"INTERACTIVE" - A USER FRIENDLY SIMULATION LANGUAGE

Ronald R. Mourant  
Department of Industrial Engineering  
and Management Information Systems  
Northeastern University  
Boston, MA 02115

Raman Lakshmanan  
School of Engineering and Computer Science  
Oakland University  
Rochester, Michigan 48063

INTERACTIVE is a Pascal based network simulation language that is designed to be used on microcomputers. The language enables simulation models to be represented in a process-oriented structure using eighteen network node symbols. The simulation of a network model is performed in four steps. First, the physical model is described as a network diagram by combining the node symbols. In step 2, the details of the network diagram are entered into the computer using a "Forms Editor." In step 3, INTERACTIVE automatically creates a network flow model. The final step is the execution of the network model. In this step, run-time status messages, summary reports, and graphical plots are presented on the computer display for monitoring the system performance and possible user interaction.

1. INTRODUCTION

The availability of relatively inexpensive microcomputers has provided the opportunity to develop application software that use the interactive features of the microcomputer. In the past, simulation languages required the use of mainframe computers and minicomputers to develop and analyze simulation models. Simulation languages such as GPSS (IBM 1981), SIMSCRIPT (Kiviat et al. 1971), GASP (Pritsker 1974), and SLAM (Pritsker and Pegden 1979) have been implemented on mainframe and minicomputers. The cost associated with model development and execution using these languages on large computers is high.

Recently three simulation languages, SIMAN (Pegden 1982), Microtote (Talavage and Lilegdon 1983), and PSIM (Mourant 1983), have been implemented on microcomputers. The use of microcomputers results in: 1) lower equipment cost, 2) easy access to the computer, 3) lower computer usage fees, and 4) interactive programming environment.

The widespread use of microcomputers has resulted in the rapid gain in popularity of the Pascal language. Pascal, initially developed to teach structured programming techniques, has been used in the development of operating systems, compilers, and application software. The two major advantages of using Pascal are:

1) the construction of clear, readable programs, and
2) significantly lower software development and maintenance costs. All the simulation languages mentioned above, with the exception of PSIM, have been written in FORTRAN or Assembly language. FORTRAN does not support advanced data structure declarations and FORTRAN programs are generally unstructured compared to those written in strongly typed languages such as Pascal, ADA, and MODULA-2. Pascal's data structures are directly applicable to define simulation model components such as queues, service facilities, scheduling techniques, and routing of an object through a system.

In order to use the network simulation languages GPSS, SLAM, SIMAN, and Micronet, the modeler needs to be familiar with node statements syntax. Also, the learning of the operating system and text editor commands greatly increase the model development time. INTERACTIVE has been designed to integrate the simulation model development and its execution. The language combines Pascal's data structures and the interactive features of the microcomputer to provide a step by step approach to simulation model development.

This paper presents the details of the network simulation language INTERACTIVE. Section 2 describes the modeling framework in INTERACTIVE. In section 3, we present the network node symbols, the simulation support functions, and the uniquely designed "Forms Editor." Section 4
presents the complete modeling approach using INTERACTIVE with the aid of an example.

2. MODELING FRAMEWORK

The modeling framework in INTERACTIVE is divided into four distinct steps. Each step consists of a "Process" which operates on an "Input" and produces an "Output." The input is from the modeler or the output from the previous step. Each process includes subprocesses to perform specific tasks with the process. Figure 1 illustrates the overall modeling framework.

In the model description step, a network block diagram of the physical model is drawn using the node symbols. There are eighteen nodes available. Node names, node designs, and the node features are carefully chosen to handle most real-world situations. Support functions are included to identify system variables, generate random samples from statistical distributions, and use mathematical functions.

The model input step is used to enter the network node diagram into the computer. This is done interactively using the forms editor. The forms editor prompts the user for appropriate entries as the node details are being entered.

The completed network is stored as a node forms file. This file is used as the input to the model structure step.

