agent systems.

### *3.1. Modeling of enterprise*

This domain aims to model the enterprises (Vernadat, 2001). It has a natural influence on $\mathcal{M}_{\mathcal{A}}\mathcal{MA}$-$\mathcal{S}$. In fact, our methodology has the purpose to support the new enterprise organizations, and the modeling of enterprise has the major goal to model this kind of structure. We study some methods and languages that are used in this domain. This work permits to highlight the concepts that must be supported by $\mathcal{M}_{\mathcal{A}}\mathcal{MA}$-$\mathcal{S}$. In the following points, we briefly discuss the advantages and the defaults of each of them according to our problems.

#### *3.1.1. General methods*

Behind this term of "general method", we put the methods and the methodologies that permit to model all the enterprises in a global way. For example, CIMOSA (AMICE, 1993; Vernadat, 1998), PERA (Williams, 1994) and UEML (Vernadat, 2001) support all the aspects of an enterprise.

However, because these methods are based on an activity modeling, they cannot be directly used in our approach. In fact, $\mathcal{M}_{\mathcal{A}}\mathcal{MA}$-$\mathcal{S}$ is a simulation method that must support more low-level concepts such as queues, processing units, resources, etc.

From this kind of method, we take a systemic approach to model the industrial systems (a system is composed of a physical, an informational and a decisional subsystem) (Le Moigne, 1977).

### 3.1.2. Information system methods and languages

The computer science produces a lot of methods and languages that can be used to model the information system of an enterprise: MERISE (Tardieu et al., 1985), UML (Booch et al., 1997), etc.

We extract from them the object modeling concepts and their life cycles (if they exist).

### 3.1.3. Specialized languages

Some languages are developed to support the specific aspects of an enterprise. For example, the IDEF (US Air Force, 1993) suite proposes a set of languages that includes IDEF0 (activity modeling) and IDEF1 (information modeling).

All these languages are interesting, but as for the general methods, they do not have the right modeling level.

## 3.2. Simulation methodology

The simulation is the natural domain of our methodological approach. We studied two simulation methodologies: CM (Nance, 1981) and ASCII (Kellert and Force, 1998; Kellert and Ruch, 1998).

The Conical Methodology (CM) is one of the oldest simulation methodologies. Its interest is in its modeling framework (life cycle, modeling stages, etc.), which inspirates the $\mathscr{M}_{\mathscr{A}}\mathscr{M}\mathscr{A}$-$\mathscr{S}$'s framework (and especially the life cycle).

The ASCI methodology proposes a way to distinguish the three subsystems of an industrial system according to the systemic approach proposed by Le Moigne (1977). Moreover, some interesting concepts are supported (human resources, etc.).

Even if they are expressive, these methods do not propose a natural way to model a distributed industrial system. To tackle this problem, our methodological approach uses the multi-agent systems, which are presented in the following section.

## 3.3. Multi-agent systems

The multi-agent systems (MAS) are a new modeling paradigm inherited from the object modeling and from the distributed artificial intelligence (Ferber, 1995; Gasser, 1991). A multi-agent system is a "distributed system composed of a whole of agents interacting according to modes of cooperation, competition and coexistence (Chaib-draa, 1994; Chaib-draa, 1996; Moulin and Chaib-draa, 1996). A MAS is distinguished from a collection of independent agents by the fact that the agents interact in order to carry out a task jointly or to achieve a particular goal jointly" (Chaib-draa et al., 2001).

The multi-agent systems are very interesting for the following reasons:

*Decisional process modeling*: The cognitive and interaction capabilities of the agents permit to model and simulate the decisional processes of an industrial system (Bournez, 2001; Kabachi, 1999; Burlat, 1996);

*Simulation model distribution*: Because the MAS are naturally distributed, the agents can support the distribution of the simulation models (Duval et al., 1997);

*Modular modeling*: The autonomy of the agents allows to our methodology to be more modular and to support a component modeling approach (Chen and Szymanski, 2001; Barbuceanu and Fox, 1995). In fact, the opening of MAS (Vercouter, 2000) could be a good way to support this feature. For example Vercouter (2000) proposes an architecture that permits to agents to enter or exit from a MAS at any time. This architecture can be adapt to support a modular modeling.

According to the different points from this section, we decide to propose a methodological approach, which is presented in the rest of this paper.

## 4. Methodological approach

In this section, we present the first results on our methodological approach. We successively expose the life cycle of the models, and his major stages.

## 4.1. Life cycle

The development of a methodological approach passes by a first major stage: the definition of the life cycle of the models. We devote this section to

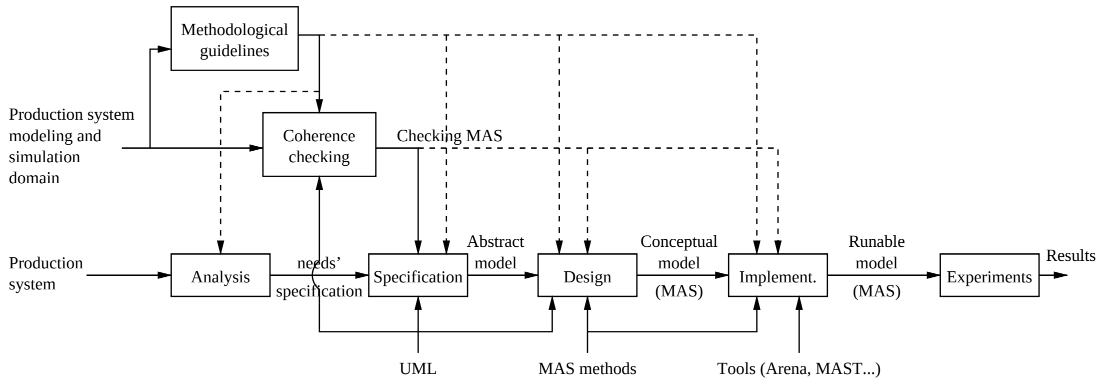


Fig. 2. Life cycle.

define this life cycle for the models of distributed industrial systems. Starting from the assets of the software engineering (Sommerville, 1998; Ghezzi et al., 1991) and the works already completed in the field of simulation (Nance, 1981; Kellert and Force, 1998; Kellert and Ruch, 1998), we propose to extend the existing approaches to take the new enterprise organizations into account.

Fig. 2 illustrates the life cycle used by $\mathscr{M}_{\mathscr{A}}\mathscr{M}\mathscr{A}$-$\mathscr{S}$. It is based on the classical life cycles (from the CM (Nance, 1981) for instance). Our contributions are restricted to the adaption of the specification, the design, and the implementation (which are presented in the following sections). We also introduce two special stages: the methodological guidelines and the coherence checking.

### 4.1.1. *Analysis*

Let us recall that, within the framework of the software engineering, the analysis is the phase where the functionalities of the system and the nonfunctional constraints must be described. For the simulation, the methodologies specialize these principles by considering that the needs' specification (produced during the analysis) must contain the abstract description of the modeled system. This description includes all information which seems necessary to the implementation of a simulation process: the description of the physical infrastructures (storage areas, processing units, etc.), the information needed to the simulation (bill of material, manufacturing routing, etc.), the information needed to drive the system (management policies, production orders, etc.), a list of experiment plans, the specification of the simulation goals.

Unfortunately, the problem of the information harvest is complex and is not yet the subject of a consensus within the community. From this heterogeneity, we decided to not tackle this problem in our works. We suppose that the phase of analysis produces the information necessary to the phase of specification.

