Proceedings of the 1988 Winter Simulation Conference
M. Abrams, P. Haigh, and J. Comfort (eds.)

Perspectives on simulation using GPSS

Thomas J. Schriber
Graduate School of Business Administration
The University of Michigan
Ann Arbor MI 48109-1234 USA

ABSTRACT

A broad overview of the simulation modeling language GPSS is given. The class of problems to which GPSS applies especially well is described, and commentary on the semantics and syntax of the language is offered. Various GPSS implementations are indicated, and vendor information is given. The GPSS learning-oriented literature is reviewed, and sources comparing GPSS and other simulation languages are cited. Professional GPSS courses are listed. A GPSS model and its output are presented and discussed in an appendix.

The GPSS tutorial at the 1988 Winter Simulation Conference will show how one goes about building GPSS models. Paper copies of the tutorial transparencies will be given out at the tutorial.

1. GPSS IN BRIEF

GPSS (General Purpose Simulation System) is a popular (Christy and Watson, 1983) 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 modeling systems in which discrete units of traffic compete for scarce resources (e.g., queuing systems), and is useful in determining how well such systems will respond to the demands placed on them. GPSS has been applied, for example, to the modeling of manufacturing systems, communication systems, computing systems, transportation systems, inventory systems, and health-care systems, and has been used in chemical engineering, mining engineering, and cancer research.

2. THE SEMANTICS AND SYNTAX OF GPSS

GPSS offers a rich set of semantics, and yet is sparse in its syntax. For example, only seven statements (plus several run-control statements) are required to model a 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 for 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 content; maximum content; average time in line), etc. This limited example is roughly suggestive of the character of GPSS. A GPSS model for the one-line, one-server system is given here in an appendix.

The sparse syntax of GPSS, coupled with its block-diagram orientation, makes it possible for the beginner to learn quickly a usable subset of the language. Because GPSS is rich and versatile, however, considerable study is required to master the language.

The GPSS world view (stylized way of looking at a problem) involves visualizing units of traffic ("transactions") which move along paths in a model as a simulation proceeds. This world view is so natural to the modeling of queuing systems that many other simulation languages have adopted it. The effect of this cross-fertilization can be found in such languages as SIMAN (Pegden 1982), SIMSCRIPT (Russell 1983), SIMULA (Birtwistle 1979), and SLAM (Pritsker 1986).

3. VARIOUS GPSS IMPLEMENTATIONS

GPSS is a multi-vendor language. (This is in contrast with such languages as SIMAN, SIMULA, SLAM, and SIMSCRIPT.) First released by International Business Machines (IBM) in 1961, GPSS evolved through a series of IBM releases (GPSS II; GPSS III; GPSS/360; and, in 1970, GPSS V (IBM 1970)), each an enhancement of its predecessor. Paralleling the IBM releases, a number of GPSS implementations were done for IBM and non-IBM hardware by non-IBM vendors. (See GPSS VENDOR INFORMATION below.)

Wolverine Software's GPSS/H, which is an upwardly compatible superset of IBM's GPSS V, is the state-of-the-art GPSS for IBM mainframes and compatible computers (e.g., the Amdahl 470 series, the Amdahl 5860, and National Advanced Systems' NAS-9000) (Henriksen et al. 1988). Written in assembly language, mainframe GPSS/H (Release 1, 1977; Release 2, 1988) also runs on the IBM PC/AT/370.

GPSS/H can be used on VAX computers as well, including the MicroVAX I, MicroVAX II, and MicroVAX 2000 ("desktop VAX"); and the 11/7xx and 8xxx machines. It also runs on Apollo; on Integrated Solutions' Optimum 5/10 and Optimum V; on the NCR Tower32; on Silicon Graphics' IRIS and IRIS Turbo; and on Sun Microsystems' Sun-3. These non-mainframe implementations are written in "C". An MS/DOS "C" implementation of GPSS/H which will run models of moderate size is also available. GPSS/H models can be transported across all GPSS/H implementations (subject to possible size constraints if downloading to MS/DOS GPSS/H).

MINUTEMAN Software vends GPSS/PC, which is a popular implementation of GPSS for the IBM PC. GPSS/PC was released in Version 1 in 1984 and in Version 2 in 1986 (Cox 1986, 1987). Version 3 is expected in 1989.

Another GPSS implementation for the IBM PC is Simulation Software Ltd.'s GPSSR/PC (Richards 1981, 1983). Simulation Software Ltd. also offers two other GPSS implementations: GPSS/VX, for VAX/VMS systems and MicroVAX systems (Martin

1981); and GPSS/C, for such 32-bit architecture computer systems as VAX UNIX, ELXSI UNIX, SUN-3 UNIX, PYRAMID UNIX, NCR Tower UNIX, Data General MV/ECLIPSE, and the HP9000 Series 500 (Richards 1984).

Comments on the GPSS-FORTRAN (Schmidt 1987) offered by a German company, Dr. Staedtler Gmbh, are given below under LANGUAGES WITH GPSS EMBEDDED.

No one keeps a comprehensive list of current GPSS implementations. In general, people not in a position to use IBM's GPSS V, MINUTEMAN's GPSS/PC, a Simulation Software Ltd. GPSS implementation, Staedtler's GPSS-FORTRAN, or Wolverine's GPSS/H, must do their own spadework to determine if a supported GPSS implementation is available for their computer system.

4. GPSS, GRAPHICS, AND ANIMATION

MINUTEMAN Software's GPSS/PC, Version 2, provides built-in graphic and simulation animation features, including showing transaction movement in block diagrams; animation of movement of objects in two-dimensional representations of the system being modeled; and dynamic display of statistical aspects of a model, e.g., histograms portraying the ongoing realization of a random variable's relative frequency function; and plots of the time series of values being taken on by variables of interest (Cox, 1987).

Simulation Software Ltd.'s GPSSR/PC also provides graphics and animation features, including many which are functionally equivalent to those described in the preceding paragraph.

Wolverine Software's GPSS/H supports TESS (The Extended Simulation System; Standridge 1985), which provides a relational database manager, a graphics generator, a forms processor, a graphical network builder, and a library of FORTRAN subroutines for manipulating individual data items or data summaries.

AutoSimulations, Inc., offers AUTOGRAM (AutoSimulations, Inc., 1986), which works with output from Wolverine Software's GPSS/H to provide three-dimensional color animation of the system being modeled. AutoSimulations, Inc., also offers AUTOMOD (AUTOmatic MODel generator), a preprocessor for the GPSS/H compiler. AUTOMOD (AutoSimulations, Inc., 1986) converts high level system descriptions into GPSS/H statements, and then passes them on to the GPSS/H compiler.

5. LANGUAGES WITH GPSS EMBEDDED

The functions performed by the GPSS blocks have been embedded in other languages in several cases. Embedding takes the form of implementing the functions of the GPSS blocks and run-control statements in a host language as subroutines which augment the power of the host language. Calling these subroutines has the effect of simulating the GPSS blocks and run-control statements. For a paper on embedding, see Rubin (1981).

An instance of such embedding is GPSS-FORTRAN (Schmidt 1987), which sees use especially in Germany and Europe. GPSS-FORTRAN, Version 3, which supports continuous modeling and combined discrete-continuous modeling, as well as discrete-event modeling, can be used in batch mode and interactively, and in real time. It can be run on computer systems which have a FORTRAN compiler.

Other examples of embedding are APL-GPSS (IBM 1977) and PL/1-GPSS (IBM 1981). (These implementations may no longer be supported by IBM.)

6. GPSS VENDOR INFORMATION

Vendor addresses and phone numbers are given below. (Please report omissions to Thomas J. Schriber.)

1. International Business Machines, Inc.
   (GPSS V)
   Contact your local IBM representative.

2. MINUTEMAN Software Inc.
   (GPSS/PC, Version 2)
   P.O. Box 171
   Stow MA 01775

   Phone: 617-897-5662

3. Simulation Software Ltd.
   (GPSS/C; GPSSR/PC; GPSS/VX)
   760 Headley Drive
   London, Ontario, Canada N6H 3V8

   Phone: 519-657-8229

4. Dr. Staedtler Gmbh
   (GPSS-FORTRAN)
   Muenchener Strasse 342
   8500 Nuernberg 50 West Germany

   Phone: 49/911/86-80-81
   (49 is West Germany; 911 is Nuernberg)

5. Wolverine Software Corporation
   (GPSS/H)
   7630 Little River Turnpike
   Annandale VA 22003-2653

   Phone: 703-750-3910

Contact vendors for current leasing and/or purchase prices and academic and quantity discount policies.