The main function of the model structure step is to convert the network nodes file into a network flow model. This conversion is done automatically by using a powerful and unique data structure design.

The model execution uses the network flow model to simulate the system performance. The execution process is completely interactive with facilities to interrupt the simulation run and request status and summary reports, and graphical plots.

3. MODEL DESCRIPTION AND INPUT

The primary means of modeling in INTERACTIVE is through the use of the network node symbols. These specialized nodes are connected to illustrate the flow and decision processes in the network. Once the network diagram is drawn, the details of the network are entered into the computer using the forms editor.

Fig. 1: Interactive Modeling Framework
3.1 Network Node Symbols

There are eighteen different node symbols in INTERACTIVE to represent system components such as queues, service facilities, and status change options. In addition, a "branch" node symbol is used to route objects leaving a network node. Figure 2 shows the node symbols in INTERACTIVE. The design and usage of the node symbols are presented by Lakshmanan (1983) and in the user's manual for INTERACTIVE (Lakshmanan and Mouriart 1983).

3.2 Simulation Support Variables and Functions

The simulation variables and functions are used in expressions to specify system properties and branching decisions from a node. Tables 1 and 2 list these variables and functions.

3.3 The Forms Editor

The forms editor is a menu driven full screen editor specially designed to simplify the network model input into the computer. The main menu selections for the editor is shown in Fig. 3. Each of these selections perform specific tasks in the network model entry. The Create/Append option is used to enter the details of a network model. The Edit option presents a list of nodes in the network and permits changes to be made in one or more nodes. The Load and Save options transfer the network model from the computer memory to a storage medium or vice versa. The Print option lists the network model details on a printer.

Fig. 2: "INTERACTIVE" Network Node Symbols
Table 1 Simulation Variables

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>DATA TYPE</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>feature</td>
<td>array [1..10] of real</td>
<td>array used to define the features of an object.</td>
</tr>
<tr>
<td>globalvar</td>
<td>array [1..10] of real</td>
<td>array to define real valued simulation variables.</td>
</tr>
<tr>
<td>glogical</td>
<td>array [1..10] of boolean</td>
<td>array to define boolean valued simulation variables.</td>
</tr>
<tr>
<td>sntime</td>
<td>real</td>
<td>current simulation time</td>
</tr>
<tr>
<td>simbegtime</td>
<td>real</td>
<td>simulation beginning time</td>
</tr>
<tr>
<td>simendtime</td>
<td>real</td>
<td>simulation ending time</td>
</tr>
</tbody>
</table>

Table 2 Simulation Functions

<table>
<thead>
<tr>
<th>MATHEMATICAL FUNCTIONS</th>
<th>RANDOM VARIABLE FUNCTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs(x)</td>
<td>random [stream]</td>
</tr>
<tr>
<td>sin(x)</td>
<td>poisson [mean, stream]</td>
</tr>
<tr>
<td>cos(x)</td>
<td>exponential [mean, stream]</td>
</tr>
<tr>
<td>exp(x)</td>
<td>normal [mean, std_dev, stream]</td>
</tr>
<tr>
<td>log(x)</td>
<td>uniform [low, high, stream]</td>
</tr>
<tr>
<td>ln(x)</td>
<td>erlang [mean, samples, stream]</td>
</tr>
<tr>
<td>atan(x)</td>
<td>lognormal [mean, std_dev, stream]</td>
</tr>
<tr>
<td>asin(x)</td>
<td>beta [alpha, beta, stream]</td>
</tr>
<tr>
<td>sqrt(x)</td>
<td>gamma [alpha, beta, stream]</td>
</tr>
<tr>
<td>sqrt(x)</td>
<td>weibull [alpha, beta, stream]</td>
</tr>
<tr>
<td></td>
<td>triangular [low, mid, high, stream]</td>
</tr>
<tr>
<td></td>
<td>userfunction [number]</td>
</tr>
</tbody>
</table>

Entering the Network

The details of the network model nodes are entered into the computer by invoking the Create/Append selection in Fig. 3. This selection presents another selection menu as shown in Fig. 4. Figure 4 lists the node symbols and a pointer to select the required symbol. When a node symbol is selected, the forms editor presents a node form for the selected node. Each node symbol has a unique node form to input the node parameters.

