AN INTERACTIVE SIMULATION DESCRIPTION INTERPRETER

BY

ROBERT M. O'KEEFE

MARCH 1986

TR-86-5

D.2.6 PROGRAMMING ENVIRONMENTS
I.6.2 SIMULATION LANGUAGES

AN INTERACTIVE SIMULATION DESCRIPTION INTERPRETER

Robert M. O'Keefe *
Department of Computer Science,
Virginia Polytechnic Institute and State University,
Blacksburg, Virginia 24061, U.S.A.

May 27, 1986

Scope and Purpose - One aim of recent research in discrete event simulation is the construction of software tools that make simulation more accessible to those without extensive programming skills. This paper discusses a software tool called Inter_SIM which provides for interactive development of, execution, and experimentation with discrete event simulations. The purpose of this paper is to present Inter_SIM via a simple example, and discuss the strengths and weaknesses of this approach to model development.

Abstract - The development of discrete event simulation programs has been aided in recent years by the proliferation of simulation programming languages and the introduction of software development aids, for instance, program generators. Inter_SIM, an interactive tool for developing and running discrete simulation models by interpreting a descriptive language, is presented. A running simulation, when halted, remains suspended. After extension, alteration, or investigation of the description, the simulation runs from the suspended state. Thus the user views the simulation model as a running model, rather than a piece of static text. The strengths and weaknesses of such an approach are discussed; other methods of interactive development are reviewed.

*Bob O'Keefe is Visiting Assistant Professor of Computer Science at Virginia Tech, on leave from the Board of Studies in Management Science at the University of Kent at Canterbury, England. He received his Ph.D. in Operational Research from the University of Southampton, England, in 1984.

INTRODUCTION

Discrete event simulation was an early application area for specialized programming languages. GPSS, SIMSCRIPT and GSP were three of the first generation of high-level languages [1]. Many other languages have since been developed: whereas true compilers can now be obtained for GPSS and SIMSCRIPT, most others are simply a collection of routines for use with a general purpose language [2,3,4], or are pre-processed into a general purpose language [5].

Simulation Programming Languages (SPLs) provide a world view in which to structure a simulation. The programmer describes a model in terms of modules (normally activities, events or processes) and activates them either implicitly or explicitly through an executive consisting of a time advance mechanism and a next event set. Further, a good SPL should provide facilities that aid the development and verification of the program, and validation of and experimentation with the model.

In addition to the considerable development of SPLs, there are a number of software systems that free the user from much of the programming effort. There have been three major approaches - program generators, analysis programs, and descriptive languages. Program generators ask the user a series of questions about their model, and then produce a program in an SPL. Examples include Clementson's CAPS [6], which produces programs in ECSL, and Mathewson's DRAFT [7], which can produce programs in a variety of languages. With analysis programs, information representing a simulation is provided as input data to a generic simulation model. Thus the simulation is constructed as a collection of data, rather than a number of programming statements. Examples include Pritsker's Q-GERT [8] and Robert's INS [9]. A descriptive language, such as the Model Description Language of Overstreet and Nance [10], is a very high level description of a simulation, which is then manually converted into an SPL program, used to generate an SPL program, or compiled directly to a running simulation. Descriptive languages are typically compact and easy to learn.

Visual Interactive Simulation

A comparitively recent development is the provision of pictorial or graphical output, where the simulation drives a continuous visual display representing the system under consideration. This is a considerable aid to verification, validation and experimentation. Perhaps of greater practical importance, pictorial colour displays helps facilitate understanding of the behaviour of the simulation model to the end user or decision maker.

User interaction with the running model allows the user to observe the effects of experiments portrayed on the screen. More importantly, the user can actually be incorporated into the model. Decisions which are too difficult to encapsulate in the model can be referred to the user at run time as they occur. When modelling systems in which a decision maker plays an important role, this can be represented by allowing the decision maker the same interaction as with the real system. This is called Visual Interactive Simulation (VIS), and owes much to the seminal work of Hurrion [11,12]. Its application is well documented (for a review, and references to a number of applications, see Bell [13]).

THE DESCRIPTION INTERPRETER APPROACH

Extensive use of VIS has highlighted the need to be able to construct simulations interactively. When a VIS is developed in a compiled language, the developer must build in facilities for the interaction. Desired interactions required by the user but not anticipated by the developer can not be met without recourse to altering and re-compiling the program text.

The tool presented here, called Inter_SIM, is an attempt to construct a tool that provides for interactive development. Inter_SIM is a descriptive language with the distinctive feature that at any time the description can be executed. It is interpreted - there is no need for any intermediate compilation or processing. The running model can be halted; it remains suspended in state and time until set running again.

Inter_SIM is activity orientated. Entities generated by a source proceed through a sequence of activities. At each activity they may be served by resources and other entities, and must queue at an activity if servers are not immediately available. Resources are passive entities that can only serve. Resources and entities are grouped into bins and classes respectively; a user describes a bin of one or more resources, a class of one of more entities.

Inter_SIM lacks a mechanism for user defined entity attributes. At the outset, it was decided to leave this out of early versions of Inter_SIM so as to make implementation easier (altering scalar attributes would require interpreting arithmetic expressions). However, entities can reside in sets, and set membership can be perceived as a boolean attribute.

The language is non-procedural. A description is composed of instances of four available language components, namely bin, class, activity and source. The details of each language component can be found in table 1. A model description is developed by adding new instances of a component, or altering the details of existing components.