### 4.1.2. *Specification*

The specification within a simulation context is the translation of the information from the needs' specification in a formal model. Indeed, the phase of analysis does not allow to obtain a formal model. It is necessary to examine the information of the needs' specification and to build a model by using a method or a technique based on a formal approach. The formalisms and the methods strongly depend on the methodological approach: for example, ASCI uses the directed diagrams Entity-Association-Attribute (EAA) and the object diagrams. If we take a methodology resulting from modeling of enterprise like CIMOSA (Zelm et al., 1995; Vernadat, 1998), the specification partially corresponds to the modeling level of the design specification. In addition, the concepts used to model an industrial system on this level are also specific to the used methodological approach. The specification stage is described in Section 4.2.

### 4.1.3. Design

The phase of design consists in the building of a data-processing model (also called conceptual) which describes in a more precise way the model resulting from the specification. The objective is not to choose the infrastructures of the data-processing execution. But, it is to build a model independent of any tool and any software platform. Thus, only the implementation choices that not influence the choice of the physical infrastructures are carried out. In general, this phase is reduced to nothing (Conical Methodology (Nance, 1981)) or is confused with the phase of implementation (ASCI (Kellert and Force, 1998; Kellert and Ruch, 1998)).

Within the framework of $\mathscr{M}_{\mathscr{A}}\mathscr{M}\mathscr{A}$-$\mathscr{S}$, we consider that the phase of design is significant. Indeed, we decided to not introduce the multi-agent systems (MAS) during the phase of specification because, as the methodologies presented in Section 3 illustrate it, we think that the choices of implementation must be hidden as much as possible to the user of the methodology. We consider that the design is the phase during which an MAS modeling could be introduced. The use of the MAS is not in contradiction with the definition of the design resulting from the software engineering (Sommerville, 1998; Ghezzi et al., 1991). Indeed, the selected MAS modeling approach (the approach "Vowels" (Demazeau, 1995; Demazeau, 1997)) is independent of any software implementation. Consequently, we can create a multi-agent model respecting this approach without choosing a platform of execution (`mast`, Aalaadin, etc.). In Section 4.3, we describe in detail this phase of design.

### 4.1.4. Implementation

The implementation is the translation of the model resulting from the design on a particular software platform. In general, the methodologies propose a specific tool to carry out the simulation process: Aren® (Kelton et al., 1998), Simple++® QNAP (Veran and Potier, 1984), etc. The result of this phase is a model which generates a set of results starting from the experiment plans defined during the analysis. We present the phase of implementation in Section 4.4.

### 4.1.5. Experiments

The phase of experiments is the stage during which the customer uses the simulation model on a set of experiment plans. We do not study this stage in the current state of our works. We assume that the existing works from the other methodologies could be used.

### 4.1.6. Methodological guidelines

The methodological guidelines are written during this stage. It permits to specify the principles of our methodology (life cycle, modeling elements, methods, etc.) that will be used to build a simulation model. The guidelines are, at the same time, the specifications of $\mathscr{M}_{\mathscr{A}}\mathscr{M}\mathscr{A}$-$\mathscr{S}$ and an user guide.

Currently, the guidelines are limited to the specification of $\mathscr{M}_{\mathscr{A}}\mathscr{M}\mathscr{A}$-$\mathscr{S}$ from Galland (2001). It will evolve according to the progresses of our works on the methodology.

### 4.1.7. Coherence checking

The coherence checking is a phase during which a multi-agent architecture was built. It aims to check the coherence of the different simulation models. This stage is not presented in this paper. You must see Galland (2001) for details.

In the following sections, we describe our propositions on the life cycle stages.

## 4.2. Phase of specification

The phase of specification is crucial in our methodological approach. Indeed, it corresponds to the moment when the first model expressed formally must be produced. In this section, we present the methodological bases and the subjacent principles of the abstract model's specification. Initially, we point out the basic concepts of discrete-event simulation. We also describe the Uml metamodel (Muller, 1997; Booch et al., 1997) which allows to define the modeling elements. In this article, we do not explain the principles of the discrete-event simulation, which are already presented by Banks (1999), Banks et al. (1996), Carson (1993).

### 4.2.1. Modeling elements

In this section, we describe the whole of the modeling elements which are used within the framework of the specification for the creation of an abstract simulation model. This building must be carried out starting from the information collected and exposed in the needs' specification. Our methodological approach considers that the distributed production system can be broken up according to the systemic approach proposed by Le Moigne (1992): an operational subsystem, an informational subsystem and a decisional subsystem. We propose modeling concepts for each of these subsystems, as well as a whole of common concepts. The abstract simulation model is composed of three strongly interdependent submodels.

### 4.2.2. Physical subsystem

*Definition*: The physical subsystem is the whole of the industrial infrastructures of the modeled system. The basic concepts supported by $\mathcal{M}_{\mathcal{A}}\mathcal{M}\mathcal{A}$-$\mathcal{S}$ are partly from Banks (1999), Banks et al. (1996), Carson (1993) and from simulation softwares as AREN® (Kelton et al., 1998) or SIMPLE++®:

- *Composition*: We include the concepts of model and submodel that already exist in simulation tools. This is necessary because of the more and more complexity of the modeled systems.
- *Critical resource*: We are only interested by the critical resources, i.e., the resources that can stop the physical flow. They are of two categories:
    - *The active resources*: They are used to realize an activity (Banks, 1999). They can be processing units—transformation of products—human resources, storage area, and transport means (conveyors, transporters or roads).
    - *The passive resources*: They are essential to an activity, but they do not give additional value to the products. In most of cases, they are used by the active resources to do these activities.
- *Queue*: The queues are modeling artifacts that represent a list of physical entities. These entities are waiting for an event (active resource release, activity end, etc.) to continue there paths along the physical flow. The management policy of the queues can be defined in the decisional subsystem.
- *Structuration of the physical flow*: To create a simulation model of the physical subsystem, we propose a set of additional modeling elements. They permit to define the physical paths used by the physical entities:
    - The *links* are the sections of the paths used by the entities to go along the physical flow;
    - The *junctions* and the *forks* permit to simulate cross-roads into the physical flow;
    - The *jumps* are an syntaxical simplification for links. They permit to define links without any explicit graphical representation;
    - The *exit* points and the *entry points* allow to the flows outside the current model to enter and exit from it.
- *Distribution*: In this category, we define the whole of modeling elements that permits to distribute simulation models. They are defined in Galland (2001): remote model, remote resource, remote bill of material, remote decision-center, etc.

We define the modeling artifacts in an extension of the UML metamodel. For instance, Fig. 3 shows the definition of a model, a stochastic law, an attribute and a remote objet. First, this metamodel defines a name space (NAMESPACE) as an object that can contain a set of modeling elements (ROOTELEMENT). Each model is defined as an extension of a name space. One type of model is describes for each subsystem (PHYSICALMODEL, DECISIONALMODEL, INFORMATIONALMODEL) and for the entire simulation model, i.e., a model that contains all the others (MAINMODEL). This metamodel also defines the stochastic laws (LAW), and the modeling elements used in each models (SIMULATIONELEMENT).

The physical subsystem metamodel is built in the same way. Fig. 4 illustrates a part of it. It corresponds to the definition of the modeling elements for the transport means. The roads (Route) permit to reach a destination. The

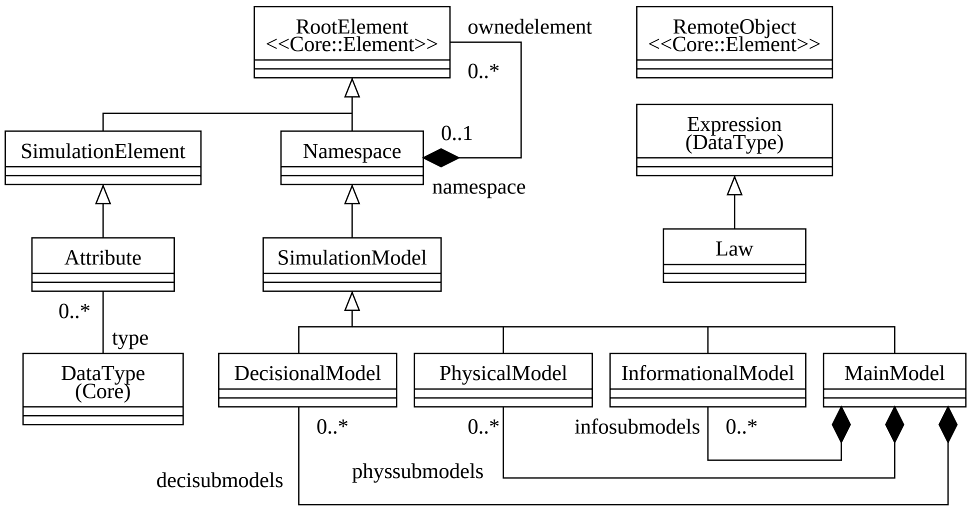


Fig. 3. Part of the common metamodel.