Figures 5 and 6 show the node forms for the generate and queue nodes. The node fields are entered by positioning the screen cursor to the requests. The values of several fields in each node form are defaulted. Also, when more than one option is possible for a given field, the options are listed on the screen. For example, the ranking options for the queue are listed in the queue node form. Based on the data type of each field, the forms editor keyboard input procedures respond to valid entries from the modeler. In addition to the error free input, the forms editor prompts for additional node parameters based on the values specified in the fields. For example, the balloing, blocking, and bumping options from a queue node are requested when the queue capacity is finite. Figure 7 shows this setup.

Editing Network Nodes

The edit selection in Fig. 3 permits a modeler to view and change node parameters in the network nodes already entered into the computer. Figure 8 illustrates the editing process in the forms editor.

The editor lists the nodes in the network along with the branching paths from each node. The options at the bottom of the screen permit different operations on the nodes. In order to view and modify a node's parameters, the "Edit" selection is made after positioning the pointer on the screen to the node. Then the forms editor presents the node form for the node with its parameters listed in the appropriate fields. This form is similar to the one presented in the node forms entering process.
**INTERACTIVE - A User Friendly Simulation Language**

**INTERACTIVE - Network Create/Edit**

Enter Selection

- Create / Append Network
- Edit Network
- Load network from a file
- Save network into a file
- Print network
- Exit
- Restart

Fig. 3: Forms Editor Selections

**NODE SYMBOLS SELECTION**

Select a node

- Assign
- Changeobj
- Find
- Generate
- Interface
- Match
- Preempt
- Queue
- Resource
- Return
- Selectfile
- Service
- Statistics
- Status
- Stop
- Terminate
- Travel

↑ and ↓ keys move the cursor
Press <RETURN> to accept indicated selection
Press <ESCAPE> to end selection

Fig. 4: Node Symbols Selection

**GENERATE NODE FORM**

Node label: [J]
Time between object generations: [++]
Generate first object at time: [0.01]
Maximum number of objects to be generated: [++]
Assign time of generation to feature: [0]

Options
Do you want branching?: [N]

Command: CONTROL-A (accept) ESCAPE (menu) CONTROL-B (branches)

Fig. 5: Generate Node Form

The Append and Insert options are used to append or insert new nodes into the network at the position of the pointer. Thus, the network can be viewed and modified with ease.

**Saving the Network**

The save selection in Fig. 3 is used to store the network model as a Pascal file of records in the computer's storage medium. This file is
read as records to edit the network or to create a flow model of the network.

The forms editor data structures eliminate the need for time consuming parsing techniques to identify network statements and detect errors in them. The combination of Pascal data types and the error free input procedures permit direct conversion of the network model into an executable form.
4. NETWORK MODEL EXECUTION

The network model is executed by selecting the "Run" command in Fig. 9. There are two steps involved in the network model execution: Flow Model Creation and Flow Model Execution.

4.1 Flow Model Creation

The network flow is created from the network nodes file. The flow model is a set of Pascal data structures which are used to represent the simulation model components and the movement of objects or entities through them. INTERACTIVE automatically creates this structure from the nodes file. Also, the network model is checked for incomplete network paths and invalid references to node functions. When an error occurs, the modeler can return to the forms editor and correct the error.

4.2 Flow Model Execution

The flow model execution process simulates the network model for the given conditions. During the simulation run the modeler can interrupt the execution and request status and summary reports, and graphical plots. Also, the simulation variables can be changed to alter the run conditions. The use of INTERACTIVE is best illustrated with an example. This example models a computer facility with a single central processing unit (CPU). The problem is taken from Law and Kelton (1982).