	Component	Definition of Component Part
	Bin	
*	Name	name of the bin
*	Number	number of resources in bin
	Histogram Utilization	histogram for resource utilization
	Show	number of resources busy entered in picture
	Class	
*	Name	name of class
*	Letter	letter to represent entities in picture
*	Colour	colour used for entities in picture
	Show	entire class entered in picture
	Individual Histograms	each entity individually entered in picture

```
    Utilization          entity utilization
    Time                 time that transient entities are in system
    Set                  number of entities in a set
*   Number               number of entities in class
    Sets                 sets associated with class, each with colour that
                         entities in that set take.

    Activity

*   Name                 name of activity
*   Next                 next activity that an entity is routed to
*   Priority             queue priority
    Balk                 maximum size of queue
      Next                 next activity for balked entities
                         co-operating resources and entities :-
    Obtain                 acquired by engaged entity but not freed after activity
    Advance                acquired by engaged entity and freed after activity
    Release                freed by entity prior to engaging in the activity
*   Duration             duration of activity
    Show                 queue for activity entered in picture
    Histograms           histograms for
      Time                  times that entity take to reach activity from source
      Queue length         queue length at activity

    Source

*   Name                 name of source
*   Class                class which generated entities will join
    Activity             first activity they will queue for
*   Distribution         inter-arrival distribution
```

Table 1: The language components.

(An asterix indicates a part of a language component description that must be present for each instance of the component; all other parts are optional.)

An Example

To introduce the system and its use, a simple example model is presented. It is a model of a bank, where arriving customers either queue for an auto-bank, for one of a number of tills, or for an enquiry desk. Following service at the enquiry desk, they may decide to queue for a till or use the auto-bank. The user of Inter_SIM would typically start by drawing an activity diagram - an activity flow diagram for the example is shown in figure 1. This would be followed by identifying the necessary components, and itemizing many of the details of each component. Here the necessary components are

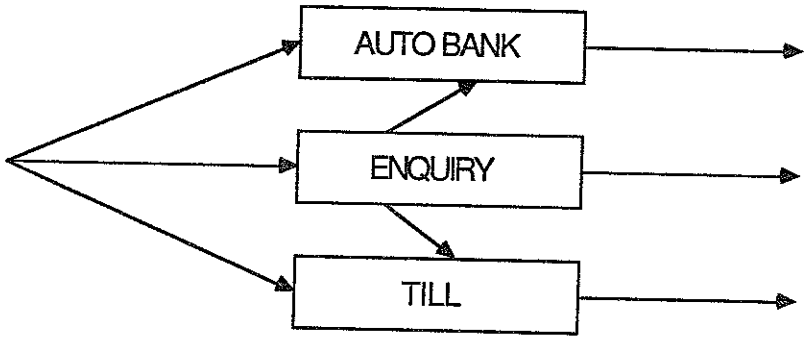

Figure 1: Customer flow through the bank.

```
Bin -       AUTO       (automatic bank)
            CASHIER    (cashiers)
            DESK       (enquiry desks)

Class -     CUSTOMER   (customers)

Activity -  ENQUIRE    (service at an enquiry desk)
            TILL       (service by a cashier)
            AUTO       (service at the auto-bank)

Source -    ARRIVAL    (arrival at the bank)
```

(The names of each component are assigned by the user.) The user is then ready to build the model. When using Inter_SIM, typically a small number of activities are described and tested, and then other activities are added until the model is complete. However, with a small model as here, the entire model would probably be described immediately.

When Inter_SIM is entered, the user is faced with a console display as shown in figure 2. A model must be built bottom up, in that it is not possible to make a reference to an instance of a component that does not exist. Thus descriptions of bins and classes are normally entered prior to any activities, and sources can only be described when the classes and activities they feed have been described. Hence the user would start by adding the bins - figure 3 shows the user adding the bin DESK.

To add the CUSTOMER class, the user would go to the class menu, and complete a description

as shown in figure 4. The decriptions for activities and sources are added last - ENQUIRE, when retrieved with the menu option "Which", would appear as in figure 5.

Printed descriptions for the entire simulation are shown in figure 6. In addition to using two sources, a dummy activity called ARRIVE (an activity with zero time duration that does not parallel an activity in the real system) has been used to route arriving customers by probability.

If the simulation is now run, it will immediately execute until any key is pressed, or a time for the end of run is reached. The console will display the picture which changes with each event. An example is shown in figure 7.

```
BIN
===

AUTO            1  1,35,7    :BLUE
CASHIER         4  1,35,13   :BLUE
DESK            2  1,35,17   :BLUE

CLASS
=====

Class     CUSTOMER
Letter    c
Colour    :GREEN
Histograms
Time           0.50     1.00
Sets      ENQUIRED:BLUE

SOURCE
======

ARRIVAL    CUSTOMER    ARRIVE       Neg Exp  (10)     0.30

ACTIVITY
========

Activity          ARRIVE     FIFO
Next ( 7)         TILL      {800}
                  ENQUIRE   {850}
                  AUTO      {1000}
Duration          Constant      0.00

Activity          ENQUIRE    FIFO
Next ( 6)         TILL      {800}
                    Gain<ENQUIRED>
                  AUTO      {850}
```