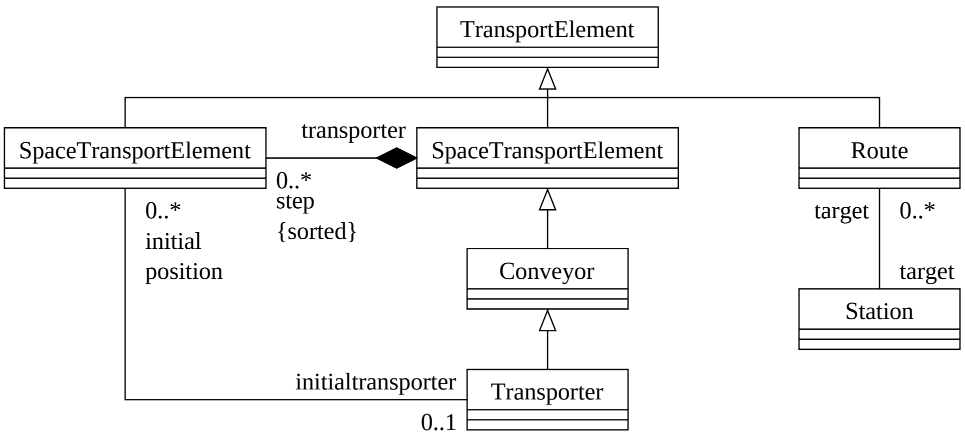


Fig. 4. Part of the physical subsystem metamodel.

elements of type TRANSPORTELEMENT contain a stochastic law for the transport duration. Thus, a road supports the temporal aspect of the transport. The two other kinds of transport means (conveyor and transporter) extend the concept of road by including a spacial aspect. The difference between a conveyor and a transporter is the limitation of the transport resources for the second.

To permit an easier design of the simulation model, we propose a graphical language attached to each object defined in the metamodel. This formalism is described in Galland (2001).

*Example*: To illustrate our remarks on the operational subsystem modeling, we propose to create a model of a simple distributed system: a consortium of three enterprises $E_1$, $E_2$ and $E_3$. The objective of this consortium is to produce movement-detecting cameras. We propose this example in order to facilitate the understanding of the used concepts and to illustrate the use of our methodological approach.

The enterprise $E_1$ produces sensors in its workshop $A$. They are sent to the second enterprise. This last assembles in its workshop $B$ the sensors with the cases which are locally manufactured.

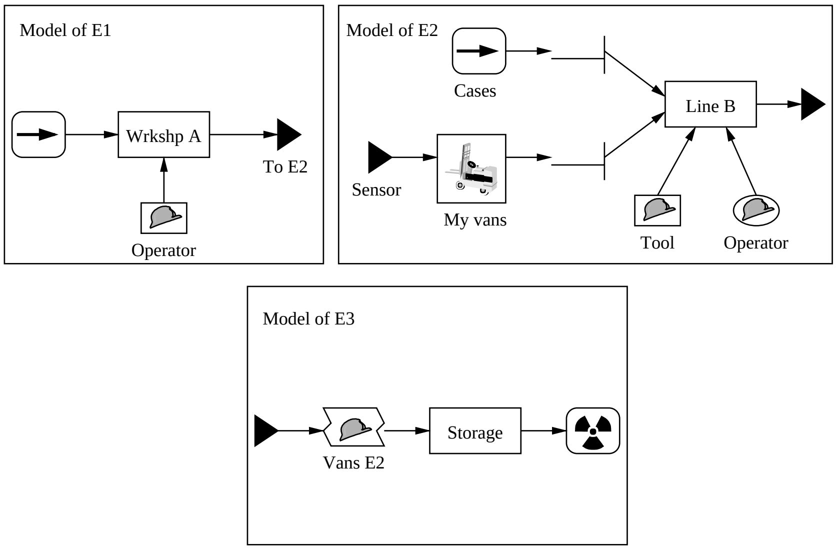


Fig. 5. Example of a physical subsystem model.

Then, $E_2$ forwards the resulting detectors to the third member of the consortium which must store the final products. The transport between these three enterprises is exclusively carried out by the transport services of $E_2$. In addition, the workshop $A$ uses a critical resource which corresponds to an operator, and workshop $B$ uses two resources: a critical resource which makes it possible to assemble the cameras, the second is a passive resource representing an operator supervising the tasks of the workshop $B$. Moreover, the agreements between the members of the consortium specify that $E_1$ does not know how the sensors are transported, and that $E_2$ does not force $E_3$ to use its transport services. These considerations enable us to put the transport modeling artifacts in the models of $E_2$ and $E_3$. Fig. 5 graphically illustrates the three models.

### 4.2.3. Decisional subsystem

*Definition*: The decisional subsystem is the whole of the organizational structures and decision-making processes of an industrial system. Our UML metamodel defines a language that permits to describe the relational structures between the *decision-making centers*. The centers can take operational, tactical and strategical decisions. The relationship between centers can be hierarchic or cooperative. This point of view is issued from the works on the organizational structures in industrial systems (Burlat, 1996; Kabachi, 1999; Berchet, 2000) and in multi-agent systems (Hannoun et al., 2000).

Each decision-making center includes at least one behavioral model. It can be taken in a whole of models where the extrems are protocolar and reactive. A protocolar model uses a communication protocol such as the contract-net protocol. A reactive model defines a set of actions that must be done when stimuli occur.

Fig. 6 shows a part of the decisional subsystem metamodel. It contains the definition of the three types of decision-making centers: operational (OperationalCenter), tactical (TacticalCenter) and strategical (StrategicalCenter). Moreover, we include the definition of the remote centers (RemoteCenter), which correspond to the centers defined in another simulation model. Each decision-making center can be attached to one or more behavioral models (BehaviouralModel).

*Example*: Let us take again the example of the consortium of three enterprises. Here, we are interested exclusively in the decisional subsystem

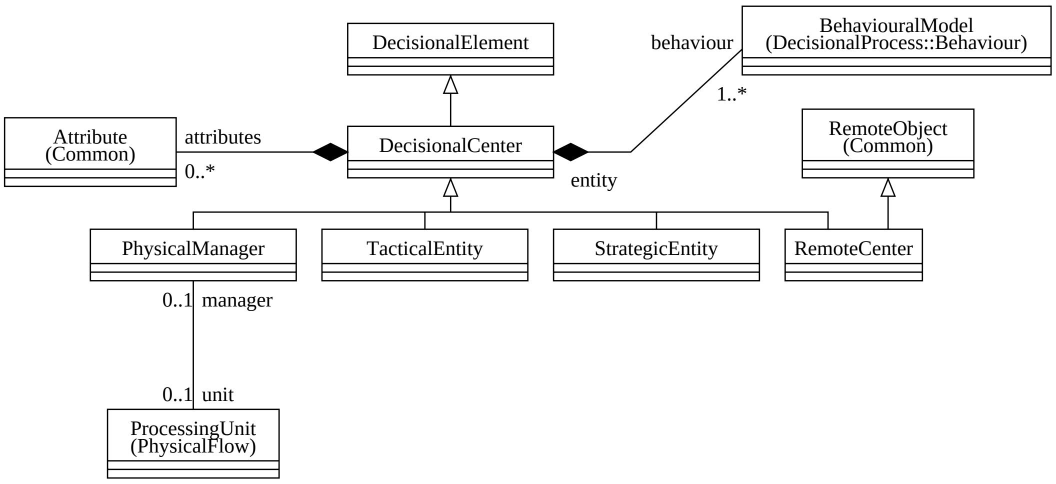


Fig. 6. Part of the decisional subsystem metamodel.