Example: Jobs arrive at a computer facility with a single CPU with interarrival times that are IID exponential random variables with a mean of 1 minute. Each job specifies upon its arrival the maximum amount of processing time it requires, and the maximum times for successive jobs are IID exponential random variables with a mean of 1.1 minutes. However, if m is the specified maximum processing time for a particular job, the actual processing time is uniformly distributed between 0.5m and 1.05m. The CPU will never process a job for more time than its specified maximum; a job whose required processing time exceeds its specified maximum leaves the facility without completing service. Simulate the computer facility until 1000 jobs have left the CPU. The jobs in the queue are processed in a FIFO manner. Estimate the average delay in queue of jobs, and the response time for each job.

Solution: Fig. 10 shows a simple layout of the computer facility. The processing time specification involves computing the actual processing time for the incoming job.

Model Development

The network diagram for this system is shown in Fig. 11. Jobs are created at the generate node "jobs" with interarrival times exponentially distributed with a mean of 1 minute. Each incoming job is then routed through a sequence...
of assign nodes to compute its actual processing time. The assign node "max time" assigns the requested maximum processing time to feature 1 of the object. The "serv time" assign node computes the actual processing time as a function of the requested maximum processing time. The actual processing time is assigned to feature 2 of the object. Objects leaving the "serv time" assign node are routed to assign node "cut time" or the queue node "cpu q" based on the branching conditions at the node. When the actual processing time is greater than the requested maximum processing time, the object is routed to the assign node "cut time" where its processing time is set to the requested maximum time; then the object is sent to the "cpu q". Otherwise the object is sent to the queue node "cpu q" directly. The service node "cpu" is used to process objects from the queue. The service time is specified as the value in feature 2 of the object being served. Objects leaving the service node pass through a statistics node "resp_time" where the response time is collected as the time in the system. Then the objects are terminated at the "exit" node. The terminate count is set to 1000 to signal the end of the simulation run.

The network model details are entered into the computer using the forms editor. The completed forms for this example are shown in Figs. 12 through 21. These forms are self explanatory and the reader can easily follow the steps involved in entering the forms. The completed forms are saved in a network nodes file and specified as the input file to the execution step.

Model Execution

Figures 22 and 23 present the flow model creation process for this example. The branching labels match with the network diagram in Fig. 11.

The simulation execution shows the status of the run as shown in Fig. 24. The modeler can interrupt the run and select the options listed. We present some of the intermediate outputs that were generated by interrupting the simulation run.

Figure 25 shows the monitor report starting at time 87.2561 minutes. The list shows the processing of objects as they flow through the nodes. This report can assist the modeler to check the logical decisions in the model. As an example, the object entering the assign node "cut time" at 90.7914 minutes needs a processing time of 0.614 minutes. Since this duration is greater than the requested processing time, the actual processing time is reassigned to the requested maximum time. Then the object joins the queue node "cpu q".

The status reports option in Fig. 24 presents menu selections shown in Fig. 26. These selections are used to view the status of simulation components. The status of the queue at 92.0630 minutes is shown in Fig. 27. Jobs 102 through 105 are waiting in the queue and their actual processing times are as indicated by feature 2. The future events in the simulation calendar at 92.0630 minutes are shown in Fig. 28. Figure 29 shows the summary report for the run at 92.0630 minutes. After viewing the interim reports the simulation execution is resumed and run is completed.

