Methodological approach for distributed simulation : Life cycle of $\mathcal{M}_{\mathcal{A}}\mathcal{MA}-\mathcal{S}$

Stéphane Galland et Frédéric Grimaud

Systèmes Industriels Coopératifs laboratory*

Ecole Nationale Supérieure des Mines

158, Cours Fauriel, Saint-Etienne,

42023 Cedex 2, France

e-mail: {galland, grimaud}@emse.fr

Abstract

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

This paper is divided into two parts. In the first part, we position us within the studying framework of a modeling and specification methodology suggested by [GGBC99]. We briefly expose the problems as well as the considered solutions. In the second part we present general concepts attached to our methodological approach $\mathcal{M}_{\mathcal{A}}\mathcal{MA}-\mathcal{S}$: a simulation model life cycle and a simple approach for distributed development. We explain the four major life cycle phases : analysis, specification, conception and implementation. We conclude this paper by some explanations on a teaching case application.

Keywords: Distributed simulation, UML Metamodel, Multi-agent systems, Methodological approach

1 Introduction

Simulation is a recognized tool and is adapted to modern industrial problems. It takes into account the dynamic aspects during production system behavioural studies. The physical, informational and decisional distributions are seldom managed within same tools. Moreover modern simulation tools are rarely accompanied by complete and adapted methodologies. In [GGBC99], we propose a multi-agent approach for simulation ($\mathcal{M}_{\mathcal{A}}\mathcal{MA}-\mathcal{S}$ [1]),which taking the three distribution aspects into account. Thus we iteratively develop the various properties of our methodology and its simulation environment.

In this article we present the general concepts used by $\mathcal{M}_{\mathcal{A}}\mathcal{MA}-\mathcal{S}$. We think that a methodology integrating physical, informational and decisional distributions of simulation models must be itself distributed *i.e.*, submodel developments can be carried out in parallel. This aspect is made possible by the use of simple project management rules. The simulation model life cycle is divided into two parts. The first one is composed by the four traditional methodological phases: analysis, specification, conception and implementation. The second part is a set of specific simulation project phases : experimental plan designs, experiments and result validation. We wish to concentrate our work on the four first stages. Indeed, we think that our contribution (management of physical, informational and decisional distributions) is concentrated on those.

The following section is a short outline of the context in which our work is located. In the section 3, we present general concepts of our methodological approach: distributed development, life cycle and analysis, specification, conception and implementation stages. Finally we conclude and expose our perspectives.

2 Background

2.1 State of the art

The advent of the simulation technics brought to the industrialists the possibility to model the behavior of their systems in much more realistic ways. Indeed other existing technics *e.g.*, the arithmetic approach, operational research or the computer-assisted production control; make it difficult to support all the behavioral aspects of an industrial system. Simulation is one of the rare approaches integrating the dynamics of a system and usable in an industrial context.

However we would like to concentrate our work on the following problems: the integra-

*SIC : Cooperative Industrial Systems

[1]**M**ulti-**A**gent **M**ethodological **A**pproach for **S**imulation

tion of distribution since the first stage of the model's construction, the fact that accompaniment of simulation tools by a methodology is still rare and the need for modularity and reutilisability of the aforementioned models.

To partly solve the problems of the decentralization of physic entities and knowledge as those of times in modelling and simulation, the scientific community has considered the implementation of distributed simulations according to two major approaches: data-processing model and knowledge distribution.

Data-processing distribution consists of a whole of simulation models that are runned on distant processes. This distribution could exist in a computer network (tcp/udp sockets, CORBA, *etc.*) or, on a same computer, in inter-process communications (sharing, ACTIVEX® controls, *etc.*). Data-processing distribution is typically included in the HLA's approach [US 96].

Knowledge distributions *e.g.*, company representation in a world context [Bur96], is composed by the distributions of informations and decisional processes. This means that models are not necessary data-processing distributed. In this context, we simply attached ourself to study informational (products, routing, nomenclatures, production technics, *etc.*) and decisional (decision-making processes, organisational structure, *etc.*) distributions inside industrial companies.

Distributed simulation, which is composed of these two axes (data-processing and knowledge distributions), makes it possible to take into account the international characteristics of companies *i.e.*, problems of technical culture, knowledge and geographical distributions can be supported by distributed simulation models. However this field has not become fully developed yet. Indeed no methodology is truly adapted to these aspects.

2.2 Proposals

We propose to conceive a methodological approach based on the multi-agent concepts. We use the vowel approach (or AEIO) defined by [Dem95] : a multi-agent system is defined according to four major axis : **A**gents, **E**nvironment, **I**nteractions and **O**rganization. Multi-agent systems enable to support the three distribution aspects of an industrial system :

physical distribution The autonomy and the interaction capacities of agents allows to carry out at the same time the distribution within a data-processing network, and the distribution of the various industrial system parts by associating each agent with the one of them;

informational distribution The cognitive and interactional agent capacities make it possible to distribute information;

decisional distribution The cognitive mechanisms composing the agents permits to set up the decision-making processes.

