Proceedings of the 1985 Winter Simulation Conference
D. Gantz, G. Blais, S. Solomon (eds.)

INTRODUCTION TO GPSS

Thomas J. Schriber
Graduate School of Business Administration
The University of Michigan
Ann Arbor MI 48109

ABSTRACT

Summary information about key aspects of the simulation modeling language GPSS is provided. The class of problems to which GPSS applies especially well is described; commentary on the semantics and syntax of the language is offered, and an example is provided; the learning-oriented literature for GPSS is summarized; various GPSS implementations are commented on; the time-sharing networks offering GPSS are cited; and public courses on the language are listed.

The GPSS tutorial itself will delve into the fundamental details of GPSS and present examples of simple GPSS models. Copies of the transparencies used for the tutorial will be distributed to those in attendance.

A BRIEF PERSPECTIVE ON GPSS

GPSS (General Purpose Simulation System) is a highly popular (1) simulation modeling language whose use greatly eases the task of building computer models for certain types of discrete-event simulations. (A discrete-event simulation is one in which the state of the system being simulated changes at only a discrete, but possibly random, set of time points, called event times.) GPSS lends itself especially well to the modeling of queuing systems (systems in which discrete units of traffic compete for scarce resources), and is generally applicable when it is of interest to determine how well a system will respond to the demands placed on it. For example, GPSS has been applied to the modeling of manufacturing systems, communication systems, computing systems, transportation systems, and inventory systems.

THE SEMANTICS AND SYNTAX OF GPSS

GPSS offers a rich set of semantics, and yet is sparse in its syntax. For example, only nine statements (plus several control statements) are required to model a simple one-line, one-server queuing system in GPSS. These statements take such simple forms as "GENERATE 18,6" and "QUEUE LINE". No read, write, format, or test statements appear in the model. And yet when a simulation is performed with the model, fixed-form, fixed-content output is produced, providing statistics describing the server (e.g., number of times captured; average holding time per capture; fraction of time in use) and the waiting line (e.g., average line content; average residence time in line; maximum line content; percent of arrivals who did not have to wait in line; and so on). This limited example is roughly suggestive of the character of GPSS. A GPSS model for the one-line, one-server system, taken from (2), is reproduced here as an Appendix.

The sparse syntax of GPSS, coupled with its block-diagram orientation, makes it possible for the beginner to learn a highly usable subset of the language quite quickly. This does not mean, however, that it is easy or straightforward to master the full set of GPSS capabilities. Considerable effort and study are needed to learn the language thoroughly.

The GPSS world view (2) involves visualizing units of traffic ("transactions") which move along from block to block in a model as a simulation proceeds. This world view is so natural to the modeling of queuing systems that several other notable simulation languages now also offer a similar view. The effect of this cross-fertilization can be found in SLAM (3), SIMAN (4), SIMSCRIPT (5), and SIMULA (6).

Disadvantages of traditional GPSS are that it has weak input/output capabilities, weak computational facilities, and a static control structure. (Each of these disadvantages has been remedied in GPSS/H, however (7, 8).) These disadvantages can be offset by interfacing a GPSS model with FORTRAN subroutines or PL/1 procedures. The GPSS HELP block is used for this purpose. Some current GPSS implementations support direct invocation of FORTRAN functions and subroutines without the need to use HELP blocks.

THE GPSS LEARNING-ORIENTED LITERATURE

There are several GPSS books (2, 9, 10, 11, 12, 13, 14). Introductions to GPSS can also be found in general simulation texts, e.g. (15, 16, 17, 18, 19, 20).

Articles demonstrating use of advanced GPSS features also occasionally appear. For example, articles illustrating HELP block use are in (21, 22, 23, 24). The GPSS user's manuals may also contain good learning-oriented material. For instance, a suggestive set of examples of HELP block use appears in (8).

GPSS is flexible enough to support taking a number of alternative approaches to modeling a system. The various tradeoffs involved are discussed and illustrated with examples in (25).

COMPARISON OF GPSS TO OTHER SIMULATION LANGUAGES

An introductory survey and description of GPSS/H, SIMAN, SIMSCRIPT II.5, and SLAM II is given in (26). The world view of each language is described, and an example problem is modeled in each language.

A qualitative comparison of GPSS/H, SLAM, and SIMSCRIPT is provided in (27), and a quantitative comparison of these three languages appears in (28). The quantitative comparison is based on a manufacturing job shop problem. "Both model size and model run length were varied to obtain data on compilation time, execution time, CPU time, memory

time and the rate of change of these variables due to changes in the simulation period" (quoted from (28, p. 45)). GPSS/H was found to compile about 50 times faster than SIMSCRIPT and about 10 times faster than SLAM. GPSS/H executed about 3.8 times faster than SIMSCRIPT and about 3.5 times faster than SLAM.