Summary of Results

The complete summary report for the run is shown in Fig. 30. The "cpu q" lists the average waiting time in the queue to be 6.93 minutes. The average response time for each job was 6.84 minutes and the CPU utilization was 85%.
Fig. 12: Project Details Form for the Computer Facility Model

Fig. 13: Generate Node "Jobs"

Fig. 14: Assign Node "max_time"

Fig. 15: Assign Node "Serv_time"
Fig. 16: Branches From Assign Node "serv_time"

Fig. 17: Assign Node "cut_time"

Fig. 18: Queue Node "cpu_q"

Fig. 19: Service Node "cpu"
**STATISTICS NODE FORM**

Node label: [resp_time]
Type: [Time in system]
(see below for selections 1-5)
Identifier: [time in cpu]

Options

Do you want a histogram? [N]

Do you want branching? [N]

Statistics types:
1. Current time
2. Time between arrivals
3. Time in system
4. Feature [1]
5. Globalvar [1]

Command: CONTROL-A(accept) ESCAPE(menu) CONTROL-B(Branches)

---Fig. 20: Statistic Node "resp_time"---

**TERMINATE NODE FORM**

Node label: [exit]
Number of objects to terminate simulation: [1000]

Command: CONTROL-A(accept) ESCAPE(menu)

---Fig. 21: Terminate Node "exit"---

---"INTERACTIVE" Network Model Building — Pass 1---

<table>
<thead>
<tr>
<th>Node Label</th>
<th>Node Type</th>
<th>Branches</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>jobs</td>
<td>Generate</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>max_time</td>
<td>Assign</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>serv_time</td>
<td>Assign</td>
<td>cut_time</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cpu_q</td>
<td>0</td>
</tr>
<tr>
<td>cut_time</td>
<td>Assign</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>cpu_q</td>
<td>Queue</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>cpu</td>
<td>Service</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>resp_time</td>
<td>Statistics</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>exit</td>
<td>Terminate</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

---Total Errors in Pass 1: 0; Press <RETURN> to continue---

---"INTERACTIVE" Network Model Building — Pass 2---

<table>
<thead>
<tr>
<th>Node Label</th>
<th>Node Type</th>
<th>Branches</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>jobs</td>
<td>Generate</td>
<td>&lt;Next&gt;</td>
<td>0</td>
</tr>
<tr>
<td>max_time</td>
<td>Assign</td>
<td>&lt;Next&gt;</td>
<td>0</td>
</tr>
<tr>
<td>serv_time</td>
<td>Assign</td>
<td>cut_time</td>
<td>cpu_q</td>
</tr>
<tr>
<td>cut_time</td>
<td>Assign</td>
<td>&lt;Next&gt;</td>
<td>0</td>
</tr>
<tr>
<td>cpu_q</td>
<td>Queue</td>
<td>&lt;Next&gt;</td>
<td>0</td>
</tr>
<tr>
<td>cpu</td>
<td>Service</td>
<td>&lt;Next&gt;</td>
<td>0</td>
</tr>
<tr>
<td>resp_time</td>
<td>Statistics</td>
<td>&lt;Next&gt;</td>
<td>0</td>
</tr>
<tr>
<td>exit</td>
<td>Terminate</td>
<td>&lt;None&gt;</td>
<td>0</td>
</tr>
</tbody>
</table>

---Total Errors in Pass 2: 0; Press <RETURN> to continue---

---Fig. 22: Flow Model Creation-Network Details---

---Fig. 23: Flow Model Creation-Node Details---
### Fig. 24: Simulation Run Status

- Run: 1
- Time: 07.2561
- Memory: 26768

Press any key to pause & select

**Selections:**
- Status Reports
- Change Variables
- Monitor the System: OFF
- Plot
- Resume Simulation
- Output Device: .console
- Quit

