*Proceedings of the 1993 Winter Simulation Conference*
*G.W. Evans, M. Mollaghasemi, E.C. Russell, W.E. Biles (eds.)*

# MONITORING MANUFACTURING SYSTEM BEHAVIOR BY CONTINUOUS DISCRETE-EVENT SIMULATION

Michel Fabre
Daniel Leblanc

Ecole Polytechnique de Montréal, Department of Industrial Engineering
P.O. Box 6079, station "A", Montréal, (Québec), H3C 3A7, CANADA

## ABSTRACT

A simulation environment (COGNOSCO) designed to monitor the evolution of manufacturing systems is presented. Assuming that today manufacturing simulation is dedicated to (long term) system design or to (short term) scheduling, this environment is an attempt to address the continuum of decisions that lie between these two extremes. Its object-oriented basis supports modular and graphical modeling. A model can be connected either to the simulation engine to evaluate decisions, or to the production system to monitor its behavior. Simple instruments are used to track the evolution of important variables, while knowledge bases will provide the user with focused information and valid candidate decisions when a shock or progressive alteration in performances arise.

## 1 INTRODUCTION

Nowadays, manufacturing simulation is used for design and capacity planning (Lenz 1985) as well as for scheduling (Bilberg, and Alting 1991) purposes. System design involves long term or at least middle term decision-making, while scheduling deals with very short term decisions (even though prior simulation of the scheduling rules is possible). Between these two extremes, a continuum of problems remains unaddressed. On another hand, production systems managers are assisted by MRP and CAD/CAM systems as well as various CIM components. These tools use the same data and generate a vast amount of information which is difficult to manage without a truly integrated global database. An ideal way to tie the loose ends would be a model collecting and arranging the information issued by the manufacturing system (Norman 1992). The simulation environment COGNOSCO (from the Latin word meaning "I learn to know" or "I learn to understand") we are currently developing addresses this question.

In this paper, we will try to show how a toolkit like COGNOSCO could be used for manufacturing system analysis. The first part is dedicated to a brief presentation of this environment and its components. The second section introduces the basis of continuous monitoring for discrete-event manufacturing systems. Finally, the instrument concept is detailed in the last section.

## 2 THE COGNOSCO ENVIRONMENT

The COGNOSCO environment belongs to the family of tools that are designed to integrate simulation and manufacturing systems (Bilberg, and Alting 1991; Mize et al. 1992; Tayanithi, Mannivan, and Banks 1992). It has been developed according to object-oriented (OO) principles in C++ and represents a set of about 50 classes and 35,000 lines of code (Figure 1). Due to these OO origins which support modularity and inheritance concepts, this environment is composed of a collection of independent skilled modules (Fabre, and Leblanc 1993). To date, the following modules have been developed:

- a manufacturing library;
- an instrument library;
- a simulation engine;
- a graphical user interface (GUI);
- a model base and its manager;
- a compiler.

Current developments involve the creation of new instruments, especially to perform ABC (Activity Based Costing) analysis, and the knowledge base module (see section 4). The COGNOSCO components are independent and reusable. The compiler is designed to establish a temporary link between them when a model is to be simulated. The information stored in the model base is then translated to create the model and its components. This translation depends on the target