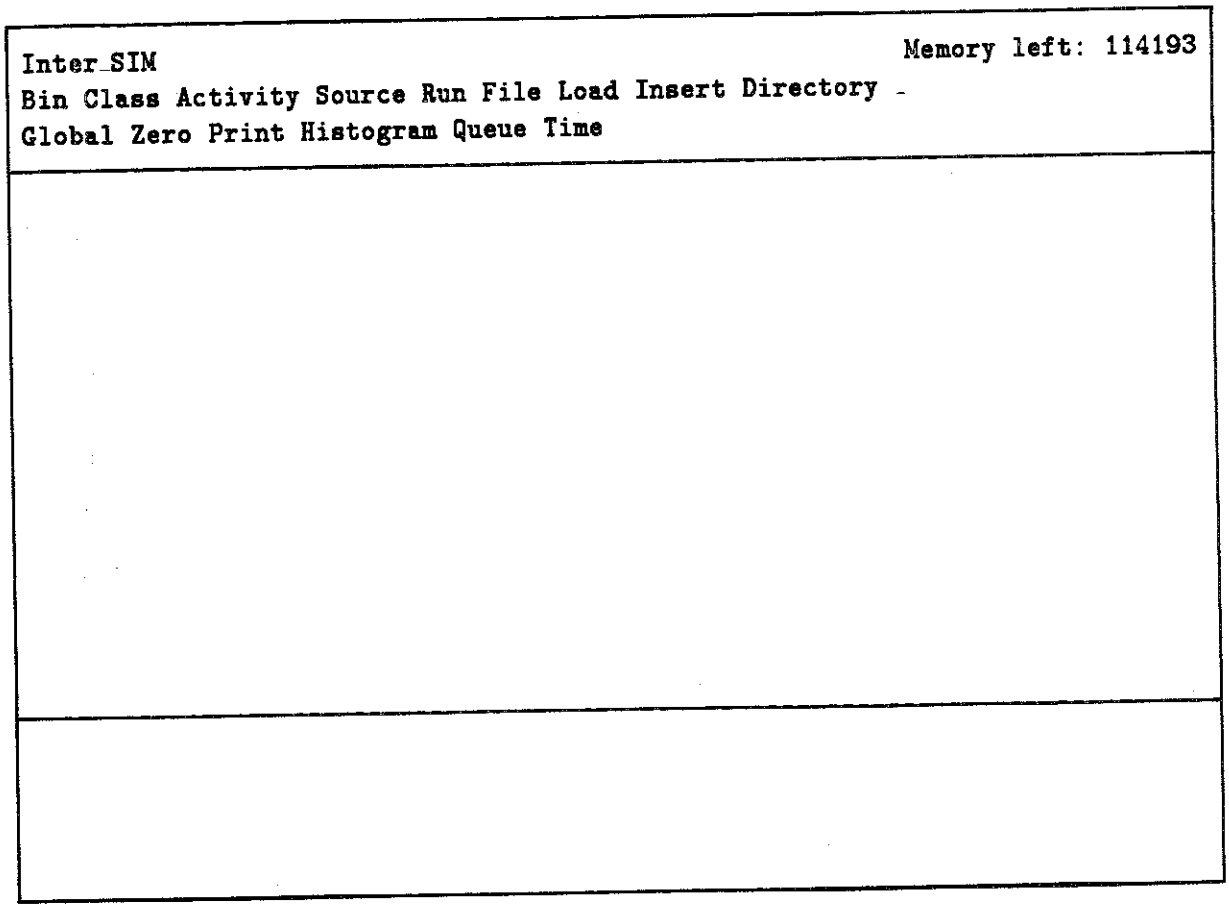

Figure 2: The opening console display.

(The screen is divided into three windows. The top window shows the available menu options, the middle window is used for entering and showing components, the bottom window is used for error messages and all other interaction with the user.)