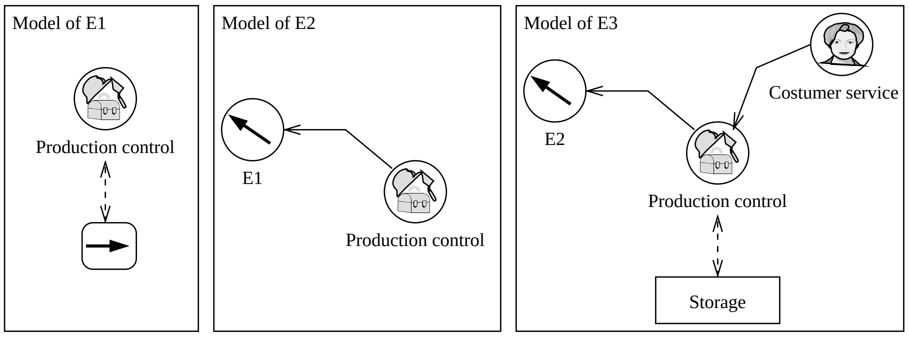


Fig. 7. Example of a decisional subsystem model.

modeling. First, we are interested in the management policy of this consortium. We consider that the system is managed a pulled and with a pushed flow management policy. Indeed, the enterprise $E_3$ is in direct relation with the “market”. Each time its stock of cameras does not make it possible to answer to an order, this enterprise sending a production order to $E_2$. This one has pulled flow management policy. For each production order coming from $E_3$, it makes correspond a production order for $E_1$. This last launches the production of the number of sensors claimed by $E_2$. The production process uses a pulled flow management policy. Fig. 7 illustrates the decisional models for each of the three members of the consortium.

Each decisional center has its own behavior (the language used is defined in Galland (2001)):

*Customer service*: The customer service generates production orders according to a given statistical law. It is a reactive behavior center. Indeed, it reacts only to one temporal event and irrevocably sends a production order to the center of production control. The behavior can be described as follows:

```
CONTEXT ‘‘Customer service’’
WHEN EACH TIME(
    INFORMATION(
    ‘‘law of customer order arriving’’))
THEN
```

```
  SEND TO ''Production control''
    TYPE ORDER
    NAMED ''Production order''
    DATA INFORMATION(''size of the PO'');
END
```

*Production control of* $E_3$: The production control allows the establishment of the link between the production orders coming from the customer service and the real state of the stock. If this last is too weak to answer to the order, the production control center sends a production order to $E_2$. The behavior of this center is:

```
CONTEXT ''Production control''
WHEN RECEIVE ''Production order''
    FROM ''Customer service''
    WITH PARAMS (''quantity to produce'')
THEN
    stock state=OPERATION(get_quantity);
    IF stock state < ''quantity to pro-
duce''
    THEN
      SEND TO E2
        TYPE ORDER
        NAMED ''Production order''
        DATA ''quantity to produce'';
    END IF
END
```

*Production control of* $E_2$: This center generates a production order of sensors towards $E_1$ for each order coming from $E_3$:

```
CONTEXT ''Production control''
WHEN RECEIVE ''Production order''
    WITH PARAMS (''size of the PO'')
THEN
    SEND TO E1
      TYPE ORDER
      NAMED ''Production order''
      DATA ''size of the PO'' *
        INFORMATION(
          bill of material camera,
          ''quantity of sensors'');
END
```

*Production control of* $E_1$: This center generates a physical entity for each sensor having to be produced. Its behavior can be defined as follows:

```
CONTEXT ''Production control''
WHEN RECEIVE ''Production order''
    WITH PARAMS (''size of the PO'')
THEN
    $i = 1$;
    WHILE($i$<=''size of the PO'')
    DO
      OPERATION(generate_entity, INFORMA-
TION(entity));
    $i = i + 1$;
  DONE
END
```

### 4.2.4. *Informational subsystem*

*Definition*: The informational subsystem contains the information used by the two other subsystems. In the modeling language, we define the concepts of *bill of material, manufacturing routing and entities* (physical or decisional). Fig. 8 shows the elements that permit to model a manufacturing routing model: production unit, raw material source, etc.

*Example*: Let us take again the example of the consortium presented in Section 4.2.2. Consider the models of bills of materials (cf. Fig. 9). Each enterprise has its own vision of the products. However, the sensor used by $E_2$ is in fact the definition being in the model of $E_1$. We have a simple example of the use of a distant definition of a product. The bill-of-material model of $E_3$ is the same as the model of the enterprise $E_2$.

Consider now the manufacturing routing. In our example, only the first two enterprises must define a manufacturing rounting model. Indeed, $E_3$ does not carry out any transformation on the products. Fig. 9 illustrates the graphical representation which we propose within $\mathscr{M}_{\mathscr{A}}\mathscr{M}\mathscr{A}$-$\mathscr{S}$ .

Thus, in $E_1$, the raw material (RM) is transformed into sensors by the workshop $A$, and in $E_2$, the cameras are obtained starting from the assembly of the sensors and the cases. Note that, for each manufacturing routing model, the treatment units must be associated to processing units from the operational subsystem (they must have

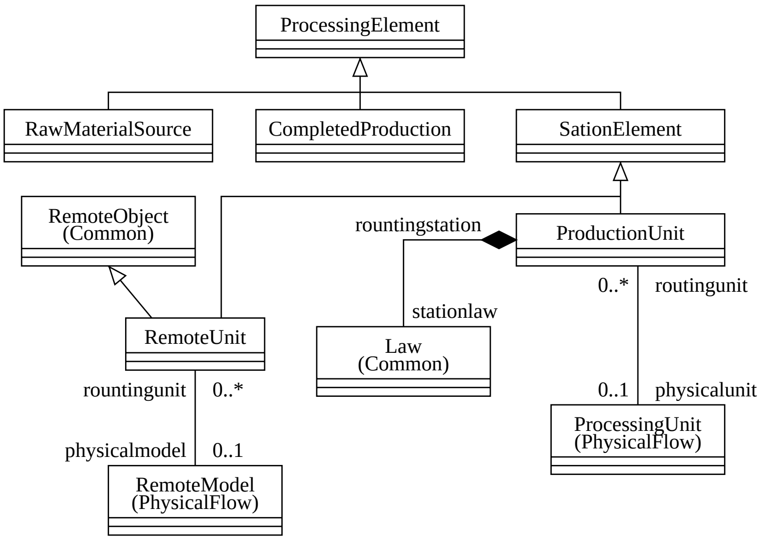


Fig. 8. Part of the informational subsystem model.