Moreover the modularity generated by the use of multi-agent systems allows us to answer another crucial point : the reuse of knowledge and already installed tools in industrial companies.

These various points led us to propose the methodological approach $\mathcal{M}_{\mathcal{A}}\mathcal{MA}-\mathcal{S}$ whose realization, illustrated by the figure figure 1, is iterative [GGBC99]. This approach permits to make gradually evolve our methodology and our simulation environment. We present the life cycle of the $\mathcal{M}_{\mathcal{A}}\mathcal{MA}-\mathcal{S}$ development in the following section.

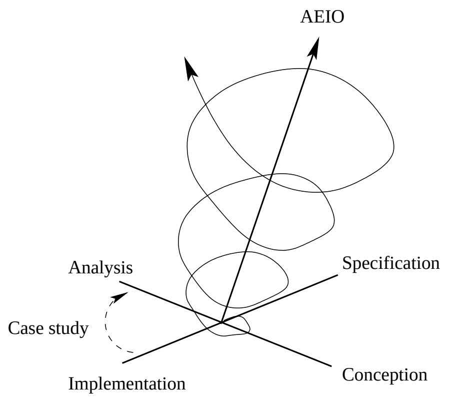


Figure 1: Problem solving evolution

3 Multi-Agent Methodological Approach

In this section, we expose the *first results* of our work on $\mathcal{M}_{\mathcal{A}}\mathcal{MA}-\mathcal{S}$: the major phases of the methodology. This one could be split into two sets of stages : the modeling and the simulation. We conclude this section with a summary of the

$\mathcal{M}_A\mathcal{MA}$–$\mathcal{S}$'s cycle of life, with some words on the distributed development and finally with a brief section on an application.

3.1 Modeling stages

One of the simulation framework central point is the distinction between modeling and simulation phases. Is this subsection only modeling stages *i.e.*, analysis and specification, are exposed.

3.1.1 Analysis

The analysis or requirement definition is the process aiming to establish which functionalities the system must provide and which non-functional constraints it must satisfy.

It is necessary to specify the level of detail to which the needs must be expressed. A problem definition must be comprehensible and used to design and carry out simulation models.

It is important to make the difference between goals and needs *e.g.*, "easy-to-use system" (non-measurable property) is a goal whereas "all user commands are selectable with popup menus" (verifiable property) is a need associated to a goal.

The analysis phase must produce a document called *requirement specification.* According to [Som98], it must satisfy the six following criteria:

- specify only the external system behavior,
- specify the realization constraints,
- easy to update,
- use as reference to the maintenance programmers,
- contain the indications concerning the later stages of the life cycle,
- specify the acceptable answers to the undesirable events.

We propose a structure for the requirement specification, which is based on the proposal of [Som98] :

1. Introduction
   This part presents the reason for being of the simulation model, its context, a short definition of the awaited functions, a presentation of the requirement specification structure and notations;
2. A physical flow description
   This part describes the whole of the physical structures, which are in the simulation model. We especially find :
   - local and distant production-line descriptions,
   - operating times,
   - breakdown durations,
   - tool-change durations,
   - used resources (single or multiple),
   - management rules;
3. An informational flow description
   This part presents informational structures of the simulation model. Various informations and types of information could be describe :
   - informal product description (manufactured or not),
   - routing and nomenclature informations,
   - decisional flow description *i.e.*, the whole of decision categories, which exist into the system;
4. A decisional system description
   This part contains the whole of information necessary to the comprehension of the simulated system oganisation *e.g.*, the organisational structures and the decision-making rules;
5. Experimental plan description;
6. Glossary and index.

3.1.2 Specification

According to the French Association of Standardization (AFNOR 1989), a specification is "a whole of activities, which define in a precise, complete and coherent way the user needs". We can affirm that the specification is a phase of the simulation model life cycle and corresponds to the definition of the essential characteristics that must have a simulated system. A specification is the requirement specification formalization and allows to validate the coherence of the brought solution with respect to the expressed needs.