VARIOUS GPSS IMPLEMENTATIONS

GPSS was originally released by IBM in 1961. It then evolved through a series of further IBM releases (GPSS II; GPSS III; GPSS/360; and, in 1970, GPSS V (29)), each offering enhancements over its predecessor. Paralleling the IBM releases, a variety of GPSS implementations was made available both for IBM and non-IBM hardware by organizations external to IBM. The state-of-the-art GPSS implementation for IBM mainframes is now GPSS/H (8), which is written in assembly language, and is an upwardly compatible superset of IBM's GPSS V (5). (Among the more significant advantages offered by GPSS/H over GPSS V are an improvement in execution speed by a factor of about five on average; the ability to interactively monitor an ongoing simulation, which greatly reduces the time required to build and debug models and achieve a detailed understanding of their behavior; the ability to read from and write to external files, which facilitates the incorporation of data into models and the passing of model outputs to post-processing software, such as graphical routines; the use of long symbolic names in extended contexts, which enhances model readability and clarity; and vastly improved ease of accessing FORTRAN subroutines and functions from an executing GPSS model.) Mainframe GPSS/H also runs on the IBM PC/XT/370, and on the IBM AT/370.

GPSS/H is available for VAX computers (8), and for microcomputers based on the Motorola 68000 chip. (These implementations are written in language C.)

Two implementations of GPSS for the IBM PC in native mode are Minuteman Software's GPSS/PC (30), and Simulation Software Ltd.'s GPSSR/PC (31). Simulation Software Ltd. also offers GPSS/VX (31), an implementation for VAX/VMS systems; GPSS/C (31), an implementation for 32-bit architecture computer systems; GPSSR (31), an implementation for DEC PDP-11 systems running RSX-11M or RSTS/E; and GPSS10 (31), an implementation for DECsystem-10 and -20 computers.

Most GPSS implementations for non-IBM mainframes are based on IBM's GPSS V (29), or its IBM predecessor, GPSS/360. Known implementations in this category include GPSS V/170 (Control Data 170 Series computer systems); GPSS/66 (Honeywell Series 60 Level 66 hardware); GPSS/UCC (University Computing Corporation's GPSS for Univac 1108 hardware); GPSSX8 (a Univac 1100-series GPSS implementation maintained at Florida Atlantic University); and GPDS (a GPSS implementation for Xerox Sigma 5-9 computers). No one is known to maintain a complete list of available and actively supported GPSS implementations. In general, those who are not in a position to use GPSS/H, GPSS/PC, IBM's GPSS V, or one of Simulation Software Ltd.'s GPSS implementations must do their own spadework to determine if a reasonably current and actively supported GPSS implementation is available for their computing environment.

ALTERNATIVE LANGUAGES WITH GPSS EMBEDDED

The functions performed by the various GPSS blocks have been embedded in other languages on some occasions. Notable here are GPSS-FORTRAN (32), APL GPSS (33), and PL/1 GPSS (34). Briefly, embedding takes the form of implementing the functions of the GPSS blocks and control statements in a host language as subroutines which augment the power of the existing language. Calling these subroutines then has the effect of simulating the behavior of the corresponding GPSS blocks and control statements. For a paper on the embedding process, see (35).

TIME-SHARING NETWORKS OFFERING GPSS

GPSS is available in the following networks: Boeing Computer Services offers GPSS/H; Computer Sciences Corporation offers GPSSTS in its Infonet System; University Computing Corporation offers GPSS/UCC on the Univac 1108; ADP-Cyphernetics offers GPSS-10 on the PDP 10; Control Data Corporation has GPSS in its Cybernet system; McDonnell-Douglas Automation Company (McAuto) offers GPSS; and American Management Systems (AMS) has a version of GPSS which can be accessed via Telenet. (This list may not be exhaustive.)

SHORT COURSES

Intensive short courses on GPSS are available from three sources. A five-day course is offered each summer in The University of Michigan's Engineering Summer Conferences (contact Prof. Thomas J. Schriber). This course is also offered in October, February, and May in the Washington D.C. area (contact Elizabeth Tucker, Wolverine Software Corporation, Annandale VA). A five-day course including GPSS is offered periodically at the State University of New York at the Center for Statistics, Quality Control and Design (contact Prof. Edward J. Dudewicz, Syracuse University, Syracuse NY). And a five-day GPSS course is given each summer at The Ryerson Polytechnical Institute (contact Prof. R. Greer Lavery, Ryerson Polytechnical Institute, Toronto, Ontario, Canada).

THE GPSS TUTORIAL

In the GPSS tutorial at the Winter Simulation Conference, the rudiments of queuing systems logic and the corresponding modeling elements offered by GPSS to implement this logic will be introduced and illustrated through a series of examples. The tutorial will make use of transparencies, copies of which will be distributed to those attending the tutorial. (There are too many transparencis to include copies of them in these proceedings.) Interested persons unable to attend the tutorial can obtain a copy of these transparencies on request from Professor Thomas J. Schriber (Graduate School of Business, The University of Michigan, Ann Arbor MI 48109-1234; 313-764-1398).

APPENDIX

The next 7 pages provide a GPSS model for a one-line, one-server queuing system. Included are a statement of the problem, a GPSS block diagram for the one-line, one-server system, the corresponding GPSS model file, program output, and discussion. These pages are reproduced from (2, pp. 51-57) with permission.

2.17 CASE STUDY 2A
A One-Line, One-Server Queuing System

(1) Statement of the Problem

The interarrival time of the customers at a one-chair barber shop is uniformly distributed over the range 18 ± 6 minutes. Service time for haircuts is 16 ± 4 minutes, uniformly distributed. Customers coming to the shop get their hair cut, first-come, first-served, then leave. Model the shop in GPSS, making provisions to collect data on the waiting line. Then run the model through 8 hours of simulated time. Interpret the output produced by the model in the context of the barber shop.

