Département de génie Industriel

Ecole Polytechnique de Montréal

Towards an Object-Oriented Simulation Environment for Manufacturing Systems Analysis

Michel Fabre, Daniel Leblanc

Septembre 1992

Dépôt légal, 3e trimestre 1992
Bibliothèque nationale du Québec
Bibliothèque nationale du Canada

Pour se procurer une copie de ce document, s'adresser:

Les Éditions de l'École Polytechnique
École Polytechnique de Montréal
C.P. 6079, Succursale A
Montréal (Québec) H3C 3A7
Tél.: (514) 340-4473

Compter 0,10 $ par page et ajouter 3,00 $ pour la couverture, les frais de poste et la manutention. Régler en dollars canadiens par chèque ou mandat-poste au nom de l'École Polytechnique de Montréal.

Nous n'honorerons que les commandes accompagnées d'un paiement, sauf s'il y a eu entente préalable dans le cas d'établissements d'enseignement, de sociétés ou d'organismes canadiens.

. Abstract

Industrial decision-makers need fully integrated simulation tools, to assist them in the modeling task and in the design of experiments and output analysis tasks. The object-oriented paradigm moves the focus towards the development of a more useful everyday-type of tool that would be organized around the model, the simulation module and a set of instruments. The whole would be linked to the manufacturing system to monitor its behavior according to certain important variables. This paper introduces COGNOSCO, a modular and evolutive environment currently under development which follows these principles. It will enhance model and components reusability and encourage the use of the model on a real-time basis to assist the control and management tasks, with the abilities to learn from both the system and the decision-maker and to propose and test solutions to the problem of recovery from shocks.

. Introduction

In a recent paper, V. B. Norman (1992) exposes his view that simulation will become an everyday tool within the next few years, serving as the nervous system of the factory in relation to the other production control and management tools. J. G. Crookes (1992) points out the emerging trend of object-oriented programming (OOP) in the field of simulation software development, which could be seen as one of the bests ways to fulfill Norman's prediction. Until recently, the products available on the market offered similar features in terms of their coverage of the simulation study cycle and were built up according to sequential programming principles. This kind of programming goes against the modularity and evolutivity of both simulation software and the models developed along with them. OOP, on the other hand, enables the creation of reusable blocks which results in more user-friendly and modular software, while addressing a wider range of studies and providing a wider range of assistance to the user for completing these studies.

This paper introduces an object-oriented manufacturing simulator developed within the framework of the ASMEMA project (*ASsistance MEcanique à la MAnutention*), the aim of which is to create a tool for the multi-criteria evaluation of manned workspaces. This simulator is the first step towards the development of a "simulation environment" for the analysis of manufacturing systems in accordance with Norman's viewpoint. In the first section, a brief overview of the uses of simulation in manufacturing and of simulation packages is presented to try to provide an understanding of the strengths and weaknesses of simulation tools. A similar overview of object-oriented modeling (OOM) and simulation (OOS) is included followed by a description of the main modules of our environment in the third section. The last section quickly introduces the developments that have been planned or completed thus for.

1. Manufacturing Simulation: uses and tools

1.1 *Application to manufacturing systems analysis*

The use of simulation as a decision support tool is still the preserve of major enterprises and consulting firms, who use it to study new systems and to fine-tune existing ones (e.g. investment and replacement projects or layout configuration). A few years ago, tools appeared to assist production control, particularly scheduling (Bilberg & Atling 1991). In this new area, simulation is not only used to perform "what if" studies, but also to allow real-time interaction with others tools, like MRP systems. Compared to these, simulation leads to more accurate results, taking into account finite-capacity constraints instead of working on the basis of the infinite hypothesis as most of these tools do. However, people still lack confidence in using this technology.

This fact was pointed out in a recent survey published in Industrial Engineering (vol 24-7, July 1992). Although this study does not reflect the thinking of the whole community, it shows that users choose simulation software first and foremost because of its flexibility and ease-of-use features, which are usually contradictory. Speed of execution, animation quality and manufacturing and material handling features are listed then, in this order of importance. This survey emphasized the importance given by users to graphics and animation, as opposed to purely statistical models, but, a few lines below, decision-makers admit that if they had the opportunity, they would prefer to take courses in statistics, and concede that their lack of understanding of this technology is a major obstacle to its use. Keller & al.

(1991) had published this result a year earlier when they outlined the three reasons why simulation fails, which are, in their opinion: the resistance to changes (skepticism), the need for education and training, especially in statistics, and the large amount of time required to complete a simulation study. In the next section, the advantages and disadvantages of the various kinds of simulation software available are discussed, based on their coverage of the steps in a simulation project.

1.2 *How simulation studies are handled by simulation tools?*

The simulation tools available on the market can be divided into two distinct categories: languages (including true simulation languages like SLAM, SIMAN, GPSS, etc. and general purpose programming languages such as Pascal, C, FORTRAN, etc.) and simulators (Witness, SimFactory, FACTOR, etc.) (WSC 1990, for example, includes documentation introducing these tools). A third category, which can be called evolutive simulation tools is discussed in the following section. Generally speaking, these tools don't cover many of the steps of a simulation experiment once the problem is defined. The main steps are: data collection and organization, model development and validation, design of experiments (DOE), simulation run(s), results analysis, conclusion and recommendations, possibly with loops between some of these stages. Evaluation of the tools can be carried out according to their ability to perform these steps.

