Virtual Reality and Multi-Agent Systems for Manufacturing System Interactive Prototyping*

Pierre Chevaillier, Fabrice Harrouet, Patrick Reignier, Jacques Tisseau

Laboratoire d’Informatique Industrielle
Ecole Nationale d’Ingénieurs de Brest, France

Abstract

This paper introduces a virtual reality platform based on a dynamic multi-agent programming language. These tools have been designed to show that simulating a multi-agent system in a virtual environment with dynamic properties can be used for interactive prototyping.

This kind of prototyping has to be considered when the designed system cannot be described as a whole but as a set of autonomous components with many interactions. Due to the fact that these interactions are very complex to model before simulating the system, we propose to let the designer enter inside the system and dynamically build, tune and mend the model.

The case of the flexible manufacturing system entirely corresponds to this approach. Actually, such a system can be described has a set of different models (from physical to behavioral) in interaction.

1 Introduction

Flexible Manufacturing System (FMS) design requires to take into account the behavior of different kinds of element: physical devices, controllers and decisional actors. This implies to use jointly different types of model which are set up by various experts, working together in a collaborative framework. Prototyping is more and more often used, especially when concurrent engineering is carried out, to reduce design cost and time-to-market. It needs specific tools to ensure rapid development, consistency and reliability of design and global life-cycle cover of systems. These tools have to be versatile, sizable, to introduce as less artifacts as possible and to enforce interactive and cooperative work. Virtual reality techniques seem to be very powerful in this context.

The aim of this paper is to point out the fact that a multi-agent approach in a virtual reality environment, as performed with our oRis/ARéVi platform, can be considered as a tool for dynamic/interactive prototyping.

The classical spiral, very often used to build a prototype of a system, consists in the following loop: 1) to generate the prototype, 2) to test the prototype, 3) to adjust the prototype according to the test’s results, 4) to go to step 2 while needed.

This approach is quite efficient in many cases where the described system can be seen as a whole corresponding to one global model. However, when the system becomes complex (i.e. made up of multiple parts in interaction), this global model is much harder to design and to tune, due to the fact that every possible interaction cannot be listed in an exhaustive way. According to this remark, our main goal is to introduce a tool which allows to handle intrinsically distributed systems and to dynamically and interactively model their behavior.

If the simulated system is complex, then its global behavior cannot be fully described. Therefore, it would be easier to give its components the appropriate behavior when peculiar situations happen to them. This can be done only when the simulation of the system is running because we are not able to know in advance every possible configuration of the system. Thus, the prototyping loop can disappear and the user and the designer of the system have to be present inside the simulated system. The use of a tool allowing to perform this kind of prototyping should consist in:

- describing an initial state of the system to work on as a set of components with their own behaviors,
- letting the tool simulate the system’s global behavior which results from interactions between components,
- allowing the user to modify or mend behaviors and structures during run-time.

In section 2, we will describe what we exactly define as interactive prototyping. The agent-oriented approach we use to achieve this goal will be presented section 3. The platform we use to perform interactive prototyping will be shown section 4. Then, the prototyping of flexible manufacturing systems will be studied section 5 and illustrated by a practical example section 6.
2 Interactive prototyping and virtual reality

Virtual reality is a term which is very commonly used, in many contexts and sometimes with different meanings. We will here find out the main meaning of this term that suits to interactive prototyping. This will be done considering first the animation meaning, then the simulation aspect, and finally the interactive point of view.

2.1 3D animation

Virtual reality is usually seen as a graphic animation in a 3D world. In this case, people only have to watch the animation but can neither do anything else nor interact with it. Sometimes, the user can travel in the virtual world, choosing its own way to look at different parts of the system under many points of view. In this case, the animation is qualified as “walk through animation”.

The quality of such a tool can be resumed to the quality of the rendering (3D images on large screens or head-mounted displays, 3D sounds . . . ). The user can only travel and look around. Therefore, this is not interactive enough to be run for interactive prototyping. Nevertheless, this kind of tool can be useful for architectural and artistic designing [1, 2].

2.2 Interactive simulation

Virtual reality can also be seen as a tool for interactive simulation. It means that in the 3D world, the user will not only be able to move but also to interact with the components of the represented system according to predefined parameters. Thus, in this situation, the only way to act on the animation, is to change the values of these parameters and nothing else.

The designer of such an animation has to think about which parameters will be accessible to the user. Then, he must give the user a way to control them. A very common way to control these parameters is to use a classical graphic user interface (control panels made up of buttons, switches, sliders . . . ), but in 3D worlds we often use specialized devices to control geometric parameters (joysticks, 3D mice, data gloves, trackers, force feedback actuators . . . ). The more accurate and easy to use these devices are, the better the quality of this tool is. This kind of animation can be useful to install premises [3], to train people to work in a specific environment (SOFI project [1]), on robotics [4], or to work in tele-operation [5]. However, the only changes to be allowed must be predefined by the designer before the user acts, and consequently, they are very often structural (as moving objects).

