BACKGROUND

Simulation languages typically are designed to substantially reduce the time and cost required to build simulation models by automatically providing commonly needed facilities, such as statistics gathering, a timing mechanism, random deviates, powerful list processing, and flexible storage management. However, the greatest benefit derived from the use of a simulation tool such as SIMSCRIPT II.5 derives from its powerful "world-view" or method of modeling. This world view provides a guide for conceptualizing the system to be modeled in a way that translates naturally into a computer language.

I. SIMSCRIPT II.5 WORLD VIEW

For more than a decade, SIMSCRIPT has been used to view the world in terms of events, entities, attributes (of entities) and sets (or lists) of entities. These concepts were originally developed in SIMSCRIPT I and expanded in the current language. Recently a further expansion to take a process view of simulation has been introduced. The process appears to be the most universally accepted element for building simulation models. In GPSS it is called a "transaction"; in SIMULA and SIMSCRIPT II.5 it is the "process."

In addition to processes, it is convenient to have a modeling element to represent static objects which in some sense provide service to the processes. In GPSS these are "facilities" or "storages"; in SIMSCRIPT II.5 they are termed "resources."

II. THE TUTORIAL

The SIMSCRIPT II.5 method of modeling is best presented in the context of an example. The classical Job Shop Model will be presented as a basis for discussion of more difficult (or more realistic) situations.

The simulated shop consists of a number of production centers. The machines in a production center are considered identical. The number of production centers is specified separately for each simulation run and remains constant throughout the run. JOBS come into the shop from outside the simulation process at arbitrary points in time. A JOB must go through a sequence of production centers, each of which performs a different TASK. The routing through the shop and the processing time at each production center are specified for each JOB and may vary for different JOBS. After a JOB is processed by one machine, it is immediately routed to its next production center. It is put into process at once if there is a machine available; otherwise it is put into the queue of orders waiting for that production center. JOBS enter and leave the queues on a first-in first-out basis.

The results desired from the simulation are the average length of time that jobs remain in the shop, and the average number of jobs in the queue of each production center. These results are to be obtained at the end of the simulation run.

The natural (English-like) style of SIMSCRIPT II.5 is illustrated by this example (Figure 1) and the ability to produce concise, readable, non-language-dependent results is illustrated by Figure 3. Figure 2 contains the data input to the model.
FIGURE 1

PREAMBLE

PROCESSES

EVERY JOB OWNS A LIST.OF.TASKS

TEMPORARY ENTITIES

EVERY TASK HAS A TASKS.TIME AND A
TASK.PERFORMER AND BELONGS TO A
LIST.OF.TASKS

RESOURCES INCLUDE MACHINE

EXTERNAL PROCESS IS JOB

ACCUMULATE AVG QUEUE.LENGTH AS THE AVERAGE OF
N.Q.MACHINE
TALLY AVG CYCLE.TIME AS THE AVERAGE AND
NO.OF.JOBS.COMPLETED AS THE NUMBER OF
CYCLE.TIME
DEFINE CYCLE.TIME AS A REAL VARIABLE

END "OF PREAMBLE

MAIN

READ N.MACHINE
CREATE EVERY MACHINE
FOR EACH MACHINE,
READ U.MACHINE(MACHINE)
START SIMULATION

PRINT 2 LINES WITH TIME.V*HOURS.V THUS
MULTI - STAGE PROCESSING EXAMPLE
REPORT AFTER ** HOURS OPERATION
SKIP 2 OUTPUT LINES
PRINT 2 LINES WITH NO.OF.JOBS.COMPLETED AND
AVG.CYCLE.TIME*HOURS.V THUS
*JOBS WERE COMPLETED
AVERAGE COMPLETION TIME WAS **HOURS
SKIP 2 OUTPUT LINES
PRINT 2 LINES THUS
AVERAGE QUEUE FOR EACH MACHINE
*MACHINE
AVERAGE QUEUE LENGTH
FOR EACH MACHINE,
PRINT 1 LINE WITH MACHINE AND
AVG.QUEUE.LENGTH(MACHINE) THUS
*

END "OF MAIN

FIGURE 2

PROCESS JOB

DEFINE TASK AND MACHINE AS INTEGER VARIABLES
DEFINE ARRIVAL.TIME AS A REAL VARIABLE

LET ARRIVAL.TIME = TIME.V

" SET UP LIST OF TASKS FOR THIS JOB (FROM DATA)

UNTIL MODE IS ALPHA,
DO
CREATE A TASK
READ TASK.PERFORMER(TASK) AND
TASK.TIME(TASK)
FILE TASK IN LIST.OF.TASKS(JOB)
LOOP

" PERFORM EACH TASK ON LIST

UNTIL LIST.OF.TASKS(JOB) IS EMPTY,
DO
REMOVE FIRST TASK FROM LIST.OF.TASKS(JOB)
LET MACHINE = TASK.PERFORMER(TASK)
REQUEST 1 UNIT OF MACHINE(MACHINE)
WORK TASK.TIME(TASK) HOURS
RELIQUESH 1 UNIT OF MACHINE(MACHINE)
DESTROY TASK
LOOP

LET CYCLE.TIME = TIME.V - ARRIVAL.TIME
END "OF PROCESS JOB

10 3 12 8 5 1 2 4 3 9 3
JOB 1 1 30
1 1.10 10 2.20 2 3.30 9 4.40 3 5.50
8 6.60 4 7.70 7 8.80*

JOB 1 1 31
1 1.10 10 2.20 2 3.30 9 4.40 3 5.50
8 6.60 4 7.70 7 8.80*

JOB 1 1 32
1 1.10 10 2.20 2 3.30 9 4.40 3 5.50
8 6.60 4 7.70 7 8.80*

JOB 1 1 33
1 1.10 10 2.20 2 3.30 9 4.40 3 5.50
8 6.60 4 7.70 7 8.80*

JOB 1 1 34
1 1.10 10 2.20 2 3.30 9 4.40 3 5.50
8 6.60 4 7.70 7 8.80*

JOB 1 1 35
1 1.10 10 2.20 2 3.30 9 4.40 3 5.50
8 6.60 4 7.70 7 8.80*
FIGURE 3
MULTI-STAGE PROCESSING EXAMPLE

REPORT AFTER 73.92 HOURS OF OPERATION

6 JOBS WERE COMPLETED
AVERAGE COMPLETION TIME WAS 43.60 HOURS

AVERAGE QUEUE FOR EACH MACHINE

<table>
<thead>
<tr>
<th>MACHINE</th>
<th>AVERAGE QUEUE LENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.04</td>
</tr>
<tr>
<td>2</td>
<td>0.</td>
</tr>
<tr>
<td>3</td>
<td>0.</td>
</tr>
<tr>
<td>4</td>
<td>.01</td>
</tr>
<tr>
<td>5</td>
<td>0.</td>
</tr>
<tr>
<td>6</td>
<td>0.</td>
</tr>
<tr>
<td>7</td>
<td>.04</td>
</tr>
<tr>
<td>8</td>
<td>.18</td>
</tr>
<tr>
<td>9</td>
<td>0.</td>
</tr>
<tr>
<td>10</td>
<td>.04</td>
</tr>
</tbody>
</table>

III. IMPLEMENTATION OF SIMSCRIPT II.5

SIMSCRIPT II.5 is a full general purpose programming language, in addition to its strong simulation capability. It has been implemented for CDC, Honeywell, IBM and UNIVAC computers. An implementation is under development for the PDP 11.

REFERENCES ON SIMSCRIPT II.5