Data collection is not assisted by either a language, or a simulator. Although simulator does provide some assistance with fill-in forms and menus, this step still remains one of the most painstaking. For example, the estimation of probability distributions based on the analysis of collected data remains an external process. For the modeling task, languages can represent a wide variety of situations (a higher degree of flexibility is obtained with programming languages), but the user must have highly developed programming skills (he might even have to model the simulation behavior using programming languages). This goes against the ease-of-use and user-friendliness features, thus widening the gap between the modeler and the decision-maker. Moreover, the models tend to be "one shot" models, and new studies must begin from scratch. This is quite a disappointing aspect considering the amount of time spent in data collection and modeling. On the other hand, simulators don't require programming skills, using modeling concepts closer to the manufacturing system and, as such are easily understood by industrial engineers. They use menus, fill-in forms and graphic-driven interfaces, and even allow some modularity, but this relative user-friendliness is provided at

the expense of modeling flexibility and tends to reduce the range of studies (Adiga & Glassey 1991). The DOE task requires expertise to define the objectives of the study and is one of the most difficult to complete. Only a few elements, such as the number of replication runs, the length of the start-up period and the type and frequency of reports are taken into account directly in both kinds of software. The simulation task itself is, of course, performed by all these tools and the results produced are also quite similar. They are based on time, time ratios and utilization percentage, and are summarized by tables or, more often now, graphically. These outputs are the only help available with these tools for the analysis of system performance. Furthermore, this step is closely related to the DOE process. Indeed, the measures have to be compared to the objectives defined in this former phase. These two steps, as well as the conclusion and recommendation stage, also require a knowledge of the manufacturing system studied and the possible ways in which it can be improved.

Finally, to summarize the ideas exposed above:

- modeling ease and tool flexibility are contradictory but mostly requested features;
- scenario design and output analysis require more (expert) assistance;
- performance measurement tools are still too simple.

Some solutions available to reduce the effects of these problems are presented below.

1.3 *Simulation support tools and new simulation environments*

Simulation support tools tend to enhance the coverage of the simulation experiments and can be arranged in three categories: external and punctual tools, "shell" tools and evolutive, integrated packages.

Among the tools in the first category, some are specially designed for simulation purposes (e.g. statistical packages), while others are general purpose tools (e.g. spreadsheets, queuing models, etc.). These tools address only one problem at a time: distribution estimation (UNIFIT II, refer to WSC 1990 for a description of this tool); modeling, allowing static design using spreadsheet models, capacity planning using queuing networks prior to simulation modeling itself (Huettner & Steudel 1992); general expert statistical output analysis (Mellicamp & Park 1989, Biles & Hatfield 1991) or focused analysis, for example, limited to important parameters isolation (Starr 1991). These tools are not well integrated with simulation packages and a great deal of data manipulation is required to use them together.

The second category of tools provides this integration. While the tools presented above can be used either with simulation languages or with simulators, these are designed for simulation languages. This category can be divided into two classes: the classical shells (e.g. TESS, Cinema, Proof) (WSC 1990) and the intelligent front ends (IFE) and simulation program generators (SPG). Those in the first class can be adapted to several languages (e.g. TESS for SLAM, GPSS and MAP/1, which is a simulator) or not (e.g. Cinema for SIMAN). They are developed according to the same sequential programming paradigm (from the user's viewpoint) as the simulation tools, although they suffer from the same shortenings in terms of evolutivity and modularity. Furthermore, they only assist the user marginally (e.g. off-line graphical design, animation, graphic presentation, model and data-base management) and still neglect key points such as results analysis or DOE. They try to confer simulator-like user-friendliness to the languages. Some of these support tools offer CAD interfacing capabilities for modeling and animation purposes. The tools from the second class tend to be more powerful, using artificial intelligence (AI) principles to support modeling (O'Keefe 1986, Haddock 1987). IFE and SPG translate fill-in forms or natural language system description (Ford & Schroer 1987) into simulation language formalism, thereby simplifying the modeling task, but reducing the study spectrum. Generally, they are confined to FMS studies, with a limited set of customizable parameters. SPG provides some (system-dependent) assistance for analysis but does not go much further than that.

The next step is handled by knowledge-based systems (KBS) which are the main constituents of the third category. These tools (e. g. ROSS, SES, KBS, etc.) like the IFE and the SPG, remain prototypes with some exceptions like Carnegie Group's SIMULATION CRAFT (Fox & al. 1989). They also make use of AI principles and are rule-based, goal driven and object-oriented. They are implemented in LISP or Prolog, which support logical programming. These tools are provided with user-friendly and graphical interfaces, different modeling levels (or abstraction levels, Bond & Soeterman 1988), rule-based validation models, scenario generation mechanisms (Roberts B. 1989), and automatic output analysis and recommendation (Shannon & al. 1986, Fox & al. 1989).

These prototypes, developed at the beginning of the last decade, represent the highest degree of integration and act as guides for further development. They are not designed to support existing languages, but redefine both the simulation tools and their use. Although they are very useful during the goal definition and results analysis processes, they don't encourage modularity and reusability, even though object-oriented programming give them this ability. They

also make conservative use of the models, which are kept in their design role rather than being fully integrated in the continuing manufacturing analysis process. This last avenue is the one several research teams have chosen (Mize & al. 1992, Bilberg & Atling 1991, Adiga & Glassey 1991) and is still well thought of by simulation software producers (Law 1990). Some of them propose evolutive products like Automod II (Norman 1992, Law 1990), which includes a scheduling and a statistical module in a powerful graphically oriented package, and Simple++ (Becker & al. 1991).

In concluding this section, two facts can be noted:

- there is a lack of tools to support the user in the modeling and analysis tasks (including scenario design);
- there is a trend emerging towards a global vision, where models are to be used on a daily basis, making the non-integrated procedural development process, which has prevailed in software and model design up to now obsolete (Mize & al. 1992, Norman 1992, Adiga & Glassey 1991, Bucki 1989).

The models developed within this kind of environment are designed to be used on a daily basis to follow manufacturing system evolution, focusing on the few variables that are important. This is the way other systems, like cars or even nuclear utilities, are piloted. The indicators depend on the system under study, its components and the level of abstraction at which the decision is to occur. This underlines once again the importance of a modular way of modeling the manufacturing process. This ideal is supported by the OOP concept.