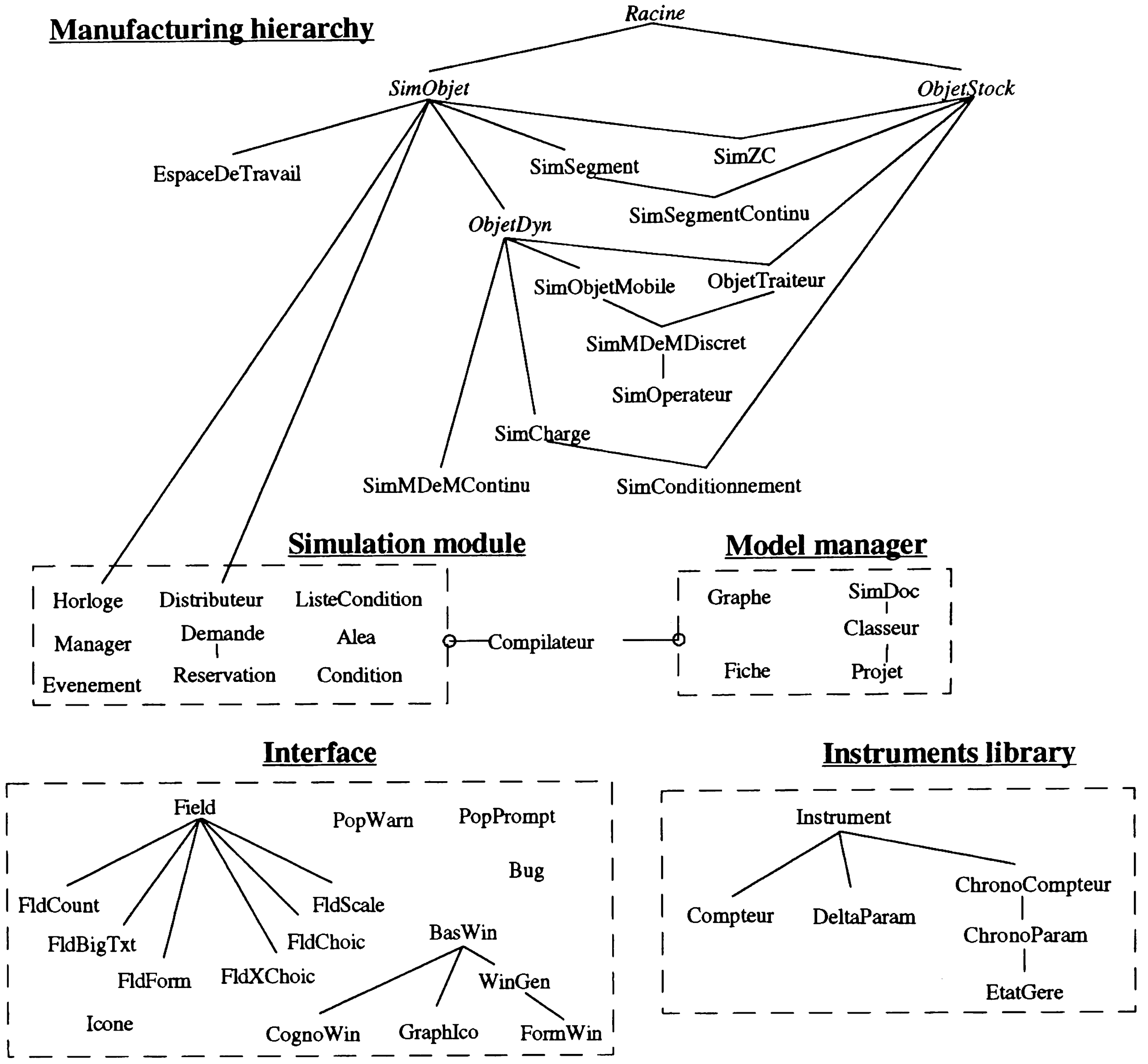

Figure 1: The COGNOSCO Classes

simulator. One possibility might be, for example, to connect the GUI and the model manager to another simulator. These modules allow model creation without coding, using only fill-in forms, graphs and icons (Fabre 1990).

A model, from the user point of view, is composed of graphs collecting icons in networks. Each icon identifies an element or another graph, allowing thus multi-level representation (Bond, and Soaterman 1988). To build a model, the user has to develop at least two graphs: one for the description of the manufacturing process for a given part, and another to represent the layout of the factory with its various resources. The description of the process is done using flow process chart representation (Fabre 1990). Two components complete a typical model: the table of schedules for the parts (using, for example, data from a production plan or probabilistic distribution to model parts arrival in the model) and the set of instruments to be used during the simulation. COGNOSCO can also be tailored to model specific manufacturing systems (e.g. FMS, warehousing systems, material handling oriented systems, etc.). Indeed, the two libraries (manufacturing components and instruments) provide a true basis for the new domain-specific sub-libraries. The inheritance concept allows us to refine the behavior of the manufacturing components classes, while assuring that the communication protocols between the objects among the whole manufacturing library will be preserved.

Traditional simulation tools (i.e. programming languages, simulation languages and simulators) used in the manufacturing domain do not allow this modular organization. Moreover, a trade-off has to be made in their selectionm between ease-of-use and flexibility. These tools are not provided with powerful design of experiment and results analysis capabilities. Much of the information generated during the simulation experiment is usually wasted, considering that decision-makers cannot cope with statistical analysis (or can, but only marginally), and continue to pay more attention to animation than statistics (Industrial Engineering, 1992). This is the main reason why they still lack confidence in simulation for decision support (Industrial Engineering, 1992; Keller, Harrel, and Leavy 1991). Attempts have been made to provide statistical analyzers (Mellicamp, and Park 1989) but these tools remain hard to use by non experts. More user-friendly tools therefore have to be developed, allowing simulation objectives and results to be accurately presented. The instruments embedded in our environment are designed to meet these requirements and facilitate the simulation analysis process.

