GSIM: A DISCRETE SIMULATOR FOR MANUFACTURING BASED ON ACTIVITIES AND SYSTEMS INTERACTION

M. J. Oliveros; C. Rudiez; F. Torres
Dpto. de Ingeniería de Diseño y Fabricación
C.P.S., Universidad de Zaragoza
C/ María de Luna, 3, 50015 Zaragoza
Spain
E-mail mjoliver@posta.unizar.es

C. Galé
Dpto. de Métodos Estadísticos
C.P.S., Universidad de Zaragoza
C/ María de Luna, 3, 50015 Zaragoza
Spain
E-mail: cgale@posta.unizar.es

KEYWORDS
Simulator, Discrete-event, Activity, System identification, Manufacturing.

ABSTRACT

In this paper we present a new methodology for modelling manufacturing discrete systems. The main contribution is to cleary differenciate the system and the activity that it performs. Such difference allows to create relationships between them, wich are essential for the model formulation. Through the proposed methodology we aim to developed an easy to use simulation tool for modelling the most usual manufacturing systems. GSIM is a Windows based system with an intuitive graphical interface. Moreover, all operations are carried out without any programming. Users introduce the required information form modelling using the menus and dialog boxes. We consider that the simulator proposed combines the advantages of a general purpose simulation language and a data-driven simulator.

INTRODUCTION

Simulation modelling tools are powerful instruments that allow analysis and evaluation of Manufacturing Systems by helping in both the identification of opportunities and the making of decisions. Two problems are traditionally found in the construction of simulation models (Law and McComas 1992). First, we have the necessary previous training required to use simulation software. The second problem consists on the difficulty to make changes in the model when it has already been created.
This new tool tries essentially to solve both problems. In order to do that, the making process has been turned upside down: before programming any simulation machine, it's important to recognise all the elements that are necessary in the model implementation, and how are they going to be used from a purely manufacturing elements systematisation perspective. For this reason, the planned structure of this new software is based on the following idea: any model consists on systems that are able to perform activities, and these activities are executed over the parts. At the same time, systems are interrelated by connection rules, which represent the way parts flow within the global system.
In section 2 the conceptual model is described and its elements are analyzed in section 3. After the different approaches to time advance, we propose a new approach in section 4 wich is implemented in the manufacturing simulator in section 5. Finally, in section 6 the conclusions are considered.

DESCRIPTION OF CONCEPTUAL MODEL

The structure of the developed conceptual model consists of a group of **systems**, so called agent systems or permanent systems, which performS activities that are executed on parts, traditionally called entities [Banks et al. 2000]. Moreover, thes systems interact each other by **connection rules**, which represent the materials flow in the manufacturing facilities. That is why we call the parts patient or temporal systems, making a clear distinction between temporal and permanent by the materials flow and behaviour concepts.
In our model, all the systems always perform an **activity**, even when they are not working, they are performing a stopped activity. The system **state** evolves through time because the activity changes that withstand. As we speak about the state of agent systems, we will also speak about the state of patient systems. In both cases, the value of the state variables of each element will give us the state **profile**.

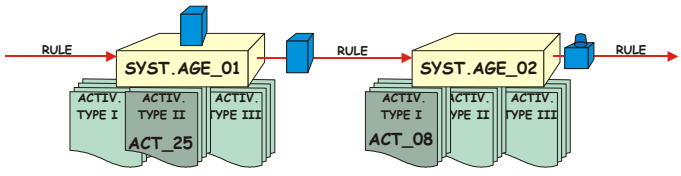


Figure 1: Description of Conceptual Model

To solve this matter, all the possible systems have been studied and also the activities that they can execute have been catalogued, in order to establish non-univocal relationships from these two kinds of elements. So, the elements of the system type of our model are independent of the elements of the activity type, making possible that a

system can execute several activities of the same or different type, and one activity can be executed by several elements

The last characteristic element of an event system that remains to be defined is the **event** or **instantaneous activity** term. Gathering the classical event definition as the instantaneous occurrence that may change the state of the system, in the formalism that follows four types of unconditional events appeared (Oliveros 2002) and other conditioned to that ones. The unconditional ones cause that systems begin performing activities and the state of the systems are changed because of the transitions generated accordingly.