2. Object-oriented modeling (OOM) and simulation (OOS)

2.1 *Principles of OOM*

OOP is one of the most powerful paradigms for representing manufacturing systems. The main features of interest of OOP in this kind of application are its ability:

- to match physical and software objects (concepts of class and encapsulation);
- to realize multiple-abstraction modeling (hierarchy and inheritance);
- to build specialized libraries from existing objects (inheritance, reusability and modularity concepts);

among other advantages, as described elsewhere (Adiga & Glassey 1991, Roberts & Heim 1988). The languages used in the manufacturing simulation domain are C++ (Eldredge & al. 1990, Choi & Minoura 1991), Smalltalk 80

(Mize & al. 1992), Objective C (Adiga & Glassey 1991). Mize & al. (1992) present a comparison of OOM and sequential modeling. In such a context and considering the concluding remarks to the first section, simulation software is to be included in a modeling environment where simulation still remains one of the main tools of analysis for the manufacturing system represented here. The modules of this environment, in terms of OOP, can be defined independently of each other, enabling a step-by-step development and integration of modules. Encapsulation and modularity allow the use of different functional representation techniques in the OOP context (e.g. rule-based programming).

The object concept is still an open one and new interpretations are made to capture the essence of manufacturing systems into a formalism that will enable easy representation of the physical relations and of the information flow between objects within the system (Bucki & al. 1989, Rodde 1989, Choi & Minoura 1991). The gains to which this kind of modeling could lead in the understanding of manufacturing system is discussed in the first two references.

2.2 *Limits of OOS*

OO models tend to be heavier than traditional ones and thus require more space. Moreover, OOS runs slower than traditional simulation, because of the message being sent between the objects during execution and because of the dynamic allocation/deallocation process tied to object creation and deletion. Some studies describe procedures for optimizing memory management, but the results are not convincing (Beaumariage & Roberts 1991).

Another avenue of interest is the one of parallel simulation. In fact, manufacturing processes are in themselves parallel, and so one would think that their parallel simulation could be easily achieved. However, synchronization problems obstruct this way and the use of complex methods, both conservative and optimistic (using the time-warp concept), is required (Fujimoto 1990). The tremendous network of direct links (i.e. successive ones) and indirect links (i.e. informative ones, to verify a condition, for example) between the different components of a system seems to be very hard to represent without restricting the modeling capability, which is not a satisfying option. Another more trivial way to exploit parallelism might be to perform simultaneous simulation runs, knowing that a simulation study is composed of several replications of the same experiment. Moreover, the constant and rapid evolution of both hardware and software leads us to believe that the slow speed of OOS is not a good reason to abandon this avenue.

3. Our simulation environment

Based on the previous analysis, our research effort has been oriented toward the development of a simulation environment that could provide a framework in which the model would be used from the design step to the operation of the system under study. Such a model, linked to the production system, would reflect its changes in real-time and monitor its evolution through a set of performance indicators. Linked to the simulation module, the model performs management and organization policy evaluations, just as in a classical simulation study, thus allowing the "learning to understand" process (i.e. what are the parameters of interest and what are the reactions to external and internal shocks to the system). Performance measures are carried out by specialized instruments.

This section presents the environment COGNOSCO designed according to the above principle. The ASMEMA project, introduced first, has supported its creation and some of the COGNOSCO elements are still modules of this software. Most of the following examples come from studies completed for this project which emphasized human material handling. This specificity does not, however, restrict the generality of the modules developed in this context.

3.1 *Development context: the ASMEMA project*

The ASMEMA software is designed to perform multi-criteria evaluations (i.e. ergonomic, economic and operational) of manned workstations and in particular, to address the problem of injuries during load handling (METIS 1992). This tool is also useful in the automation/non-automation decision process, taking into account the benefits of mechanical assistance in material handling (Fabre & al. 1992) and including the ergonomic aspect in the process of evaluating alternatives. This tool is composed of two main modules: the graphical modeling and ergonomic analysis module, and the simulation and operational and economic analysis module. The information from the first, modeling the physical features of objects, are reused in the simulation module, while information provided by the simulation experiments (e.g. frequency estimation, total weight carried, weight variation, etc.) are used by ergonomic instruments. This software runs on a Silicon Graphics platform, which was selected because of its graphics capabilities. It is implemented in C++ (Wiener & Pinson 1988), using the NIHCL library (Gorlen & al. 1988), the X Window programming system (Barkakati 1991) and the Motif toolkit (OSF 1991). The next paragraph provides the description of the simulation module.

3.2 *Main components*

The main components of COGNOSCO have now been realized within the ASMEMA context.