With this assistance, manufacturing simulation may become used efficiently on a more regular basis and, as suggested above, for a wider range of studies.

## 3 CONTINUOUS SYSTEM MONITORING

The monitoring and control of manufacturing systems using simulation is a relatively new issue in the field of manufacturing simulation in comparison with design and scheduling. Most of the work in this area is oriented toward the control of FMS or automated cells (Bilberg, and Alting 1991; Mize et al. 1992; Tayanithi, Mannivan, and Banks 1992; Harmonosky 1990). The first aim of the COGNOSCO environment is to achieve continuous monitoring of manufacturing systems.

This presupposes that the system studied is adequately equipped with captors tracking and transmitting its evolution in real-time. Otherwise, only selective monitoring can be achieved (mostly when problems arise), but data collection in this case could be a too long process to allow for rapid and efficient reaction. The large amount of information to handle and its dynamism make traditional mathematical optimization methods cumbersome. Therefore, a simulation model seems the most appropriate way to treat the evolution of the manufacturing system as well as changes in the user's objectives.

Our environment is organized around the model that will be at the heart of the manufacturing information system (Figure 2). The objective is to allow

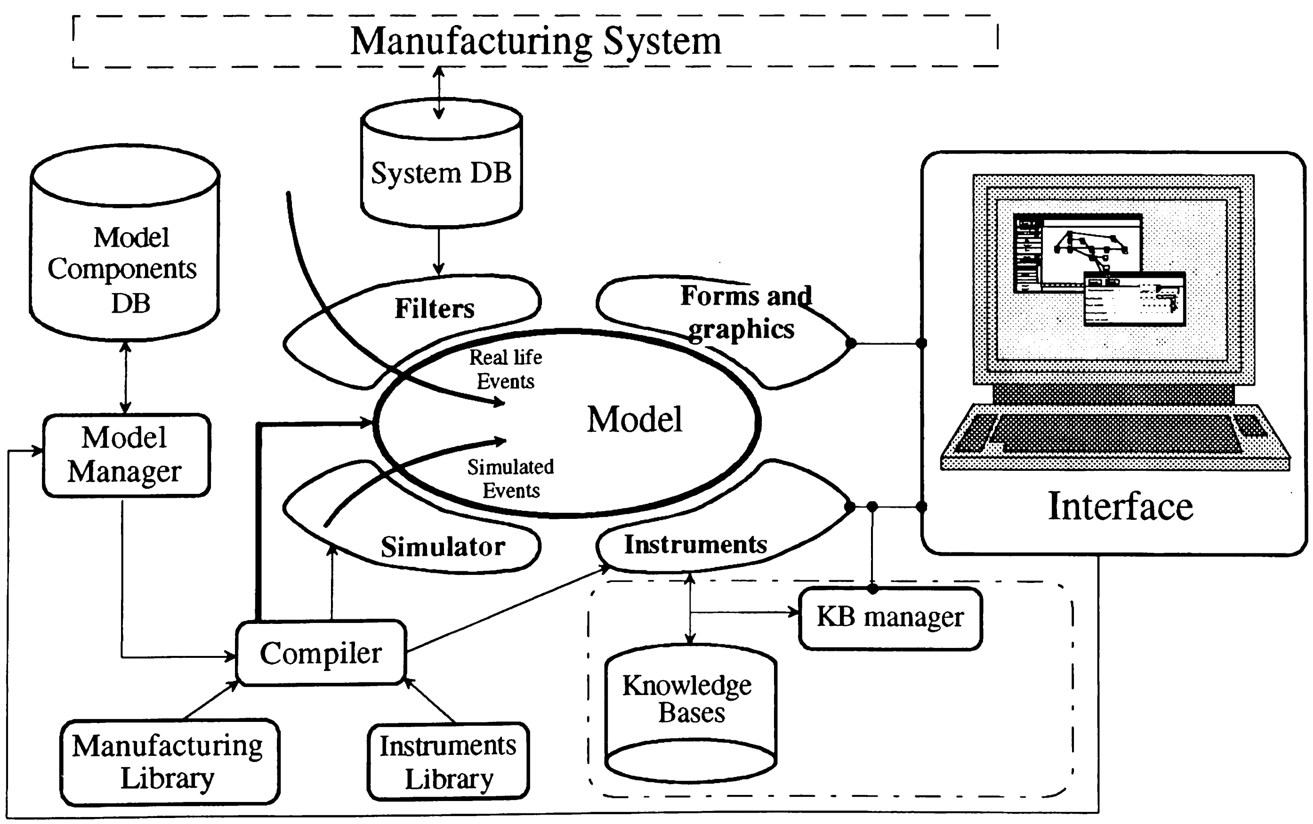

Figure 2: Organization of the COGNOSCO Components

manufacturing engineers to constantly monitor the production system, provided with gauges reflecting its behavior in real-time. Once the model has been developed and validated with the simulator, constant monitoring will be helpful for:

- training (from machine level, to factory level);
- diagnosis support;
- scheduling;
- costing and accounting, etc.

Simulation-based monitoring, with the support of the model, will help users to understand the dynamic behavior of the system and the interactions among its components. It appears to be a preliminary condition for achieving accurate real-time control.

This depends on the constant evolution of the model in accordance with the changes in the system. Once validated, the model is disconnected from the simulator engine and is constantly updated by events arriving from the system via the global database and its filters. This presupposes, as we pointed out earlier, that data can be collected throughout the factory, but does not imply that the system is fully automated.