(2) Approach Taken in Building the Model

This model is easily constructed as a single sequence of Blocks, excepting the run-control component. The order in which the Blocks appear corresponds to the sequence of stages through which customers move in the real system. Customers arrive; if necessary, they wait their turn; then they engage the barber, get their hair cut, release the barber; and leave. Except for the GENERATE and TERMINATE Blocks, this sequence has already been displayed and discussed in Figure 2.19.

To control the duration of the run, a two-Block "timer segment" can be used. In Figure 2.10, a segment accomplishing the objective required here was presented and discussed, under the assumption that the implicit time unit in effect is 1 minute. That segment will be used for this model.

(3) Table of Definitions

Time unit: 1 Minute

TABLE 2A.1 Table of Definitions for Case Study 2A

GPSS Entity	Interpretation
Transactions	
Model Segment 1	Customers
Model Segment 2	A timer
Facilities	
JOE	The barber
Queues	
JOEQ	The Queue used to gather statistics on the waiting experience of customers

(4) Block Diagram

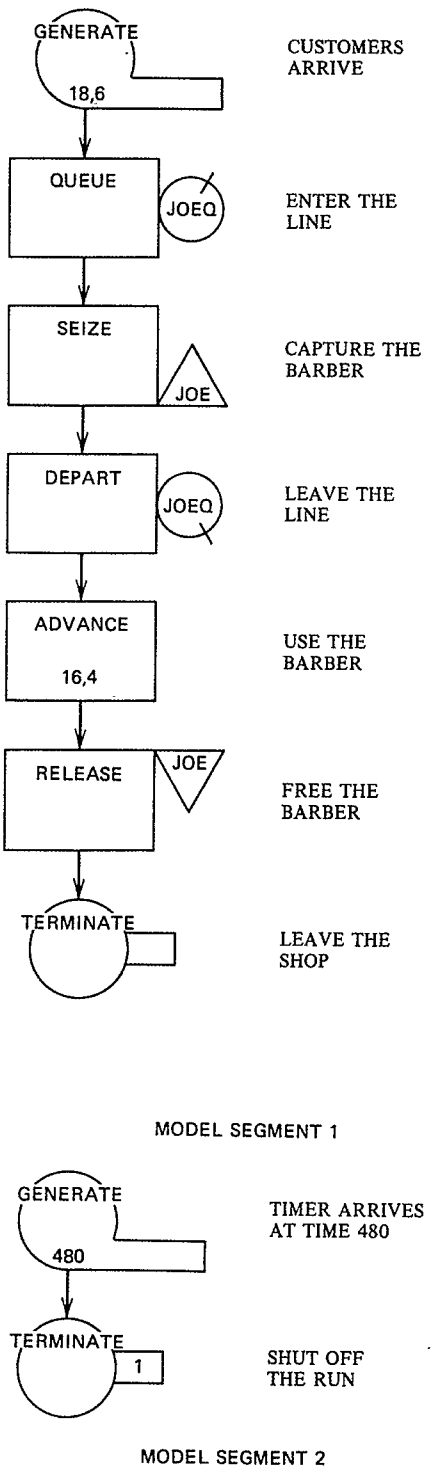


FIGURE 2A.1 Block Diagram for Case Study 2A

(5) Extended Program Listing

LOCATION	OPERATION	A,B,C,D,E,F
	SIMULATE	
*		
*	MODEL SEGMENT 1	
*		
	GENERATE	18,6 CUSTOMERS ARRIVE
	QUEUE	JOEQ ENTER THE LINE
	SEIZE	JOE CAPTURE THE BARBER
	DEPART	JOEQ LEAVE THE LINE
	ADVANCE	16,4 USE THE BARBER
	RELEASE	JOE FREE THE BARBER
	TERMINATE	LEAVE THE SHOP
*		
*	MODEL SEGMENT 2	
*		
	GENERATE	480 TIMER ARRIVES AT TIME 480
	TERMINATE	1 SHUT OFF THE RUN
*		
*	CONTROL CARDS	
*		
	START	1 START THE RUN
	END	RETURN CONTROL TO OPERATING SYSTEM

(a)

BLOCK NUMBER	*LOC	OPERATION	A,B,C,D,E,F,G	COMMENTS	CARD NUMBER
		SIMULATE			1
	*				2
	*	MODEL SEGMENT 1			3
	*				4
1		GENERATE	18,6	CUSTOMERS ARRIVE	5
2		QUEUE	JOEQ	ENTER THE LINE	6
3		SEIZE	JOE	CAPTURE THE BARBER	7
4		DEPART	JOEQ	LEAVE THE LINE	8
5		ADVANCE	16,4	USE THE BARBER	9
6		RELEASE	JOE	FREE THE BARBER	10
7		TERMINATE		LEAVE THE SHOP	11
	*				12
	*	MODEL SEGMENT 2			13
	*				14
8		GENERATE	480	TIMER ARRIVES AT TIME 480	15
9		TERMINATE	1	SHUT OFF THE RUN	16
	*				17
	*	CONTROL CARDS			18
	*				19
		START	1	START THE RUN	20
		END		RETURN CONTROL TO OPERATING SYSTEM	21