### Fig. 25: Simulation Monitor Report

<table>
<thead>
<tr>
<th>SIM Time</th>
<th>At Node</th>
<th>Type</th>
<th>Obj Time</th>
<th>Obj #</th>
<th>...</th>
<th>Object Features...</th>
</tr>
</thead>
<tbody>
<tr>
<td>07.2561</td>
<td>serv_time Assign</td>
<td>IN</td>
<td>87.256</td>
<td>103</td>
<td>0.687</td>
<td>0.0</td>
</tr>
<tr>
<td>07.2561</td>
<td>cpu_q Queue</td>
<td>IN</td>
<td>87.256</td>
<td>103</td>
<td>0.687</td>
<td>0.447</td>
</tr>
<tr>
<td>08.0431</td>
<td>jobs Generate NEW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>08.0431</td>
<td>max_time Assign</td>
<td>IN</td>
<td>88.043</td>
<td>104</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>08.0431</td>
<td>serv_time Assign</td>
<td>IN</td>
<td>88.043</td>
<td>104</td>
<td>0.187</td>
<td>0.0</td>
</tr>
<tr>
<td>08.0431</td>
<td>cpu_q Queue</td>
<td>IN</td>
<td>88.043</td>
<td>104</td>
<td>0.187</td>
<td>0.166</td>
</tr>
<tr>
<td>88.5517</td>
<td>cpu Service OUT</td>
<td>76.256</td>
<td>95</td>
<td>2.197</td>
<td>1.922</td>
<td></td>
</tr>
<tr>
<td>88.5517</td>
<td>resp_time Statistics</td>
<td>IN</td>
<td>76.256</td>
<td>95</td>
<td>2.197</td>
<td>1.922</td>
</tr>
<tr>
<td>88.5517</td>
<td>exit Terminate</td>
<td>IN</td>
<td>76.256</td>
<td>95</td>
<td>2.197</td>
<td>1.922</td>
</tr>
<tr>
<td>88.5517</td>
<td>cpu Service CHK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>88.5517</td>
<td>cpu Service</td>
<td>IN</td>
<td>76.811</td>
<td>96</td>
<td>2.201</td>
<td>1.562</td>
</tr>
<tr>
<td>90.1138</td>
<td>cpu Service OUT</td>
<td>76.811</td>
<td>96</td>
<td>2.201</td>
<td>1.562</td>
<td></td>
</tr>
<tr>
<td>90.1138</td>
<td>resp_time Statistics</td>
<td>IN</td>
<td>76.811</td>
<td>96</td>
<td>2.201</td>
<td>1.562</td>
</tr>
<tr>
<td>90.1138</td>
<td>exit Terminate</td>
<td>IN</td>
<td>76.811</td>
<td>96</td>
<td>2.201</td>
<td>1.562</td>
</tr>
<tr>
<td>90.1138</td>
<td>cpu Service CHK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>90.1138</td>
<td>cpu Service</td>
<td>IN</td>
<td>78.450</td>
<td>97</td>
<td>0.322</td>
<td>0.222</td>
</tr>
<tr>
<td>90.3359</td>
<td>cpu Service OUT</td>
<td>78.450</td>
<td>97</td>
<td>0.322</td>
<td>0.222</td>
<td></td>
</tr>
<tr>
<td>90.3359</td>
<td>resp_time Statistics</td>
<td>IN</td>
<td>78.450</td>
<td>97</td>
<td>0.322</td>
<td>0.222</td>
</tr>
<tr>
<td>90.3359</td>
<td>exit Terminate</td>
<td>IN</td>
<td>78.450</td>
<td>97</td>
<td>0.322</td>
<td>0.222</td>
</tr>
<tr>
<td>90.3359</td>
<td>cpu Service CHK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>90.3359</td>
<td>cpu Service</td>
<td>IN</td>
<td>80.216</td>
<td>98</td>
<td>1.020</td>
<td>1.020</td>
</tr>
<tr>
<td>90.7914</td>
<td>jobs Generate NEW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>90.7914</td>
<td>max_time Assign</td>
<td>IN</td>
<td>90.791</td>
<td>105</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>90.7914</td>
<td>serv_time Assign</td>
<td>IN</td>
<td>90.791</td>
<td>105</td>
<td>0.605</td>
<td>0.0</td>
</tr>
<tr>
<td>90.7914</td>
<td>cut_time Assign</td>
<td>IN</td>
<td>90.791</td>
<td>105</td>