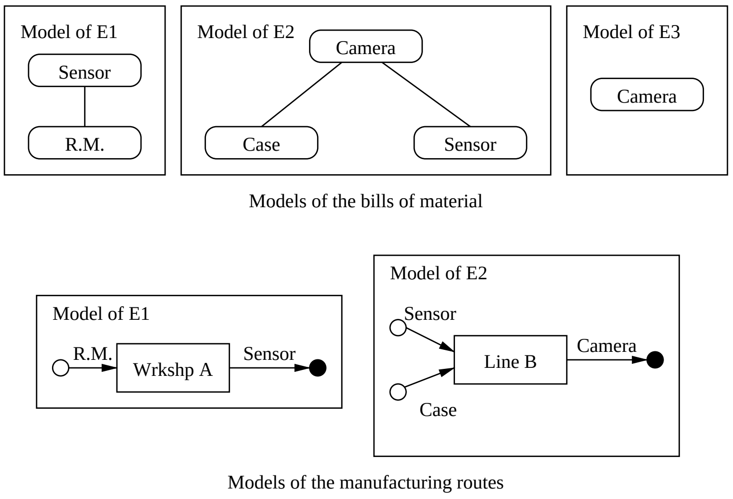


Fig. 9. Example of a informational subsystem model.

the same name). In addition, a treatment must define a whole of times necessary to model the treatments' durations. In the example, all the operations obey to a normal law of average 10 and standard derivation 0.5 (`NORMALE`(10,0.5)).

In this section, we presented the phase of specification. It allows to obtain an abstract simulation model independent of any software platform. In the following section, we present the phase of design, which permits to build a multi-agent model that corresponds to the abstract model previously obtained.

### 4.3. Design

The conceptual model is a multi-agent system (MAS) that corresponds to the abstract model shown in the previous section. It describes a structural organization of agents. But, it does not force to use a particular multi-agent platform or a particular simulation tool. The only one constraint is that it must respect a specification according to the approach "Vowels" (or AEIO) (Demazeau, 1995).

In the following sections, we describe the MAS context and the two models, which are used to build a multi-agent model, and we illustrate the design on the same example from the previous sections.

### 4.3.1. *Multi-agent infrastructure*

To permit the simulation process with an agents' society, we propose an infrastructure, which is illustrated in Fig. 10. It is mainly composed of interconnected agents. This principle permits to distinguish two classes of agents:

- The facilitators (AGF) facilitate the exchange of messages between simulation agents. They are intermediaries between agents' subsocieties. Moreover, the facilitators manage a knowledge database of resources and services (resources, processing unit names, decisional centers, etc.). The AGF route a message to an agent which proposes the desired service.
- The agents for simulation (AGS) support the simulation process. We propose an architecture that permits to interact with AGF, with other AGS, or with objects from the MAS environment. Our architecture is a "white box" that must be filled with the behavior defined in the abstract simulation model.

The facilitators are the only ones agents with a static definition in our approach, i.e., their architecture and their interacts are all defined in $\mathscr{M}_{\mathscr{A}}\mathscr{M}\mathscr{A}$-$\mathscr{S}$, and cannot be extend by an user definition. Only AGS can be extend by user models. Fig. 10 illustrates another aspect of our approach: the recursivity of the architecture. In fact, each agent or environment object can be also an agents' society.

*Definition of facilitators*: We assume that the facilitators are a mean to exchange data between agents. They allow a better modularity of simulation models. They also apply the capability to build a recursive agents' society. We define a facilitator with the approach AEIO:

(1) *Facet "Agent"*: A facilitator is composed by a behavioral module that permits to an agent to be a gateway and a facilitator. We assume that the facilitators are friendly (Vercouter, 2000) with other AGF. Fig. 11 illustrates this architecture:
   (a) *Knowledge*: Knowledge base needed by the raisoning model: services offered by AGS registered in this AGF, etc.
   (b) *Friendly knowledge*: Whole of knowledge needed to realize a friendly behavior. This knowledge is divided into two parts: the

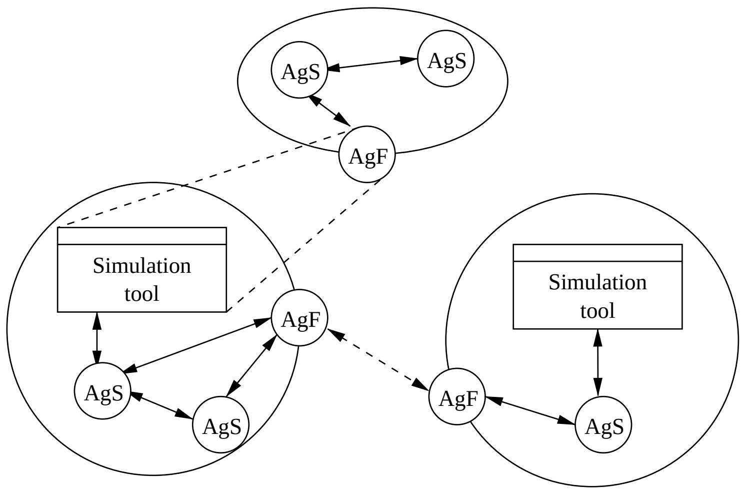


Fig. 10. Simulation model architecture.

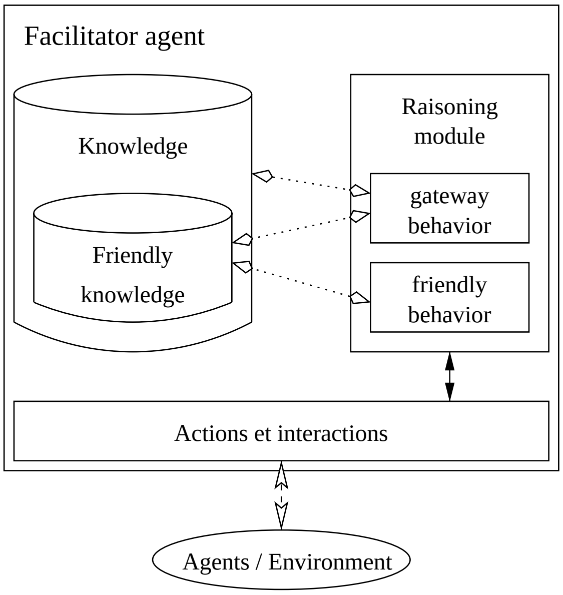


Fig. 11. Architecture of an facilitator.

"model of other AgF" and the common agent knowledge (action or plan semantic, etc.) (Vercouter, 2000).

(c) *Friendly behavior*: This module permits to agents to be friendly for other facilitators. When an AgF would enter to the system, it present itself to other facilitators with the method proposed by Vercouter (2000). When it is accepted, the agent maintains the coherences of its knowledge about their AgS and about other AgF.

(d) *Gateway behavior*: This module permits to agent to be a gateway between agents' subsocieties. It uses the "models of others" from the friendly knowledge.

(e) *Actions and interactions*: This module allows to make actions decided by the cognitive process, and to interact with the environment of the agent.

(2) *Facet "environment"*: The facilitator does not have any relationship with the MAS environment. This facet is ignored in this model.

(3) *Facet "interaction"*: The facilitator have two main classes of interaction: interactions with other facilitators, and interactions with agents for simulation. In addition of the friendly protocols, we define two other protocols:

- the protocol to send messages to other AgF,
- the protocol that manages messages from AgS, which is illustrated in Fig. 12(a).

These protocol use a state-transition graph formalism. The links are labeled with the messages sended (prefixed by +) or received (prefixed by !). Moreover, each message can be postfixed by parameters.

*Other protocols can be used by the facilitators*: Search of a service in facilitators' knowledge databases, etc.

The exchanged messages during the two previous protocols could be expressed with FIPA-ACL (FIPA, 1998), FIPA-SL (FIPA, 2001; Carron et al., 1999).

(4) *Facet "organization"*: Facilitators are gateways between the agents' societies. They have accointances with other facilitators and authority relationships with AgS. The expression of this kind of organization can be done with MOISE (Hannoun et al., 2000).

*Definition of simulation agents*: This kind of agent is not entirely specified in $\mathscr{M}_{\mathscr{A}}\mathscr{M}\mathscr{A}$-$\mathscr{S}$. We propose an architectural skeleton of AEIO's facets that permits to the agents to interact with the facilitators, and, in most of cases, with the environment objects. The interaction facet is composed of two protocols: send messages to and receive messages from facilitators. Fig. 12(b) illustrates the protocol, which is used to send messages to AgF.