7. FIVE MYTHS ABOUT GPSS

GPSS was first released (by IBM) in 1961. Those familiar with early versions of the language (including IBM's GPSS V, released in 1970, and now no longer up to date), but who haven't kept up with state-of-the-art GPSS, may suffer from one or more major misconceptions ("myths") about GPSS (Henriksen 1983):

1. **Misconception:** "GPSS is inherently slow."

   Observation: Many early interpretive versions of GPSS *are* slow. In contrast, some state-of-the-art implementations produce compiled code and provide performance far superior to old versions. (For example, Wolverine Software's GPSS/H executes 5 to 6 times faster on average than IBM's GPSS V.) Some other state-of-the-art versions, although not producing compiled code, generate an intermediate level of code, making it unnecessary to reinterpret each statement each time it is acted upon (e.g., MINUTEMAN Software's GPSS/PC). (For a published comparison of compilation and execution rates of Wolverine Software's GPSS/H vs. SIMSCRIPT and SLAM, see GPSS AND OTHER SIMULATION LANGUAGES, below.)

2. **Misconception:** "To do anything sophisticated in GPSS, HELP blocks must be used to combine GPSS models with FORTRAN routines."

   (Note: "HELP blocks" can be used in GPSS modeling to interface an executing GPSS model with one or more external routines written in such other languages as FORTRAN, C, PL/1, or assembly language.)

   Observation: The power of state-of-the-art implementations of GPSS is such that FORTRAN (or other) routines are rarely required. Some current implementations (e.g., Wolverine Software's GPSS/H) include general purpose I/O statements, for example, which make it unnecessary to use HELP Blocks for I/O. And, when the use of FORTRAN routines is convenient for such things as obtaining t or z statistics, these routines can be directly invoked, without resorting to the use of HELP Blocks.

3. **Misconception:** "GPSS is trivial to learn."

   Observation: GPSS is trivial to learn only to a superficial depth. While rudiments of GPSS can be learned in a day, real mastery of GPSS requires considerable study (study at least equivalent to taking a three-credit course, or a five day intensive course) *and* practice.

4. **Misconception:** "Modeling difficulties arise more frequently due to language shortcomings than due to lack of modeler expertise."

   Observation: Misconceptions about the lack of power of GPSS come from people with an insufficient grasp of the language. According to Geoffrey Gordon (1978), who originally conceived GPSS, misconceptions about lack of power, where the real problem is a lack of user expertise, have been commonplace since the earliest versions of GPSS.

5. **Misconception:** "GPSS is batch oriented."

   Observation: It is of course true that early versions of GPSS (from circa 1961 to 1977) were batch oriented. In contrast, current versions are designed both for interactive and batch use (e.g., GPSS/H; GPSS/PC; GPSS/VX; GPSS/C; GPSSR/PC; GPSS-FORTRAN). State-of-the-art versions offer powerful interactive monitoring capabilities which greatly speed up the process of building GPSS models, debugging them, and verifying them.

8. THE GPSS LEARNING-ORIENTED LITERATURE

There are several GPSS books (Bobillier, Kahan, and Probst 1976; Cummings 1986; Donovan 1976; Gordon 1975; Greenberg 1972; Schmidt 1987; Schriber 1974; Sulzer and Bouteille 1970; Weber, Trzebiner, and Tempelmeier 1983). Overviews of GPSS can also be found in general simulation texts, e.g. Banks and Carson (1984); Bratley, Fox, and Schrage (1987); Fishman (1978); Law and Kelton (1982); and Solomon (1983).

The GPSS user's manuals may also contain good learning-oriented material. For example, an instructive set of HELP block examples and of built-in I/O use is given in Henriksen et al. (1988).

GPSS is flexible enough to support taking a number of approaches to modeling a system. Tradeoffs involved are discussed in Henriksen (1981; 1986), and in Henriksen and Schriber (1986).

The *Proceedings of the 1988* (or 1987, 1986, etc.) *Winter Simulation Conference* are good sources of papers on simulation applications, including applications of GPSS. Until sold out, copies of these proceedings can be purchased from The Society for Computer Simulation (P.O. Box 17900, San Diego, California 92117; phone 619-277-3888).

9. GPSS AND OTHER SIMULATION LANGUAGES

Introductory descriptions of Wolverine Software's GPSS/H, and of SIMAN, SIMSCRIPT II.5, and SLAM II, are given in Banks and Carson (1985). The world view of each language is described, and one and the same problem is modeled in each language.

Qualitative and quantitative comparisons of GPSS/H, SLAM, and SIMSCRIPT are given in Abed, Barta, and McRoberts (1985a, 1985b). 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 the 1985b article, p. 45). GPSS/H compiled *50* times faster than SIMSCRIPT and *10* times faster than SLAM. GPSS/H executed *3.8* times faster than SIMSCRIPT and *3.5* times faster than SLAM.

Guidelines for evaluating simulation software, and a good comparison and contrast of various simulation languages (including Wolverine Software's GPSS/H and MINUTEMAN Software's GPSS/PC) and packages in terms of these guidelines, can be found in Haider and Banks (1986).

10. PROFESSIONAL GPSS TRAINING COURSES

GPSS training courses are available from these five sources:

1. A four-day course featuring use of MINUTEMAN's GPSS/PC is offered every several months in Corvallis, Oregon. Contact:

   Mr. Gerald Airth
   West Coast GPSS Training
   1463 SW "A" Street
   Corvallis, Oregon 97333

   Phone: 503-754-7919

2. A five-day GPSS course is offered in November and March at the Georgia Institute of Technology. Contact:

   Professor Jerry Banks
   School of ISYE
   Georgia Institute of Technology
   Atlanta GA 30332

   Phone: 404-894-2312

3. Five-day GPSS courses are given each May at The Ryerson Polytechnical Institute in Toronto, Ontario, Canada. Contact:

   Professor R. Greer Lavery
   Ryerson Polytechnical Institute
   Math, Physics, and Computer Science
   350 Victoria
   Toronto, Ontario, Canada M5B 2K3

   Phone: 416-979-5000 XT 6972

4. Professor Thomas J. Schriber teaches intensive five-day GPSS courses five times each year. For information about the July offering, held in Ann Arbor, Michigan, phone The University of Michigan's Engineering Summer Conferences at 313-764-8490, or contact:

   Professor Thomas J. Schriber
   Computer and Information Systems - GSBA
   The University of Michigan
   Ann Arbor MI 48109-1234

   Phone: 313-764-1398

   For information about the November, February, and May offerings, held in such places as Washington, D.C. (November and May) and San Diego (February), contact either Professor Schriber or:

   Ms. Elizabeth Tucker
   Wolverine Software
   7630 Little River Turnpike
   Annandale VA 22003-2653

   Phone: 703-750-3910

   Professor Schriber's five-day course is also taught each June in Leuven (Louvain), Belgium (just outside Brussels). Taught in English, the course is hosted by Professor Dr. Maurice Verhelst and the University of Leuven's Dept. of Applied Economics. Contact Professor Schriber or Professor Dr. Verhelst:

   Professor Dr. Maurice Verhelst
   University of Leuven
   D.T.E.W.
   Dekenstraat 2
   B3000 Leuven Belgium

   Phone: 32/16/22-75-17
   (32 is Belgium; 16 is Leuven)

5. GPSS courses emphasizing applications in mining engineering are taught periodically in Australia and Las Vegas, Nevada. Contact:

   Professor John R. Sturgul
   School of Mining and Metallurgy
   South Australian Institute of Technology
   P.O. Box 1
   Ingle Farm 5098 South Australia

   Phone: 08/343/3248
   (08 is Australia)

11. THE GPSS TUTORIAL

In the GPSS tutorial at the Winter Simulation Conference, the fundamentals of queuing system logic and the modeling elements offered by GPSS to implement this logic will be introduced and illustrated. The tutorial will make use of transparencies, paper copies of which will be distributed to those in attendance. Others can obtain these materials from Professor Thomas J. Schriber (Graduate School of Business, The University of Michigan, Ann Arbor MI 48109-1234; 313-764-1398).