Within the framework of our methodological approach, we propose a formal definition of the specification progress, which is illustrated

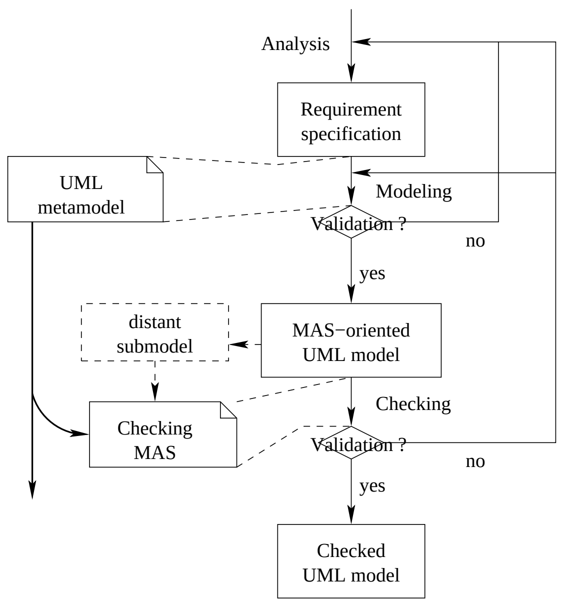


Figure 2: Specification phase

by figure figure 2 on following page. Two concepts are used during this specification phase : multi-agent systems and the UML metamodels [Mul97].

We use a multi-agent system to check the coherence of model with respect to distant model definition *e.g.*, a resource, which is defined in the model M_2, must be used by the model M_1 according to its definition in M_2. But the whole of constraint attach to distributed modeling elements does not allow to carry out a complete checking by the multi-agent system *during* the conceptual model construction *e.g.*, if a modeling componant, which corresponds to a distributed resource R, is "put" in the simulation model before the machine M, which uses it, the multi-agent system is incompetent to carry out the correlation between the resource use and its definition. To solve this problem, let us split the specification phase into two successive stages : the modeling and the checking.

The modeling is based on the UML metamodel use. It defines construction rules for the conceptual simulation model. Thus the UML mechanisms enables to create models, which respect a static preset formalism. Thus developers are able to put modeling elements in an unspecified order. They use the construction rules enacted in the UML metamodel. Then we obtain a static conceptual model called *MAS-oriented* UML *model* or MaUm.

To illustrate the role of the UML metamodel, we define a toy model showed in figure figure 3 : a processing unit could use a set of resources and a resource must be used by at least one processing unit. The metamodel represents statical modeling constraints. In our example, a resource element could not exist in a simulation model if it is not linked to a processing unit modeling element.

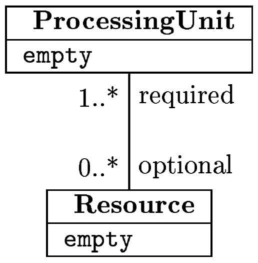


Figure 3: UML metamodel Example

But a UML metamodel is not sufficient to entirely checkout a simulation model. Because we allow the use of distributed modeling elements, it is necessary to verify it according to the distant definitions. This checking is carried out by a specific multi-agent system. Checking agents have the role to contact remote agents and, by the way of conversations, verify using correctnesses of modeling elements. But there agents are only definition customers. They does not provide any information about their associated modeling elements. Then we introduce information source agents, which are associated to distributable modeling elements. We suppose that the description providers are allways woke up. The problem of information disposal *i.e.*, existence of agents, is partly solve in the section 3.4.

The result of this phase is a model called *Checked* UML *model* or CdUm, which respects at the same time the formalism enacted by the UML metamodel and well-formed communications with the remote modeling elements (checking by the specific multi-agent system).

3.2 Simulation stages

The second part of our methodological approach is composed by simulation stages : conception, implementation, experiments and result validation. The last two steps are not approach is this paper. We concentrate our explanations on the conception of a multi-agent system dedicated to the simulation and on its implementation on simulation softwares.

3.2.1 Conception

The figure figure 4 illustrates the *conception* phase progress. From the CdUm, the development team works on the generation of a equivalent multi-agent model called *Instanciable Simulation Multi-Agent System* or iSMaS. This phase is partly automated by the use of a whole of translation rules from conceptual modeling elements to agent societies. Developers can modify this set of rules in order to enter in adequacy with the simulation problem solving. The resulting model is multi-agent platform-independent.

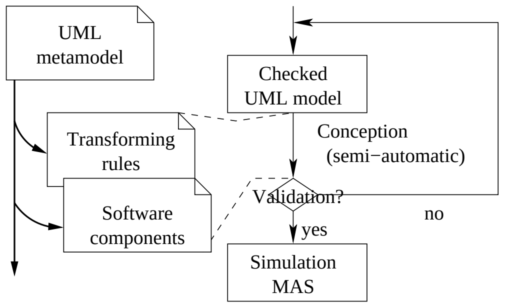


Figure 4: Conception stage

To illustrate the transformation rules, lets us considere the following BNF-like grammar :

```
rule ::= "AGENT" [ agentname  ]
         '←' description ';' |empty

description ::=
    'A' '{' pseudo-code '}' '∧' 'E' '{' pseudo-code '}' '∧' 'I' '{' pseudo-code '}' '∧' 'O' '{' pseudo-code '}' '

agentname ::= string
```

It defines the syntax of a agent generation rule. We considere too that agent are composed by four modules : **A**gent, **E**nvironment, **I**nteraction and **O**rganization [BD94]. Each of them manages an AEIO axis [Dem95]. In addition, we include another module called **U**ser definition. It permits to introduce some specific features into the agent definition (variables, functions, *etc.*).

