

GPSS/VI™

Duane Ball

Advanced System Technologies, Inc.
Suite 514, 5113 Leesburg Pike
Falls Church, VA. 22041, U.S.A.

ABSTRACT

This paper describes an implementation of the General Purpose Simulation System (GPSS) simulation programming language, called *GPSS/VI™*, with a Visual model development and an Interactive simulation environment. The objectives of *GPSS/VI™* were to: (1) be easy to use; (2) eliminate most model specification errors; (3) greatly reduce model development time; (4) enhance model verification; and (5) permit ad-hoc model experimentation. *GPSS/VI™* met these objectives and has been used by AST for two years.

1. INTRODUCTION

Our company applies high fidelity discrete-event simulation to real-time information systems design problems. In a transition from mainframe to desktop computers, we found ourselves without a simulation package. After reviewing available modeling software, we determined that a language "like" GPSS would best satisfy our needs. However, we found the model development environments for existing implementations of GPSS primitive. Our experience has been that cost effective simulation requires tools that permit a high degree of user interaction and simplify the task of visualizing the model. As noted by Kiviat (1991) effective simulation practitioners must be "process explorers" rather than parameter "estimators":

We should be seeking qualitative understanding through quantitative calculation, and constantly asking the question, "Why is this happening?" in addition to the classic, "What happened?"

As part of a NASA-sponsored research project we developed a visual programming environment using context-sensitive hierarchical two-dimensional drawings to represent programs. A review of this programming environment convinced us that it was suitable for entering and editing GPSS block diagrams. We decided to perform a pilot development effort to determine the feasibility of implementing a GPSS that would tell us both what happened and why.

At the time of the pilot study, we were starting a

detailed communications protocol analysis. Our plan was to implement "just enough of GPSS" to perform the analysis. We began by writing a model of the protocol in GPSS using the minimum number of blocks required for fidelity. After approximately one man-month of effort, an initial prototype version of *GPSS/VI™* was operational. On the basis of this early success, we decided to develop a complete simulation package.

2. REQUIREMENTS FOR GPSS/VI

Michael Rooks (1991) has developed four requirements for a complete visual interactive (VI) simulation system:

Specification The analyst must be capable of specifying model parameters, in accordance with the objectives for analysis of the model.

Intervention The analyst must be provided with an effective means of initiating interaction with the model. Modes of interaction include inspection, specification, and visualization.

Inspection The analyst must have access to all model data relevant to the experiments to be performed on the model...

Visualization The analyst must be capable of viewing model data in ways that illustrate the model dynamics and relationships of interest...

We did not have Rooks' article during the *GPSS/VI™* design phase; however, his principles precisely summarized our goals. This article shows how *GPSS/VI™* attempts to satisfy these four requirements.

3. MODEL SPECIFICATION IN *GPSS/VI™*


While there have been many versions and dialects of GPSS, we took the description of the language given in *General Purpose Simulation System V User's Manual* by IBM Corporation (IBM 1971) to be a definitive language

specification. In the remainder of this article, when we reference GPSS we mean IBM's GPSS/V

We classified elements of GPSS into: (1) essential features of the language; (2) artifacts that reflected obsolete programming and execution environments; (3) language elements which, while not inherently obsolete, would become obsolete in a VI environment; and (4) features that require a new form for VI simulation. An example illustrating each type of language element follows.

3.1 Essential: The Block Diagram

The block diagram is the best example of an essential GPSS language feature. GPSS was probably the first visually defined programming language. Many analysts first represent GPSS models as two-dimensional block diagrams and then manually translate the block diagrams into textual form. Each block in the diagram corresponds to a model statement and arrows between blocks denote the flow of transactions. As illustrated in Figure 1 (adapted from Schriber 1974), *GPSS/VI™* permits direct entry and revision of block diagrams. The two columns of icons appearing at the left of the figure are a palette of block types. The user adds a block to the model by using a mouse to select the block's icon from the palette and then placing the icon in the block diagram to the right.

To manage large models, *GPSS/VI™* allows multi-page model specification (letter or legal size with either landscape or portrait orientation). A connection between two pages is made using PageTransfer blocks (denoted by the icon ).