The amount of information can rapidly become overwhelming, even with the use of filters to reduce and format this information. Simulation will be useful here as well. In fact, we believe that only a small number of indicators is relevant, given a set of objectives for the manufacturing system to fulfil. This is the way cars are piloted and the way nuclear power utilities and many continuous process systems are run. But, continuous processes rely on models using differential equations, and therefore, monitoring in this case is easy to achieve. This is not true for discrete-event manufacturing systems where no mathematical equation can be easily retrieved and relevant variables are harder to isolate.

Thus, simulation is used to determine the variables of interest before the model is connected to the manufacturing system. The composition of this set of pertinent variables is dynamic and can evolve. Assuming that the system is in a steady state, simulation will have to be performed at defined intervals to update this list. At the same time, however, the evolution of the system must still be monitored to avoid losing information (Harmonosky 1990). A copy of the model in its current state is issued for simulation purposes, while the model itself continues to monitor the system in the background.

In the monitoring mode, the important variables are constantly displayed on the screen and their dynamic behavior is studied by instruments. The evolution of the other variables is recorded by these instruments and does not appear constantly on the screen. This record is printed as a summary when the results are reported, unless the variables leave their authorized range of variation during the study. In this case, a warning is issued to the effect that the system is evolving outside its limits. This situation can be the consequence of a progressive divergence or due to an unexpected event.

When unexpected events, shocks to the system or changes in the objectives occur, simulation will also be necessary. Unexpected events can be classified into two categories (external and internal) and are the key issue to address in constant monitoring. External shocks, for example, are those generated by variations in the forecasted demand (e.g. cancellation of an order or new order with a high priority), in the material or in the manufacturing resources received (e.g. in time or in quality). Internal shocks are mostly the result of machine breakdowns and accidents. Tayanithi, Mannivan, and Banks (1992) propose a knowledge-based on-line simulation system to handle machine breakdowns and deal with this kind of internal shocks. The distinction between external and internal shocks is valuable at the various levels of abstraction, from the system level to the machine level. When a shock occurs, not only do the important variables have to be updated, but the objectives have to be redefined. Simulation is used to determine which strategy is the best suited for an "efficient" recovery as shown in Figures 3 and 4.

At this point, we introduce the knowledge bases (KB), a part of the COGNOSCO environment

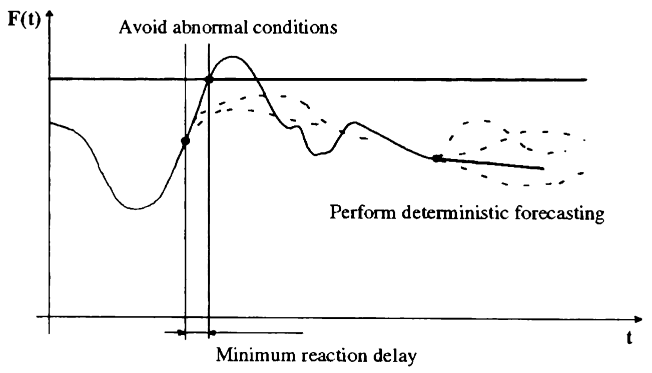

Figure 3: Continuous Monitoring Output (1)