ELEMENTS OF CONCEPTUAL MODEL

Systems

The first and more important step is to define the elements that a simulation model needs to work with. By definition, a model is always an abstraction at some level, preserving the essential aspects of the system and leaving out the rest. So, the semantic of the systems in the discrete even simulation model introduced in this work, is oriented to the **productive behaviour of the main elements of a manufacturing plant**, specially in the mechanical industry. Thus, it will not be studied elements or behaviours of continuous systems or management systems.

The main criteria to classify the systems is the property that each agent system modifies the parts. Because of the initial and primordial objective in this kind of processes is to obtain final products from raw material using manufacturing resources that perform several tasks or activities.

The final result is that the global system, the manufacturing plant, is composed of subsystems, called from this point systems. According to their material flow, these are classified in **operands** and **agents**.

Operand system

It is the one that may flow through the permanent systems of the system as it appears and disappears of the model, it's also called temporal system. In our model there is only one system of this type.

Part. System that moves around the model, being liable to be modified by the rest of the elements. This modifications will be reflected in the values that of its attributes. It is a key element in the model evolution, because it is the basic connection between the systems, and it has a special relationship with activities because its presence will determine if the system performs one or another activity. This elements have several attributes that characterize them and depending on the attribute modified the activity in progress will be one or another.

Agent system

It is a permanent system that is able to perform activities or that is necessary for the activities to be done. In our model we classify the permanent systems as follows:

Buffer. It modifies the time attribute of the a part.

Machine. It modifies the structure attribute of a part.

Transport elements. They modify the position attribute of a part. There are two different kind of transport elements attending to their functionality:

Conveyor. It is a fixed loading and unloading point.

Vehicle/*Track*: It is a variable loading and unloading point. The task of these systems is to connect several manufacturing systems in a manufacturing unit.

Labour. System needed to perform an activity together with other elements. In addition, it is able to perform activities by itself.

Parts generator. It is a system that creates all the parts that go into the model from outside. For instance, the external supply of raw material for production.

Parts exit. When all the operations are finished parts have to leave the model. This system is defined as a kind of outgoing buffers and let us modelize the exit of parts. It could be thought as a part sink.

Activities

An activity represents, and so it is modeled, a period of time during systems perform an specific task. The beginning and ending of all activities comes from an event.

All systems are always making an activity that belongs to one of the following categories:

Type I: waiting.

Type II: operating.

Type III: being stuck.

The type II activities of the permanent systems can be classified bearing in mind the usual tasks of a manufacturing process:

Action activities made by the agents over the operands

Processing.

Moving by conveyors and vehicles.

Placing (loading / unloading).

Storing parts.

Testing parts (quality control) by machines.

Interruption activities performed by the agents over themselves.

Reparing breakdowns (broken down).

Carrying out (set up).

Moreover, it is possible to establish non univocal relationships between agents or systems and the activities that they can perform. In Table 1 can be seen that not all the systems may perform activities by themselves, and that there are activities like maintenance and breakdowns that are made over the systems and do not need parts being present. Finally, there are also activities performed by systems like processing, changing position, and so on.

Table 1: Relationships between Systems and Activities

ACTIVITIES \ SYSTEMS	BUFFERS	MACHINES	VEHICLES	CONVEYORS
PROCESSING		✸		
TESTING		✸		
STORING	✸	✸		
MOVING			✸	✸
PLACING		✸	✸	
CARRYING OUT		✸	✸	✸
REPARING		✸	✸	✸

The most important fact shown in Table 1 is that **the elements of type system are independent from the elements of type activity**. Because of that a system may perform several activities and an activity can be performed by several systems being the effects of the same activity in each system quite different. For example, the change of position activity can be made by several conveyors and/or vehicles with clear difference in the activity effects and its characteristics.

So, in order to create and select activities the natural process follows these rules (see figure 2):

Checking the type of activities that the system is able to perform.

Defining parameters for each activity in the selected types of activities.

Selecting the particular activities defined that can be made by each system.

Choosing between the activities selected the ones that are being executed in the model.