(b)

FIGURE 2A.2 The Case Study 2A model as submitted, and the corresponding Extended Program Listing (a) Completed coding sheet for the punchcard version of the model. (b) Extended Program Listing produced for the model in (a)

(6) Program Output

```
*
*       MODEL SEGMENT 1
*
  1     GENERATE    18     6
  2     QUEUE       1
  3     SEIZE       1
  4     DEPART      1
  5     ADVANCE     16     4
  6     RELEASE     1
  7     TERMINATE
*
*       MODEL SEGMENT 2
*
  8     GENERATE    480
  9     TERMINATE   1
*
*       CONTROL CARDS
*
        START       1
```

(a)

```
FACILITY SYMBOLS AND CORRESPONDING NUMBERS

                1    JOE
```

(b)

```
QUEUE SYMBOLS AND CORRESPONDING NUMBERS

                1    JOEQ
```

(c)

```
RELATIVE CLOCK          480  ABSOLUTE CLOCK          480

BLOCK COUNTS
BLOCK  CURRENT    TOTAL     BLOCK  CURRENT    TOTAL     BLOCK  CURRENT    TOTAL
    1        0       27
    2        1       27
    3        0       26
    4        0       26
    5        1       26
    6        0       25
    7        0       25
    8        0        1
    9        0        1
```

(d)

FACILITY	AVERAGE UTILIZATION	NUMBER ENTRIES	AVERAGE TIME/TRAN	SEIZING TRANS. NO.	PREEMPTING TRANS. NO.
JOE	.860	26	15.884	3	

(e)

QUEUE	MAXIMUM CONTENTS	AVERAGE CONTENTS	TOTAL ENTRIES	ZERO ENTRIES	PERCENT ZEROS	AVERAGE TIME/TRANS	$AVERAGE TIME/TRANS	TABLE NUMBER	CURRENT CONTENTS
JOEQ	1	.160	27	12	44.4	2.851	5.133		1

$AVERAGE TIME/TRANS = AVERAGE TIME/TRANS EXCLUDING ZERO ENTRIES

(f)

FIGURE 2A.3 Selected Program Output for Case Study 2A. (a) Assembled model. (b) Symbol dictionary for Facilities. (c) Symbol dictionary for Queues. (d) Clock values and Block Counts. (e) Facility statistics. (f) Queue statistics

(7) Discussion

Model Logic. In the model presented here, no provision is made for "removing customers from the barber shop" when the simulation shuts off at time 480. If the barber were to be true to the model, he would simply have to "walk out of the shop" at the end of his 8-hour day. Conversely, if the model were to be true to the barber, it would simulate locking the door after 8 hours, but the simulation would not stop until all customers already in the shop at that time had been serviced. It will eventually be seen how this latter approach can be implemented in GPSS.

Model Implementation. The coding sheet from which the punchcard version of the model was prepared is shown in Figure 2A.2(a). The corresponding Extended Program Listing produced by the Processor appears in Figure 2A.2(b). Notice how the Processor has augmented the original information in producing the Extended Program Listing. The extensions consist of the "Block Number" and "Card Number" columns appearing at the extreme left and right, respectively, in Figure 2A.2(b). Inspection of the Block Number column shows that Block Numbers have been assigned, in sequence, to each punchcard representing a Block image. In the Card Number column, note that each card in the deck has been assigned a sequence number.

"Comments" have been used liberally to document the model. Cards 2, 3, 4, 12, 13, 14, 17, 18, and 19 in Figure 2A.2(b) are comments cards which set off the model segments, and the Control Card segment. An asterisk (*) has been entered in column 1 on each of these cards. The Block-image punchcards also carry comments in the Operands field. These comments are identical to the annotations written next to the corresponding Blocks in the Figure 2A.1 Block Diagram.

Card 1 in Figure 2A.2(b) is the SIMULATE card. If the analyst is submitting a deck to have a run made, this card usually must be the first one the Processor encounters when it inputs the deck. The card consists of the single word SIMULATE, punched in the Operation field. If the SIMULATE card is absent, the Processor checks the deck for violations of the language rules, but makes no run with the model.

As stated in Section 2.9, the Processor starts the simulation when it finds a START card in the model. A START card has been placed, then, at the end of the model (Card 20). A "1" has been entered as the A Operand on the START card.

After a run shuts off, the computer session is not necessarily finished. Many additional options remain open to the analyst. Whether or not these options are exercised, the analyst eventually reaches the point at which all instructions for the run have been included in the deck. At this point, he puts in an END card. This card instructs the Processor to return control to the operating system. The END card appears after the START card in Figure 2A.2(b). It consists of the word END, punched in the Operation field.

The order of the cards *within a model segment* is critical, but the relative ordering of model segments within the card deck is not. For example, the timer segment could have been placed ahead of the major segment in Figure 2A.2 without having any effect on the model. If this had been done, the Extended Program Listing would appear as shown in Figure 2A.4.

Program Output.[11] It is not evident from examining either the Block Diagram or the Extended Program Listing how any output is produced by the model. At the end of a simulation, the GPSS Processor *automatically* prints out an extensive set of information pertaining to the model. This information includes statistics for each of the various entities used, i.e., for Facilities and Queues (and other entity types not yet discussed).