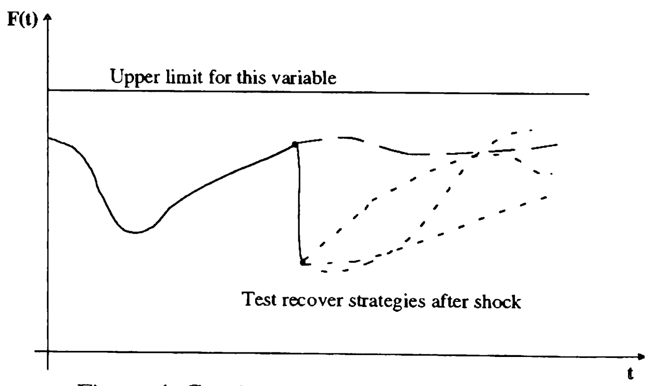

Figure 4: Continuous Monitoring Output (2)

providing expert assistance in decision-making in association with the instruments.

The KB will help the user in his efforts to "learn to know" the system, managing both instrument information and previous (accurate or validated) decisions. They have to be viewed as an expert advisor for the user as well as a tutor that will also learn from its pupil.

When it can be stated that a decision has failed to meet the required objectives and when a good decision can be retrieved (directly or by simulating candidate decisions), the user must modify the record of his previous choice and update the KB with a good set of parameters/decision. Because it is constantly growing, the knowledge will have to be maintained too. Inconsistencies must be avoided and higher-level knowledge is to be generated, aggregating existing knowledge. This leads to rule generation (i.e. move from factual to global or general knowledge of the manufacturing system) supervised by the user who is responsible for validating the candidate rule. The rules belong to two categories: static and transient. Those in the first category are time-independent and are usually system-independent (e.g. when a machine has broken down, make an attempt to reassign products or a rule to recognize bottlenecks). Those belonging to the second category (e.g. rules dealing with rerouting entities) are affected by the evolution of the system and have to be weighted according to their current pertinence.

These rules are used to recognize problems as well as to propose solutions. Some problems are easy to forecast (as in Figure 3, when the curve tends to reach the upper limit) and warnings can be issued before they occur, given the upper and/or lower limit for this variable. In this case, candidate strategies can be simulated and implemented before this limit is reached.

When an unavoidable problem arises (Figure 4), the user will be given a choice of possible decisions, the number of which depends on the quality and amount of knowledge included in the KB. In some cases, there will be no solution proposed, or, in other cases, only one pertinent solution will exist. The simulator will be helpful to test the candidates lying between these two extremes.

Problems that can affect the system are global, due to a conjunction of small local dysfunctions that are not really problems (i.e. many indicators are close to one of their limits), or local, with or without global consequences (chain reaction). The first case is the most difficult to treat, because no reason can be easily retrieved. The bounds on the variables are perhaps need to be revised or extended to a range, rather than limited to a single value. In the second case, self-owned, induced and structural problems (Rodde 1989) have to be recognized. Those in the first category are critical and must be solved. Those in the second reveal bottlenecks and the links existing between the various manufacturing components. The third category contains problems mostly related to external shocks or global events (e.g. end of shift).

OO manufacturing components and instruments permit the isolation of local self-owned or induced problems. Indeed, the local behavior of manufacturing components is continuously monitored, even though aggregate results are dispatched on screen. Thus, problems can be recognized as they arise, considering desegregated information from the lower-level instruments when a warning has been issued.

## 4 THE INSTRUMENTS

The instrument concept, that we are using in COGNOSCO and introduced in Fox et al. (1988), is quite simple. As in a real-life system, skilled instruments (e.g. clocks, counters and related products) are linked to the objects they study (an instrument can be linked to one or several objects). This is the first level of instrumentation. The highest level contains more sophisticated tools which will be able to provide the KB with treated information (i.e. expert or aggregated).