We should express the transformation rule of an Uml element Resource as follow :

$$
\begin{array}{l}
\texttt{AGENT} \\
\texttt{["Resource\_" + Resource.name]} \leftarrow \{ \\
\texttt{A} \left\{ \begin{array}{l} \texttt{Definition of :} \\ \texttt{the current resource user} \\ \texttt{a list of awaiting agents} \end{array} \right\} \wedge \\
\texttt{E} \, \{ \, \} \\
\texttt{I} \left\{ \begin{array}{l} \texttt{seized} \xrightarrow{\text{want release}} \texttt{released} \\ \texttt{seized} \xrightarrow{\text{want seize}} \texttt{wait \& seized} \\ \texttt{released} \xrightarrow{\text{want seize}} \texttt{seized} \end{array} \right\} \wedge \\
\texttt{O} \, \{ \, \} \\
\texttt{U} \left\{ \begin{array}{l} \texttt{Definition of methods} \\ \texttt{to seize, release and} \\ \texttt{waiting for resource} \end{array} \right\} \wedge \\
\}
\end{array}
$$

The formal definition of the BNF grammar, and his semantic, is currently in progress. It will permit to write rules for generating a multi-agent system, which will be independant of any multi-agent platform and simulation tool.

3.2.2 Implementation

We considere the implementation as an entirely automated phase. Indeed, it is possible to pass from the iSMaS to a particular implementation on a multi-agent platform as SWARM [Bur94], ARéVi [DMR+97] or MAST [BBP+98]. In addition, implementing rules can take into account the use of commercial simulation software as Arena® or Simple++®. Currently, there rules are the result of a pragmatic approach. As transformation rules in the conception phase, they does not have a formalized form.

We can illustrate the behaviour of the implementing rules by the following informal example :

$\texttt{AGENT["Resource\_"*]} \xrightarrow{\text{Arena}} \{$
```
model = getCurrentModel("Arena")
if ∄ model.ressources then
model.add("ressources")
model.ressources.add($name)
}
```
$\texttt{AGENT["ProcessingUnit\_"*]} \xrightarrow{\text{Arena}} \{$
```
model = getCurrentModel("Arena")
if $resourcecount > 0 then
model.add("seize","delay","release")
else model.add("delay") ...
}
```

The first rule create, for each conceptual resource agent ("Resource_"*), a resource declaration in a ARENA® model. The second permit to generate a processing unit in the same ARENA® model. Identifiers, which begin with $, represent properties of the current transformed agent. Applications of there kind of rules permit to generate, from the simulation multi-agent system (ISMAS), a runnable multi-agent system for the production system simulation (RSMAS).

3.3 Life cycle

In this section, we define the life cycle used by MAMA–S. The figure figure 8 on page 10 illustrates the simulation model life cycle. It is closed to the "water-fall" definition [Som98] and is especially based the Conical Methodology [Nan81]. In addition to analysis, specification, conception and implementation phases defined is previous sections, we bring simulation characteristics : experiment plan design, experiments and simulation result validation. We focus our work on the first four phases. Indeed, we think that our contribution, based on the multi-agent systems and UML, appears during these phases. The Petri net formalism, used in the figure figure 8 on page 10, permits to represent the behavior of our methodological approach. By associating each mark to a development team activity, we can model distributed developments of simulation models. Our life cycle allows returns to previous stages in case of validation failures (transitions t_{12}, t_{13}, t_{14}, t_{14}, and t_{15}).

Until this point we did not clearly introduce relation-ships between models in the cycle of life. The figure figure 5 on following page presents dependences of the various produced documents with respect to multi-agent system and UML concepts. Our approach is based on the use of a UML metamodel. It defines the *modeling elements*, which are specific to production systems, as well as a *formalism* [BJR+97]. This metamodel is used and respected by all the componants of our methodology. MAMA–S produces a set of documents specific to each life cycle phase. A multi-agent oriented UML model (MAUM) is built during the *modeling*. It is the formalized translation of informations contained in the requirement specification. It uses the modeling elements and the formalism defined in the UML metamodel. The main characteristic of the producted UML model is the respect of the structural constraints imposed by the metamodel[2] *i.e.*, the using coherence of distributed modeling elements cannot be carried out directly. To resolve these problems, we define a specific multi-agent system. During the *checking* phase, this system checks the MAUM. If no anomaly or error was detected, the checking system produces a checked UML model (CDUM). This last is not an instance of a multi-agent model. It is made up only of conceptual modeling elements.