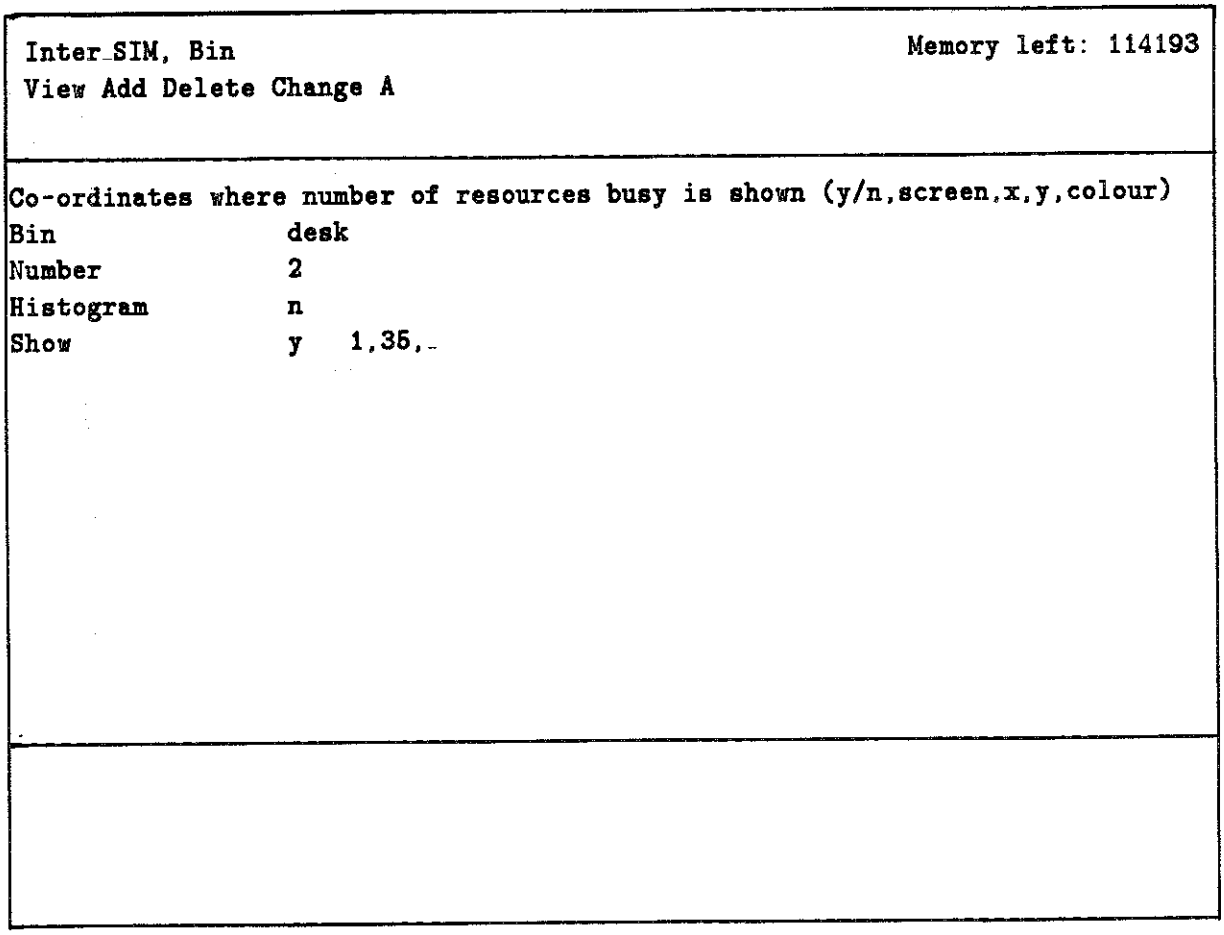

Figure 3: Entering a bin.

(The user is completing the description of DESK by specifying screen co- ordinates where the number of busy desks will be shown. The first number is a logical screen. The help line, which is the top line of the middle window, specifies the responses required from the user.)

```
Inter_SIM, Class                                          Memory left: 114165
View Which Next Add Delete Change A

Class              customer
Letter             c
Colour             :green
Show               n
Individual         n
Histograms
Time               y    0.50 1.00
Utilisation        n
Set                n
Number             0
Sets               enquired     :blue
```

Figure 4: A completed description for the CUSTOMER class.

(Entities from this class will be represented by the letter 'c' in the picture, and will appear green unless they have membership of the set ENQUIRED, in which case they will appear blue. Note that a colon is always a prompt to provide a colour. A histogram for the times that entities from the class remain in the system has been specified with a cell base of 5.0 and cell width of 2.0. When a histogram is part of a component description, automatic statistics collection takes place.)