Our methodological approach also proposes a whole of definitions of simulation agents: agent that send physical entities to another model, resource management center, etc. These different models are directly usable. They are the first level of dedicated object templates.

### 4.3.2. *Some rules to translate the abstract model*

To facilitate the building of the MAS model, $\mathscr{M}_{\mathscr{A}}\mathscr{M}\mathscr{A}$-$\mathscr{S}$ proposes a whole of methodological rules, which permit to translate the abstract model

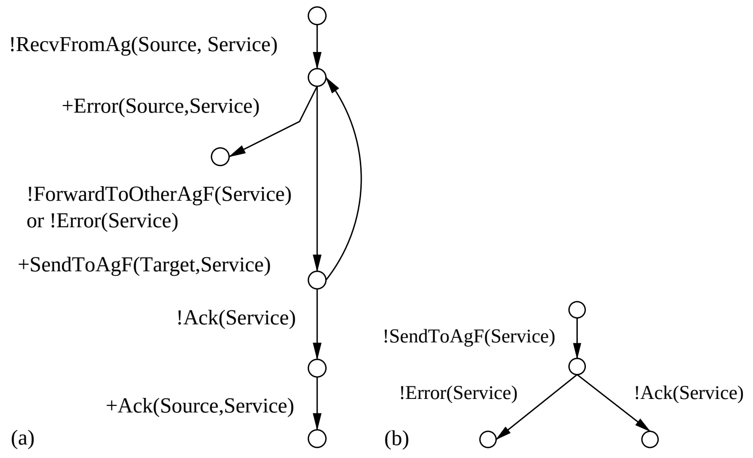


Fig. 12. Protocols for agents.

into a MAS model. These rules are currently under development (Galland, 2001). They respect the following grammar:

```
rules::= ‘‘AGENT’’ [‘[’ agent_name‘]’]
        ‘←’abstract_object_name ‘=’
        description ‘;’ rules
        |
        empty
description::= ‘A’ ‘{’ pseudo-code ‘}’ ‘∧’
        ‘E’ ‘{’ pseudo-code ‘}’ ‘∧’
        ‘I’ ‘{’ pseudo-code ‘}’ ‘∧’
        ‘O’ ‘{’ pseudo-code ‘}’ ‘∧’
        ‘U’ ‘{’ pseudo-code ‘}’
agent_name::=string
abstract_object_name::=string
```

This grammar describes a rule that permit to create an agent named agent name from objects of the abstract simulation model object (abstract object name). This description is based on the approach “Vowels” (Demazeau, 1995), and defines, by the way of pseudo-code, the content of each facet of the new agent. The pseudo-code can have any form, e.g., MOISE for the organization and FIPA-ACL for interaction.

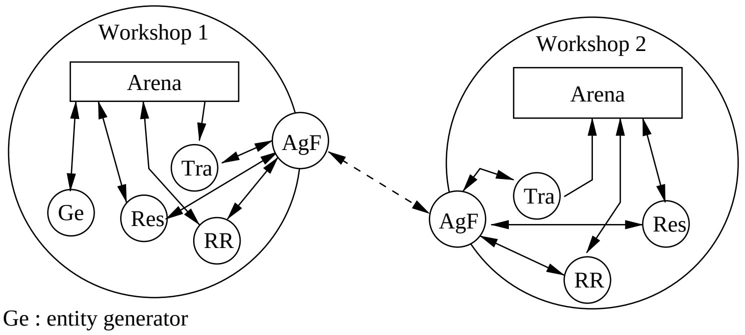
$\mathscr{M}_{\mathscr{A}}\mathscr{M}\mathscr{A}$-$\mathscr{S}$ propose some simple rules that allow to quickly build a skeleton of MAS model. But, these rules are not exhaustives and must be updated by $\mathscr{M}_{\mathscr{A}}\mathscr{M}\mathscr{A}$-$\mathscr{S}$ developers and users (Galland, 2001). For example, we propose the following rule:

RULE 1 :

*We consider that physicial production units can be entirely translated into the MAS environment.*

*ENVIRONMENT* ← *ProcessingUnit* =

$$\left\{ \left\langle \begin{array}{l} \_.\mathtt{name}, \\ \left\{ \begin{array}{l} \langle \mathtt{getState}, method, \emptyset \rangle, \\ \left\langle \begin{array}{l} \mathtt{beginProcess}, event, \\ \{\langle \mathtt{entit\acute{e}}, \mathtt{entit\acute{e}}, \alpha \rangle\} \end{array} \right\rangle, \\ \left\langle \begin{array}{l} \mathtt{endProcess}, event, \\ \{\langle \mathtt{entit\acute{e}}, \mathtt{entit\acute{e}}, \gamma \rangle\} \end{array} \right\rangle, \end{array} \right\}, \\ \_.\mathtt{getAttributes()} \cup \\ \left\{ \begin{array}{l} \left\langle \begin{array}{l} \mathtt{delay}, Law, \\ \mathtt{"Law\_for\_"+\_.name} \end{array} \right\rangle, \\ \left\langle \begin{array}{l} \mathtt{fail}, Law, \\ \mathtt{"Fail\_Law\_for\_"+\_.name} \end{array} \right\rangle \end{array} \right\} \end{array} \right\rangle \right\}$$



Ge : entity generator
Res : competence manager
Tra : entity transfert
RR : remote competence manager

Fig. 13. Example of a multi-agent model.

### 4.3.3. *Example*

From the example presented during the specification phase, we want to highlight the following agent classes:

- the agents corresponding to resource management decision centers,
- the agent corresponding to the entity generation center,
- the agents that permits to send physical entities from a simulation model to another,
- the agents representing the "remote" resource managers.

Fig. 13 illustrates the structure of the multi-agent system. $\mathcal{M}_{\mathcal{A}}\mathcal{MA}$-$\mathcal{S}$ considers that agents are white boxes, which must be filled with the parts of the abstracts models. The means to interact between agents is proposed by $\mathcal{M}_{\mathcal{A}}\mathcal{MA}$-$\mathcal{S}$ (message syntax, service specification, etc.).

## 4.4. *Implementation*

### 4.4.1. *Definition*

The implementation is the last phase which we want to adapt to the support of the modeling and the simulation of distributed industrial systems. The objective is here to transform the model of multi-agent system previously obtained into a software simulation model. Within this intention, this phase aims to choose the tools which will have to carry out simulation (ARENA® (Kelton et al., 1998), SIMPLE++®, QNAP (Veran and Potier, 1984), etc.) and the multi-agent platforms being used as support of the execution of the agents (mast (Boissier et al., 1998), ARÉVI (Duval et al., 1997), SWARM (Burkhart, 1994), AALAADIN (Ferber and Gutknecht, 1998), etc.).