2.3 Interactive prototyping

We will now get rid of the main limitation of the above kind of tool: a restricted expression of changes, due to the fact that the designer should think about every possible change the user could try. Since prototyping represents the action of designing, building the model of the system, the previous tools cannot be run for this. They can be used to test the prototype in several predefined situations, but they do not allow to build such a prototype. The designer builds the system, the user tests it and the designer re-builds the system until the user is satisfied of his tests’ results. This approach can be described as “prototyping”
but cannot be qualified as “interactive” (it corresponds to the prototyping spiral seen section 1).

Then, if we want to have an interactive prototyping process, we must give the user the opportunity to substantially change the behavior of the system’s components whenever and however he wants. This can be done in order to try many behaviors against each other, to simulate defaults or mistakes in a process, to watch how the system reacts to an unknown situation, to simulate the reconfiguration of an industrial process, and many operations which cannot be done on a pre-established simulation.

Therefore, if we want to give the user the full control of his experiments on the system, he has to be able to use the same expressions and ways to act than those used by the components in the system. This means that if a language is used by the designer to describe the system’s components and their behavior, the user has to use the same language during the experiment. In this case, more than testing the component’s behavior, he will be able to tune or to mend them in situation, while running them.

Thus, we can say that the user is in “immersion through the language” so that there is no longer any limit between him and the designer of the simulated system. This is the main meaning we give to interactive prototyping, and that’s the way we consider virtual reality too (as much as being in sensitive immersion). The three main aspects of virtual reality can be summarized as shown figure 1. Further explanations on this systemic approach of virtual reality and interactive prototyping are presented in [6].

![Figure 1: Systemic approach of virtual reality](image)

### 3 Agent paradigm and virtual reality

Since we have established that the user should be able to dynamically build the model he is working on, we will now think about the way the model can be described. The first approach to be considered is an object-oriented one. This is a natural choice concerning virtual reality due to the fact that the simulated system can be described as a set of components with their own properties.

However, these components are not only static objects but also have a strong dynamic behavior. More than just looking like real world’s objects, the virtual components must behave as real ones. This remark drives us to describe the system conforming to an agent-oriented approach which is an extension to the object paradigm [7].

The designer cannot know in advance how the system will change during the simulation. The global evolution of the system is the consequence of many interactions between its components. These ones react according to local information, so that it is very difficult
to deduce a global result from so many local data. In this case, the only way to determine
the global behavior of the represented system is to let it work.

Thus, the simulated system is a multi-agent system in which each component is an
agent endowed with its own goal and behavior. It implies that the behavioral/dynamic
aspect of the virtual components must be taken under consideration as much as their
structural/static aspect. The designer of such a system cannot model it as a whole but has
to describe each agent’s behavior using the following sequence:

▷ perception, in the virtual environment (other agents or objects),
▷ decision-making, according to its own perception, state and goal,
▷ action, in the environment and on itself.

During prototyping, the user can introduce changes and make the simulated system
go in a state which may never have been reached without its intervention. A possible
way, for the user, to act in the virtual world, is to use an “avatar” able to perform several
actions, and seen as an agent in the system. But more than that, the user must be able to
dynamically change every part of the system by rewriting their behavior when needed.

If we consider all the above requirements, we can list the properties we expect to find
in a tool allowing interactive prototyping on multi-agent systems:

▷ an object-oriented environment allowing to consider each component as a whole,
▷ a concurrent environment to execute each active object’s activity,
▷ various communication means (peer-to-peer, broadcast . . . ) to be used by the
agents,
▷ highly dynamic properties allowing to interactively build or tune the agents’ behav-
iors.

These points (and especially the last one) drove us into the realization of a language,
oRis, which should help the interactive prototyping of multi-agent systems. Of course,
some of these points can be found in many different languages but we must ensure that
the environment keeps consistent when we use all these properties together.

This language has been made as dynamic as possible while keeping a strongly typed
approach. It does not only consist in loading shared libraries at run-time (as in C/C++
or Java), but it allows to rewrite code of existing classes or instances. The scheduler has
been designed so that concurrency is as fair as possible and does not introduce a bias in
the simulation. Further explanations on the oRis language can be found in [8].