Most of the output produced by running the Figure 2A.2(a) model is shown in Figure 2A.3. In part (a) of that figure is displayed the assembled model. It has four noticeable features.

1. The absolute Block Numbers assigned by the Processor appear in the assembled model. The numbers 1 through 9 in the left column in Figure 2A.3(a) are these Block Numbers.

2. Instead of appearing in consecutive columns and being separated by commas, the Operands have been printed left-justified in adjacent six-column fields, and the commas have been eliminated. (It is not immediately evident in Figure 2A.3(a) that six-column fields have been used to display the Operands.)

3. All symbolic entity names in the model have been replaced with the corresponding numeric equivalents assigned by the Processor. Hence, the A Operand of the QUEUE Block (Block 2) is 1, not "JOEQ"; the A Operand of the SEIZE Block (Block 3) is 1, not

[11] The total CPU time required by the simulation on an IBM 360/67 computer was 1.6 seconds. Computer time requirements for GPSS simulations are discussed in Chapter 4.

```
BLOCK                                                                    CARD
NUMBER   *LOC   OPERATION   A,B,C,D,E,F,G          COMMENTS              NUMBER
                SIMULATE                                                    1
         *                                                                  2
         *      MODEL SEGMENT 2                                             3
         *                                                                  4
1                GENERATE    480           TIMER ARRIVES AT TIME 480       5
2                TERMINATE   1             SHUT OFF THE RUN                6
         *                                                                  7
         *      MODEL SEGMENT 1                                             8
         *                                                                  9
3                GENERATE    18,6          CUSTOMERS ARRIVE                10
4                QUEUE       JOEQ          ENTER THE LINE                  11
5                SEIZE       JOE           CAPTURE THE BARBER              12
6                DEPART      JOEQ          LEAVE THE LINE                  13
7                ADVANCE     16,4          USE THE BARBER                  14
8                RELEASE     JOE           FREE THE BARBER                 15
9                TERMINATE                 LEAVE THE SHOP                  16
         *                                                                  17
         *      CONTROL CARDS                                               18
         *                                                                  19
                START       1             START THE RUN                   20
                END                       RETURN CONTROL TO OPERATING SYSTEM 21
```

FIGURE 2A.4 Extended Program Listing for Case Study 2A, with Model Segments interchanged

"JOE", and so on. (In Chapter 4, the method the Processor uses to establish a correspondence between symbolically named entities and their numeric equivalents will be described.)

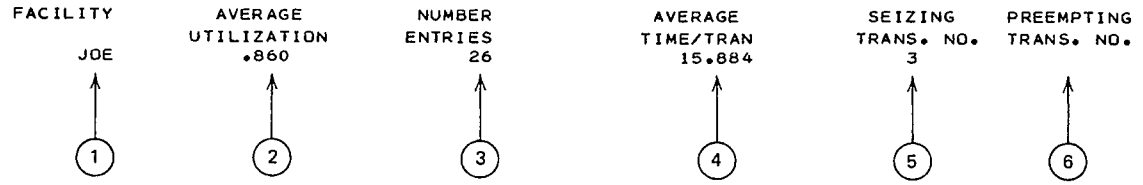
4. "Comments" entered on punchcard images of Blocks have been suppressed. "Pure" comments cards (that is, cards with an asterisk entered in card column 1) have not been suppressed, however. They are reproduced in their entirety in the printout of the assembled program.

Parts (b) and (c) in Figure 2A.3 show symbol dictionaries for Facilities and Queues. In the symbol dictionary for Facilities, the numeric equivalent assigned by the Processor for all symbolically named Facilities is shown. Hence, the Facility symbolically named JOE is Facility 1 in the assembled model; and the Queue symbolically named JOEQ is Queue 1. This is consistent with the A Operands for the SEIZE–RELEASE and QUEUE–DEPART Blocks in Figure 2A.3(a). If any Block Locations had been named symbolically in the model, a corresponding symbol dictionary also would have been provided in the output. Actually, if symbolic Location names have been used, the correspondence between them and Location numbers is apparent in the Extended Program Listing.

Figure 2A.3(d) shows clock values and Block Counts. As indicated in the top line of that figure, there are two clocks, the "Relative Clock" and the "Absolute Clock." The distinction between these two clocks will be explained later. For now, it is enough to note that both clocks show values of 480 in Figure 2A.3(d). This simply means that the simulation shut off at simulated time 480.

Immediately under the clock line in Figure 2A.3(d) are shown the Block Counts. This information appears in three columns, "Block Numbers" (labeled simply as BLOCK in the figure), "Current Count" (shown as CURRENT), and "Total Count" (shown as TOTAL). The Block Numbers correspond to those shown in Figure 2A.3(a). The Current Count is the count of Transactions in the corresponding Blocks at the time the simulation shut off. The Total Count is a count of the total number of Transactions which entered the corresponding Blocks during the simulation, *including* those that are still in the Block (if any). For example, the Total Count at Block 1 is 27, meaning that 27 Transactions entered the model through the Location 1 GENERATE Block. Similarly, the Total Count at Block 2 is 27, meaning that 27 Transactions moved into the QUEUE Block in Location 2. The Current Count at Block 2 is 1, meaning that one Transaction is still in the QUEUE Block, i.e., one customer was waiting for the barber when the model shut off. At the Block in Location 5, the ADVANCE Block, the Current Count is 1 and the Total Count is 26. That is, 26 customers have captured the barber; of the 26, one still has him captured. The Total Counts at the SEIZE and RELEASE Blocks are 26 and 25, respectively, which is consistent with the ADVANCE Block Counts.