To tackle the translation of the multi-agent model, we propose a set of constraints that must be respect by the softwares:

*For the simulation tools*: The simulation tools are objects in the MAS environment:

- They must propose an interface of communication usable by the agents. This interface is based on the exchange of messages. Thus, for each service of the type method (resp. event), the object must be able to treat the reception (resp. the emission) of a tagged message by the name of the service. The syntax used by the messages depends on the selected tools.
- The objects of the environment must establish behaviors compatible with those awaited in the conceptual model.
- Any object is independent of the other objects, except if a relationship is explicitly described in the conceptual model. In this case, the platform must be able to support the relations between the objects.
- The objects should not endanger the course of simulation. Thus, it must be in conformity with

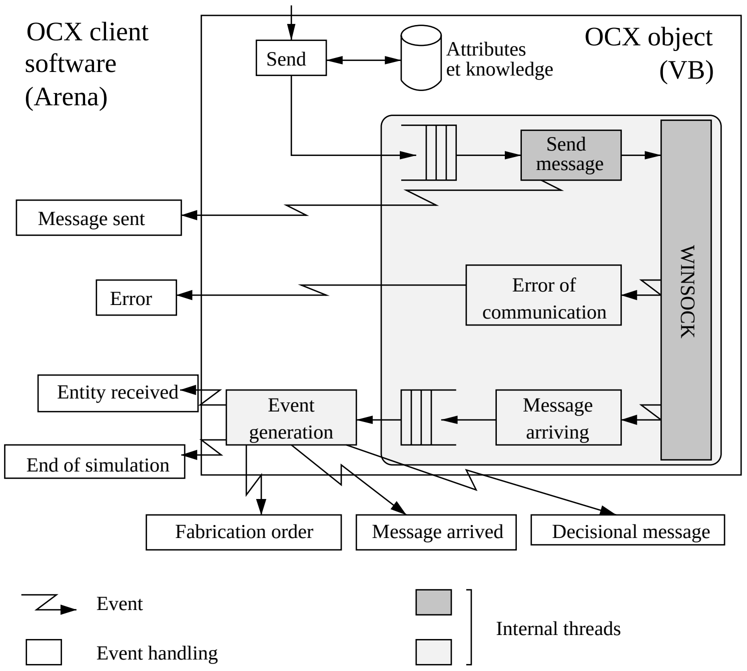


Fig. 14. AREN$_{®}$ ↔ $ MAS interface.

the policy of synchronization of the models (the constraints of causality and promptness must be respected).

*For the agents*: The agents result directly from the conceptual model. Thus, the constraints of implementation are common to any multi-agent system:

- autonomy of reasoning of the agents;
- capacities of interaction;
- data-processing distributivity of the system;
- modeling according to the approach Vowels.

To these constraints, we add that the achievements of the agents must respect their conceptual definition: they must exclusively propose and use the same services. The passage from the conceptual model to the software model is much simpler to realize for agents than for the objects of the environment. Note that the agents must respect the protocols and the formats of messages used by the objects of the environment. In addition, we use the same notation to describe the messages exchanged by the various agents (facilitators or not).

### 4.4.2. *Example*

The previous multi-agent model example must be instanced. We choose to use the simulation software AREN$_{®}$ with a simple implementation of a multi-agent platform in VISUAL BASIC$_{®}$. Fig. 14 illustrates the architecture of an OCX object, which is used to support interactions between AREN$_{®}$ and the multi-agent system.

To obtain the runable simulation model, we assume that the physical subsystem is translated into AREN$_{®}$, the decisional into the multi-agent system, and the informational subsystem into AREN$_{®}$ and the multi-agent system.

## 5. Conclusion and perspectives

In this paper, we propose a methodological approach for the creation of simulation models of complex and distributed systems. This approach, named $\mathscr{M}_{\mathscr{A}}\mathscr{MA}$-$\mathscr{S}$, was introduced in Galland et al. (1999) and Galland et al. (2000a), and specified in Galland (2001). It facilitates the modeling and the simulation of decision-making processes, whether centralized or distributed. It also provides better reaction capacity in terms of modeling (systemic approach, etc.). Lastly, one of its strengths is its capacity to produce re-usable models. Nevertheless, some applications (teaching application in the simulation scope (Galland et al., 2000b) and cyclic scheduling of a production system (Campagne et al., 2001) enabled us to highlight certain weaknesses in our approach. First of all, collaborative modeling is not fully taken into account by $\mathcal{M}_{\mathcal{A}}\mathcal{M}\mathcal{A}$-$\mathcal{S}$. Indeed, we only propose a basic architecture capable of supporting this approach. In addition, we have yet to propose more highly developed modeling components (dedicated template, etc.), which would make modeling easier. Moreover, this paper presents the first works completed on $\mathcal{M}_{\mathcal{A}}\mathcal{M}\mathcal{A}$-$\mathcal{S}$. It still remains of many points for which a study proves to be necessary (synchronization of the simulation models, methodological guidelines, etc.).

Finally, our methodological approach needs to move towards the attempts at standardization that concern our areas of investigation: (i) the Unified Enterprise Modeling Language (UEML) for modeling distributed production systems, (ii) the Discrete-Event systems Specification (DEVS) to support distributed and interoperable simulation, (iii) the Foundation for Intelligent Physical Agents (FIPA) whose aim is to define the set of components of a multi-agent system platform. The previous points are currently being developed within the frame of an extension of the $\mathcal{M}_{\mathcal{A}}\mathcal{M}\mathcal{A}$-$\mathcal{S}$ methodological approach.

## References

AMICE, 1993. CIMOSA: CIM Open System Architecture, 2nd revised and extended edition. Springer, Berlin.

Banks, J., 1999. Introduction to simulation. In: Winter Simulation Conference, Prentice-Hall, Englewood Cliffs, NJ, pp. 7–13.

Banks, J., Carson, J., Nelson, B., 1996. Discrete Event System Simulation. Prentice-Hall, Englewood Cliffs, NJ.

Barbuceanu, M., Fox, M.S., 1995. The Architecture of an Agent Building Shell. In: Intelligent Agents II: Agent Theories, Architectures and Languages. Lecture Notes in Artificial Intelligence, Vol. 1037, Springer-Verlag, Berlin.

Berchet, C., 2000. Modélisation pour la simulation d'un système d'aide au pilotage industriel, Université de Savoie.

Boissier, O., Beaune, P., Proton, H., Hannoun, M., Carron, T., Vercouter, L., Sayettat, C., 1998. The Multi-Agent System Toolkit, SIC/ENSM-SE.

Booch, G., Jacobson, I., Rumbaugh, J., et al., 1997. Unified Modeling Language 7 Specifications—version 1.1, UML Consortium–Object Management Group.

Bournez, C., 2001. Une architecture multi-agents réflexive pour le contrôle de systèmes de production distribus et hétérogènes, Institut National des Sciences Appliquées (Lyon).

Burkhart, R., 1994. The SWARM multi-agent simulation system. In: OOPSLA Workshop on "The Object Engine".

Burlat, P., 1996. Contribution à l'Évaluation Économique des Organisations Productives: vers une modélisation de l'entreprise-compétences, Université Lyon 2.

Butera, F., 1991. La métaphore de l'organisation: du château au réseau. In: Les Nouveaux Modèles d'Entreprise: Vers l'Entreprise Réseau., Éditions d'Organisation, Paris.

Campagne, J.-P., Grimaud, F., Hacid, S., 2001. Production cyclique: Application et évaluation par simulation chez un équipementier automobile. In: 3ème Conférence Francophone de MOdélisation et SIMulation, Troyes, pp. 965–972.

Carron, T., Proton, H., Boissier, O., 1999. A temporal agent communication language for dynamic multi-agent systems. In: Modelling Autonomous Agents in a Multi-Agent World, Spain, pp. 115–127.

Carson, J., 1993. Modeling and simulation world views. In: Winter Simulation Conference, Prentice-Hall, Englewood Cliffs, NJ, pp. 18–23.

Chaib-draa, B., 1994. Distributed artificial intelligence: An overview. In: Encyclopedia of Computer Science and Technology, Vol. 31, Marcel Dekker, New York, pp. 215–243.

Chaib-draa, B., 1996. Interaction between agents in routine, familiar and unfamiliar situations. International Journal of Experimental and Theorical AI (JETAI) 1 (5), 7–20.

Chaib-draa, B., Jarras, I., Moulin, B., 2001. Systèmes multi-agents: Principes généraux et applications. In: Agent et Systèmes Multi-Agents, Hermès, Paris.

Chen, G., Szymanski, B.K., 2001. Component-based simulation. In: Proceedings of the 15th European Simulation Multiconference, Prague, pp. 68–75.