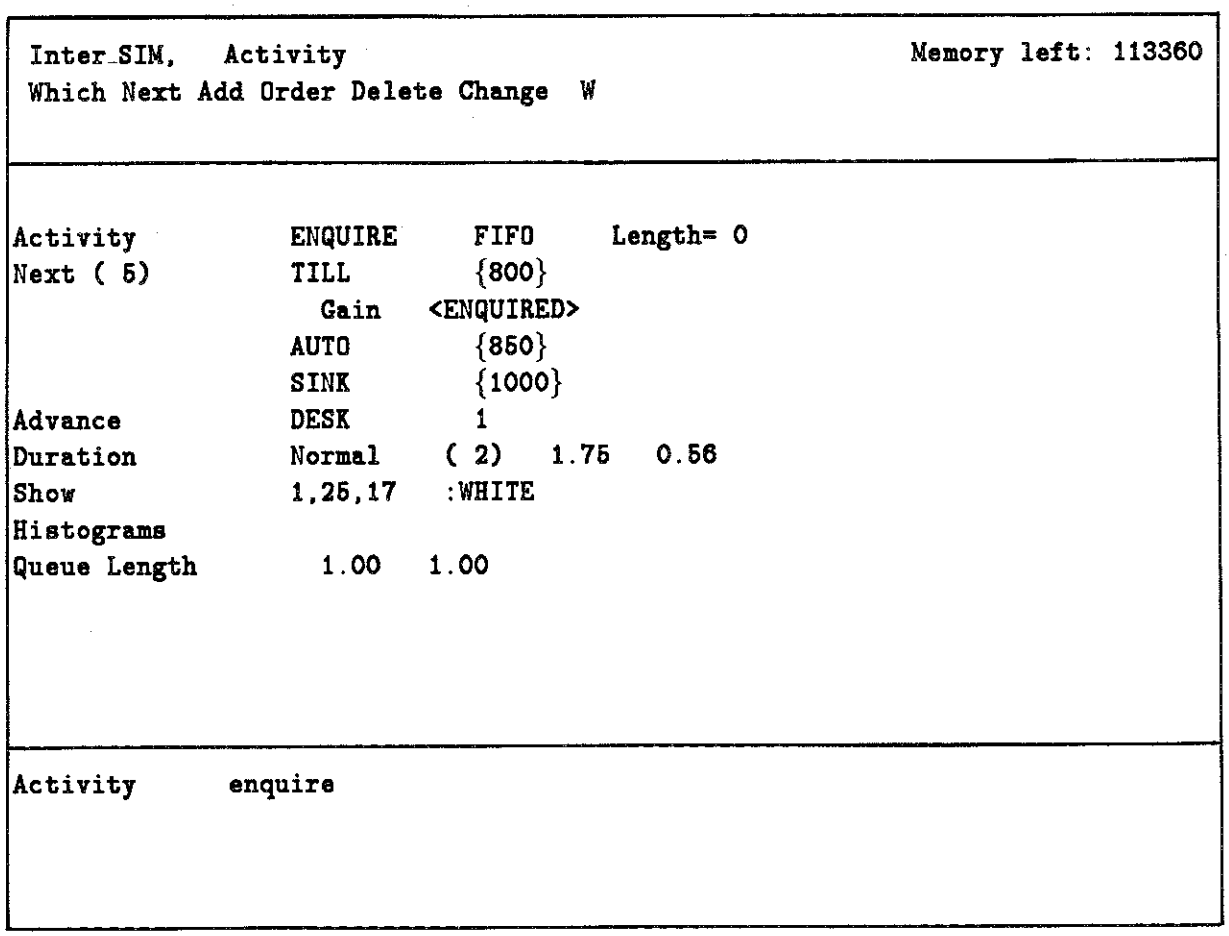

Figure 5: Using the Which option to retrieve the activity for ENQUIRE.

(The description shows (a) a queue priority rule (FIFO) (b) the entity is routed by probability, where probabilities are expressed in thousandths, are drawn from strean number 5, and are cummulative (each possible branch is tried in turn, where entities routed to queue for a till gain membership of ENQUIRED) (c) a single DESK is the resource constraint, where the DESK will be released at the end of the activity (d) a duration (normal distribution with a mean of 4.3, standard deviation of 1.7, and samples taken using random numbers from stream number 2) (e) the queue entered in the picture, on screen 1, and with a white background (f) a histogram for queue length (with cell base 1.0 and cell width 1.0).)

```
                SINK     {1000}
Advance         DESK       1
Duration        Normal   ( 4)    1.75    0.56
Show            1,25,17  :WHITE
Histograms
Queue Length     1.00    1.00

Activity        TILL      FIFO
Next ( 5)       SINK     <ENQUIRED>
                ENQUIRE  {50}
                SINK     {1000}
Advance         CASHIER    1
Duration        Normal   ( 2)    0.45    0.12
Show            1,25,13  :WHITE

Activity        AUTO      FIFO
Next            SINK     {1000}
Advance         AUTO       1
Duration        Neg Exp  ( 1)    1.20
Show            1,25,7   :WHITE
```

Figure 6: Printed descriptions for the example.

If the running simulation is suspended, the user can take menu options concerned with the investigation of a running model. For instance, the Histogram option leads to a further menu which allows users to display, print or file any histogram that is part of any description. The Time option reveals a further menu concerned with all aspects of time - viewing the calendar of future events, rescheduling the events, setting the time at which the running simulation will end. Figure 8 shows an example view of the calendar.

The actions of the other top level menu options are :-

Queue Provides a view of all entities in an activity queue.

Print All the descriptions are sent to the printer or a file. (Figure 6 was produced by Print.)

Global Enables the alteration of parameters concerned with the display and picture, for example: (1) turn the picture on/off, (2) specify a file for the static background, (3) set a damper for the speed the picture advances.

Zero Leads to a menu concerned with resetting all or part of a simulation. Streams can be reseeded and histograms emptied - this is necessary for statistical experimentation. The time can be set back to zero. Queues can be emptied.