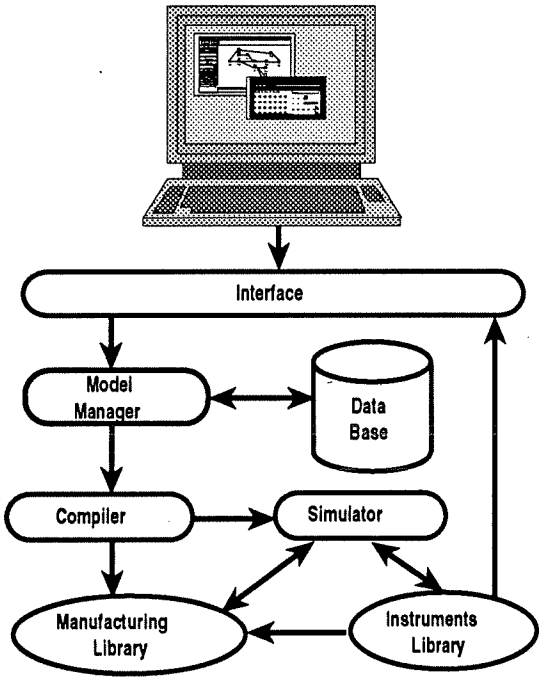

Figure 1: Simulation module organization

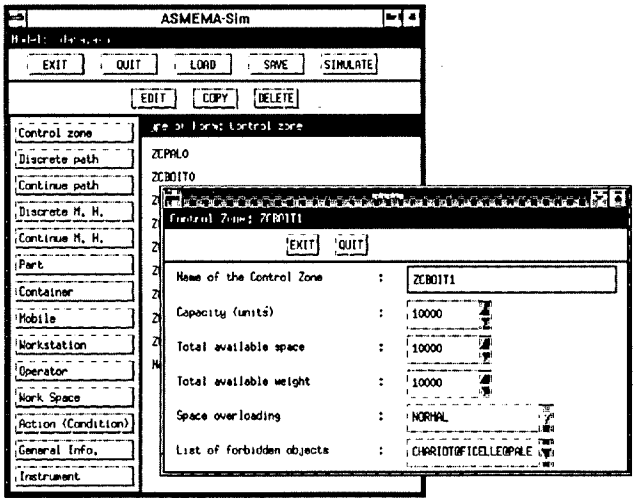

Figure 2: User-interface (main window and CZ form)

There are seven distinct elements (Figure 1): the user interface, a model base and its manager, a compiler, a simulation module, a generic manufacturing library and an instruments library.

3.2.1 User-Interface

The user interface (Figure 2), is still a prototype and has allowed us to test the other environment's modules. It will be refined to ensure the modularity between the model components. However, the use of fill-in forms will remain. All the forms corresponding to the different types of objects are organized the same way (Figure 2). The top row contains the EXIT and QUIT buttons. The left column contains the comments regarding the fill-in field which is on their right. These fields are provided with mechanisms (e.g. scales, pop-up lists, multiple-choice pop-up lists, form EDIT button, etc.) to facilitate the recording of informations and they "know" the nature of the information they may receive, allowing simple error-checking. Furthermore, the comments on the left give access to an on-line help feature for the field in question. The contents of the comments and the help messages, as well as the type of field are defined in customizable ASCII files.

3.2.2 Model manager

Like the user-interface, this module is still a prototype performing model saving and retrieving operations. When they are saved, the contents of the forms is checked with a knowledge of their fields relationships, when possible. This kind of checking is also performed when the complete model is saved to ensure that all the requisite forms and all the forms referenced by the user are filled in. Two kinds of partial reutilization are possible: functional and hierarchical. The first organizes the forms in three libraries: the product library, the resource library and the goal definition library (including both scenario description and instruments). The second allows the aggregation of components into sub-models which can also be treated as model components.

Both the interface and the model manager are independent of the other modules. After minor modifications, they can be customized to meet the needs of other software. Only the compiler, which provides the link with the application, must be redesigned. In fact, these two modules were first developed to provide a user-friendly interface to the simulator MAP/1 (Fabre 1990) and were customized for this new application.

3.2.3 Compiler

The compiler creates the manufacturing objects and the series of actions representing the system behavior using the information in the forms and checking for logical errors. During the process of action creation, it adds implicit actions and conditions (see paragraph 3.5). This kind of "expertise" doesn't allow the modification of dynamic parameters during a simulation run and, in the future, only static characteristic modification will be permitted.

3.2.4 Simulation control module

This module and the class it contains are independent of the kind of simulation study. The simulation classes are organized as follows (Figure 3):

- the events and their related conditions (Evenement and Condition classes, see paragraph 3.4)
- the calendar of events and its management mechanism (Manager class);
- the auxiliary mechanisms associated with the manager (Horloge and Distributeur classes);
- the trouble generator for the modeling of breakdown, maintenance and operator's break (Alea class).

The object *manager,* the unique instance of the Manager class, manages the calendar of events. Each event received is ranked according to its due date and, in case of conflict, to its priority. This object take into account the end-of-shift rule to process its events. The *horloge,* the unique instance of its class, maintains the current time and the end of simulation date. It is in charge of the verification of the time-based conditions (see Table 2). The *distributeur*, also the unique object of its class, deals with the verification of probabilistic conditions (see Table 2). It also generates random numbers according to eight distributions (i.e. uniform, triangular, exponential, normal, erlang, beta, gamma, weibull). Moreover, it can read values modeling some phenomenon in a user-supplied ASCII file. The objects of the Alea class (*panne, maintenance, maintenanceHS* and *pauseOp*) generate blocking events according to user-supplied information on resources. These events are immediately effective (breakdowns, operator's "hard" breaks) or delayed until the end of the current action (maintenance, operator's "cool" breaks). All these classes are independent of the manufacturing hierarchy and have only the knowledge of the abstract classes (the ones within the dashed boxes in Figure 2). The "virtual" concept of the OO procedures (or methods) and the inheritance allows dynamic binding, and so the appropriate treatment, defined for a particular class, is always performed.

3.2.5 Instrument library

The instrument concept was introduced with KBS (Fox & al. 1989). In our environment, instruments are tailored to measure nearly everything, even allowing a focus on, for example, the 37th object of a particular type. They look like their physical homonyms and are ranked from the simple counter to the expert analyzer (which can propose improvements and explain its diagnostic). The library contains only simple instruments of the following types:

- event-tracing instrument (TRACE);
- operational and economic instruments;

These instruments must be linked with the objects they analyze but are not confined to the one simulation process and are also used when the model is coupled with the manufacturing system. Thus, they monitor the behavior of the real objects. Their OO development allows this rapid creation of new instruments based on the root class of this hierarchy (Instrument class), which defines all the external mechanisms for data extraction and presentation. This can be done without modifying either the objects of the manufacturing library or the classes of the simulation module.