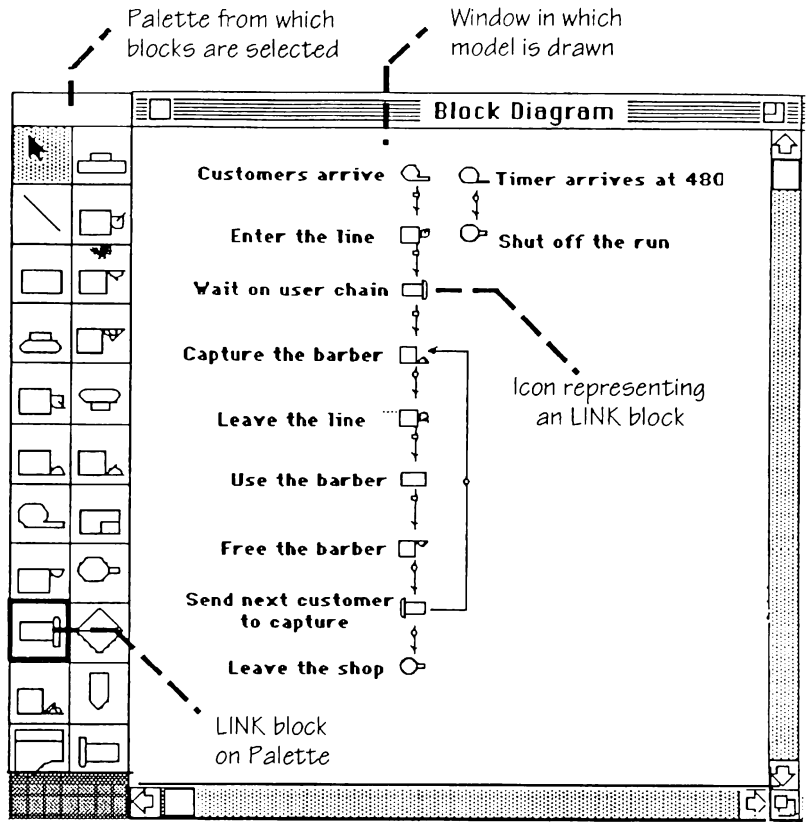


Figure 1: Joe the Barber Revisited

3.2 Artifacts: Operands and Data Types

In contrast to the block diagram, the use of positional operands in GPSS blocks is an artifact of obsolete computing technology. Misplaced commas separating positional operands have caused countless GPSS model specification errors. The differences between "GENERATE ,,1" and "GENERATE ,,,1" illustrate this point.

As shown in Figure 2, there is a dialog box for each block in the *GPSS/VI™* block diagram. Operands are entered on a keyword, rather than positional, basis. In addition, each dialog box acts as a context sensitive editor. Consider the PREEMPT block's operand context restrictions (Schriber 1974):

Figure 2: No more ".,,"

Restriction 1. If a default is taken on the PREEMPT block's B Operand then the Processor ignores the C,D, and E Operands.

Restriction 2. If a default is taken on the PREEMPT Block's C Operand then the Processor ignores the D and E Operands.

GPSS/VI™ reflects these context restrictions by hiding the C, D, and E operands when they are inapplicable.

The data type micro-management (e.g., use of the halfword), integer clock, and lack of useful mathematical expressions in GPSS are other relics of the computing hardware and compiler technology that existed when the language was designed. GPSS/VI™ has: floating point, Boolean, and character data types; a floating point clock; and expressions which are dynamically evaluated during the simulation.

3.3 Obsolete in VI : The GATE Block

Some GPSS language elements, while not inherently antiquated, become so in the GPSS/VI™ environment. Consider the GATE block which is used as an alternative to the TEST block in situations: (1) where Boolean variables are used; and (2) the analyst knows that until the first transaction waiting to enter the block is admitted, attempts to advance subsequent transactions queued for the block will be futile. In GPSS/VI™, TEST block operands can be logical expressions. Thus, the only use for the GATE block becomes simulator efficiency.

GPSS/VI™ identifies expressions that will not change during a scan of the current events chain. Such expressions are called *scan-invariant expressions*.¹ The

GPSS/VI™ implementation of the TEST block uses logical expressions and the notion of scan-invariance to provide all the functionality of the GATE block while automatically recognizing opportunities for increased execution efficiency².

3.4: New Form: The MATRIX

Finally, there are essential elements of GPSS that require a new form for VI simulation. A good example of this type of language element is the MATRIX whose natural VI implementation is an EXCEL-like (EXCEL is a registered trademark of the Microsoft Corporation) spreadsheet. As illustrated in Figure 3, in GPSS/VI™ each MATRIX is represented by a two-dimensional

1 For example, a function of the status of facilities will not change during a scan of the current events chain (a change in status resets the scan) and is scan-invariant. A function in which any term is a random variate is an example of a non-scan-invariant function.