The semi-automatic generation of the multi-agent simulation model (ISMAS) is carried out by the application of translation rules and the use of preset software components (cooperation protocols, production facility agents, *etc.*). This *conception* permits to completely define a multi-agent model dedicated to simulation and independent of any platform and tool. User software components correspond to definitions of modeling elements and are defined by development teams *e.g.*, decision-making processes could be modeled with this kind of componants.

The last phase of the model build is the *implementation*. The runnable simulation multi-agent system (RSMAS) is made up from the ISMAS and with a whole of specific agent platforms and simulation tools *e.g.*, platform MAST and tool ARENA®. The section 3.5 presents a brief example of a multi-agent system in combination with ARENA®.

The dedicated editor has a particular status. It allows to create and publish the various models of the MAMA–S lige cycle. For that, it exposes the formalism contained in the UML metamodel and permits to call upon the other methodology components.

3.4 Distributed development

In this section, we introduce distributed model development. We assume that a model development should begin at anytime. The main problem is the disposal of a model description when it is needed by another one. We suggest a simple approach that consists in create model development processes when their needed are detected. The figure figure 6 on following page illustrates an example. During the A's analysis phase, we found a dependency with a nonexistent model named B. A new model development begins and evolves parallely to A's one. But the needed dectection is not limited to analysis phase. It could

[2]UML model checking is defined in OMG specifications and implemented in many tools as Rational Rose

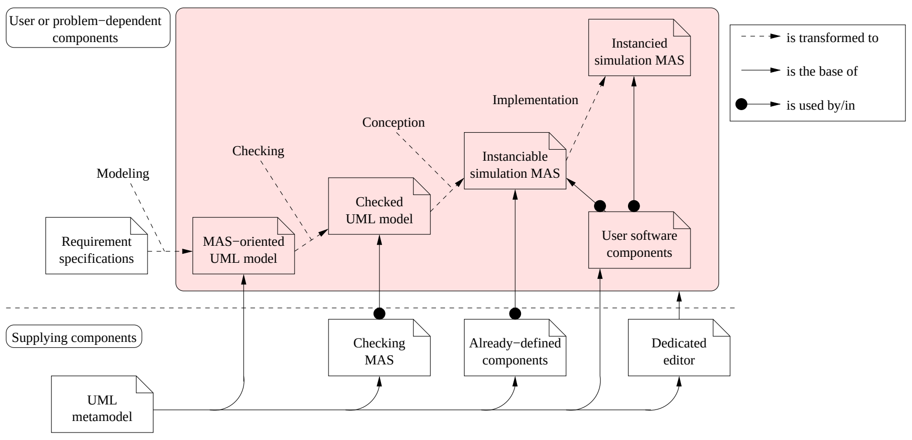


Figure 5: $\mathcal{M_AMA}$–$\mathcal{S}$ life cycle dependences

be reached during the other methodology steps. Informations placed at the disposal of the distant models, have various types and are function of phases:

- an "remote" analysis must shared modeling element descriptions by checking-agent intermediary;
- the "remote" conception must place simulation-agent descriptions at the disposal;
- the implementation phases of the various distributed models are completely independent. Only one constraint is necessary : all runnable simulation models must be finished before the experiment phase.

There are another problem : does we go to next methodology stage without waiting for submodel results ? It is difficult to check and validate models if submodel results are unknow. To solve this problem, development team must wait for distant submodel results. In our example, the A's conception end is reached only when B's conception was finished.

We will refined the distributed development behaviour to make our methodology much more user-friendly and to highlight links with other multi-agent and simulation methodologies as HLA.

3.5 Instanciation example

Within the iterative development framework of our methodological approach [GGBC99], we designed a simple multi-agent model. It is composed of agents, which are intermediaries between various simulation models, as show in figure figure 7 on following page. There last are built in a specific simulation software (Arena®). This multi-agent model permits to build simulation models, which take into account physical and decisional flow distributions. However we limit the modeling capabilities of this multi-agent model to flowshop cases *i.e.*, neither cycle nor flow junction.

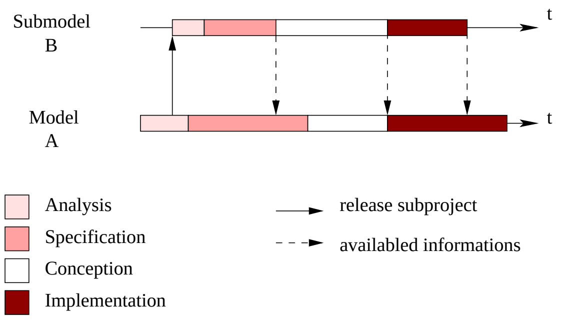


Figure 6: Example of distributed modeling with $\mathcal{M_AMA}$–$\mathcal{S}$

Within a validation framework, we apply this first model on a teaching case. Learners deals with a physical or decisional parts of the simulation model. Each learner group manages his part independently and parallely to other groups. This application highlights local decision-making problems within production systems and allows to create interesting teaching situations.