APPENDIX A: A GPSS MODEL FOR A ONE-LINE, ONE-SERVER QUEUING SYSTEM

This appendix shows a GPSS model for a one-line, one-server queuing system. The appendix consists of these sections:

A.1 Statement of the Problem

A.2 The Approach Taken in Building the Model

A.3 The GPSS Block Diagram for the Model

A.4 The GPSS Model File

A.5 Discussion of Selected Simulation Output

A.6 Replications in GPSS

A.1 Statement of the Problem

In a manufacturing system, castings are sent to a drilling machine, where each casting is to have a hole drilled in it. The interarrival time of castings at the machine is uniformly distributed over the interval 15.0 $\pm$ 4.5 minutes. The time required to drill a hole in a casting is 13.5 $\pm$ 3.0 minutes, uniformly distributed. Castings are processed in first-come, first-served order. Model this system in GPSS, making provision to collect queuing statistics for castings waiting their turn to be drilled. When the simulation starts, no castings are to be waiting to use the drill, and the drill is to be idle. Perform a single simulation with the model, simulating until holes have been drilled in 100 castings. Discuss the output produced at the end of the simulation. Finally, perform eight independent simulations with the model under the conditions described. Use the resulting output to compute 90% confidence intervals for the expected values of these three dependent random variables: (a) the time required to drill holes in 100 castings; and, during the time needed to drill holes in 100 castings: (b) the average number of castings in line; (c) the average time castings spend waiting in line.

A.2 The Approach Taken in Building the Model

Consider the time-ordered series of events associated with a casting as it moves through the one-line, one-server system:

1. The casting arrives at the system.

2. The casting requests the machine.

3. The casting waits, if necessary, to capture the machine. (If the machine is idle when the casting arrives, waiting time will be zero.)

4. When its turn comes, the casting captures the machine.

5. The casting holds the machine in a state of capture while the machine drills a hole in the casting.

6. The casting gives up control of the machine.

7. The casting leaves the system.

Castings can be thought of as units of traffic which move through the castings-and-machine system. The units of traffic in this system are conveniently simulated in GPSS by language elements known as "transactions". Transactions are units of traffic which are created and introduced into a model from time to time, move along a path in the model as the simulation proceeds, and then eventually are destroyed (leave the model). The experiences of transactions as they go through their life cycle in the castings-and-machine *model* are analogous to the experiences of castings as they go through the castings-and-machine *system*. Positioned on the path along which transactions move are *blocks*. Each block represents a subroutine. Movement of a transaction into a block causes the subroutine represented by the block to be executed. By choosing appropriate types of blocks, the GPSS modeler can easily build an appropriate path (sequence of blocks) for casting-transactions to move along to mimic the sequence of events outlined above.

The sequence of blocks begins with the type of block used to create transactions from time to time during a simulation and introduce them into a model, the GENERATE block. The time that elapses between introduction of consecutive transactions into a model by a GENERATE block is "interarrival time." In this model, the interarrival time random variable is uniformly distributed over the interval 15.0 $\pm$ 4.5 minutes. (15.0 $\pm$ 4.5 describes the interval ranging from 10.5 to 19.5.) The values 15.0 and 4.5 are provided in the model as GENERATE block *operands*. In programming language terms, a block's operands correspond to the arguments whose values are passed to a subroutine at the time of subroutine execution. (In general, arbitrarily complicated interarrival time distributions can be modeled at GENERATE blocks. This is done by defining functions which describe the applicable distribution, then using these functions as GENERATE-block operands.)

The sequence of blocks ends with a TERMINATE block. When a transaction moves into a TERMINATE block, the block subroutine destroys the transaction. A counter can be used with a TERMINATE block so that, after a specified destroy count has been reached (a count of 100 in this problem), a simulation will stop. (More generally, arbitrarily complicated stopping conditions can be specified in GPSS models.)

A SEIZE block is included in the sequence. A transaction requests control of a single server by trying to move into a SEIZE block. A SEIZE block operand is used to identify the single server. If the server is idle when a transaction requests it, the requesting transaction moves into the SEIZE block without delay and takes control of the server. But if the server is currently under the control of one transaction when another requests it, the requesting transaction cannot move into the SEIZE block. Instead, it remains in its current block and waits its turn to capture the server. In the simplest case, turns come in the order of first-come, first-served. (In general, arbitrarily complicated rules can be specified in GPSS to control the sequence in which servers are captured by requestors.)

A RELEASE block is also included in the sequence. A transaction which is in control of a single server gives up control by moving into a RELEASE block. A RELEASE block operand is used to identify the server involved.

GPSS automatically collects (and then, when a simulation stops, prints out) statistical information about single servers modeled with use of SEIZE and RELEASE blocks. (See section A.5 for an example of these statistics.)

An ADVANCE block is used to delay transaction movement along its path for a specified simulated time. In this model, an ADVANCE block can be used to simulate the time required for the machine to drill a hole in a casting ("service time"). The service time random variable in this model is uniformly distributed over the interval 13.5 $\pm$ 3.0 simulated minutes. The values 13.5 and 3.0 are provided in the model as ADVANCE block operands. (Arbitrarily complicated service time distributions can be modeled at ADVANCE blocks, of course. This is done by defining functions which describe the applicable distribution.) By placing an ADVANCE block on the path between SEIZE and RELEASE blocks, simulated time delays between server capture and release can be modeled.

By moving into a QUEUE block, a transaction initiates membership for itself in a *queue*, or waiting line. This membership continues until the transaction brings its queue membership to an end by eventually moving into a DEPART block. An operand is used at the QUEUE and DEPART blocks to indicate the particular queue involved. By placing a SEIZE block between QUEUE and DEPART blocks, transactions will be members of a queue while waiting their turn to capture a server. GPSS automatically collects (and then, when a simulation stops, prints out) statistical information about such queues. (See section A.5 for an example of these statistics.)

Limited space does not permit a more complete explanation here of the GPSS approach to modeling a one-line, one-server system. For a detailed explanation, see chapter 2 in Schriber (1974).

Note that seven types of GPSS blocks have been commented on in this section (GENERATE; TERMINATE; SEIZE; RELEASE; ADVANCE; QUEUE; DEPART). In total, there are more than *fifty* types of blocks in GPSS. By appropriate use of these block types, GPSS models of extremely complex systems can be built with considerable ease.

A.3 The GPSS Block Diagram for the Model

The model described above is shown in the form of a block diagram in Figure A.1 (see the next page). The block diagram consists of a sequence of seven Blocks. (Each block type in Figure A.1 has its own unique, arbitrary geometry.) A simulation performed with the model will start with an empty queue and an idle server, as requested. (See Schriber (1974), chapter 2, for particulars.)

The Figure A.1 block diagram assumes implementation of the model in Wolverine Software's GPSS/H, Release 2 (1988), which uses a floating point simulated clock and therefore permits specification of floating-point interarrival times at GENERATE blocks and holding times at ADVANCE blocks. (In versions of GPSS which use an integer clock, only integer-valued interarrival times and holding times can be realized. In integer-clock versions of this model, units registered by the simulated clock could then have the implicit dimension of *tenths of minutes* (instead of *minutes*), and the GENERATE and ADVANCE block operands could be stated as "150,45" and "130,35," respectively.)

The text appearing adjacent to the blocks in Figure A.1 (e.g., "castings arrive"; "check into the drill queue") is not part of the model, but is simply commentary which has been (optionally) provided as documentation.

A.4 The GPSS Model File

Figure A.1 shows the *block diagram* for a GPSS one-line, one-server model. To perform a simulation with this model, the *statement version* of the Figure A.1 block diagram must be prepared, and then supplemented with additional types of statements used to control compilation and execution of GPSS models. The resulting collection of statements must then be arranged in a model file. The model file is simply a computer file which can be used as the basis for performing one (or more) simulations.

Figure A.2 shows the model file corresponding to the Figure A.1 block diagram. The statements making up the model file are shown against a "background" consisting of column identifiers (e.g., LABEL; OPERATION; and OPERANDS) and horizontal and vertical lines. The background is provided here only as a guide for the eye. The model file statements themselves have such simple forms as "SIMULATE"; "GENERATE 15.0,4.5"; etc.