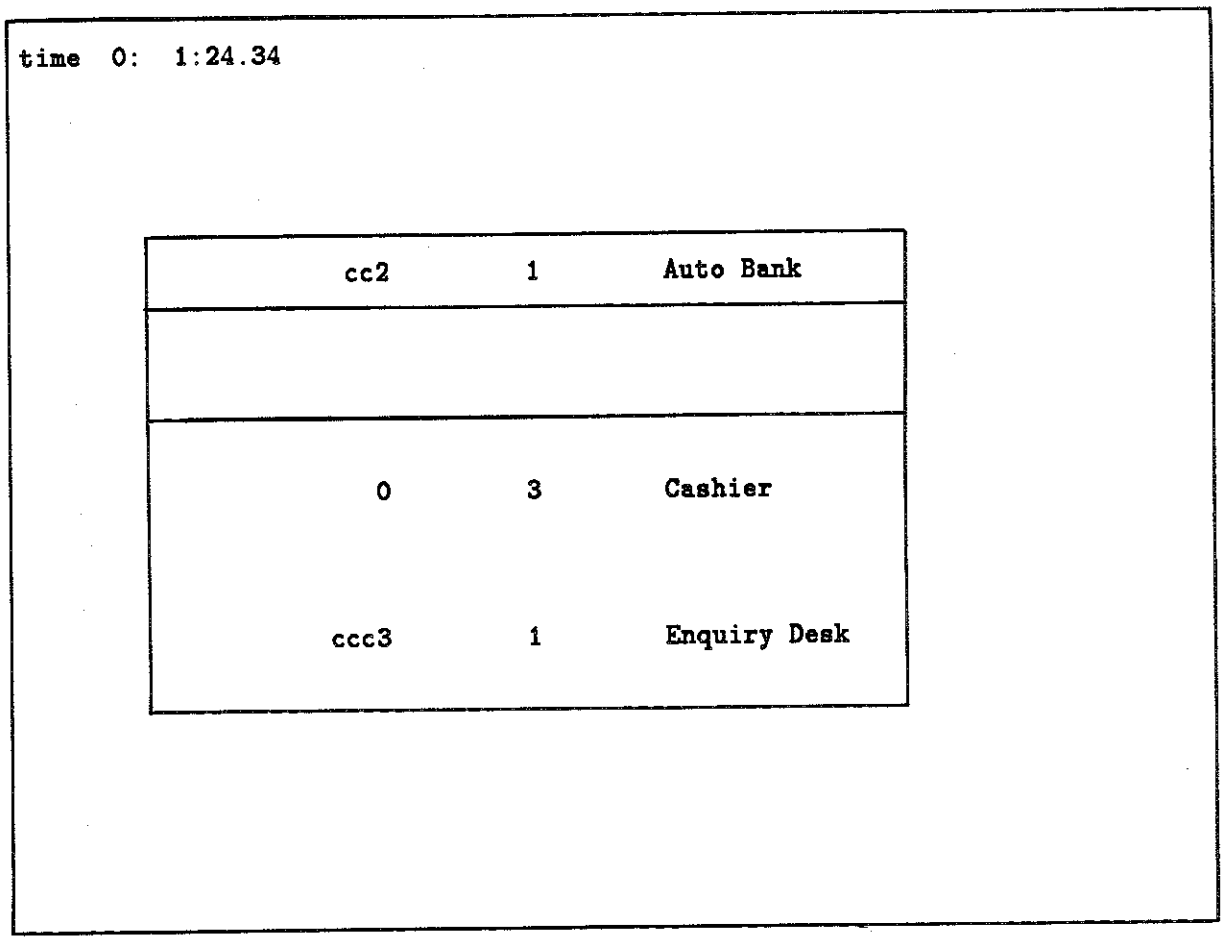

Figure 7: Visual display for the example.

(Each queue entered in the picture is shown as the queue length preceeded by letters representing the queueing entities, where the letter identifies their class, and colour represents their set membership. Bins are represented as the number of resources presently being used. The static text, which annotates the moving part of the display, has been provided by specifying a text file that is prepared externally to Inter_SIM.)

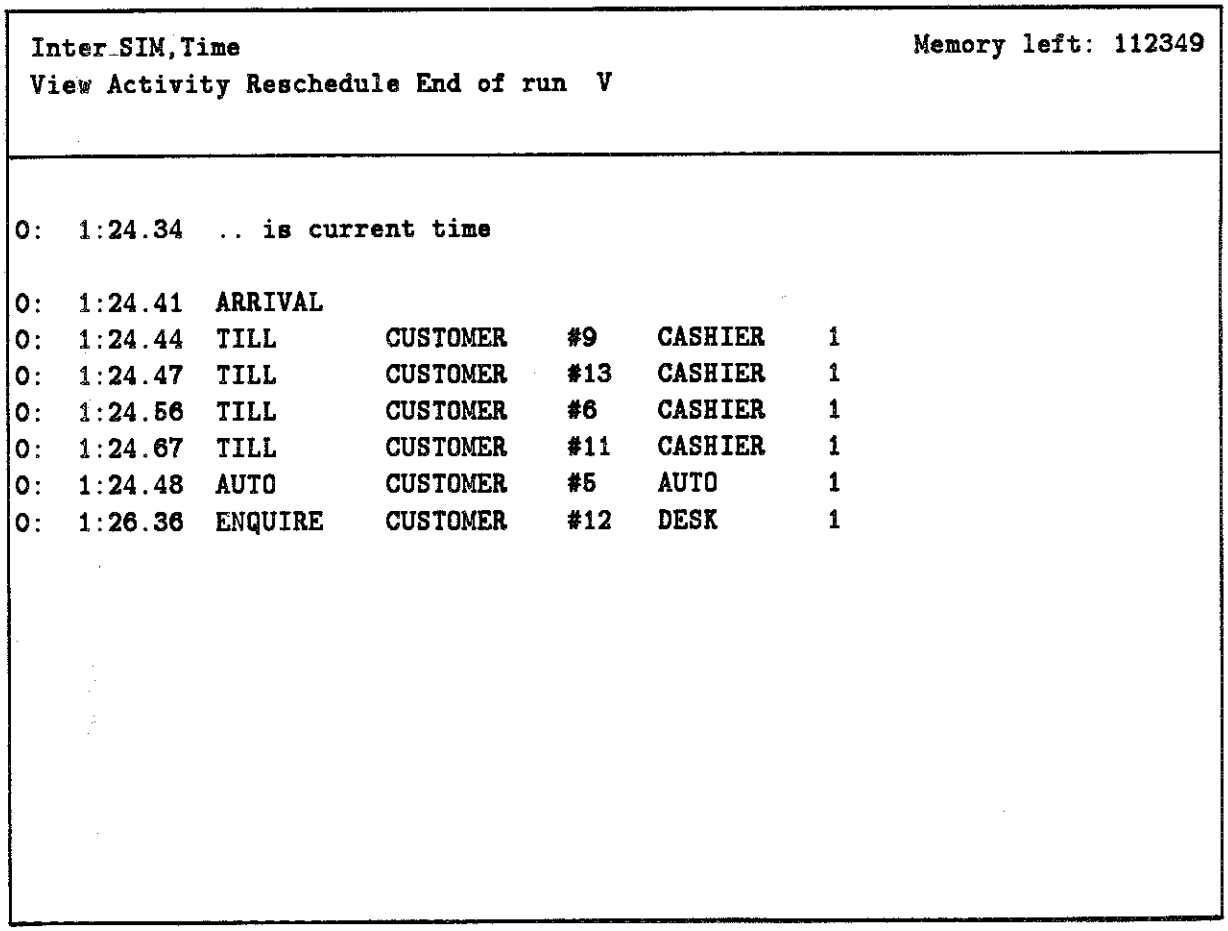

Figure 8: A view of the next event calendar.