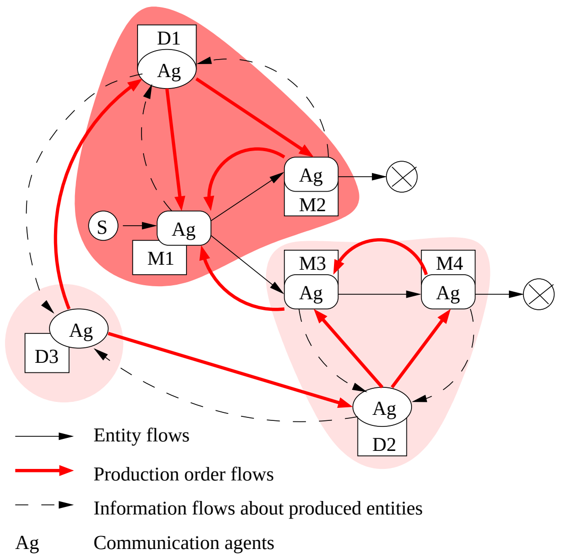


Figure 7: Instanciation example

The figure figure 10 on page 11 is a ARENA® modeling example of the processing unit M_1. It is composed by a generator of production order (`GenOF`), a production unit model (`M1`), a branching choice (`Choix`) and two modules of entity sending (`vers_M2` and `vers_M3`).

The sending-to-M2 submodel (`vers_M2`) is illustrated by the figure figure 9 on page 10. His main modeling element is a VBA component. It permits to construct a message from entities and send it to the M_2 model. This work is doing by ACTIVEX® controls, which are reactive agents. ARENA® models call an agent's sending method for each of there entities. In the other side, another reactive agents wait for messages. They generate VISUAL BASIC® events into attached ARENA® models.

Because we would study interaction capabilities between agents and ARENA® models, we introduce some problem simplification :

- we use a simple pessimistic synchronization between agents. It is simple because agent ignore messages, which have a too-old stamp-date;
- we modelize only flowshops. Then production systems with junctions or loop back are not supported;
- we does not made assumption about ARENA® model contents.

It is not the purpose of this paper to completely present this first multi-agent model and its application. They are developed in [GGC00].

4 Conclusion and perspectives

Simulation is recognized as a tool adapted to the study of industrial system behaviors. But even if this technic takes into account dynamic aspects, it seldom allows physical, informational and decisional distributions. In addition, modern tools are rarely accompanied by adapted methodologies. With this observation, we had proposed in [GGBC99] a methodological approach for the simulation based on the multi-agent concepts : $\mathcal{M_A MA}$–$\mathcal{S}$. An iterative development of this methodology was adopted *i.e.*, the various functionalities evolve gradually during our work.

In this article we present the first results on the development of our approach. We think that a methodology integrating physical, informational and decisional distributions must be itself distributed *i.e.*, the various submodels necessary to a simulation model could be developed in parallel. Once to expose our point of view on this problem, we define the simulation model life cycle. It is made up primarily by the traditional phases of analysis, specification, conception and implementation. We add to it the three phases of experimental plan design, expriments and simulation result validation, which are more specific to simulation problems. We consider that our contribution is in the level of the four first phases.

In the future, we will defined the various concepts used and usable during the analysis (requirement specification), the specification (UML metamodel, checking multi-agent system), the conception (multi-agent simulation model) and the implementation (multi-agent platforms). We will checked our theories on industrial and teaching case applications. And we will highlighted connections between our approach and other simulation methodologies as HLA. We find that your current works are enought abstract to permit this.

References

[BBP$^+$98] Olivier Boissier, Philippe Beaune, Hubert Proton, Mahdi Hannoun, Thibaut Carron, Laurent Vercouter, and Claudette Sayettat. The multi-agent system toolkit. Technical report, SIC/ENSM-SE, 1998.

[BD94] Olivier Boissier and Yves Demazeau. An architecture for social and individual control and its application to computer vision. In *MAAMAW*, Odense, Danemark, August 1994.

[BJR$^+$97] Grady Booch, Ivar Jacobson, Jim Rumbaugh, et al. Unified modeling language specifications - version 1.1. Technical report, UML consortium - Object Management Group, 1997.

[Bur94] Roger Burkhart. The swarm multi-agent simulation system. In *OOPSLA Workshop on "The Object Engine"*, 1994.

[Bur96] Patrick Burlat. *Contribution à l'Évaluation Économique des Organisations Productives : vers une modélisation de l'entreprise-compétences.* PhD thesis, Université Lyon 2, January 1996.

[Dem95] Yves Demazeau. From interactions to collective behaviour in agent-based systems. In *European conference on cognitive science*, Saint-Malo, France, April 1995.