2 The 2-nd through n-th transactions waiting for a scan-invariant TEST block are made scan inactive and remain so until their immediate predecessors enter the block.

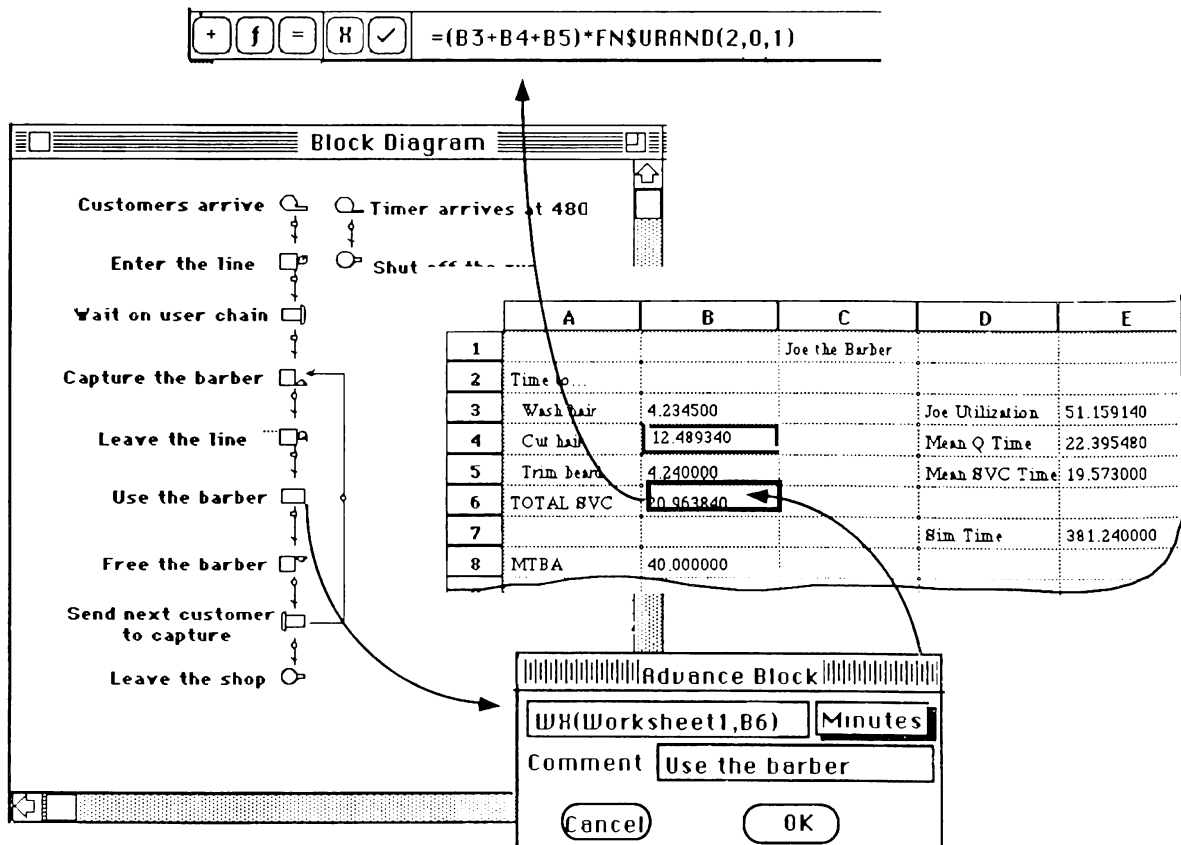


FIGURE 3: The Ubiquitous Worksheet

spreadsheet. Spreadsheet entries can contain numeric constants, strings constants, string expressions, and mathematical expressions. Standard numerical attributes (SNA) can appear as terms in the mathematical expressions along with a complement of mathematical functions. As can be seen in the Formula Bar in Figure 3, spreadsheet entries can reference one another. Standard spreadsheet editing commands such as Copy, Paste, and Fill are supported. Any numeric or logical block operand can reference the spreadsheet.

Two spreadsheet features that received particular attention in GPSS/VI™ are *dynamic addressing* and *incremental updating*. Dynamic addressing allows a block operand to reference a spreadsheet entry whose address is determined at the time of reference. Successive executions of the block may reference different entries. For example, the spreadsheet entry used in a block operand may vary depending on a property of the transaction being processed such as priority. Another example of dynamic addressing is the spreadsheet MATCH function which is a powerful generalization of the GPSS FUNCTION.