(Each line shows the time of the event, the name of the activity or source which has scheduled it, and (if scheduled by an activity) the entity engaged in the activity (class name and entity number) with any co- operating resources or entities.)

File Store the entire simulation in its present state.

Load Retrieve a previously filed simulation.

Directory Show a directory of all simulations.

Insert Call a user programmed insert.

Any part of any component description can be changed. New activities could be added, the source changed, parts of the picture placed on different parts of the screen, histograms added or deleted. The only restriction is that the name of any component can not be altered.

Incorporating the User in the Simulation

Suppose the manager of the bank reviews the state of all queues approximately every hour, altering the number of cashiers as he thinks fit. This can be modelled by placing the manager in front of the running simulation, allowing him to stop the model as desired, and use menu options to change the number of resources in the CASHIER bin. The manager can try out various experiments, for example, investigating the effect of a sudden batch arrival.

The above are examples of user determined interaction. Often it is useful to specify model determined interaction. This can be achieved in Inter_SIM by use of the keyword ASK, which when encountered has the effect of suspending the running simulation and prompting the user for an appropriate decision. For instance, given

```
Activity        ARRIVE      FIFO
Next (6)        TILL        {750}
                ASK
                NONE
```

then when routing a CUSTOMER with this list, if the uniform sample from stream 6 is greater than 0.75 the user will be prompted with

```
CUSTOMER #13 finished ARRIVE
Next Activity ?
```

in the bottom window.

Programmed Extensions

The level of complexity of models that can be directly constructed in Inter_SIM is comparitively low. Thus facilities are provided for extending simulation with programmed inserts. This is achieved by re-programming a number of Pascal routines and relinking Inter_SIM.

The two most straightforward routines are

```
procedure main_insert;
procedure user_picture;
```

where main_insert is invoked by the menu option Insert, and user_picture is called when ever the simulation is run. There are also a number of hooks which are always called by the executive, for instance,

```
type identifier=string[8];

procedure user_start_activity(i:identifier);
procedure user_end_activity(i:identifier);
```

are always called when an activity is started or ended, with i as the name of the activity.

A programmed insert can also be called by specifying the word USER in some parts of a component description, followed by a user number in the range 1 to 64. The user number is passed to the appropriate routine

```
const user_num_max=64;
      stream_max=33;

 type user_num=1..user_num_max;
      stream_num=1..stream_max;

function user_sample(num:user_num;st:stream_num):real;
function user_priority(num:user_num):boolean;
function user_next_activity(num:user_num;e:pt_integer):boolean;
```

where user_sample is called for USER as an activity or source duration, user_next_activity for part of a Next activity list, and user_priority for a queue priority rule. Note that both user_priority and user_next_activity must return true if they actually remove an entity from the queue or route an entity to another activity. Typically the user number is used in a case statement to invoke another user provided routine.

Routines are provided to help the programmer with programming inserts. Most simply allow access to information about the simulation, for instance

```
function queue_size(a:identifier):integer;
```

returns the present queue size for activity a. Thus, given the example and appropriate declaration,

```
k:=queue_size('TILL     ');
```

is valid.

OTHER METHODS OF INTERACTIVE DEVELOPMENT

There are a number of other software tools that have arisen out of research into providing for interactive development of VIS. Withers and Hurrion [14] produced software for the development of continuous models based upon objects called vessels. (A vessel can be visualized as a storage tank with a capacity and flow into and out of the tank.) The user is asked questions about the model, much as in a program generator, and the model can then be immediately executed. The system is somewhat simpler than the one presented here, the behaviour of vessels being fairly easy to define.

Another approach is to provide software for a specific application area. This has the advantage that the software can converse with the user in terminology specific to the area, rather than in the abstract terminology of simulation. An example of such work is the VISUALPLAN system for modelling Flexible Manufacturing Systems produced by da Silva and Bastos [15]. Like the system of Withers and Hurrion, VISUALPLAN asks the user a series of questions, following which the model can be executed.

Ultimately, a user should be able to draw a simulation model. Saduba [16] produced a system called Q-GRAF for the production of Q-GERT models on high resolution graphics screens. The user positions network nodes and edges with a mouse, adding necessary textual information at the keyboard. The network can be immediately processed by the Q-GERT analysis program. Hence the development cycle is far superior to standard Q-GERT, allowing the user to think and work purely in terms of the network diagram. However, interaction with the running model is not possible. More recently, a similar system has been produced by Melamed and Morris [17] at Bell Laboratories, where models of communication systems can be drawn using a number of representative icons. Further, it does allow for extensive user interaction.

CONCLUSIONS

Inter_SIM has been used by the author and others to build a number of simulation models, in the areas of communication networks, production layout, maintainability analysis, and health care. In most cases the provision of a visual display has been of paramount importance. Interaction with the running model has been important as a means of investigating and validating its performance, and thus the ability to be able to halt the running model as desired has been seen as very useful by most users. As yet no model has been constructed where model determined interaction is crucial.

The main strengths of the description interpreter approach are accessibility and rapid development. Inter_SIM models have been developed by a number of users whose previous experience with simulation is fairly limited. In a number of instances, completed models have been handed over to end users. Interpretation removes the development time associated with compilation or any other form of translation, and fosters incremental development. Working proto-types have often been developed very quickly at the keyboard.