[DMR$^+$97] T. Duval, S. Morvan, P. Reignier, F. Harrouet, and J. Tisseau. ARéVi : Une boîte à outils 3d pour des applications coopératives. *"La coopération"*, 9(2):239–250, June 1997.

[GGBC99] Stéphane Galland, Frédéric Grimaud, Philippe Beaune, and Jean-Pierre Campagne. Multi-agent methodological approach for distributed simulation. In Graham Horton, Dietmar Möller, and Ulrich Rüde, editors, *Simulation in Industry - 11th European Simulation Symposium*, pages 104–108, Erlangen - Germany, October 1999. Society for Computer Simulation.

[GGC00] Stéphane Galland, Frédéric Grimaud, and Jean-Pierre Campagne. Multi-agent architecture for distributed simulation : Teaching application for industrial management. In Rik Van Landeghem, editor, *Simulation and Modelling : Enablers for a better quality of life – 14th European Simulation Multiconference*, pages 756–762, Ghent, Belgium, May 2000. Society for Computer Simulation.

[Mul97] Pierre-Alain Muller. *Modélisation objet avec UML.* Eyrolles, 1997.

[Nan81] R.E. Nance. Model representation in discrete event simulation: The conical methodology. Technical Report CS-81003-R, Department of Computer Science, Virginia Polytechnic Institute and State University, Blackburg, USA, 1981.

[Som98] I. Sommerville. *Le Génie Logiciel et ses applications.* InterEditions, 1998.

[US 96] US Department of Defense. High level architecture federation development and execution process (fedep) model, version 1.0. Technical report, Defense Modeling and simulation Office, September 1996.

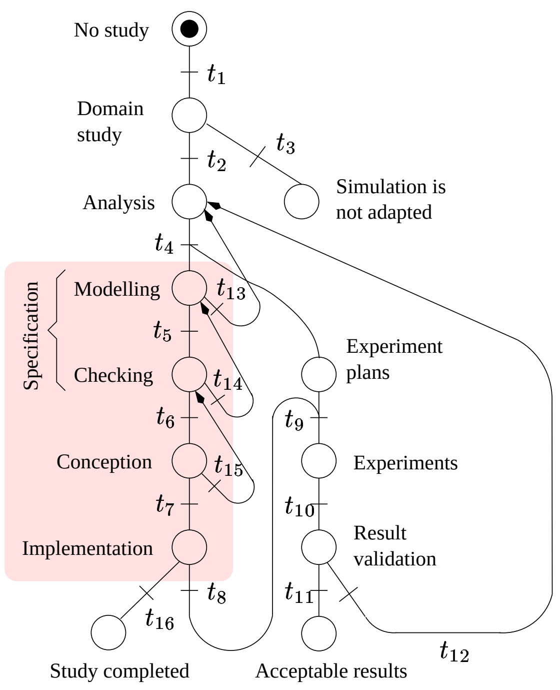


Figure 8: $\mathcal{MAMA}$–$\mathcal{S}$ life cycle course

t_1	Problem is highlighted
t_2	Simulation is selected
t_3	Simulation is unadapted
t_4	Requirement specification is written
t_5	MAS-oriented UML model is built
t_6	UML model is checked
t_7	Instanciable MAS is ready
t_8	Experiments needed
t_9	Ready to experiment
t_{10}	Experiments are realized
t_{11}	Result validation is reached
t_{12}	Requirement specification changes are needed
t_{13}, t_{14}, t_{15}	Validation failure is happened
t_{16}	Runnable system ready