A column of *statement numbers* ("STMT NO.") has been appended at the far left in the Figure A.2 model file to support discussion here. Statements 7 through 13 correspond to the blocks in Figure A.1. These statements (optionally) include documentation text identical to that appearing in Figure A.1. For example, the text "castings arrive" has been appended to statement 7, but is not an operational part of the statement, and could be deleted.

Statements 1, 19, and 21 in Figure A.2 are examples of statements used to control the compilation and execution of GPSS models. They have been specified in Figure A.2 in such a way that when the model file is submitted for execution, only one simulation will take place. The simulation will stop when the 100th casting has been drilled.

(Limited space doesn't permit detailed discussion of GPSS run-control statements here. In general, however, flexible run control is easily achieved in GPSS. As will be shown in section A.6, for example, only a few changes need be made in the Figure A.2 model file to specify that a series of independent simulations (replications) is to be performed when the file is submitted for execution. Results from these independent simulations can then be statistically analyzed. See Kelton (1986) and Law and Kelton (1983)).

Any model-file statement beginning with an asterisk (*) is a *comments statement*. Comments statements can (optionally) be included in a model file to make it easier (for a person) to read the model file. In Figure A.2, statements 2 through 6, 14 through 18, and 20 are examples of such statements.

A.5 Discussion of Selected Simulation Output.

Selected output automatically produced at the end of the simulation when the Figure A.2 model file was submitted for execution is displayed in Figure A.3 (please turn the page for Figure A.3). The displayed output consists of: (a) clock values; (b) block counts; (c) server statistics; (d) queue statistics; and (e) random number statistics, and will be discussed in that order.

(a) Clock Values

As indicated in Figure A.3(a), GPSS maintains two simulated clocks: a RELATIVE CLOCK; and an ABSOLUTE CLOCK. The ABSOLUTE CLOCK measures the simulated time that has elapsed since the simulation began (that is, since simulated time 0.0). The value of the ABSOLUTE CLOCK at the end of the simulation was 1488.9+. In other words, it took 1488.9+ simulated minutes to drill holes in 100 castings in this replication.

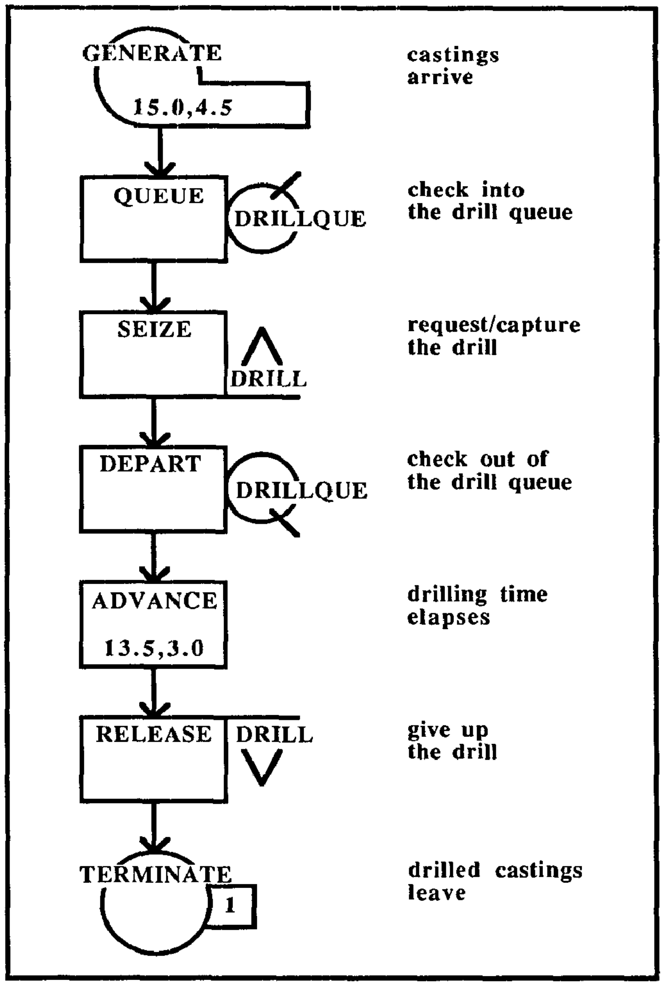

Figure A.1: GPSS Block Diagram for a One-Line, One-Server Queuing System

The RELATIVE CLOCK has no special meaning *unless* one or more RESET run-control statements are used in the model file. RESET statements have not been used here, and so the RELATIVE CLOCK has no special meaning in Figure A.3(a).

(When a RESET statement is included in a model file and is executed, *statistical* aspects of the model are reinitialized, but *units of traffic* (transactions) are left intact wherever they are in the model at the time of RESET statement execution. RESET statements are a useful tool for eliminating biased statistical observations in cases when a simulation proceeds through transient conditions and into a steady state of operation. The RELATIVE CLOCK tells how much simulated time has elapsed since a RESET statement was most recently executed. When there are no RESET statements in a model file, the RELATIVE and ABSOLUTE CLOCKs have identical values, as in Figure A.3(a). For more particulars, see chapter 2 in Schriber (1974)).

STMT NO.	1	LABEL	1 0	OPERATION	2 1	OPERANDS →
1 →				SIMULATE		1S set a 1-CPU-Second time trap
2 →	*					
3 →	*	**				
4 →	*	Model Segment 1 (Movement of Castings Through the System) *				
5 →	*	**				
6 →	*					
7 →				GENERATE		15.0,4.5 castings arrive
8 →				QUEUE		DRILLQUE check into the drill queue
9 →				SEIZE		DRILL request/capture the drill
10 →				DEPART		DRILLQUE check out of the drill queue
11 →				ADVANCE		13.5,3.0 drilling time elapses
12 →				RELEASE		DRILL give up the drill
13 →				TERMINATE		1 drilled castings leave
14 →	*					
15 →	*	**				
16 →	*	Run-Control Statements *				
17 →	*	**				
18 →	*					
19 →				START		100 start the simulation
20 →	*					
21 →				END		end of model-file execution

Figure A.2: A GPSS Model File for the Figure 1 Block Diagram

(b) Block Counts

Blocks in a model are assigned *location numbers* as part of model compilation. These numbers are assigned serially, from 1 forward, in the top-down order in which blocks (block statements) appear in the model file. In Figure A.3(b), the leftmost column (the column labeled BLOCK) contains the numbers 1 through 7, corresponding to the 7 blocks in the Figure A.2 model file. The GENERATE Block is in location 1, the QUEUE block is in location 2, ... , the TERMINATE block is in location 7.

In Figure A.3(b), the second column (the column labeled CURRENT) shows the *counts* of the *number of transactions* currently in the corresponding blocks at the time the output was produced. When the CURRENT count is zero, printing of the zero is suppressed. The only block with a nonzero CURRENT count in Figure A.3(b) is the block in location 2, the QUEUE block. (When the Figure A.3 printout was produced, there was 1 transaction in the QUEUE block, simulating a casting waiting its turn to use the drilling machine.)

The third column in Figure A.3(b) (the column labeled TOTAL) shows the counts of the number of transactions which moved into the corresponding blocks during the simulation. For example, the TOTAL count at the location 1 GENERATE block is 101, indicating that 101 casting-transactions came into the model through that block. The TOTAL count at the location 2 QUEUE block is also 101, indicating that all 101 of these casting-transactions initiated membership for themselves in the queue of castings waiting their turn to use the machine. The TOTAL count at the location 3 SEIZE block is 100, indicating that 100 of these casting-transactions captured the machine during the simulation. (Of the 101 casting-transactions which moved into the location 2 QUEUE block, 100 eventually moved into the location 3 SEIZE block, and one is still in the QUEUE block.)

In general, CURRENT and TOTAL block counts indicate the current state and total extent of traffic movement along the various paths in a model. This information can be of considerable use in analyzing model behavior. Furthermore, CURRENT and TOTAL block counts can be accessed by transactions during the course of a simulation (as values of GPSS *standard numerical attributes*, or reserved words). Such block-count information can be used to support "real time" decision making on the part of transactions as a simulation proceeds, so that transaction movement and path selection can depend on the state of the model at the time the movement and path selection are taking place.