In Figure 2A.3, parts (e) and (f) show the statistics gathered for the Facility JOE and the Queue JOEQ. The Facility statistics are shown again in Figure 2A.5, where the columns have been numbered for ease of reference. The Table appearing in the lower part of Figure 2A.5 in-



Column	Significance	Column	Significance
1	Names (numeric and/or symbolic) of the various Facilities used in the model	5	Number of the Transaction (if any) which currently has the Facility captured. (Transaction numbers are discussed later in this chapter.)
2	Fraction of the time that the corresponding Facilities were in a state of capture during the simulation	6	Number of the Transaction (if any) which currently has the Facility preempted. (Preemption will not be explained until Chapter 7.)
3	Number of captures		
4	Average holding time per capture		

FIGURE 2A.5 Interpretation of the information shown in Figure 2A.3(e)

dicates the significance of the entries in the various columns. Similarly, in Figure 2A.6, the Queue statistics have been repeated with column numbers included. The Table at the bottom of that figure indicates the meaning of the various Queue statistics. The tables in Figures 2A.5 and 2A.6 should be studied, making reference to the output immediately above them in the process. Note these features of the information provided in those figures.

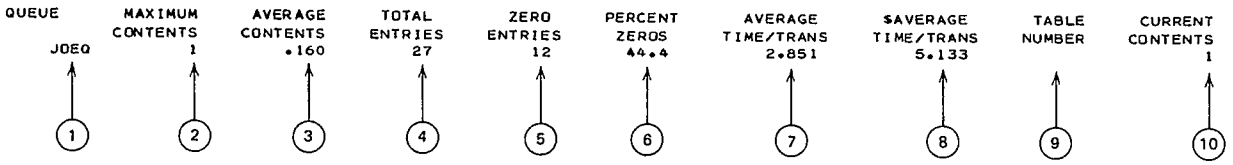
1. Joe was in use 86 percent of the time (AVERAGE UTILIZATION = .860).

2. JOE was captured 26 times (NUMBER ENTRIES = 26). This is consistent with the previously noted Total Count of 26 for the SEIZE Block.

3. The average holding time per capture of JOE was 15.884 minutes (AVERAGE TIME/TRAN = 15.884).

4. Transaction number 3 had JOE in a state of capture when the simulation shut off (SEIZING TRANS. NO. = 3). The fact that JOE was "in use" when the simulation shut off is consistent with the previously-noted Current Count of 1 at the ADVANCE Block. As for Transaction "numbers," they will be discussed in Section 2.21.

5. There was never more than one customer in the Queue JOEQ (MAXIMUM CONTENTS = 1).



Column	Significance	Column	Significance
1	Names (numeric and/or symbolic) of the various Queues used in the model	7	Average time that each Queue entry spent waiting in the Queue (zero entries are *included* in this average)
2	Largest value the record of Queue content ever assumed	8	Average time that each Queue entry spent waiting in the Queue (zero entries are *excluded* from this average)
3	Average value of the Queue content	9	Name (numeric and/or symbolic) of the GPSS Table in which the distribution of Queue residence time is being tabulated. (The Table concept is not discussed until Chapter 4.)
4	Total number of entries to the Queue		
5	Total number of entries to the Queue which experienced no waiting ("zero entries")	10	Current value of the Queue content
6	Percentage of total Queue entries which experienced no waiting		

FIGURE 2A.6 Interpretation of the information shown in Figure 2A.3(f)

6. The average number of customers in the waiting line was .160 (AVERAGE CONTENTS = .160).

7. The total number of entries to the waiting line was 27 (TOTAL ENTRIES = 27).

8. Included among the 27 total entries to the waiting line were 12 zero entries (ZERO ENTRIES = 12).

9. Of the total entries to the waiting line, 44.4 percent of them were zero entries (PERCENT ZEROS = 44.4).

10. The average residence time in the waiting line per entry (*including* zero entries) was 2.851 minutes (AVERAGE TIME/TRANS = 2.851).

11. The average residence time in the waiting line *per nonzero entry* was 5.133 minutes ($AVERAGE TIME/TRANS = 5.133).

12. At the time the simulation shut off, there was one Transaction in the waiting line (CURRENT CONTENTS = 1). This is consistent with the previously-noted Current Count of 1 at the QUEUE Block.

The statistical measures in Figures 2A.5 and 2A.6 are highly intuitive in meaning, then, and are almost without need of definition. This is especially true for *Facilities*. Because only one Transaction at a time can use a Facility, NUMBER ENTRIES is a direct count of the number of Transactions which captured the Facility, and AVERAGE TIME/TRAN is the average time that each capturing Transaction held the Facility. The same simple comments apply to Queue statistics *if* the B Operand at the QUEUE and DEPART Blocks is 1 (as is true by default in the Case Study 2A model). Recall from the section on Queues, however, that the Processor computes Queue statistics with respect to "units of content," *not* with respect to "Transactions." In Case Study 2A (and throughout this book), each Transaction moving through a Queue contributes exactly one unit of content. If this were not the case, the following extended interpretation would have to be applied to Queue statistics.