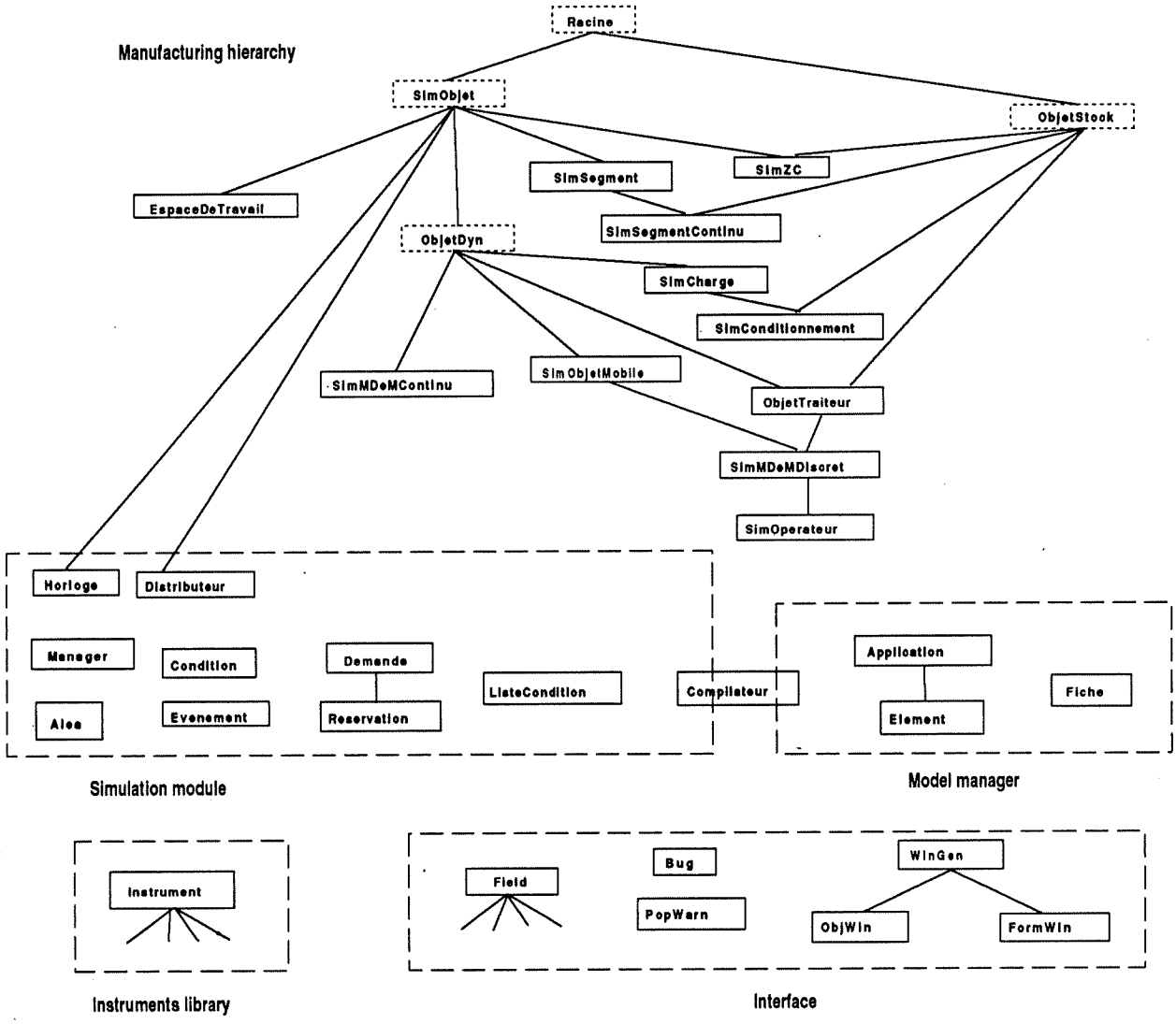

Figure 3: COGNOSCO classes hierarchy

3.3 *Manufacturing classes hierarchy*

Figure 3 also presents the kernel of generic manufacturing classes, enabling the representation of manufacturing systems at a first level of abstraction. Even though the EspaceDeTravail class can now realize the aggregation of simple components into a submodel, the multi-abstraction mechanism is not fully operational and this hierarchy must be completed. However, the integration of domain-specific libraries (e.g. warehousing, job shop, distribution, services, etc.) and typical resources libraries (e.g. for conveyors, CN machine tools, etc.) is now possible and based on the generic classes of this hierarchy that enhance this kind of flexibility. These libraries are modular, independent and compatible, inheriting from the same basic elements (even several of them).

3.4 *Types of events and behavior modeling*

From the dynamic viewpoint, fifteen types of actions, including two for animation purposes (Evenement class) and nine types of conditions (Condition class) were defined (Table 1 and 2). The actions and their related list of conditions (ListeCondition class) are cristalized on the control zone or CZ (SimZC class). These objects can be viewed as logical nodes and system decision points, as well as storage places. For the moment, the actions are generated and stored by the compiler (It will be possible with the use of an expert on-line analyzer to generate alternative actions in response to unexpected events, providing the objects with alternative paths in real time and, in addition, to test such rerouting policies, for example). The actions can take effect immediately or are delayed until a set of conditions is verified. The objects are notified by the CZ when they arrive. The CZ also asks the concerned objects to verify the conditions (Demande and Reservation classes) for all the simultaneously possible actions. An action is performed every time its set of conditions is true. Once the new event has been sent to the *manager* and its parameters updated, the object cancels every condition verification request made in its name. It then ask its CZ to make the next requests for information, and so on. Several actions can be defined for a given class, as well as several conditions for a given action, using the "and" (ET) and "or" (OU) links.