(c) Server Statistics

Figure A.3(c) shows server (drill) statistics accumulated during the simulation. The columns in the figure have been numbered (not by the GPSS software, but after the fact) to make it easy here to refer to the information they contain. The meaning of the information in each of these columns will now be indicated by column number:

(1) The FACILITY column lists the identifier used in the model for the single server (the DRILL, in this case) for which statistics are being reported.

(In GPSS, the *facility* entity used to model single servers. A single server is referred to as a "facility." The postsimulation statistical report contains one row of information for each single server, or facility, contained in a model.)

(2) The --AVG-UTIL-DURING-- TOTAL TIME column shows the *fraction* of *total simulated time* that the server was in a state of capture. In this case, the DRILL was in use 91.7% of the time.

(Note that the *expected value* of the utilization random variable in this model is 0.90. This expected value is computed by dividing the expected service time, 13.5, by the expected interarrival time, 15.0.)

(3) The --AVG-UTIL-DURING-- AVAIL TIME column shows the fraction of *available simulated time* that the server was in a state of capture. A server's "available simulated time" is the amount of simulated time during which the server was "in working order" (or "on duty") during a simulation.

In general, a server in a system is usually not in working order or on duty *all* the time, and cannot be expected to provide service when not in working order or not on duty. For example, if a machine breaks down, it cannot be expected to provide service until after it has been put back into working order. As another example, even though a machine has not broken down, its services might be withdrawn temporarily so that preventive maintenance can be performed on it. While the preventive maintenance is taking place, the machine cannot be expected to provide service. In GPSS, a server which is in working order (on duty) is said to be *available*, and a server not in working order (not on duty) is said to be *unavailable*. (Note that the concept of "available/unavailable" is *not the same* as the concept of "idle/captured.") GPSS provides a rich capability for moving servers back and forth between states of "availability" and "unavailability," in the sense just described, and produces server statistics accordingly. This "available/unavailable" capability has *not been used* in the simple castings-and-machine model here. As a result, the simulated drilling machine was in working order during the entire simulation.

Because *available* simulated time matches *total* simulated time in this model, the entries in columns 2 and 3 in Figure A.3(c) are logically identical. As a result, the --AVG-UTIL-DURING-- AVAIL TIME column has been left blank by the GPSS software.

(4) The --AVG-UTIL-DURING-- UNAVL TIME column shows the fraction of *unavailable simulated time* that the server was in a state of capture. A server's "unavailable simulated time" is the total time the server was "not in working order".

For an *unavailable* (not in working order, or off duty) server to be in a *state of capture* (and therefore doing useful work) seems logically impossible. But there are some systems in which this situation can occur. For example, even though a worker has officially gone off duty because the end of a workshift has come, the worker might continue to work *on his/her own time* for a while to complete an unfinished task. (This might not happen when unions are involved, but it might happen in a worker-owned company.) As another example, the time for scheduled machine maintenance might come (with a period of official machine unavailability starting as a result), but before maintenance actually begins, the machine *might continue to be used* until an ongoing piece of work it is doing has been finished.

Because the drilling machine was never out of working order in this model, its utilization during unavailable time was zero, and so the --AVG-UTIL-DURING-- UNAVL TIME has been left blank in Figure A.3(c) by the GPSS software.

(5) The ENTRIES column indicates the number of times the server was put into a state of capture during the simulation. This statistic is a *capture count*. In Figure A.3(c), the capture count is 100. (After the 100th casting-transaction to take control of the drill gave up control and terminated, the simulation immediately stopped.)

(6) The AVERAGE TIME/XACT column shows the average holding time per capture of the server. (XACT is an abbreviation for transaction.) The AVERAGE TIME/XACT in Figure A.3(c) is 13.6+. (Note that the *expected value* of the holding time random variable is 13.5. This value has been supplied as the first of the ADVANCE block's operands.)

(7) The CURRENT STATUS column indicates the server's "in working order" ("on duty") vs. "not in working order" ("off duty") status at the time the statistical report was produced. AVAIL means "in working order," whereas UNAVAIL means "not in working order." Figure A.3(c) shows that the DRILL was AVAIL at the end of the simulation. (In this model, it was AVAIL during the entire simulation.)

(8) The PERCENT AVAIL column shows the fraction of total simulated time that the server was "in working order" ("on duty"). Figure A.3(c) shows that the DRILL was "available" 100% of the time during the simulation.

(9) The SEIZING XACT column shows the number of the transaction (if any) holding the server in a state of capture when the statistical report was produced. If a server is not captured, the SEIZING XACT column is blank, as in Figure

```
RELATIVE CLOCK:    1488.9629          ABSOLUTE CLOCK:   1488.9629
(a) Clock Values

BLOCK   CURRENT   TOTAL
1                  101
2             1    101
3                  100
4                  100
5                  100
6                  100
7                  100
(b) Block Counts

               (2)    (3)    (4)
  (1)        --AVG-UTIL-DURING--     (5)        (6)          (7)        (8)        (9)          (10)
FACILITY    TOTAL  AVAIL  UNAVL    ENTRIES    AVERAGE      CURRENT    PERCENT    SEIZING    PREEMPTING
             TIME   TIME   TIME              TIME/XACT     STATUS      AVAIL      XACT         XACT

 DRILL      .917                     100       13.655        AVAIL      100.0
(c) Drilling-Machine Statistics

   (1)        (2)          (3)         (4)        (5)        (6)          (7)          (8)          (9)
  QUEUE     MAXIMUM      AVERAGE      TOTAL       ZERO     PERCENT      AVERAGE     $AVERAGE     QTABLE
            CONTENTS     CONTENTS    ENTRIES    ENTRIES     ZEROS      TIME/UNIT    TIME/UNIT    NUMBER

DRILLQUE         2          .215        101         42        41.6         3.172        5.430
(d) Queue Statistics

  (1)          (2)           (3)         (4)         (5)          (6)
RANDOM     ANTITHETIC     INITIAL     CURRENT      SAMPLE     CHI-SQUARE
STREAM      VARIATES      POSITION    POSITION     COUNT      UNIFORMITY

   1           OFF         100000      100202        202          0.70
(e) Random-Number Generator Statistics
```

Figure A.3: Selected Simulation Output

A.3(c). (The simulation stopped immediately when the 100th casting to be drilled *gave up control of the drill* and terminated. This explains why the drill was not in a captured state when the simulation stopped, even though a casting-transaction was waiting at that time to capture the drill. Had the simulation continued at clock time 1488.9+, the waiting casting-transaction would have captured and started to use the drill at that clock time, etc. For particulars on the *chronological order* in which individual steps are carried out when a GPSS model is updated at a given simulated time, see chapter 2 in Schriber (1974).)

As suggested above, transactions have unique *numbers*. They can also have many other individual properties, or attributes, just as the individual units of traffic moving through a real system often have individual properties. In a manufacturing system, for example, units of work-in-process might have properties such as a priority level, a job-type designation, an order number, a customer number, a routing sequence, and a due date. Information of this sort can be attached to transactions in GPSS to support modeling systems in which units of traffic have individual characteristics. For details, see *standard numerical attributes* in general, and *priority levels* and *transaction parameters* in particular, in chapter 4 et seq. in Schriber (1974).

(10) The PREEMPTING XACT column shows the number of the transaction (if any) holding the server in a state of preemption at the time the statistical report was produced. A server is put into a state of preemption if a transaction takes the server away from another transaction. If a server is not in a state of preemption, the PREEMPTING XACT column is blank, as in Figure A.3(c). (The potential for preemptive use of the drill has not been modeled here.)

(In many systems, preemptive use of some servers is permitted. For example, suppose a doctor working in a hospital emergency room is attending a patient who has sprained his ankle. Suddenly another patient is brought in, needing immediate attention as the result of an automobile accident. If the doctor interrupts his or her work on the sprained-ankle patient and begins working immediately on the automobile-accident patient, preemptive use of the server (the doctor in this case) is being made. GPSS provides an extended capability for modeling preemptive use of servers. For particulars, see chapter 7 in Schriber (1974).)