<td>0.605</td>
<td>0.0</td>
</tr>
<tr>
<td>90.7914</td>
<td>cpu_q Queue</td>
<td>IN</td>
<td>90.791</td>
<td>105</td>
<td>0.605</td>
<td>0.614</td>
</tr>
<tr>
<td>91.3359</td>
<td>cpu Service OUT</td>
<td>80.216</td>
<td>98</td>
<td>1.020</td>
<td>1.020</td>
<td></td>
</tr>
<tr>
<td>91.3359</td>
<td>resp_time Statistics</td>
<td>IN</td>
<td>80.216</td>
<td>98</td>
<td>1.020</td>
<td>1.020</td>
</tr>
<tr>
<td>91.3359</td>
<td>exit Terminate</td>
<td>IN</td>
<td>80.216</td>
<td>98</td>
<td>1.020</td>
<td>1.020</td>
</tr>
<tr>
<td>91.3359</td>
<td>cpu Service CHK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>91.3359</td>
<td>cpu Service</td>
<td>IN</td>
<td>80.239</td>
<td>99</td>
<td>0.431</td>
<td>0.422</td>
</tr>
<tr>
<td>91.7785</td>
<td>cpu Service OUT</td>
<td>80.239</td>
<td>99</td>
<td>0.431</td>
<td>0.422</td>
<td></td>
</tr>
<tr>
<td>91.7785</td>
<td>resp_time Statistics</td>
<td>IN</td>
<td>80.239</td>
<td>99</td>
<td>0.431</td>
<td>0.422</td>
</tr>
<tr>
<td>91.7785</td>
<td>exit Terminate</td>
<td>IN</td>
<td>80.239</td>
<td>99</td>
<td>0.431</td>
<td>0.422</td>
</tr>
<tr>
<td>91.7785</td>
<td>cpu Service CHK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>91.7785</td>
<td>cpu Service</td>
<td>IN</td>
<td>82.146</td>
<td>100</td>
<td>0.215</td>
<td>0.196</td>
</tr>
<tr>
<td>91.9742</td>
<td>cpu Service OUT</td>
<td>82.146</td>
<td>100</td>
<td>0.215</td>
<td>0.196</td>
<td></td>
</tr>
<tr>
<td>91.9742</td>
<td>resp_time Statistics</td>
<td>IN</td>
<td>82.146</td>
<td>100</td>
<td>0.215</td>
<td>0.196</td>
</tr>
<tr>
<td>91.9742</td>
<td>exit Terminate</td>
<td>IN</td>
<td>82.146</td>
<td>100</td>
<td>0.215</td>
<td>0.196</td>
</tr>
</tbody>
</table>
You may view/print the:
- Calendar Events
- Final Reports
- State Info.
- Queue Info.
- Resource Info.
- Server Info.

or you may
- Exit to the simulation menu

---

**Fig. 26: Status Report Menu**

---

**"INTERACTIVE" -- Objects in Queue: cpu_q**

---

**Fig. 27: "cpu_q" Status**

---

**"INTERACTIVE" -- Simulation Calendar**

---

**Fig. 28: Simulation Calendar Status**

---

**Fig. 29: Interim Summary Report**
5. SUMMARY AND CONCLUSIONS

This paper presented only a brief review of INTERACTIVE. The language has been used to model several large and complex systems. The interactive features used in the development of simulation models and their execution make the language easy to learn and use. INTERACTIVE can be used by engineers and decision makers to analyze discrete systems in a short amount of time. Also, the language can be used as an effective teaching aid in simulation modeling courses.

The approach used in the design of INTERACTIVE is the first step in development of an integrated simulation software system. The Pascal language permits a flexible design to allow future enhancements to the language. Efforts are underway to include a continuous system modeling capability and a data base support to the language.

REFERENCES


Pritsker AAB, Pegden CD (1979), Introduction to Simulation and SLAM, Systems Publishing Co., West Lafayette, IN.