4 The ARéVi/oRis platform

As seen above, the respect of the agent paradigm is a very important point to take under
consideration in virtual reality. This means that everything concerning the rendering must
be strongly coupled with the agent-oriented simulator to obtain a homogeneous environ-
ment. This remark motivated us to develop a specialized virtual reality platform, named
ARéVi — Atelier de Réalité Virtuelle — [9], which is more focused on the behavior of
its components than on the quality of the rendering. Obviously, the rendering is very im-
portant for virtual reality, but our platform embeds what has already been developed in
specialized libraries (Open Inventor [10], OpenGL Optimizer [11]).
In ARéVi, we use an agent-oriented approach for every object used. Of course, the simulated system can be described as a multi-agent system, but more than that, the tool itself can be seen as set of agents, each of them having a specific task to perform. For example, a joystick can be seen as an agent in charge of giving instructions to a 3D entity, a viewer can change on its own the quality of its graphic rendering (no texture, wire frame ...) according to the frame rate, the graphical user interface can change when facing particular events in the simulation, and many other possible behaviors can be given to these tools so that the virtual reality platform can be adapted to each particular simulation planed. All these agents’ behaviors are described using the oRis language, which is embedded in ARéVi, to allow the dynamic changes implied by interactive prototyping. Actually, since it is very easy with oRis to add, remove agents and to change their behavior while running, it seems to be very interesting to use this property on the tool itself (to try one device instead of another or to compare different rendering methods ...).

Because ARéVi relies on specific libraries, the quality or the speed of the rendering are equivalent to those provided by these latter. The supported sensors and actuators are those supported by these libraries. However, it is not always possible to mix different libraries in the same application. To go beyond in the rendering, these libraries should give us the access to the low level informations they use (to support deformable shapes for example). Since this platform is used for prototyping, the speed of the execution is not a critical point, then, the fact that the language is entirely dynamic (thus, not very fast) is not an important limitation. Temporal consistency is not directly ensured by the platform, although we have the full control of the scheduling policy, but a timer agent corresponding to the specific application can be introduced.

5 Flexible Manufacturing System prototyping

Flexible Manufacturing System (FMS) design needs to take into account a large variety of models corresponding to the different subsystems.

- physical model of the operative part elements,
- formal model of the control,
- decision-making models for planning, scheduling and resource allocation rules.

These different kind of models must be introduced in the virtual prototype because there are strong interactions between subsystems. So, the performance of the system has to be validated as a whole.

It appears that the physical modeling (which could be performed by large with CAD tools) is only one aspect of the requirements of FMS prototyping. There are strong needs concerning the behavioral aspects of the system. CAD tools have too strong limitations in this field and virtual reality techniques are required. One important point is that the different models may not be introduced at once but following an incremental and interactive process.
The agent-oriented approach seems to be efficient in this frame. It allows to express
the different kind of expertise.

▷ Reactive agents are used to model the behavior of the active elements of the opera-
tive subsystem (sensors and actuators) and the reactive elements of the control.
▷ Cognitive agents are used to model the decision-making processes.
▷ Adaptive agents are used to simulate learning processes or to assist avatars.

This approach is also suitable because it enforces modularity and robustness of the
prototype, which reduces dramatically design costs. The prototype’s architecture is char-
acterized by aggregates of agents slightly coupled between them. Thus, this archite-
cture allows to perform an incremental design of the prototype, the global functioning of
which is obtained through the interactive and incremental programming of the interac-
tions between agents. Each element of the FMS are supposed to be autonomous since
they are able to execute a set of actions when receiving external requests. For example,
a computer-operated arm carries out its duty as soon as it perceives a piece and only if
its internal rules allow it. Each element is compounded of sensor and actuator agents,
reactive controller agents and, when needed, decisional agents.

The next section presents some decision elements concerning the operative part mod-
eling; the next one explains how controllers have been implemented.

5.1 Behavioral model of operative part elements

The first step consists in modeling the behavior of the operative part which corresponds
to each element of the flexible manufacturing system. The model provides a geometrical
representation of a physical element designed to carry out one function. This function
can be compounded of several tasks which may be programmed. The simulation of this
function is the element’s behavioral model. The exploitation of the model is directly
dependent from the different factors which influence the behavior of the entities integrated
in it. The user/designer has to work out the following characteristics:

▷ Which components must be represented by evaluating their contribution to both the
flexible manufacturing system and the sensorial and behavioral rendering.
▷ Which are the physical phenomena to reproduce: mechanical links, collisions, grav-
ity, magnetic field . . .
▷ Which sequences can be computed in advance: the transformation of a piece on
a machine can be done by a sequence of animation, or during the progression by
boolean operations between solids or by extrusion of a primitive shape.
▷ Whether it is necessary to model the functioning of an actuator or the result of its
action on the physical system.
▷ The discretization of the kinematics.
▷ The autonomy of action given to the element’s components. For example, a detec-
tion cell is modeled as an autonomous entity (an agent) which permanently scans
its environment in order to detect some objects within its field of perception.
▷ Which sensors and which actuators each element has to be composed of to be driven
by the control system.