Incremental updating is an extension to standard spreadsheet technology needed for simulation. Normally when an entry in a spreadsheet is changed either: (1) all the entries in the spreadsheet are updated, or (2) only the changed entry is updated (the user must explicitly restore spreadsheet consistency). In a typical simulation, hundreds of thousands of spreadsheet updates may occur. An entry must be consistent whenever it is used (i.e., is evaluated as a block operand). The CPU cost of restoring the consistency to the entire spreadsheet each time an entry is used would be prohibitive. During the simulation, GPSS/VI™ uses incremental updating, a technique that recursively updates only those spreadsheet entries required to restore the consistency of the value currently being accessed.

4. USER INTERVENTION IN GPSS/VI™

User intervention allows the analyst to interact with the model as it executes. Rooks' criteria for true visual interactive simulation requires "capabilities for accessing model-specific parameters, and for the execution of model-specific functions that are programmed by the modeler." GPSS/VI™ supports both direct and conditional user intervention. Direct intervention allows the user to stop simulation execution immediately. Any model parameter can be updated at this time. In addition, the user can edit the Current Transaction's priority and parameters, or the Current Transaction can be moved from the current to the future events chain (i.e., a

dynamic ADVANCE) or terminated. The user may then resume the simulation.

In addition to direct user-intervention, GPSS/VI™ allows the user to define a set of conditions that, when satisfied, cause the simulation to stop. These conditions, called *conditional breakpoints*, can be attached to any connector between blocks in the model. When a breakpoint condition is satisfied, the block that contains the Current Transaction is highlighted and the window containing it is brought to the front of the screen. As with direct intervention, the Current Transaction and any model parameter may be inspected and changed before resuming simulation.

A second type of conditional user intervention supported by GPSS/VI™ causes the simulation to stop whenever a transaction is accepted by a new block. This kind of intervention is called *single-stepping*.

5. DATA INSPECTION IN GPSS/VI™

A data inspection capability for VI simulation must allow the user to understand what is happening in the simulation and why it is happening. GPSS/VI™ provides several complementary views of the simulation state data including: spreadsheets; standard GPSS reports; direct visual feedback to the block diagrams; and an annotated trace. This section describes GPSS/VI™'s annotated trace.

The annotated trace, as illustrated in Figure 4, gives

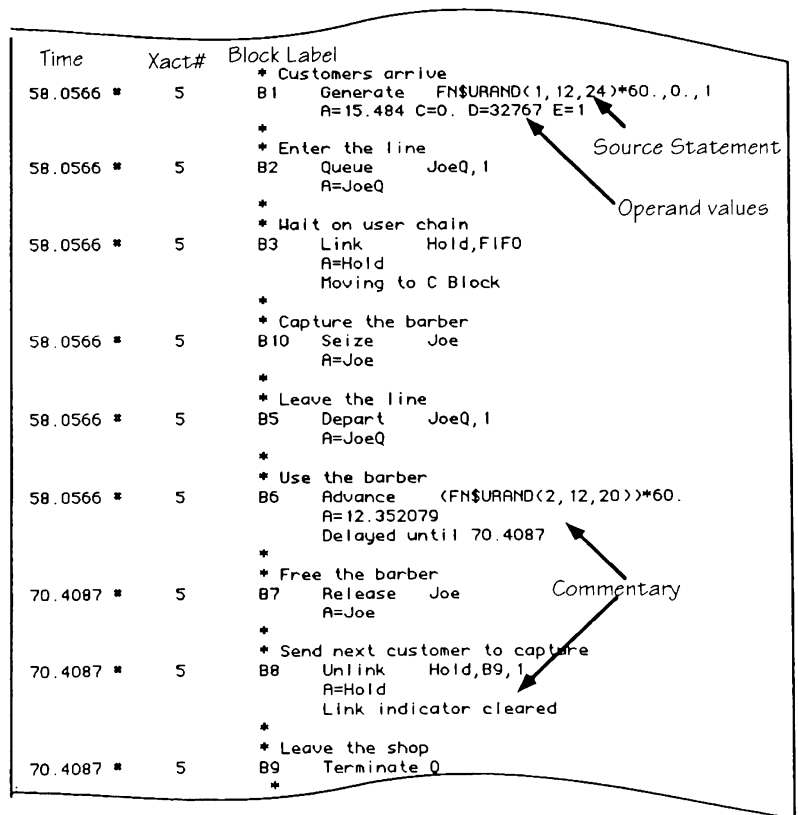


Figure 4: Transaction 5's View of "Joe the Barber"