The Instrument class (Figure 5) is the root of this hierarchy. Its attributes offer basic support for the other instruments, from the simple counter to the expert analyzer. The `pamList` attribute contains the list of parameters the instrument needs to be operational. Once the instrument is embedded in a model, the `listeValeur` attribute is used to record the observations made by the instrument. Each instrument can be linked to a graphical output manager (`presentateurAssoc`) that formats and shows the results on the computer screen using graphs, histograms, pie charts, Kiviat graphs or simple tables (by default). The `collecteur` attribute eventually points towards another instrument. It can be an instrument of the same type, when the measures concerning all the objects of a given family must be displayed in a single indicator, or an instrument from a higher level, thus allowing multi-level measurement and aggregation of results (this characteristic is important for economic measures in a activity-based costing environment, as described below). Instruments are updated when the object(s) they are studying is (are) modified by an event. Every object that has to be studied is provided with a list of pointers to related instruments and sends them a message using the `metAJour()` method when its state is modified.

The `metAJour()` mechanism is blind in the sense

```
class Instrument: public virtual Chose {
  protected:
      bool          unique;                  //global instrument or not
      bool          imprim;                  //results shown on screen?
      float         lastImprim;              //date of last ostream update
      float         datAff;                  //time of first ostream update
      float         pasAff;                  //time between ostream updates
      OCltn         listeValeur;             //collection of measures
      OCltn         pamList;                 //collection of parameters
      Instrument    *collecteur;             //instrument of upper level
      XFiche        *presentateurAssoc;      //personal screen manager for
                                             //data output and format
  protected:
      virtual void          ajusteValeurs(bool affiche) {}
  public:
      Instrument();
      Instrument(MyString& nm, int typ, int ind, bool unq);
      Instrument(MyString& nm, int typ, int ind, bool unq, Instrument& in, XFiche& pr);
      virtual ~Instrument();
      virtual void          presenteResultat();                //format results to show them
      virtual void          affiche();                         //print measures on screen
      virtual void          affiche(ostream* ) {}              //print measures on O/stream
      virtual void          affiche(const char* ) {}           //save results on file
      virtual void          metAJour(Chose& obj);              //update measures
      virtual void          ajouteInfo(Instrument& inst);      //collect data from instruments
      virtual void          compileResultats();                //compile results
      virtual bool          verifieValidite(int &, Chose* = NULL) {}      //validate data
      virtual Instrument    *dupliqueInstrument();             //duplicate this instrument
      ...
};
```

Figure 5: Declaration of the Instrument Class

that there is no assurance that all the object's instruments have to be updated but, by this means, we achieve independence of manufacturing objects and instruments. The argument of this method is the object itself. The instruments are thus loosely tied to their related object(s), but, provided with the parameters of the `pamList` attribute, they are able to get the relevant information from among the object's parameters. This standard instrument pattern allows and encourages the multi-criteria evaluation of models. Evaluations of projects have been done according to operations, economics and ergonomics criteria (Fabre et al. 1992).

The lower-level instruments (Figure 1) have been completed and allow us to produce reports like those generated by traditional simulators and simulation languages. These instruments belong to the operations instrument family and constitute the basis for the other categories. The economics family of instruments (Table 1) has been designed as well. These cost measurement instruments, tied to simple objects, will be used to perform activity-based costing (ABC) measurement, simple cost monitoring and tailored pricing in accordance with system changes. McNair (1990) as well as Raffish, and Turney (1991) propose a general presentation of the ABC concepts. Both tangible and intangible (e.g. quality, flexibility) costs can be tracked and reported. This complies with the emerging trend of reforming traditional manufacturing accounting systems which became obsolete with the introduction of new technologies where indirect cost grows while labor cost diminishes. Abundant literature introduces new approaches to take into account this evolution (see for example: Arbel, and Seidman 1984; Azzone, and Bertele 1989; Park, and Son 1988; Suresh 1990; Kaplan 1990).

## 5 PLANNED DEVELOPMENTS

An evolutive environment (COGNOSCO) for the continuous monitoring of manufacturing systems has been introduced in this paper. Indeed, due to the modular OO structure of COGNOSCO, new modules can be easily integrated with the existing ones. New components are still under development. The most important among those are the specialized instruments that will be added to assist the user in the decision-making process. These instruments will also manage the knowledge bases dedicated to monitoring. Artificial intelligence and the cognitive sciences (connectionist networks) are among the avenues we are considering for developing these instruments.

## REFERENCES

Arbel, A., and A. Seidman. 1984. Performance evaluation of FMS. *IEEE Transactions on Systems, Man and Cybernetics* 14:118-129.