Actions (english translation)	User-defined parameters
APPARAIT (appear)	first_apparition_date frequency e.g. APPARAIT [0 CONS(250)]
DISPARAIT (disappear)	disparition_frequency (or nothing) e.g. DISPARAIT []
CHARGE (load)	workstation_to_load e.g. CHARGE[MACHINE1]
PORTE (handle)	container_or_MH_or_operator_to_load e.g. PORTE[PALETTE]
DEPOSE (unload)	---
ASSEMBLE (assemble)	object_created {component(quantity)} ex: ASSEMBLE[BOITE FD(2) BR(5)]
DEMOLIT (produce)	{object_produced(quantity)} ex: DEMOLIT[PL_1(2) PL2_(10) BD(4)]
MODIFIE (modify)	modif._duration {*parameter* absolute_flag value} ex: MODIFIE[UNIF(25, 40) POIDS OUI 25]
ATTEND (wait)	waiting_time ex: ATTEND[CONS(250)]
PART (leave)	leaving_path ex: PART[TRNCN_23]

Internal actions: STOPPE (stop) to end simulation; BLOQUE (block) for an Alea event, DEPLACE (move) for the new spatial position, DESSINE (draw) for graphical and animation updating and ARRIVE (arrive) for object arrival over a CZ.

Table 1: Types of actions

Conditions (english translation)	User-defined parameters
HORAIRE (time)	quantifier date e.g. HORAIRE > 125
PROBABILITE (probability)	quantifier value e.g. PROBABILITE >= 0.87
BLOCAGE (blocked state)	CZ_name e.g. BLOCAGE CZ_1
ESPACE (space)	CZ_name path_name e.g. ESPACE CZ_2 TRNCN_1
PRESENCE (presence)	place object quantif. amount e.g. PRESENCE PORTEUR OBJET_3 >= 10
PARAMETRE (parameter)	place object quantif. amount parameter para._quantif. para._vlue e.g. PARAMETRE CZ_1 COLIS == 1 POIDS < AUTOREF
RSRV+PRESENCE (preempt+presence)	(see PRESENCE) réservation e.g. RSRV+PRESENCE CZ_34 CHARIOT == 1 GLOBAL
RSRV+PARAMETRE (preempt+param.)	(see PARAMETRE) reservation_ e.g. PARAMETRE CZ_1 FIXTURE == 1 OCCUPATION < 10 LOCAL

Table 2: Types of conditions

The actions are considered in their modeling order, with the highest priority given to the first in case of conflict. The demands can be informatives (Demande class) or consumers (Reservation class). The first ask only for information about the object's state, while the second preempt this object (P) in the name of the one making the request (A) until A sends a release message to P. This kind of demand is used, for example, to claim a discrete manufacturing handler to pick-up a load and deliver it somewhere else. Thus, material handlers and operators are guided through the path network by the loads they are carrying. Furthermore, they can choose the next reservation to process according to seven rules (i.e. FIFO, LIFO, due date, type-of-load-based priority, CZ-based priority, farthest load, nearest load).

3.5 *Key features*

This set of modules represents about 28,000 lines of C++ code. Its speed varies from 20 to 60 events per second (roughly) depending on whether or not the "TRACE" instrument is used (Figure 4). Three types of end-of-shift rules can be modeled. A shortest-path algorithm is used to find the way to a pick-up CZ. Space and path checking are done implictly before an object is allowed to leave for another CZ. A batch of new objects can be released once during the run, or according to certain conditions or frequency. The user can call on 40 parameters for the actions and conditions, and can refer to the object itself or its carrier, either on a local basis or throughout the entire workspace. Workstations and "discrete" material handlers (i.e. carts, cranes, etc., as opposed to "continuous", like conveyors) claim their

pilots implicitly, release them at the end of a task or when a breakdown occurs. In this case, they request throughout the entire space for repairs and, when they are operational again, they look for their pilot to finish the interrupted task, if required. These objects as well as the operators can perform set-up operations between different or similar loads.

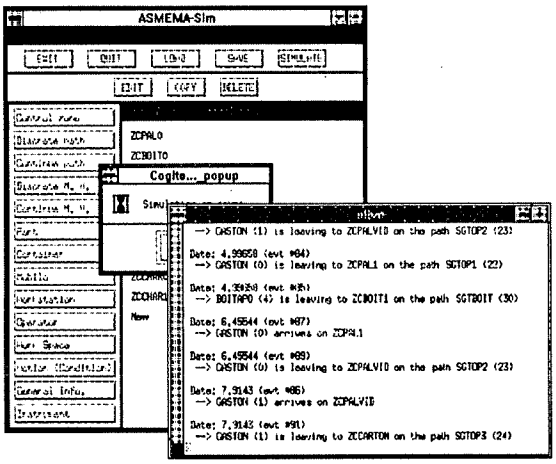

Figure 4: TRACE output

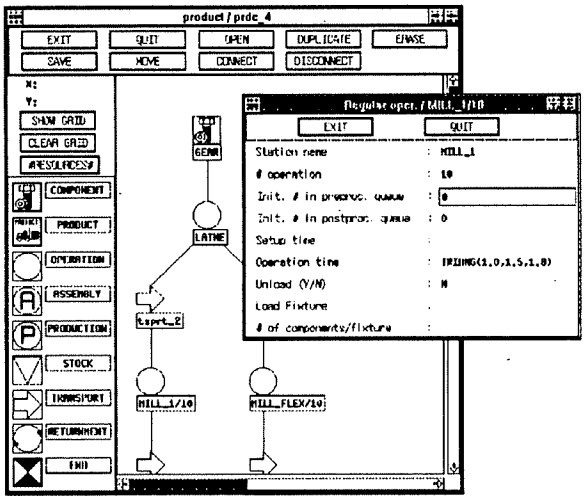

Figure 5: Process chart representation for products

4. Future enhancements

Our environment does not address all the functions presented in the first section. Although it permits modular and evolutive modeling, the model manager is still unfinished. However, the instrument concept is now defined and we can use a generic manufacturing hierarchy. So, the current development stages emphasize the DOE and output analysis skills.