1. TOTAL ENTRIES is the number of "units of content" which entered the Queue. More precisely, TOTAL ENTRIES is the value of a counter initialized at zero, and incremented by an amount equal to the QUEUE Block's B Operand each time the QUEUE Block is executed. Except when the QUEUE Block's B Operand is 1, this value does not equal the total number of Transactions which became Queue members during the simulation.

2. ZERO ENTRIES is the number of "units of content" which spent zero residence time in the Queue. More precisely, ZERO ENTRIES is the value of a counter initialized at zero, and incremented by an amount equal to the DEPART Block's B Operand each time the DEPART Block is executed by a Transaction whose residence time in the Queue is zero. Except when the DEPART Block's B Operand is 1, this value does not equal the number of Transactions which became Queue members, and then experienced zero residence time in the Queue.

3. AVERAGE TIME/TRANS is the average residence time in the Queue *per "unit of content."* Fortunately, this value is identical to the "average Queue residence time *per Transaction*," assuming that each Transaction moving through the Queue *decrements* the "current content" by the same amount that it *incremented* the "current content" earlier. If this condition is not satisfied, then the label AVERAGE TIME/TRANS is misleading.

4. Similarly, the labels MAXIMUM CONTENTS, AVERAGE CONTENTS, PERCENT ZEROS, $AVERAGE TIME/TRANS, and CURRENT CONTENTS must all be interpreted with respect to "units of contents," and not with respect to "Transactions," except of course when QUEUE and DEPART B Operands are 1.

Another feature of both Facility and Queue statistics should be noted. If a Facility is in a state of capture when Facility statistics are printed out, there is a *downward bias* in the AVERAGE TIME/TRAN statistic. This is because AVERAGE TIME/TRAN is computed by dividing NUMBER ENTRIES into the total simulated time during which the Facility was in a state of capture. If there is a current user who is not yet done when the simulation stops, then what would have been his *entire* holding time is not taken into account in computation of the AVERAGE TIME/TRAN statistic. The same observation can be made with respect to Queues. AVERAGE TIME/TRANS is computed for Queues by dividing TOTAL ENTRIES into total Queue residence time. If the Queue has CURRENT CONTENTS when the simulation stops, then they have not yet contributed their full measure to total Queue residence time. This results in a downward bias in AVERAGE TIME/TRANS, and also in $AVERAGE TIME/TRANS.

Finally, as will be explained in Section 2.21, the earliest simulated time at which Transactions can experience movement in a model is 1. This means that the content of all Queues in a model is necessarily zero during the simulated time interval from 0 to 1, all Facilities are necessarily "available" during the simulated time interval from 0 to 1, etc. Because statistics such as "average Queue content," "Facility utilization," and so on, are computed as though the simulation started at time 0, a slight bias may consequently be introduced into these statistics.

REFERENCES

(1) Christy, D.P., and H.J. Watson, "The Application of Simulation: A Survey of Industry Practice," Interfaces, October, 1983.

(2) Schriber, T.J., Simulation Using GPSS, John Wiley & Sons, New York NY, 1974.

(3) Pritsker, A.A.B., Introduction to Simulation and SLAM II, 2nd ed., Halsted Press, New York NY, 1984.

(4) Pegden, C.D., Introduction to SIMAN, Systems Modeling Corporation, State College PA, 1982.

(5) Russell, E.C., Building Simulation Models with SIMSCRIPT II.5, CACI Inc., Los Angeles CA, 1983.

(6) Birtwistle, G.M., Discrete Event Modeling in Simula, McMillan and Co., London, England, 1979.

(7) Henriksen, J.O., "State-of-the-Art GPSS," In: Proceedings of the 1983 Summer Computer Simulation Conference, Society for Computer Simulation, San Diego CA, 1983.

(8) Henriksen, J.O., and R.C. Crain, GPSS/H User's Manual, 2nd Ed., Wolverine Software Corporation, Annandale VA, 1983.

(9) Bobillier, P.A., B.C. Kahan, and A.R. Probst, Simulation with GPSS and GPSS/V, Prentice-Hall, Englewood Cliffs NJ 1976.

(10) Donovan, T.M., GPSS Simulation Made Simple, Wiley-Interscience, Chichester, England, 1976.

(11) Gordon, G., The Appliclation of GPSS V to Discrete Systems Simulation, Prentice-Hall, Englewood Cliffs NJ, 1975.

(12) Greenberg, S., GPSS Primer, Wiley-Interscience, New York NY, 1972.

(13) Sulzer, J.R., P. Bouteille, et. al., La Simulation: Initiation Pratique au GPSS, Enterprise Moderne D'Edition, Paris, 1970.

(14) Weber, K., R. Trzebiner, and H. Tempelmeier, Simulation mit GPSS, Verlag Paul Haupt, Stuttgart, West Germany, 1983.