the user a "transaction's eye view" of the simulation. Each entry in the trace includes the: (1) simulation time; (2) transaction's number; (3) GPSS source block being executed; and (4) value of the operands passed to the block. In addition, when the effect of executing the block depends on system or transaction state information not derived from its operands, *commentary lines* are added to complete the description. A simple example of a commentary line is the one appended to each trace entry for ADVANCE blocks. The effect of an ADVANCE block is to move a transaction onto the future events chain until a future simulation time. It can be useful to follow the progress of the transaction after the expiration of the advance time. To simplify this process, the commentary line appended to the trace entry includes the time when the transaction should reenter the current events chain. In more complicated cases, such as processing pending Preempts, several lines of commentary are appended to the trace entry.

The annotated trace is updated whenever a transaction whose TI flag is set enters a block. Just as with conditional breakpoints, the trace flags can be set or cleared whenever a transaction traverses an interblock connector. Interblock connectors set trace flags in two modes: unconditional and conditional. In the unconditional mode, any transaction that traverses the connector has its trace flag set. In the conditional mode, a logical expression attached to the connector is evaluated. If the expression, which can be composed of predicates dependent on both general system state and the particular transaction being evaluated, is true then the TI flag is set. Similarly, both conditional and unconditional clearing of TI flags is possible.

One of the most powerful ways to perform process exploration with *GPSS/VITM* is to combine transaction tracing with single stepping. Both the block diagram and the trace can be viewed simultaneously. When the analyst moves the current transaction one block forward, the block is highlighted on the block diagram and all the information being used to process the block appears in the trace window.

6. VISUALIZATION IN *GPSS/VITM*

Rooks groups visual model display into two categories: representational and abstract. The primary representational display in *GPSS/VITM* is dynamic feedback to the block diagram. Whenever a transaction is admitted to a block, the icon representing the block is updated with block counts and summary statistics describing the average transaction holding time for the block. In addition, in single-step mode, the icon representing the block being entered is highlighted and the operands being passed to the block are displayed in a special transcript window.

GPSS/VITM interfaces with a wide variety of abstract visualization packages (e.g., statistical analysis systems) and animation systems. The spreadsheet and PRINT

block are the primary mechanisms for transferring data to visualization packages for post-processing. Early in the design of *GPSS/VITM*, we decided that data analysis and animation would be done using packages especially designed for those purposes. Unlike the block diagram and spreadsheet which are tightly coupled to the model building and verification process, simulation and analysis and display of the results are loosely coupled. Our goal has been to develop flexible interfaces to statistical analysis and animation packages we can obtain "off-the-shelf." To date, several statistical analysis and graphing packages have been used with *GPSS/VITM*. In addition, *GPSS/VITM* can exchange data directly with other spreadsheet systems such as EXCEL.

7. FUTURE DIRECTIONS

Our immediate plans are: (1) to find an opportunity to use an animation package in with *GPSS/VITM*; (2) to develop the a "graphics macro" that will allow users to define new blocks with existing blocks; and (3) porting *GPSS/VITM* to additional desktop platforms. We intend to make a low-cost student version available for graduate-level simulation courses.

REFERENCES

- IBM, 1971. *General Purpose Simulation System V User's Manual*. International Business Machines Corporation.
- Kiviat, P. 1991. Simulation, Technology, and the Decision Process. In *ACM Transactions on Modeling and Computer Simulation*, Vol. 1 No 2:89-98.
- Rooks, M. 1991. A United Framework for Visual Interactive Simulation. In *1991 Winter Simulation Conference Proceedings*, ed. B.L. Nelson, W.D. Kelton, and G.M. Clark, 1146-1155. Association for Computing Machinery, Baltimore, MD.
- Schriber, T. 1974. *Simulation Using GPSS*. New York: John Wiley and Sons.

AUTHOR'S BIOGRAPHY

DUANE R. BALL is a Principal Engineer at Advanced System Technologies, Inc. His research interests include simulation, computer performance engineering, and automated reasoning. He is a member of the IEEE Computer Society and the ORSA Air Traffic Control Special Interest Group.