(d) Queue Statistics

Figure A.3(d) shows queue (waiting-line) statistics accumulated during the simulation. The columns in the figure have been numbered (not by the GPSS software, but after the fact) to support discussion. The meaning of the information in each of these columns will now be indicated by column number:

(1) The QUEUE column lists the identifier used in the model for the queue (the DRILLQUE, in this case) for which statistics are being reported.

(In GPSS, the *queue* entity is used to gather waiting-line statistics. The postsimulation statistical report contains one row of information for each queue in a model.)

(2) The MAXIMUM CONTENTS column indicates the maximum length of the waiting line (this statistic has the value 2 in the case of the DRILLQUE).

(3) The AVERAGE CONTENTS column shows the average length of the waiting line (0.215 in the case of the DRILLQUE).

(4) The TOTAL ENTRIES column shows the count of the number of times transactions joined the waiting line (101 in the case of the DRILLQUE).

(5) The ZERO ENTRIES column shows the count of the number of transactions which passed through the waiting line in zero simulated time (42 in the case of the DRILLQUE). (A transaction passes through a waiting line in zero simulated time by initiating and then ending waiting-line membership at one and the same simulated time. In the Figure A.1 model, note that each casting-transaction passes through the waiting line, whether or not it has to wait to take control of the drill.)

(6) The PERCENT ZEROS column shows the percentage of transactions which passed through the waiting line in zero simulated time (41.6 in the case of the DRILLQUE). In other words, PERCENT ZEROS is the percentage of castings which did not have to wait to take control of the drill.

(7) The AVERAGE TIME/UNIT column shows how much time transactions spent resident in the waiting line on average (3.172 in the case of the DRILLQUE). (Here, the term "UNIT" in the AVERAGE TIME/UNIT label means "transaction.")

(8) The $AVERAGE TIME/UNIT column shows how much time transactions spent in the waiting line on average, *excluding* any transactions which passed through the waiting line in zero simulated time. To put this in other words, the $AVERAGE TIME/UNIT (5.43 in the case of the DRILLQUE) is the average time in line for transactions which *did have to wait* to take control of the drill.

(9) If a *qtable* is used in connection with a queue, the QTABLE NUMBER column gives the number (or symbolic name) of the qtable.

(The column (7) AVERAGE TIME/UNIT statistic only provides an estimate of the *expected value* of the "queue residence time" random variable. A qtable is a *tabular histogram* for the "queue residence time" random variable. Information contained in a qtable includes not just the *average* in a sample of queue residence times, but also includes the *sample standard deviation*, and the *relative frequencies* with which the sampled queue residence times fell into various frequency classes prescribed by the model builder. A qtable could have been requested in this model by including one additional statement in the model file. (See chapter 4 in Schriber (1974).)

(10) The CURRENT CONTENTS column shows the number of transactions which were members of the waiting line at the time the Figure A.3(d) report was produced *This column has been cropped from Figure A.3(d) because of space limitations here*. DRILLQUE had a CURRENT CONTENTS of 1 in Figure A.3(d).

(e) Random Number Statistics

Figure A.3(e) shows statistical information for the random numbers used to drive the simulation. The columns in the figure have been numbered (after the fact) to support discussion.

Before discussing Figure A.3(e), some things should be said about the use of random numbers in a simulation. It's often necessary in the Figure A.1 model to sample from the distribution of casting interarrival times, and of drilling times. This sampling involves two steps: (1) A value is drawn from the population uniformly distributed on the 0-1 interval; (2) This value is then converted into a value from the population of interest, e.g., interarrival times uniformly distributed on the 15.0 ± 4.5 interval.

To support step (1) above, GPSS provides built-in 0-1 uniform random number generators. (Many older versions of GPSS have 8 such generators; MINUTEMAN Software's GPSS/PC and Wolverine Software's GPSS/H have a virtually unlimited number of such generators. The motivation for having more than one such generator is explained and demonstrated in chapter 3 of Schriber (1974).) These generators are numbered 1, 2, 3, 4, etc. In the Figure A.1 model, uniform random number generator 1 is used to sample from both the interarrival-time and the service-time distributions. (For particulars, see chapter 3 in Schriber (1974).)

The 0-1 uniform random number generators often built into simulation software typically use a deterministic algorithm to produce what are called *pseudo*-random numbers. These numbers aren't *truly* random, because they are computed by a reproducible deterministic procedure. This raises a question about how "good" a sample of such pseudo-random numbers is, statistically speaking. This matter of statistical goodness can be tested in one or more ways by simulation software for the pseudo-random numbers actually used during the course of a simulation. Results of such a test (or tests) can then be reported as part of the simulation results.

Suppose the pseudo-random numbers contained in the samples coming from the 0-1 generators used in a simulation (replication) don't test well for 0-1 uniformity. The modeler might then decide to throw away the results from that particular simulation (replication).

The "suspect" (or low probability) results can be replaced by using *other* pseudo-random numbers to carry out another replication. ("Other" pseudo-random numbers would be obtained by specifying a different *starting point* for the random number generator or generators. For particulars, see chapter 3 in Schriber (1974).)

As part of its postsimulation output, GPSS/H (Release 2, but not Release 1) includes a statistical report on the 0-1 uniform random numbers used in the simulation just completed. Referring to Figure A.3(e) by column number, the following information is contained in this report:

(1) The RANDOM STREAM column gives the number of the 0-1 generator to which the report applies (generator 1 in Figure A.3(e)).

(2) The ANTITHETIC VARIATES column indicates whether the random numbers themselves *or their antithetic equivalents* coming from the indicated generator were used. Column entries of OFF and ON are used to indicate whether the random numbers themselves (OFF) or their antithetic equivalents (ON) were used. For the case at hand, the random numbers themselves were used.

(For a 0-1 uniform distribution, an "antithetic random number" is 1.0 minus the random number. For example, 0.15 is antithetic to 0.85. In some circumstances, antithetic random numbers can be used to reduce the variance of one-population estimators. For particulars, see Law and Kelton (1983).)

(3) INITIAL POSITION indicates the ordinal position (in the time series of random numbers produced by the generator) from which the *first* random number came (position 100,000 for the case at hand).

(4) CURRENT POSITION indicates the ordinal position from which the *next* random number will come for the generator (position 100202 for the case at hand) if the simulation is resumed.

(5) SAMPLE COUNT indicates how many random numbers were sampled from the generator (202 in this case).

(6) CHI-SQUARE UNIFORMITY gives the *achieved significance level* of a Chi-square goodness-of-fit test for the uniformity of the random numbers sampled from the generator. An achieved significance level is a probability. The purpose of the chi-square goodness-of-fit test is to examine the hypothesis that the sampled random numbers come from a source of random numbers uniformly distributed on the 0-1 interval. If the "Chi-square uniformity" number is 0.05 or less, then the probability is 0.05 or less of drawing a sample with *this* sample's Chi-square statistic from a source of *true* 0-1 uniform random numbers. In such a low probability case, the modeler *might* want to throw away the results of the simulation, substituting the results of another replication instead. (The CHI-SQUARE UNIFORMITY statistic in the Figure A.3(e) replication is 0.7.)

A.6 Replications in GPSS

This section briefly reviews the concepts of point and interval estimates of the expected value of a random variable (or, more generally, of an unknown population parameter), provides numeric examples for these concepts in the setting of the section A.1 one-line, one-server model, and introduces the use of the GPSS CLEAR statement as a means for carrying out a series of independent simulations (replications) whose results can be used to form interval estimates.

Figure A.3 provides *point estimates* of the expected values of such dependent random variables as the time required to drill holes in 100 castings (ABSOLUTE CLOCK); the average length of the line of castings waiting for the drill (AVERAGE CONTENTS); and the average time castings spend waiting for the drill (AVERAGE TIME/UNIT). Recall (from a first course in statistics) that a point estimate is a *single number* used as an estimate of the value of an unknown population parameter (e.g., an expected value). The point estimates in Figure A.3 result from *one* simulation, or replication. In general, if a *series* of independent simulations is performed, the value of a point estimate will vary from simulation to simulation. *One* point estimate provides no information about the *variability* of the point estimator, and so it can be misleading to use a *single* point estimate to estimate the expected value of a random variable.