These choices are not irrevocable and at any time they can be challenged interactively
by the designer. The solution does not have to be uniform. For example, the gravity can
be taken into account only for the free transfer between equipments and can be ignored the rest of time. The ability to modify in-line the model itself strongly reduces the development costs and allows to focus the design on the system’s behavioral problems towards its control.

As soon as all the elements have been modeled individually, they are placed into the same scene to build the FMS prototype. Thus, the general behavior of the operative system is obtained. The spatial design is either carried out by placing the equipments according to a planned setting up diagram or by solving a constraint system on the relative positions and eventually on some equipments’ dimensions. Once the elements are in position, their movements are interactively parameterized. For example, the course of a sliding element is set up according to external constraints. This setting up may require to adjust the sensors’ attributes. The collision spaces as well as the actions required during the collisions are also defined.

At this stage, the flexible manufacturing system behavior can be observed by interacting directly on its elements through different ways: panel of control, peripherals or oRis language. This allows the primary validation of the design.

## 5.2 FMS control modeling

Because they are often critical systems, the design of FMS control must rely on formal models. To be efficient, prototyping has to integrate the business languages of automation like Grafce, Petri nets, Statecharts . . . To be consistent, the design process of a FMS needs also validation techniques (which are not in the field of virtual reality) and to reduce the gap between models. These constraints led us to implement controllers as synchronized interpreted Petri nets using an agent-oriented algorithm [12].

Communications between operative parts and controller are implemented by peer-to-peer message exchange between agents. Synchronizations between controllers are supported by message broadcasting.

The interest of virtual reality relies on the fact that it is possible to interact dynamically on the system when it is under control. For example, it is possible to move a sensor, to disrupt it, to change its mode of functioning or to put an obstacle in the field of action of a mobile. This allows to test the robustness of the control system, in condition which eventually could not be executed in real situations (such as scenarios which can threaten the integrity of equipments or people).

## 6 Case study : KorSo

The approach proposed in this paper has been applied to the case study of the production cell defined in the German KorSo project — Korrecte Software — [13]. It is a quite simple system which is nevertheless interesting due to the variety of equipments and synchronizations involved (figure 2).

The first step consisted in the design of the physical properties of the different equipments making up the production cell.

Then, the intrinsic behavior of each equipment has been tested. At this stage, event
occurrences were produced by keyboard events. For complex simulations, a Simulator agent has been implemented. Its role was to simulate the delivery of external messages. Since the behavior of each equipment seemed pertinent, the element could be integrated into the whole system.

The last step was to validate the global behavior of the production cell. Both stand-alone simulations and scenarios involving unpredictable user’s actions were performed. The system’s response to these different scenarios was then studied. For example, to estimate the robustness of the design, new blanks were introduced during the simulation, or the location of some sensors was changed. Since oRis is a dynamically interpreted language, these actions were not necessary foreseen.

Interactive prototyping has been performed to deal with two kinds of problem: the configuration of the production cell and the tuning of controllers. For example, it has been decided to add a second press. To do that, a new instance of KorsoPress has been instantiated. Its position has been defined by the user so that the robot could reach it. This operation has been performed without disturbing the running simulation. The next step consisted in changing the robot controller: the new controller had to manage two distinct loading/deposit positions. The controller of the press and the other elements were not affected by this change.

A very useful feature of ARéVi was the ability to instantiate dynamically new cameras to observe the scene during prototyping or to dynamically link a peripheral to an object. For example, it was possible, at any time, to connect a 3D-mouse object to the agent responsible for the movement of one arm of the robot. It allowed to test the behavior of the control in unexpected configurations.

7 Conclusion

Our objective was to show that a multi-agent approach in a virtual reality environment could be performed for interactive prototyping. The main point of interactive prototyping concerns the use of dynamic modeling capabilities in a virtual environment designed as a multi-agent system.

As far as we are concerned, the oRis/ARéVi platform we provide seems to be an
ideal tool to interactively design and tune a distributed system. The complex behavior of such a system can be described by many agents whose interactions ensure a consistent representation of the global virtual world. The dynamic properties of the oRis language allow the user to decide what becomes the model he is working on.

These characteristics are very interesting for flexible manufacturing system prototyping. Due to the fact that the use of several model is allowed in the same simulation, it is possible to build step by step a prototype of a flexible manufacturing system. The agent formalism suits to behavioral part (controller and decision-making), but we can easily introduce, for example, Petri net or Grafcet players if some previous study has been made with these formalisms. Concerning the operative part of the flexible manufacturing system, it could be interesting to integrate an agent-oriented approach of some physical models (mechanical models used in CAD for example).

References