(15) Banks, J., and J.S. Carson, Discrete-Event System Simulation, Prentice-Hall, Englewood Cliffs NJ, 1983.

(16) Bratley, P., B.L. Fox, and L.E. Schrage, A Guide to Simulation, Springer-Verlag, New York NY, 1983.

(17) Fishman, G.S., Principles of Discrete Event Simulation, Wiley-Interscience, New York NY, 1978.

(18) Maisel, H., and G. Gnugnoli, Simulation of Discrete Stochastic Systems, Science Research Associates, Palo Alto CA, 1972.

(19) McMillan, C., and R.F. Gonzalez, Systems Analysis, 3rd Ed., Richard D. Irwin, Homewood IL, 1973.

(20) Solomon, S., Simulation of Waiting Lines, Prentice-Hall, Englewood Cliffs NJ, 1983.

(21) Andrews, R.W., and T.J. Schriber, "Interactive Analysis of Output from GPSS-based Simulations," In: H.J. Highland, N.R. Nielsen, and L.G. Hull (eds.), Proceedings of the 1978 Winter Simulation Conference, Society for Computer Simulation, San Diego CA, 1978.

(22) Degen, R.J., and T.J. Schriber, "On the Feasibility of Using GPSS Models in the Simulation of Hierarchical Control Systems," In: H.J. Highland, T.J. Schriber, and R.G. Sargent (eds.), Proceedings of the 1976 Winter Simulation Conference, Society for Computer Simulation, San Diego CA, 1976.

(23) Lefkowits, R.M., and T.J. Schriber, "Use of an External Optimizing Algorithm with a GPSS Model," In: L. Boelhouwer (ed.), Proceedings of the 1971 Winter Simulation Conference, Association for Computing Machinery, New York NY, 1971.

(24) Schriber, T.J., and R.W. Andrews, "Interactive Analysis of Simulation Output by the Method of Batch Means," In: H.J. Highland, M. Spiegel, and R. Shannon (eds.), Proceedings of the 1979 Winter Simulation Conference, Association for Computing Machinery, New York NY, 1979.

(25) Henriksen, J.O., "GPSS - Finding the Appropriate World-View," In: T.I. Oren, C.M. Delfosse, and C.M. Shub (eds.), Proceedings of the 1981 Winter Simulation Conference, Society for Computer Simulation, San Diego CA, 1981.

(26) Banks, J., and J.S. Carson II, "Process-Interaction Simulation Languages," Simulation, Vol. 44, No. 5, May 1985.

(27) Abed, S.Y., T.A. Barta, and K.L. McRoberts, "A Qualitative Comparison of Three Simulation Languages: GPSS/H, SLAM, SIMSCRIPT," Computers & Industrial Engineering, Vol. 9, No. 1, 1985.

(28) Abed, S.Y., T.A. Barta, and K.L. McRoberts, "A Quantitative Comparison of Three Simulation Languages: GPSS/H, SLAM, SIMSCRIPT," Computers & Industrial Engineering, Vol. 9, No. 1, 1985.

International Business Machines, GPSS V User's Manual, (SH20-0851), IBM Inc., Armonk NY, 1970.

(30) Cox, S., GPSS/PC User's Manual, Minuteman Software, Stow MA, 1984.

(31) Martin, D., GPSSR/PC, GPSS/VX, GPSS/C, GPSSR, and GPSS10 Reference Manuals, Simulation Software Ltd., London, Ontario, Canada.

(32) Schmidt, B., GPSS-FORTRAN, John Wiley & Sons, Inc., New York NY, 1980.

(33) International Business Machines, APL GPSS (G320-5745 and SH20-1942), IBM Inc., Armonk NY, 1977.

(34) International Business Machines, PL/1 GPSS (G320-6398), IBM Inc., Armonk NY, 1981.

(35) Rubin, J., "Embedding GPSS in a General Purpose Language," In: T.I. Oren, C.M. Delfosse, and C.M. Shub (eds.), Proceedings of the 1981 Winter Simulation Conference, Society for Computer Simulation, San Diego CA, 1981.

THOMAS J. SCHRIBER is Professor and Chairman of Computer and Information Systems at The University of Michigan. His principal area of interest is discrete-event simulation, and includes the development, delivery, and dissemination of teaching materials in this area; the application of simulation modeling in problem-solving situations; simulation-related research; and professional service. He has written articles and a book on modeling using GPSS, and regularly teaches intensive courses on GPSS-based simulation. He has consulted with such organizations as Monsanto Chemical, CPC International, SRI International, ITT, Ford Motor Company, Exxon, General Motors, and Occidental Petroleum, both in a problem-solving and a teaching capacity. His simulation-related research currently centers on the analysis of output from simulation models. Professor Schriber has been a National ACM Lecturer, and has participated in the Joint Science and Technology Exchange Agreement between the United States and the U.S.S.R. He is the ACM member of the Board of Directors of the Winter Simulation Conferences, is a former Board Chairman, and was Program Chairman of the 1976 Bicentennial Winter Simulation Conference as well as co-editor (with Harold H. Highland and Robert G. Sargent) of the proceedings of that conference. His professional affiliations include ACM, AIDS, SCS, and TIMS.

Graduate School of Business Administration
The University of Michigan
Ann Arbor MI 48109-1234
313-764-1398