The weakness of the description interpreter approach is lack of flexibility. Most models constructed with Inter_SIM have required the programming of some user inserts. Although considerable extensions can be accomplished via the various facilities for inserts and hooks, this is dependent on

knowledge of Pascal, and the process of relinking Inter_SIM with the inserts is not painless. The programming is not always straight- forward, mainly due to having to access the Inter_SIM data structures, and frequently involves more coding than the equivalent extension in an SPL would entail. Further, having to temporarily halt development so as to extend the system with some programmed inserts is frustrating. Inserts could have been avoided in many instances if Inter_SIM were more powerful, particularly if a means of handling user defined entity attributes was present. (Future development of Inter_SIM will see the addition of integer and real attributes to the class component, and a full expression facility to alter attribute values within activity descriptions). Given its inflexibility, some users see Inter_SIM simply as a tool for rapid proto- typing, the Inter_SIM model being manually converted into an SPL or general purpose language.

There are a number of other areas where Inter_SIM needs futher development. The enforcement of bottom-up description is too strict. Frequently a user wishes to refer to a as yet undescribed resource, entity or activity whilst describing an activity. Top-down description should be supported. An implementation using full windowing would provide a richer environment, for example, displaying a histogram over the picture of a suspended run would be a useful aid to working with a model.

Allowing the user unconstrained interaction with the running model presents a number of problems. A criticism often levelled against Visual Interactive Simulation is that the user will determine the results of experiments by looking at the picture for a small amount of time, and accumulating histograms that are invalid due to user interaction altering various parameters in the midst of statistics collection. Bell [13] discusses some of these problems, and the need for a methodology. Most interactive experimentation should be accompanied by detailed statistically valid experimentation, perhaps performed by a specialist.

One encouraging conclusion from experience with Inter_SIM is the value of the menu interface. Even if much of the model must be provided by Pascal inserts, once these have been developed and tested, the interface allows other analysts to quickly complete development, and end users to interactively design and execute appropriate experiments. This is in contrast to program generators, which support development but not experimentation, and analysis programs, where it is often difficult to alter the input data so as to alter the model or run experiments.

Acknowledgements

Part of this work was supported by a grant from the Science and Engineering Research Council of the United Kingdom, supervised by the late Professor K.D. Tocher (Professor of Operational Research at the University of Southampton, England), Professor David Barron (Computer Studies Group, University of Southampton, England) and John Crookes (Department of Operational Research, University of Lancaster, England), and submitted as a Ph.D. thesis [18]. This paper benefited from the comments provided by Professor Peter Brown (Computer Laboratory, University of Kent at Canterbury, England) and Professor Richard E. Nance (Computer Science Department, Virginia Polytechnic Institute and State University, U.S.A.).

References

1. *Proceedings of the IBM scientific computing symposium on simulation models and gaming*, IBM, New York (1966).

2. A.A.B. Pritsker and P.J. Kiviat, *Simulation with GASP II*, Prentice- Hall, Englewood Cliffs, New Jersey (1969).

3. P.R. Hills, SIMON - a computer simulation language in Algol 60, in *Digital Simulation in Operational Research (Hollingdale, Ed.)*, English Universities Press, London, England (1965).

4. G.M. Birtwistle, *Discrete Event Modelling on SIMULA*, Macmillan, London, England (1979).

5. A.T. Clementson, Extended Control and Simulation Language, *Computer Journal* **9**, 216-220 (1966).

6. A.T. Clementson, ECSL, in *Proceedings of the 1978 UKSC Conference on Computer Simulation*, IPC Press, Guildford, England (1978).

7. S.C. Mathewson, User acceptance: design considerations for a program generator, *Software - Practice and Experience* **13**, 101-117 (1983).

8. A.A.B. Pritsker and C.E. Sigal, *Management Decision Making - a Network Simulation Approach*, Prentice-Hall, New Jersey (1983).

9. S.D. Roberts and T.E. Sadlowski, INS: Integrated Network Simulator, in *Proceedings of the Winter Simulation Conference*, The Society for Computer Simulation, La Jolla (1975).

10. C.M. Overstreet and R.E. Nance, A specification language to assist in analysis of discrete event simulation models, *Comm. of the ACM* **28**, 190-201 (1985).

11. R.D. Hurrion, *The Design, Use and Required Facilities of an Interactive Visual Computer Simulation Language to Explore Production Planning Problems*, Ph.D. thesis, University of Warwick, England (1976).

12. R.D. Hurrion, An investigation of visual interactive simulation methods using the job-shop scheduling problem, *J. Opl. Res. Soc.* **29**, 1085-1093 (1978).

13. P.C. Bell, Visual Interactive Modelling in Operational Research: successes and opportunities, *J. Opl. Res. Soc.* **36**, 975-982 (1985).

14. S.J. Withers and R.D. Hurrion, The interactive development of visual simulation models, *J. Opl. Res. Soc.* **33**, 973-976 (1982).

15. C. Moreira da Silva and J. Mesquita Bastos, *VISUALPLAN: a Visual Interactive Simulation System for FMS Modelling*, Progress report number 3, GEIN, Universidade do Porto, Portugal (1985).

16. J. Saduba, *A Study of Q-GERT Modeling and Analysis Using Interactive Computer Graphics*, M.Sc. thesis, Purdue University (1977).

17. B. Melamed and R.J.T. Morris, Visual simulation: the performance analysis workstation, *IEEE Computer* **18**, 87-94 (1985).

18. R.M. O'Keefe, *Developing Simulation Models: an Interpreter for Visual Interactive Simulation*, Ph.D. thesis, University of Southampton, England (1984).