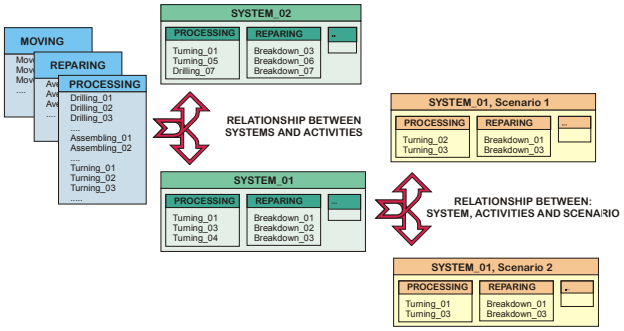


Figure 2: Selection of Activities

Once created and selected the activities, when it is possible to execute more than one, the order of execution is established by the usual manufacturing programming methods. All the priority rules considered have been described in (Ochoa and Arana 1997; Royo 2001). In particular, the cycle time shorter (SPT) or larger (LPT), FIFO, random and planner priority based on some attribute are the most extensivily used in this context.

Connection Rules

A system is a group of sections related by its operations. In order to simulate these relationships is necessary to define the connection rules that control the flow of parts between the systems. These rules are specified in the permanent systems.

When selecting a rule to characterize a material flow, it is convenient to bear in mind not only the manufacturing process but also the transport process. For example, dealing with lots of products, we have to distinguish between the manufacturing and the transporting lots. The connection rules implemented takes into account this difference. So, these rules describe how the parts flow between the systems and how the vehicles deal with the requests of the manufacturing elements. In addition, the type and attributes of the part also determine its flow.

The role played by the vehicles in the proposed model is different from the other models in the literature. The vehicles are defined as systems that perform a pseudoactivity called **connect**. Because of that these systems can only be executed to carry parts from one place to another.

Events and state transitions

Each particular element is used like an isolated entity that communicates itself with the others by detecting events an answering to them. Events represent the stimulus that a system can detect and that cause a state change. So, each fact that can have an effect on the system is characterized as an event.

The four unconditional events defined in this model are:

Part generation. When a part is generated lively by the generator element, its input to the model is characterize as an event. The generated parts go to a physical store. If there is not enough space, remains in the generator until some space is found. The input of parts to the system cause that elements which are waiting can take them and initialise their cycle.

Activity end. In the proposed model, the begining of the activity to be perfomed is an conditional event, because it depends on the temporal elements present in that moment in the system. However, once an activity is being performed the activity end is an unconditional event. This event has different meanings depending on the activity being executed in that particular system. For instance, in a store occurs when the maximum store time is reached and in a machine could be the end of a process or a loading or an unloading or a quality control and so on.

Set-up and breakdown. These events cause the stop of those activities whichever there were. Afterwards, some of the interruption activities will begin. Due to the random behaviour related to these events, the beginning and the end of these events are unconditional.

For instance, the diagram of transitions related to the Set-up and breakdown event is shown in the Figure 3.

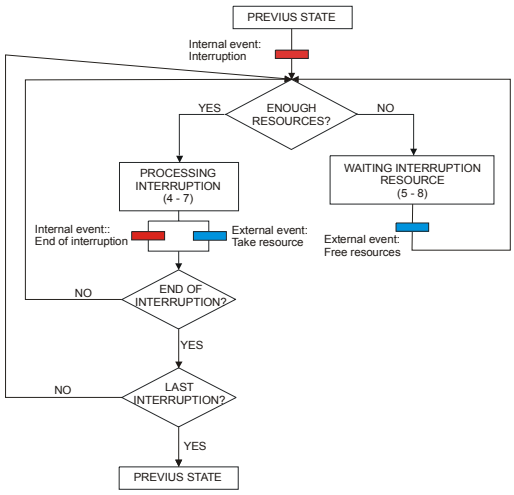


Figure 3: Diagram of Transitions related to a machine

TIME ADVANCE: PROPOSED SOLUTION

Time advance

Due to difficulties to choosing the time interval length properly, it is better to use an asynchronous algorithm, in which the time interval length is variable. In the proposed algorithm, the state of the model is examined and updated only when an event cause a state change, that is, each time an event occurs.