4.1 Interface and model manager

An earlier project (Fabre 1990) has shown the usefulness of process charts for representating the manufacturing process (i.e. the series of actions and conditions for the loads). This kind of graphical modeling (Figure 5) is more user-friendly than the current pseudo-language that will remain as an alternative modeling method. It allows the aggregation of operations which enables multiple abstraction levels from the process perspective.

It will also simplify the modeling of the information flow and will be tailored to minimize the information explicitly asked of the user. This graphical modeling will make use of the knowledge of the paths available between two locations, for example, to choose the right material handler at compilation time. The user can, however, fix this

choice explicitly at the time of modeling. Resource descriptions will also make a better use of the graphical representation, allowing 3D animation and reuse of the graphical generic elements from the ASMEMA project.

4.2 *Manufacturing library*

During the analysis step of this library, a design tool has been designed to build resources, using simple geometric blocks and allowing them to perform some simple function (e.g. lift, translate, rotate, etc.). It permits the development of specialized manufacturing libraries, as well as the rapid creation of graphical user-defined resources, and so will be fully operational prior to the creation of new libraries.

4.3 *Instrument libraries*

The operational instrument library curently contains simple instruments, but permits the realization of the same measures as classical simulation tools, using counter, chronometer, and "state chronometer" instruments. For the ASMEMA project, the economic library put the emphasis on operator-related instruments. However, in line with an emerging trend in the field of manufacturing cost estimation (Arbel & Seidman 1984, Azzone & Bertele 1989, Park & Son 1988), the new instruments will try to emphasize the related intangible costs (e.g. quality, flexibility and customer satisfaction costs) according to the ANSI Z94.5 (1988) standard. Simulation will make it possible to take into account of both the uncertainity factors and the intangibles in cost-related studies (Park & Son 1988, Suresh 1990).

4.4 *Scenarist tool and output analysis assistant*

The few existing tools in these categories are still incomplete, considering the level of knowledge they require from the statistical and the manufacturing system perspective (Figure 1 in Mellicamp & Park 1989). The tools in these two categories are interrelated: the scenarist defines the objectives that the analyst will compare with the results obtained with the instruments proposed during the scenario phase. This step can be viewed as follows:

- definition of the duration of the simulation experiment;
- number of replications of the experiment;
- modifications to parameters between experiments;
- definition of the objectives to be met.

In the last step, the objectives can be divided into two categories: the fuzzy objectives and the precise ones. The first relate to global experiments (e.g. new system design, machine replacement projects). Thus, the evaluation tools monitor general parameters in an effort to point out bottlenecks, bad inventory levels, high cycle times or poor machine utilization, for example, and report these facts. The second deal with more specific studies. The user must be provided with an "objective explorer" that will make it possible to choose the key factors interactively: the objects to study, the kind of study for each of them and the set of instruments to use. During these three steps, the tool must suggests choices, but the user still controls the final decision. The evaluation process follows, again including the summary reports, but supplying the user with detailed performance measures concerning the objects under study, and even proposals for improvements (Fox & al. 1989).

A knowledge base (KB) needs to be linked with this tool. This KB will "learn" from the results of the simulation experiments and from the user. In all situations, the KB manager checks for the consistency of the rules and ensures the homogeneity of the KB. The KB and its learning mechanism can allow rapid and accurate recovery after shocks affecting the manufacturing system. Given different alternatives, the user can simulate some of them to select the most appropriate in this particular case. The reaction speed when an external shock occurs (e.g. delivery delayed, new order, etc.) or an internal one (e.g. breakdown), depends on the amount of knowledge available, and so forth, reflects the accuracy of the performance indicators and the degree of knowledge the user has of the manufacturing system he is trying to manage. It can be pointed out that there is an upper limit to the amount of knowledge the system can maintain, and that the possible uses of "meta-knowledge" remain to explore.

4.5 *Production system interfaces*

Connecting points must be defined to ensure the link between the information system, the model and the instruments and to follow the real-time model evolution according to the system state. These interfacing points must filter and arrange the data to send events (Evenement class) to the model. They depend on the nature of the data received, and so, will be implemented when the tool is integrated into a manufacturing information system. An example of such an integrated communication system for the process industry can be found in Barcelo (1992).

. Conclusion

Following an overview of the main features of existing simulation softwares, an OOS environment has been introduced consisting of a simulator, a manufacturing library and an instrument library. A DOE module and a performance analysis module are now under development and will be included in this system. A model created in this environment can perform studies throughout the life of the manufacturing system (from rough design to operation control). It must be able to provide assistance for quick recovery from unexpected shocks. To achieve this, an evolutive knowledge base with a learning capability will also be included. Ultimately, this tool will make it possible to pilot a production system in the same way a car is driven: based on the analysis of a small number of pertinent indicators.