Chevaillier, P., Querrec, R., Tiseau, J., 1997. Modélisation multi-agents d'une cellule de production. In: 2ème Colloque National de Productique, Casablanca, Maroc, pp. 1–10.

David, R., Alla, H., 1992. Petri Nets and Grafcet—Tools for Modeling Discrete Event Systems. Prentice-Hall, Englewood Cliffs, NJ.

Demazeau, Y., 1995. From interactions to collective behaviour in agent-based systems. In: European Conference on Cognitive Science, Saint-Malo, France.

Demazeau, Y., 1997. Steps towards multi-agent oriented programming. In: Proceedings of the First International Workshop on Multi-Agent Systems (IWMAS), Boston.

De Rosnay, J., 1979. The Macroscope: A New World Scientific System. Harper & Row Publishers, New York.

Dupont, L., 1998. La Gestion Industrielle. Hermès, Paris.

Durand, D., 1975. La systèmique. In: Que Sais-je? Presses Universitaires de France, Paris.

Duval, T., Morvan, S., Reignier, P., Harrouet, F., Tisseau, J., 1997. ARéVi: Une boîte à outils 3D pour des applications coopératives. La coopération 9 (2), 239–250.

Ettinghoffer, D., 1992. L'Entreprise Virtuelle ou les Nouveaux Modes de Travail. Éditions Odile Jacob, Paris.

Ferber, J., 1995. Les Systèmes Multi-Agents–Vers Une Intelligence Collective. InterEditions.

Ferber, J., Gutknecht, O., 1998. A meta-model of the analysis and design of organizations in multi-agent systems. In: Proceedings of the Third International Conference on Multi-Agent systems (ICMAS), pp. 128–135.

FIPA, 1998. Agent Communication Language. Foundation for Intelligence Physical Agents.

FIPA, 2001. The official site of the Foundation for Intelligent Physical Agents (FIPA), http://www.fipa.org/.

Galland, S., 2001. Approche multi-agents pour la conception et la construction d'un environnement de simulation en vue de l'évaluation des performances des ateliers multi-sites. École Nationale Supérieure des Mines et Université Jean Monnet, Saint-Étienne.

Galland, S., Grimaud, F., Beaune, P., Campagne, J.-P., 1999. Multi-agent methodological approach for distributed simulation. In: Simulation in Industry–11th European Simulation Symposium, Erlangen, Germany, pp. 104–108.

Galland, S., Grimaud, F., Campagne, J.-P., 2000a. Methodological approach for distributed simulation: General concepts for $\mathcal{M}_{\mathcal{A}}\mathcal{M}\mathcal{A}$-$\mathcal{S}$. In: Simulation and Modelling: Enablers for a Better Quality of Life—14th European Simulation Multiconference, Ghent, Belgium, pp. 77–82.

Galland, S., Grimaud, F., Campagne, J.-P., 2000. Multi-agent architecture for distributed simulation: Teaching application for industrial management. In: Simulation and Modelling: Enablers for a Better Quality of Life—14th European Simulation Multiconference, Ghent, Belgium, pp. 756–762.

Gasser, L., 1991. An overview of DAI. In: Distributed Artificial Intelligence. Theory and Praxis. Kluwer Academic Publishers, Dordrecht.

Ghezzi, C., Jazayeri, M., Mandrioli, D., 1991. Fundamentals of Software Engineering. Prentice-Hall, Englewood Cliffs, NJ.

Goranson, T., Johnson, M., Presly, A., Rogers, H., 1997. Metrics for the agile virtual enterprise case study. In: Proceedings of the Sixth Annual National Agility Conference.

Grimaud, F., 1996. Conception d'une base de composants logiciels pour l'évaluation des performances des entreprises manufacturières, Université Blaise Pascal, Clermont-Ferrand.

Hannoun, M., Boissier, O., Sichman, J.S., Sayettat, C., 2000. MOISE: An organizational model for multi-agent systems. In: Advances in Artificial Intelligence, Lecture Notes in Artificial Intelligence, Vol. 1952 (Brazil), Springer-Verlag, Berlin, pp. 156–165.

Kabachi, N., 1999. Modélisation et apprentissage de la prise de décision dans les organisations productives. Université Jean Monnet/ENSM.SE, Saint Etienne, France.

Kellert, P., Force, C., 1998. Méthodologie de modélisation orientée objets de systèmes de production—application à une chaîne d'étuvage de bobines d'allumage. Journal Européen des Systèmes Automatisés 32 (1), 107–131.

Kellert, P., Ruch, S., 1998. Méthodologie de modélisation orientée objets de systèmes de production—un processus de construction/validation du modèle générique orienté objets d'un système de production. Journal Européen des Systèmes Automatisés 32 (1), 51–105.

Kelton, W., Sadowski, R.P., Sadowski, D.A., 1998. Simulation with Arena. McGraw-Hill, New York.

Knuth, D.E., 1998. The Art of Computer Programming. Addison-Wesley, Reading, MA.

Law, A., Kelton, W., 1991. Simulation Modeling and Analysis. McGraw Hill, New York.

Le Moigne, J.-L., 1977. La Théorie du Système Général. Théorie de la Modélisation. Presses Universitaires de France, Paris, France.

Le Moigne, J.-L., 1992. La Modélisation des Systèmes Complexes. Editions Dunod.

Moulin, B., Chaib-draa, B., 1996. An overview of distributed artificial intelligence. In: Foundations of Distributed AI. Chichester, England, pp. 3–54.

Muller, P.-A., 1997. Modélisation Objet avec UML. Eyrolles.

Nance, R., 1981. Model Representation in Discrete Event Simulation: The Conical Methodology. Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA.

Nunes, P., 1994. Formes PME et ogranisation en réseaux, INIST-CNRS.

Sommerville, I., 1998. Le Génie Logiciel et ses Applications, InterEditions.

Tardieu, H., Rochfeld, A., Colleti, R., 1985. La Méthode MERISE—Tome I: Principes et Outils, Edition d'Organisation, Paris.

US Air Force, 1993. Integrated Computer Aided Manufactured Definition Language (IDEF methods). Department of Commerce, National Institute of Standards and Technology, Computer Systems Laboratory, Gaithersburg, USA.

US Department of Defense, 1996. High Level Architecture Federation Development and Execution Process (FEDEP) Model, Version 1.0, Defense Modeling and Simulation Office.

Veran, M., Potier, D., 1984. QNAP 2: A portable environment for queueing system modelling. In: Colloque International sur la Modélisation et les Outils d'Analyse de Performance, Paris.

Vercouter, L., 2000. Conception et mise en oeuvre de systèmes multi-agents ouverts et distribués. École Nationale Supérieure des Mines, Saint-Étienne, France.

Vernadat, F., 1998. La modélisation d'entreprise par la méthodologie CIMOSA. In: Modélisation d'entreprise

(CNRS PROSPER "Systèmes de production"), Roisy-en-France.

Vernadat, F., 2001. UEML: Towards a unified entreprise modelling language. In: 3ème Conférence Francophone de MOdélisation et de SIMulation, Vol. 1, Troyes, France, pp. 3–10.

Wang, J., 1998. Timed Petri Nets, Theory and Application. Kluwer Academic Publishers, Dordrecht.

Williams, T., 1994. The Purdue enterprise reference architecture. Computers in Industry, pp. 141–158.

Ye, X., 1994. Modélisation et simulation des systèmes de production: une approche orientée-objets. École Nationale Supérieure des Mines de Saint Etienne.

Zelm, M., Vernadat, F.,Kosande, K., 1995. The CIMOSA business modelling process. Computers in Industry 21 (2).

Zimmermann, A., 1994. A modeling method for flexible manufacturing systems based on colored Petri nets. In: International Workshop on New Directions of Control and Manufacturing, pp. 147–154.