Next-event

The revision of the classical approaches to the problem (Banks et al. 2000; Fishman 2001; Garcia 1990; Law and Kelton 2000; McCormack and Sargent) let us establish the following classification:

Event Scheduling (ES) is based on an unconditional event sequence through time, being the key element an event list ordered by occurrence time and the priority of the generator elements. When an event is read the related event routine is executed, so it is necessary to implement routines for each of the unconditional type of events that appears in the system.

Activity Scanning (AS) is based on the activities list of a model and those conditions, simple or complex, that allow an activity to begin. At each clock advance, the conditions for each activity are checked and if the conditions are true, then the corresponding activity begins.

In the three-phases approach, events are considered to be activities of duration zero time units, and the total activities (events + activities) are divided en two categories: unconditional and conditional.

Process Interaction (PI) approach defines the simulation model in terms of process. A process is a time-sequence list of events, activities and delays, including demands for resources, that define the life cycle of one entity as it moves through a system.

The approach proposed in the model is based on the following lists:

Future event list (FEL) like the ES aproach.

A dinamic list of systems that execute activities (DSL). Conceptually DSL contains all the permanent systems in the model that performs activities, because of all the systems are always executing an activity. This list sustitutes the usual Current Event List (CEL).

DSL is implemented as a pointer list to this type of systems in the model, ordered by decreasing priority. That means that the number of pointers are the same as the number of systems in the model that could change their state and so the list length is finite and small.

Another important difference with classical strategies is the control of evolution of the model. This control is performed by the systems pointed by the DSL list and not through parts, in order to avoid the duplication of information produced when you have to store it on each part, specifying the activity in progress and the new possible ones.

As we said before, the DSL is sorted by decreasing priority. It is possible that more than one system have the same priority. The user can define priority groups, so all the elements in a group have the same priority and the order of revision in each group is the one defined by the user. Note that, depending on which system is the first revised the system evolution can be different.

The proposed strategy modifies the three phases approach, analyzing the systems that can evolve at the moment to different states instead of the activities that can be performed which are fixed.

In Figure 4 this strategy is represented by a block diagram and the pseudocode with a description of the working logic is presented is described in Figure 5.

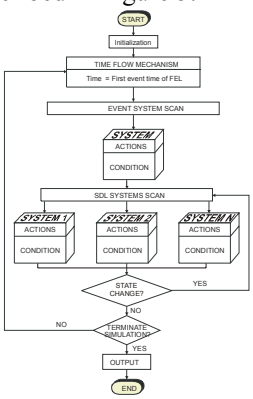


Figure 4: Block Diagram of the Proposed Strategy

ALGORITM FOR THE PROPOUND SOLUTION

```
Perfom initialization (set beginning_time, ending_time, file
initial events into FEL and permanet systems into DSL,
initialize component state descriptors);
BEGIN
  To annotate in FEL the first events;
   (The event notice consists of:  event_time, event_type,
                                   element)
  Recover and clear the event with the first occurrence in
  FEL;
  TIME := first_event_time;
  Revision_number_DSL = 1;
  Operand = first_event_element;
  While (TIME <= ending_time) do
    If T[operando] < TIME and cond_operand_routine=true
        execute activity_operand_routine;
    End_If;
    Repeat
      change=false
      For j:= highest_priority to lowest_priority
        i:= index to system with priority j;
        If T[i] < TIME and cond_i_routine = true
          then
            execute_activity_i_routine;
            If change_state_i = true
              then
                change= true;
            End_If;
        End_If;
      End_For;
    Until change=false
    Recover and clear the event with the first occurrence in
    FEL;
    TIME :=  next_event_time;
  End_While
END
```

Note:
- Each procedure cond_i_routine evaluates the specific conditions to change the state of the permanent system i, and returns as a result true or false.
- Each procedure activity_i_routine models the state transition of the element and insert, if there are any, the unconditional events in FEL.