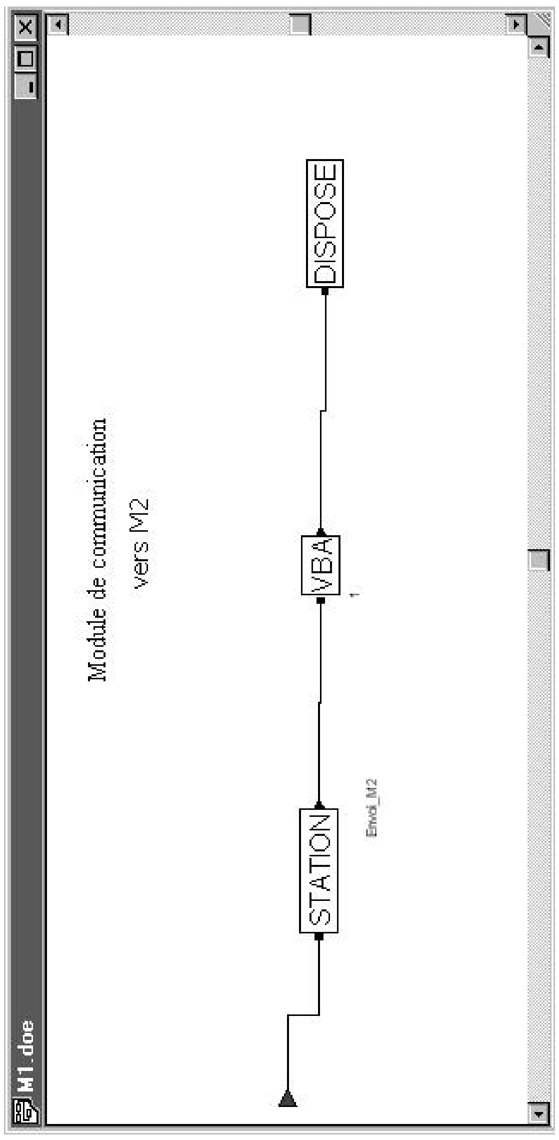
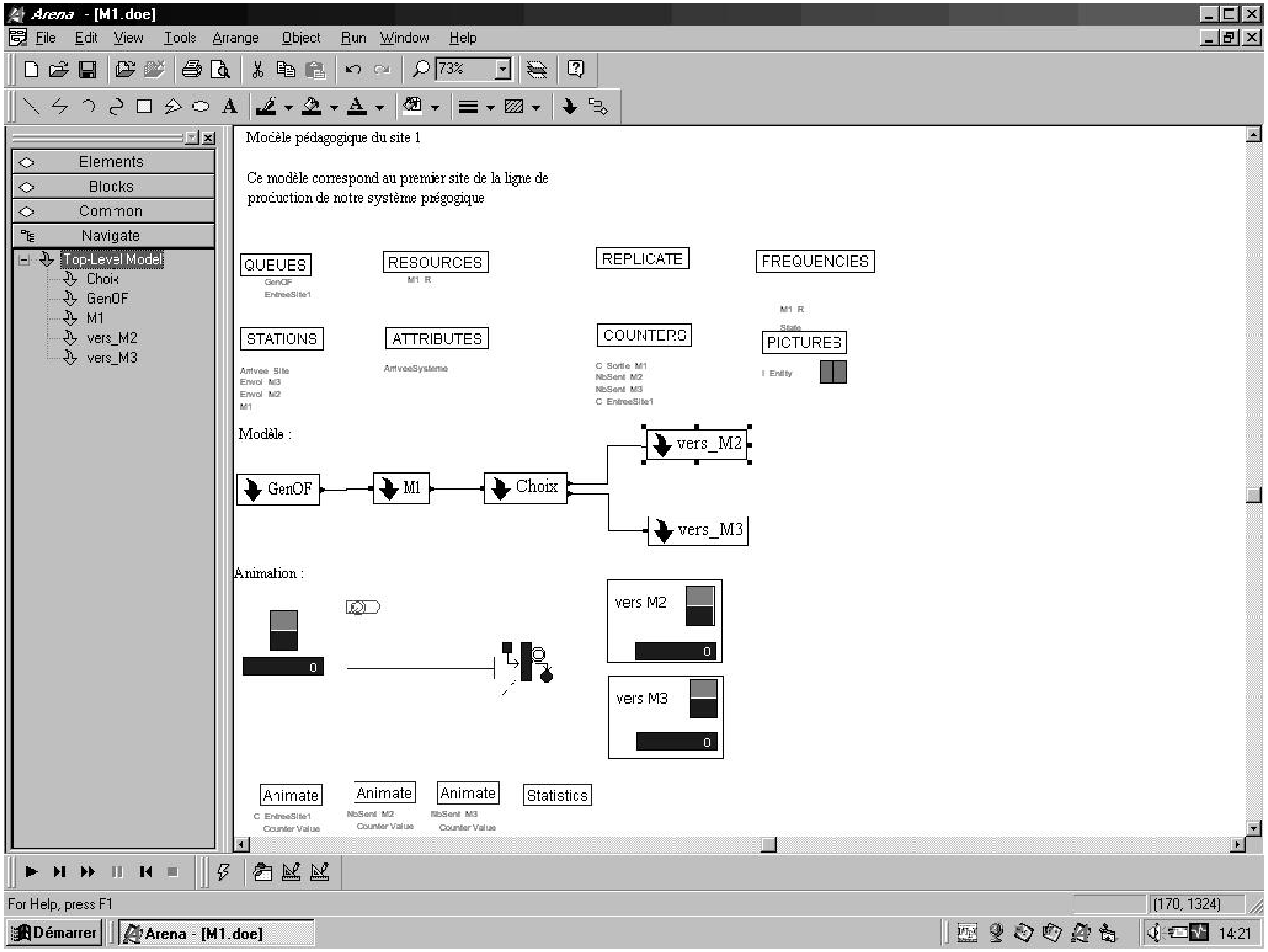


Figure 9: M_1 to M_2 entity sending



Modèle pédagogique du site 1

Ce modèle correspond au premier site de la ligne de production de notre système prégogique

Modèle :

Animation :

Figure 10: Arena® model of M_1