Azzone, G., and U. Bertele. 1989. Measuring the economic effectiveness of flexible automation: a new approach. *International Journal of Production*

Table 1: Costs Measured by Economic Instruments

| Costs | Load | Operator | M.H. System | Workstation | Workspace |
|---|---|---|---|---|---|
| Added value | x | | | | |
| Inventory cost | x | | | | |
| Scrap cost | x | | | | |
| Work cost | | x | x | x | x |
| Training cost | | x o | | | |
| Space cost | | | x | | x o |
| Queues cost | | x o | x | x | |
| Idle time cost | | x | x | x | |
| Setup cost | | | x o | x o | |
| Maintenance cost | | | x o | x o | x o |
| Breakdown cost | | x o | x o | x o | x o |
| Induced idle time cost | | x | x | x | |
| Stop cost | | | | | x |

x: relevant cost
o: opportunity cost

*Research* 27(5):735-746.

Bilberg, A., and L. Alting. 1991. When simulation takes control. *Journal of Manufacturing Systems* 10(3):179-193.

Bond, A. H., and B. Soaterman. 1988, Multiple abstraction in knowledge-based simulation. In *AI and Simulation: the Diversity of Applications*. ed. T. Henson, 61-66. Society for Computer Simulation, San Diego, California.

Fabre, M., and D. Leblanc. 1993. Towards an object-oriented simulation environment for manufacturing systems analysis, In *Proceedings of the 26th Annual Simulation Symposium*, 247-256. Institute of Electrical and Electronics Engineers, Washington, DC.

Fabre, M., J. C. Pitre, M. Levasseur, D. Leblanc, R. Gilbert, and M. Normandin. 1992. The decision of non-automation in manned workstations. Technical Report EPM/RT-92/36, Ecole Polytechnique de Montréal, Montréal, Québec, Canada.

Fabre, M. 1990. Une interface orientée objet pour la construction modulaire de modèles en simulation manufacturière. Master's thesis, Industrial Engineering Department, Ecole Polytechnique de Montréal, Montréal, Québec, Canada.

Fox, M. S., N. Husain, M. McRoberts, and Y. V. R. Reddy. 1989. Knowledge-based simulation: an artificial intelligence approach to system modeling and automating the simulation life cycle. In *AI, Simulation and Modeling*. ed. L. E. Widman, K. A. Loparo, and N. Nielsen, 447-486. New York: John Wiley & Sons.

Harmonosky, C. M. 1990. Implementation issues using simulation for real-time scheduling, control and monitoring. In *Proceedings of the 1990 Winter Simulation Conference*. ed. O. Balci, R. P. Sadowski, and R. E. Nance, 595-598. Society for Computer Simulation, New Orleans, Louisiana.

Industrial Engineering. 1992. Majority of IEs think graphics/animation "very important". 24(7):43.

Kaplan, R.S. 1990. New systems for measurement and control. *The Engineering Economist* 36(3):201-218

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

Lenz, J. E. 1985. FMS: what happens when you don't simulate. In *Proceedings of the 1st international conference on simulation in manufacturing*. ed. W. B. Heginbotham, 297-311. IFS, Stratford-upon-Avon, United Kingdom.

McNair, C. J. 1990. Interdependence and control: traditional vs. activity-based responsibility accounting. *Cost Management for the Manufacturing Industry*, Summer: 15-24.

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

Mize, J. H., H. C. Bhuskute, D. B. Pratt, and M. Kamath. 1992. Modeling of integrated manufacturing systems using an object-oriented approach. *IIE Transactions* 4(3):14-25.

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

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

Raffish, N., and P.B.B. Turney eds. 1991. Glossary of activity-based costing. *Cost Management for the Manufacturing Industry,* Fall: 53-63.

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

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

Tayanithi, P., S. Manivannan, and J. Banks. 1992. A knowledge-based simulation architecture to analyze interruptions in a Flexible Manufacturing System. *Journal of Manufacturing Systems* 11(3):195-213.

## AUTHOR BIOGRAPHIES

**MICHEL FABRE** is a PhD candidate in the Department of Industrial Engineering of Ecole Polytechnique de Montréal, Québec, Canada. His research interests include object-oriented simulation, artificial intelligence, information management, and manufacturing decision-support systems.

**DANIEL LEBLANC** is a Associate Professor in the Department of Industrial Engineering of Ecole Polytechnique de Montréal, Québec, Canada. His research interests include simulation as decision-support system for strategic real-time production and management decisions.