Figure 5: Pseudocode Algorithm to Model

GSIM: A MANUFACTURING SIMULATOR

After planning the model, a simulator tool based on it has been programmed, called GSIM, using a general purpose programming enviroment: C++ Builder v5.0 (see Figure 6). Model development is completely graphical and object-oriented. To the greatest extent possible, all inputs are provided graphically with information being grouped by object type and presented in a "spreadsheet-like" format for quick and intuitive access. For example, when the modeler defines a machine he can also define the machine's icon, type, intrinsic parameters, input and output rules, activities and so on.

In the Figure 6, the logic order in the elements definition is presented. Firstly, the systems (intrinsic characteristics an representation), secondly the activities, and finally, the relationships between this two elements.

In order to validate and verify the model and the simulation tool, we followed the steps proposed by (Barceló 1996) with several problems proposed by commercial software and training problems in learning simulation tools. Following, we describe the steps:

First step. The program is validated from the modelling point of view, that is, if diverse systems can be modelled, and the complexity of these models and their logic.

Second step. Once the model is defined, each type of event, the states of the systems and actions that cause, is checked to see if the evolution is correct.

Third step. The last step is revision of the results. This is made getting the value of the different indicators planned in the GSIM models and comparing them with the ones providing by a commercial program: Witness®.

Once the program has been validated, we obtain important conclusions, specially in the Witness comparation. In some examples we detect differences, mainly with maintenance and breakdown activities. These differences exist as the time between two consecutive activities is different in both cases:

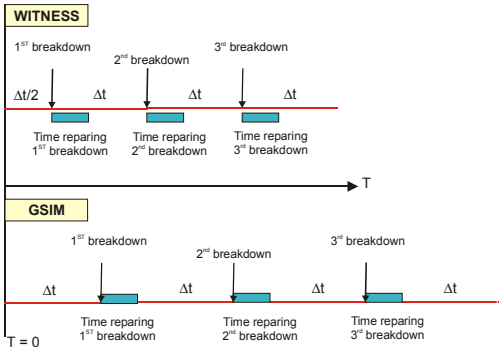
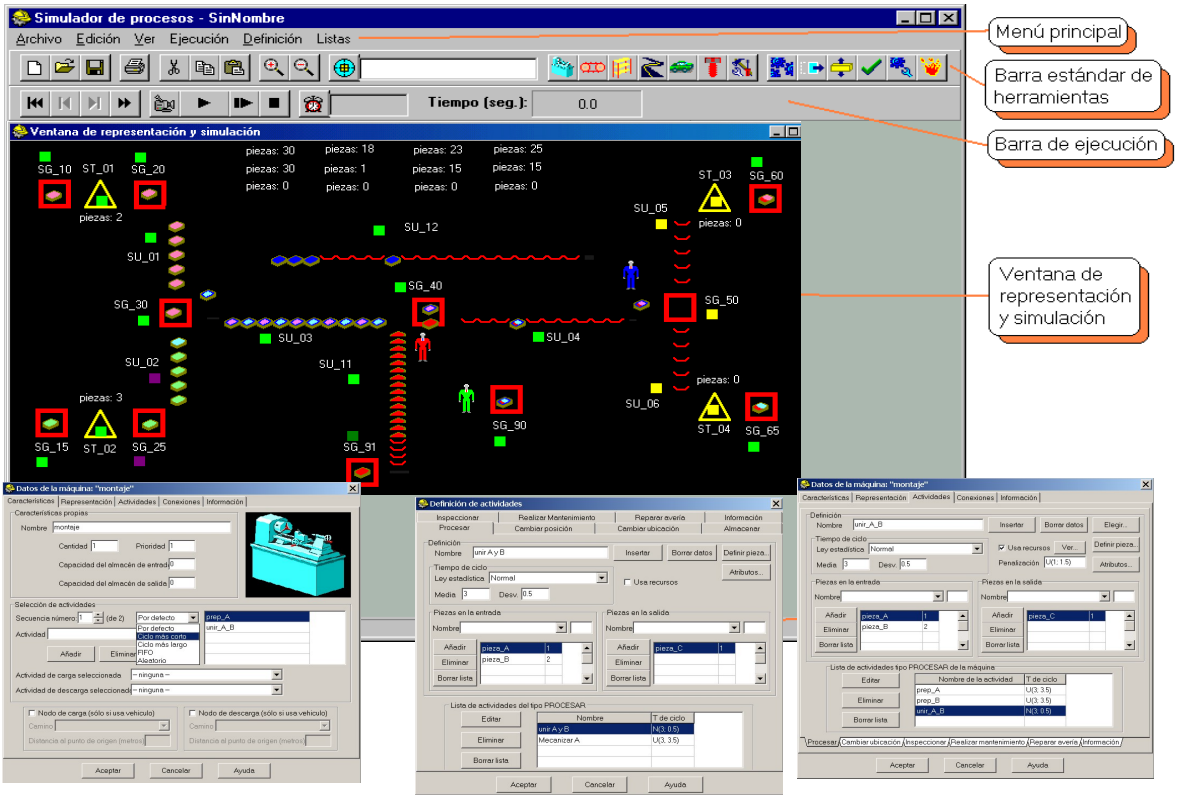