By way of example, consider Table A.1, which shows the values of eight point estimates of: (a) the time required to drill holes in 100 castings; (b) the average length of the line of castings waiting for the drill; and (c) the average time castings spend waiting for the drill. The Table A.1 values result from a series of eight independent replications performed by using a slightly modified version of the Figure A.2 model file. (The modifications made in the Figure A.2 model file to produce the Table A.1 results are discussed below.) The variability in the point estimates from replication to replication is evident in Table A.1. For example, the time to drill 100 castings ranges from about 1488 minutes (replication 3) to about 1561 minutes (replication 7), with a sample mean (for the sample of eight replications) of 1518.1 minutes and a sample standard deviation of 25.97 minutes. Here, the sample standard deviation is relatively small (just under 2% of the sample mean).

Similarly, for the eight replications in Table A.1, the average number of castings waiting for the drill ranges from 0.049 (replication 7) to 0.215 (replication 1), with a sample mean of 0.139 and a sample standard deviation of 0.062. Here, the sample standard deviation is quite large (just greater than *40%* of the sample mean).

Table A.1: Summary Statistics for Eight Independent Replications with the One-Line, One-Server Model

Replication Number	Time to Drill 100 Castings	Castings Waiting for the Drill	
		Average No. Waiting	Average Time Spent Waiting
1	1489.0	0.215	3.172
2	1544.7	0.130	1.995
3	1488.6	0.193	2.851
4	1522.8	0.064	0.970
5	1507.8	0.199	3.001
6	1527.5	0.110	1.667
7	1561.2	0.049	0.770
8	1503.0	0.152	2.259
Mean:	1518.1	0.139	2.09
Standard Deviation:	25.97	0.062	0.910

Finally, the average time castings spent waiting for the drill ranges from 0.770 minutes (replication 7) to 3.172 minutes (replication 1) in Table A.1, with a sample mean of 2.09 and a sample standard deviation of 0.910. Here, the sample standard deviation is again quite large (about *45%* of the sample mean).

The variability evident in the Table A.1 point estimates can be taken into account quantitatively by using the *sample standard deviations* to form *interval estimates* for the expected values of the corresponding random variables. Recall (from a first course in statistics) that an interval estimate of a population parameter is a *pair of numbers* determining an interval within which the value of the parameter *may* lie. The interval which the pair of numbers determines is called a *confidence interval*. A confidence coefficient, such as 90% or 95%, is attached to this interval to indicate the *confidence level*, or degree of confidence we have that the population parameter *does* lie within the confidence interval.

Table A.2 shows the 90% confidence intervals computed from the set of eight replications given in Table A.1. For example, the Table A.2 90% confidence interval for the time required to drill 100 castings is [1499.5, 1536.7]. In other words, we are 90% confident that the expected value of the "time to drill 100 castings" random variable falls somewhere in the interval between 1499.5 and 1536.7.

Table A.2: 90% Confidence Intervals Resulting from the Table A.1 Replications

	Castings Waiting for the Drill	
Time to Drill 100 Castings	**Average No. Waiting**	**Average Time Spent Waiting**
[1499.5, 1536.7]	[0.094, 0.184]	[1.434, 2.737]

(Recall that a given confidence interval either *does* or *does not* contain the expected value of the population parameter being estimated. Each number in the pair determining a confidence interval is a *random variable*. This means that if we produced another 8 independent replications, then computed the resulting 90% confidence intervals as in Table A.2, they would, in general, differ from the Table A.2 confidence intervals. Suppose we *repeatedly* formed 90% confidence intervals for the problem at hand, each based on *another* set of eight replications. Then, *among all such* confidence intervals, *90%* of them *will contain* the value being estimated. This is what it means to say we are "90% confident" that any one such confidence interval does contain the value being estimated.)

(The steps followed in computing confidence intervals can be found in any introductory textbook on statistics, and in any general simulation textbook. It is recommended that interested persons consult a general *simulation* textbook, which will not only summarize how to compute confidence intervals but, perhaps more importantly, will also discuss the issues involved in producing statistically valid results via simulation. A simulation text will also discuss the distinction between "terminating" and "steady state" simulations, will explain various alternative statistical methodologies for analysis of simulation output (e.g., the method of replications; the method of batch means; time series methods), and so on.)

Now consider the operational aspects of producing replications in GPSS simulations. The replications whose results are summarized in Table A.1 were produced with use of a GPSS CLEAR statement. The CLEAR statement is a run-control statement. When a GPSS model is "cleared" (that is, when a CLEAR statement is executed), the following two actions occur:

1. All transactions in the model (if any) are destroyed.

2. Statistical aspects of the model are reinitialized (e.g., facility capture counts are set back to zero; facility total time captured is set back to zero; queue entry counts are set back to zero; total queue residence time is set back to zero; the RELATIVE and ABSOLUTE CLOCKs are set back to zero; etc.).

CLEARing a model has the effect, then, of returning the model to its original starting point, *with one important exception*. Executing a CLEAR statement does *not* cause the setting of the random number generator(s) being used in a model to be reinitialized. Instead, the random number generators are left "as is." The result of CLEARing a model, then, is to set the stage for carrying out another simulation with the model, a simulation which will be independent of the one or more immediately preceding simulations because the 0-1 uniform random numbers used to drive the simulation will (or should) be independent of those used to drive the preceding simulation(s).

For example, suppose the START statement (STMT NO. 19) in Figure A.2 were followed by a CLEAR statement, and then by another START statement, and then by an END statement. This sequence of four statements would take the form shown in Figure A.4. If the Figure A.2 model file were modified accordingly, and were submitted for execution, *two* replications would be performed with the one-line, one-server model. By including another six "CLEAR/START" statement pairs in the model file, then submitting the resulting model file for execution, the eight replications whose results are summarized in Table A.1 would be carried out.

```
        START     100     start the 1st replication
*
        CLEAR             clear the model
*
        START     100     start the 2nd replication
*
        END               end of model-file execution
```

Figure A.4: A Modified START/.../END Sequence for the Figure A.2 Model File

If the technique described in the preceding paragraph is followed, then there will be seven consecutive "CLEAR/START" pairs preceding the END statement at the bottom of the model file. In Wolverine Software's GPSS/H, a single "CLEAR/START" statement pair can be made the subject of a DO/ENDDO loop, with control then traversing the loop any number of times (such as 7 times) specified by the modeler. This is but one example in a large set of capabilities included in GPSS/H to facilitate the execution of run-control statements in GPSS modeling.

Another way to produce confidence intervals in GPSS modeling is to design the model file in such a way that confidence intervals themselves (rather than just the individual results from a series of replications) are reported out at the end of a simulation. This can be done in GPSS/H, for example, with the use of LET statements (which are computational statements), PUTPIC statements (which are general purpose output statements), and external ampervariables (which make it possible to invoke an external routine to obtain dynamically the t statistic(s) needed to compute confidence intervals). This can also be done in MINUTEMAN Software's GPSS/PC, for example, by combined use of the RESULT command (to put simulation results into a specified file) and of the ANOVA command (for postsimulation analysis of variance).

REFERENCES

Abed, S. Y., Barta, T. A. and McRoberts, K. L. (1985a). A qualitative comparison of three simulation languages: GPSS/H, SLAM, SIMSCRIPT. *Computers & Industrial Engineering* **9**, 35-43.

Abed, S. Y., Barta, T. A. and McRoberts, K. L. (1985b). A quantitative comparison of three simulation languages: GPSS/H, SLAM, SIMSCRIPT. *Computers & Industrial Engineering* **9**, 45-66.

AutoSimulations, Inc. (1986). *AUTOGRAM User's Manual.* AutoSimulations, Inc., Bountiful, Utah.

AutoSimulations, Inc. (1986). *AUTOMOD User's Manual.* AutoSimulations, Inc., Bountiful, Utah.

Banks, J. and Carson, J. S. (1984). *Discrete-Event System Simulation.* Prentice-Hall, Englewood Cliffs, New Jersey.

Banks, J. and Carson, J. S. (1985). Process-interaction simulation languages. *Simulation* **44**.

Barnette, D. T., and Sommerfeld, J. T. (1987). Discrete-event simulation of a sequence of multicomponent batch distillation columns. *Computers and Chemical Engineering* **11**.

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

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

Bratley, P., Fox, B. L. and Schrage, L.E. (1987). *A Guide to Simulation*, 2nd ed. Springer Verlag, Berlin and New York.