Bibliography

ANSI Z94.5 Engineering Economy Subcommittee, 1988, Notation and terminology standards for engineering economy, *Engineering Economist,* 33 (2), pp 145-173

Adiga, S., Glassey, R., 1991, Object-Oriented Simulation to support research in manufacturing systems, *International Journal of Production Research,* 29 (12), pp 2529-2542

Arbel, A., Seidman, A., 1984, Performance evaluation of FMS, *IEEE Transactions on systems, man and cybernetics,* 14, (July - August 84), pp 118-129

Azzone, G., Bertele, U., 1989, Measuring the economic effectiveness of flexible automation: a new approach, *International Journal of Production Research,* 27 (5), pp 735-746

Barcelo, Y., 1992, L'information en temps réel, *PLAN,* Sept. 92, pp 16-18

Barkakati, N., *X Window System™ Programming,* Macmillan Computer Publishing, Carmel, In, 1991

Beaumariage, T.E., Roberts, C.A., 1991, Object-Oriented modeling: attempts at improving model execution speed, Simulation series 23 (3), R. Ege ed., pp 93-99

Becker, D., Walderich, W., Noyes E.L., 1991, Efficient modeling and simulation of logistic processes with an object-oriented system, Simulation series 23 (3), R. Ege ed., pp 87-92

Bilberg, A., Alting, L., 1991, When simulation takes control, *Journal of manufacturing systems,* 10 (3), pp 179-193

Biles, W.E., Hatfield, I.T., 1991, Simultaneous factor screening and region reduction in computer simulation experi-

ments, *Computers and Industrial Engineering,* 21 (4), pp 617-621

Bond, A.H., Soaterman, B., 1988, Multiple abstraction in knowledge-based simulation, in 'AI and simulation: the diversity of applications', T. Henson ed., SCS, pp 61-66

Bucki, J., Semeteys A., Lasoudris, L., 1989, Architecture des systèmes décisionnels, cahiers 1 et 2, Institut de Génie Décisionnel, Paris

Choi, S., Minoura, T., 1991, Active Object System for discrete system simulation, Simulation series 23 (3), R. Ege ed., pp 209-215

Crookes, J. G., 1992, Simulation in 1991: ten years on, *European Journal of Operation Research,* 57, pp 305-308

Elderedge, D.L., Mc Gregor, J.D., Summers M.K., 1990, Applying the object-oriented paradigm to discrete event simulation using the C++ language, *Simulation,* 54(2), pp 83-91

Fabre, M. 1990, Une interface orientée objet pour la construction modulaire de modèles en simulation manufacturière, M. Sc. A. thesis, Ecole Polytechnique de Montréal

Fabre, M., Pitre, J.C., Gilbert, R., Leblanc, D., 1992, The decision of non-automation in manned workstation, POMS 3rd annual meeting, Oct. 18-21, Orlando, Fl

Ford, D.R., Schroer, B.J., 1987, An expert manufacturing simulation system, *Simulation,* 48(5), pp 193-200

Fox, M.S., Husain, N., Mc Roberts, M., Reddy, Y.V.R., 1989, Knowledge-based simulation: an artificial intelligence approach to system modeling and automating the simulation life cycle, in 'AI, Simulation and modeling', L.E. Widman, K.A. Loparo, N.R. Nielsen eds, John Wiley & Sons, pp 447-486

Fujimoto, R., 1990, Parallel discrete event simulation, *Communication of the ACM,* 33(10), pp 31-53

Gorlen, K.E., Orlow, S.M., Plexico, P.S., *Data abstraction and object-oriented programming in C++,* John Wiley & Sons, Chichester, 1988

Haddock, J., 1987, An expert system framework based on a simulation generator, *Simulation,* 48(2), pp 45-53

Huettner, C.M., Steudel, H.J., 1992, Analysis of a manufacturing system via spreadsheet analysis, rapid modeling and manufacturing simulation, *International Journal of Production Research,* 30(7), pp 1699-1714

Keller, L., Harrel, C., Leavy, J., 1991, The three reasons why simulation fails, *Industrial Engineering,* 23(4), pp 27-31

Law, A.M., 1990, Simulation software for manufacturing applications: the next few years, *Industrial Engineering,* 22(6), pp 14-15 and 22(7), pp 18-20

Mellicamp, J.M., Park, Y.H., 1989, A statistical expert system for simulation analysis, *Simulation,* 52(4), pp 134-139

METIS, 1992, Rapport de présentation du projet et du logiciel ASMEMA, *unpublished technical report,* Ecole Polytechnique de Montréal

Mize, J.H., Bhuskute, H.C., Pratt, D.B., Kamath, M., 1992, Modeling of integrated manufacturing systems using an object-oriented approach, *IIE Transactions*, 24(3), pp 14-25

Norman, V.B., 1992, Future directions in manufacturing simulation, *Industrial Engineering,* 24(7), pp 36-37

O'Keefe, R., 1986, Simulation and expert systems: a taxonomy and some examples, *Simulation,* 46(1), pp 10-16

OSF, *OSF/Motif™ programmer's reference (rev. 1.1),* Prentice Hall,Englewood Cliffs, NJ, 1991

Park, C.S., Son, Y.K., 1988, An economic evaluation model for advanced manufacturing systems, *Engineering Economist,* 34(1), Fall 88, pp 1-26

Roberts, B., 1989, Scenario generation for knowledge-based simulation, in 'Advances in AI and Simulation', SCS, pp 214-218

Roberts, S.D., Heim J., 1988, A perspective on object-oriented simulation, in 'Proceedings of the 1988 Winter Simulation Conference', SCS, pp 277-281

Rodde, G., *Les systèmes de production: modélisation et performances,* Hermès, Paris, 1989

Shannon, R.E., Mayer, R.J., Phillips, D.T., 1986, Knowledge based simulation techniques for manufacturing, SME technical paper, *proceedings of the Ultratech Conference,* SME, pp 2.237-2.261

Starr, P.J., 1991, Integration of simulation and analytical submodels for supporting manufacturing decisions, *International Journal of Production Research,* 29(9), pp 1733-1746

Suresh, N.C., 1990, Towards an integrated evaluation of flexible automation investments, *International Journal of Production Research,* 28(9), pp 1657-1672

Wiener, R.S., Pinson, L.J., *Introduction to object-oriented programming and C++,* Addison-Wesley Publishing Co., Reading, Ma, 1988

WSC, 1990, *Proceedings of the 1990 Winter Simulation Conference*, IEEE, O. Balci, R.P. Sadowski & R.E. Nance eds