Figure 7: Differences between Witness and GSIM

Figure 6: GSIM Environment. Forms to Data Input



For the sake of brevity, we have not presented others results or discussion about the validity of the simulator that can be found in (Oliveros 2002). In the thesis the simulator validity is verified by obtaining a good level of confidence in the representativeness of the simulation model against real-life situations.

CONCLUSIONS

In this paper, a new methodology is proposed with the aim of representing the productive behaviour of the elements in an industrial installation. This model put special emphasis on the difference between system and activity, and allow the systems to execute different tasks, as it occurs in real situations.

In order to take into account the difference between system and activity, we have to designed a new programming strategy. We consider that it is optimum among the basic strategies on discrete events simulation programming examined.

Finally, the new methodology and the proposed approaches have been implemented in a simultation tool easy to be used by inexperienced operators. Following a validation method described in the literature, it has been demonstrated that the performance of the simulator GSIM was the expected.

REFERENCES

Balci, O. 1997. "Verification, Validation and Accreditation of Simulation Models". *Proceedings of the 1997 Winter Simulation Conference*, 135-141.

Banks, J.; J.S. Carson; B.L. Nelson; and D.M. Nicol, 2000. *Discrete-Event System Simulation, 3rd Edition*. Prentice Hall, New Jersey.

Banks, J.; E. Aviles; J.R. McLaughlin; and R.C. Yuan. 1991. "The simulator: new member of the simulation family". *Interfaces 21 (2)*. pp 76–86.

Barceló, J. 1996. *Simulación de sistemas discretos*. Isdefe, Madrid.

Fishman, G.S. 2001. *Discrete-Event Simulation. Modeling, Programming and Analysis*. Springer.

García, M.R. 1990. "Discrete Event Simulation Methodologies and Formalisms". *Proceedings of the Winter Simulation Conference, 1990*.

Law, A.M.; and W.D. Kelton, 2000. *Simulation modeling & Analysis. 3rd Edition*. McGraw Hill.

Law, A.M.; and McComas, M.G. 1992. "How to select simulation software for manufacturing applications". *Industrial Engineering 24* (7), 29–35.

McCormack, W.M,; and R.G. Sargent, "Analysis of future event set algorithms for discrete event simulation". *ACM, 24*, 801-812.

Ochoa, C.; and P. Arana. 1997. *Gestión de la producción.* Editorial Donostiarra, San Sebastián.

Oliveros, M.J. 2002: *Simulación de instalaciones productivas de flujo discreto mediante un modelo de categorías universales.* PhD thesis. C.P.S., University of Zaragoza, Spain.

Oliveros, M.J.; Rudiez, C.; and Torres, F. 2001: "GSIM: Un simulador basado en la interacción de actividades y sistemas." *Workshop en Metodología de Modelado y Simulación de Sistemas. U.A.B.*

Pidd, M. 1992: *Computer Simulation in Management Science. 3rd Edition*. John Wiley.

Robinson, S. 2002. General concepts of quality for discrete-events simulation. *European Journal of Operational Reseach. Vol. 138*, 103-117.

Royo, J. 2001: *Optimización de la planificación de recursos y programación de operaciones según listas de selección*. PhD thesis. C.P.S., University of Zaragoza.

Zeigler, B.P. 1990: *Object-oriented simulation with hierarchical modular models*. Academic Press.