Christy, D. P. and Watson, H. J. (1983). The application of simulation: a survey of industry practice. *Interfaces* **13**, 47-52.

Cox, S. (1986). *GPSS/PC User's Manual*, 2nd ed. MINUTEMAN Software, Stow, Massachusetts.

Cox, S. (1987). Interactive graphics in GPSS/PC. *Simulation* **48**, 9

Crain, R. C., Brunner, D. T. and Henriksen, J. O. Advanced use of GPSS/H. In: *Proceedings of the 1987 Winter Simulation Conference* (A. Thesen, H. Grant, and W. D. Kelton, eds.). The Society for Computer Simulation, San Diego, California.

Cummings, G. F. *GPSS/PC Simulation Tutorials.* MINUTEMAN Software, Stow, Massachusetts.

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

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

Gordon, G. (1975). *The Application of GPSS V to Discrete Systems Simulation.* Prentice-Hall, Englewood Cliffs, New Jersey.

Gordon, G. (1978). The development of the General Purpose Simulation System (GPSS). In: *SIGPLAN Notices (History of Programming Languages Conference)* **13**, No. 8. Association for Computing Machinery, New York, 183-198.

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

Haider, S. W. and Banks, J. (1986). Simulation software products for analyzing manufacturing systems. *Industrial Engineering*, July, 98-103.

(Several figures in this article are typeset incorrectly. Corrected versions of these figures are given in *Industrial Engineering*, September 1986, pp. 86-87. However, the corrected figures *do not show* that animation and a special-purpose database are available in MINUTEMAN's GPSS/PC, Version 2; and that animation and a database are available to Wolverine's GPSS/H through TESS.)

Henriksen, J. O. (1981). GPSS - finding the appropriate world view. In: *Proceedings of the 1981 Winter Simulation Conference* (T. I. Oren, C. M. Delfosse, and C. M. Shub, eds.). The Society for Computer Simulation, San Diego, California, 505-516.

Henriksen, J. O. (1983). State-of-the-art GPSS. In: *Proceedings of the 1983 Summer Computer Simulation Conference.* The Society for Computer Simulation, San Diego, California, 918-933.

Henriksen, J. O. (1986). You can't beat the clock: studies in problem solving. In: *Proceedings of the 1986 Winter Simulation Conference* (J. R. Wilson, J. O. Henriksen, and S. D. Roberts, eds.). The Society for Computer Simulation, San Diego, California, 713-726.

Henriksen, J. O. et al. (1988). *GPSS/H User's Manual*, 3rd ed. Wolverine Software Corporation, Annandale, Virginia.

Henriksen, J. O. and Schriber, T. J. (1986). Simplified approaches to modeling accumulating and nonaccumulating conveyor systems. In: *Proceedings of the 1986 Winter Simulation Conference* (J. R. Wilson, J. O. Henriksen, and S. D. Roberts, eds.). The Society for Computer Simulation, San Diego, California, 575-593.

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

International Business Machines (1977). *APL GPSS* (G320-5745). IBM Inc., Armonk, New York.

International Business Machines (1981). *PL/1 GPSS* (G320-6398). IBM Inc., Armonk, New York.

Kenvin, J. C., and Sommerfeld, J. T. (1987). Discrete-event simulation of large-scale poliomyelitis vaccine production. *Process Biochemistry* **22**, No. 3, 74-77.

Kelton, W. D. (1986). Statistical analysis methods enhance usefulness, reliability of simulation models. *Industrial Engineering*, September, 74-84.

Law, A. M., and Kelton, W. D. (1982) *Simulation Modeling and Analysis*. McGraw-Hill, New York.

Martin, D. (1981). *GPSS/VX User's Manual*. Simulation Software Ltd., London, Ontario, Canada.

Murphy, W. H. and Schriber, T. J. et al. (1982). Analysis of the immune response to malignant cells using computer simulation. In: *Proceedings of the 6th Annual Symposium of Computer Applications in Medical Care*. IEEE Press, New York.

Passariello, I. and Sommerfeld, J. T. (1987). GPSS simulation of chocolate manufacturing. In: *Tools for the Simulation Profession*. The Society for Computer Simulation, San Diego, California, 1-6.

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

Pritsker, A. A. B. (1986). *Introduction to Simulation and SLAM II*, 3rd ed. Pritsker and Associates, West Lafayette, Indiana.

Richards, C. B. (1983). *GPSSR/PC User's Manual*. Simulation Software Ltd., London, Ontario, Canada.

Richards, C. B. (1984). *GPSS/C User's Manual*. Simulation Software Ltd., London, Ontario, Canada.

Richards, C. B. and Mullin, J. K. (1981). GPSSR: General Purpose Simulation System for mini/micro machines. In: *Modelling and Simulation on Microcomputers*. The Society for Computer Simulation, San Diego, California.

Roberts, S. D. and England, W. L. (1981). *Survey of the Application of Simulation to Health Care*. SCS Simulation Series **10**, No. 1. The Society for Computer Simulation, San Diego, California.

Rubin, J. (1981). Embedding GPSS in a general purpose language. In: *Proceedings of the 1981 Winter Simulation Conference*. (T. I. Oren, C. M. Delfosse, and C. M. Shub, eds.). The Society for Computer Simulation, San Diego, California, 33-40.

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

Schmidt, B. (1987). *Model Construction with GPSS-FORTRAN (Version 3)*. Springer Verlag, Berlin and New York.

Schriber, T. J. (1974). *Simulation Using GPSS*. John Wiley & Sons, New York. (Russian edition, 1980, Mashinostroyenie Press, Moscow.)

Schriber, T. J. (1985). A GPSS/H model for a hypothetical flexible manufacturing system. *Annals of operations research* **3**, 171-188.

Schriber, T. J. (1987). The nature and role of simulation in the design of manufacturing systems. In: *Simulation in Computer Integrated Manufacturing* (K. E. Wichmann, ed.). The Society for Computer Simulation, San Diego, California, 5-18.

Schultheisz, D. J. and Sommerfeld, J. T. (1988). Discrete-event simulation in chemical engineering. *Chemical Engineering Education* Spring 1988.

Solomon, S. (1983). *Simulation of Waiting Lines*. Prentice-Hall, Englewood Cliffs, New Jersey.

Standridge, C. (1985). A tutorial on TESS: The Extended Simulation System. In: *Proceedings of the 1985 Winter Simulation Conference* (D. Gantz, G. Blais, S. Solomon, eds.). The Society for Computer Simulation, San Diego, California, 73-79.

Sturgul, J. R. (1987). Building simulation models of surface coal mines using the GPSS computer language. *The Coal Journal* **15**, 11-17.

Sturgul, J. R. (1987). Simulating mining engineering problems using the GPSS computer language. *Bulletin of the Proceedings of the Australian Institute of Mining and Metallurgy* **292** (4), 75-78.

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

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

ACKNOWLEDGMENTS

This article has benefitted from thoughtful comments generously provided by MINUTEMAN Software's Alice Cox and Springer Cox, Simulation Software Ltd.'s David M. Martin, and Wolverine Software's Daniel T. Brunner and Robert C. Crain.

AUTHOR'S BIOGRAPHY

THOMAS J. SCHRIBER is a Professor of Computer and Information Systems in the Graduate School of Business at The University of Michigan. He holds a B.S. (University of Notre Dame) and M.S.E., M.A., and Ph.D. (University of Michigan). Prof. Schriber teaches, does research, and consults in the area of discrete-event simulation. He has authored or co-authored several dozen articles, and has authored or edited nine books. He regularly teaches intensive courses on GPSS-based simulation, and has consulted with such organizations as Monsanto Chemical, CPC International, ITT, Ford Motor Company, Exxon, General Motors, and Occidental Petroleum, both in a problem-solving and a teaching capacity. 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. From 1977-86 he was the ACM member of the Board of Directors of the Winter Simulation Conferences, serving as Board Chairman two years. He was Program Chairman and Proceedings Co-Editor of the 1976 Bicentennial Winter Simulation Conference. His professional affiliations include ACM, DSI, IIE, ORSA, SCS, SME/CASA, and TIMS.

Thomas J. Schriber
Graduate School of Business
The University of Michigan
Ann Arbor MI 48109-1234 USA
(313) 764-1398