Proceedings of the 1991 Winter Simulation Conference
Barry L. Nelson, W. David Kelton, Gordon M. Clark (eds.)

GPSS/H IN THE 1990s

Daniel T. Brunner
Robert C. Crain

Wolverine Software Corporation
4115 Annandale Road
Annandale, Virginia 22003

ABSTRACT

The 1980s were a decade of change in the discrete event simulation software industry. Simulation languages have been joined by animators, by "packages," and by "simulators" in the simulationist's bag of tricks. The expanding array of available tools reflects recognition by the marketplace that the simulation "user" may be an executive, an operations manager, a project engineer, a simulation engineer, a programmer, or a CAD operator. Each individual brings a different set of skills and expectations to the table. In this paper, we distinguish the "simulation end user" from the "simulationist," recognizing that these may sometimes be the same individual.

For a language product such as GPSS/H, this new environment presents challenges. However, the use of discrete event simulation languages has continued and has probably increased, even while the non-language tools have flourished. Many applications seem better suited for the language approach. In this paper, we describe the history of GPSS/H and the role of GPSS/H in the 1990s environment. We detail several near-and medium-term enhancements to GPSS/H that will strengthen this popular language during the coming years.

1 SIMULATION SOFTWARE IN THE 1980s

1.1 Languages

The decade of the 1980s began with four major languages having 1960s roots—SIMSCRIPT, GPSS, SLAM (descended from GASP), and SIMULA—and no other commercial discrete simulation software to speak of. However, by the end of the 1980s, there had been a tremendous flowering of products. Among simulation languages, GPSS had splintered into at least five separate implementations, supported by as many vendors. SIMAN had grown from birth to a major language.

Thanks in part to animation, simulation languages were at least as widely used in 1990 as they were in 1980. In some fields, such as manufacturing, the use of simulation languages mushroomed. Although languages are not directly useful to some simulation end users (executives and operations managers), there are plenty of engineers and analysts who prefer the language approach.

1.2 Animation

The news of the 1980s, however, lay not with the languages but with other simulation software. Beginning with SEE WHY, AutoMod, and GPSS/PC, and continuing with PC Model, Cinema, and TESS, there were at least six animation programs commercially available by the end of 1985. Purists (including, we must say, Wolverine) scoffed at the idea that pretty pictures (in some cases not so pretty) could add anything meaningful to the statistical science of simulation. But the purists were wrong, and now are few and far between. Animation has heightened interest in simulation among managers, has made simulationists more effective promoters of their ideas, and has even, in our opinion, become a significant tool for model verification and validation. With Wolverine's release of Proof Animation in 1991, every major discrete event simulation software vendor offers animation.

1.3 Packages

The other revolution in the 1980s was the birth of the "package" genre. We contrast the term "package" with the term "simulator," preferring to use the latter term to describe simulation tools that are tied to a particular narrow application, perhaps even to a single application or project (see below).

The packages are characterized by software interfaces that hide the underlying language (if there is one), and by promotional claims that "no programming is required." The intent seems to be to involve the busy manager or

the harried project engineer directly in simulation without the time commitment necessary in the past. Some packages try to help the simulation end user to become the simulationist, while others are engineering tools intended only to save a simulationist's time.

The real package revolution is still underway, having begun in the latter half of the 1980s. Some packages were built from scratch, but most derive either their concepts or their underlying engine from previous efforts. Packages introduced during the 1980s include XCELL, Witness, AutoMod II, SIMFACTORY, and ProModel. (We do not use the term "package" for tools that are really just graphical program editors in which individual statements are depicted as boxes on the screen. Some level of conceptual aggregation is necessary in order to have a "package.")

The pitfalls of packages are well known. They include added difficulty in programming complex constructs due to the trading of flexibility for ease of use; hidden assumptions that can render models invalid without suspicion by the user; and difficulty in validation due to lack of debugging tools. Still, it is safe to say that packages will continue to gain popularity assuming that vendors strive address these concerns.

1.4 Simulators

1.4.1 Tailoring the Model

When a simulationist develops a model and turns it over to the model end user for experimentation, he or she has delivered a *simulator*. Although it is a goal of most *packages* to "be" the completed model right out of the package, some development work is nearly always necessary in order to tailor the model for the situation at hand. It is impossible for a simulation software vendor to anticipate all possible scenarios. For a typical application, development work is necessary to create a simulator using either a language or a package.

The simulationist will have to do the work of tailoring the specific model. This raises several questions about how difficult it is to tailor the package or language. Must one (one should also ask, *can* one) program in FORTRAN? In a simulation language? By dragging boxes with a mouse? Is it even possible at all to tailor the model in the necessary ways? The answers depend on the simulation software.

1.4.2 Tailoring the End User Interface

Another component to a simulator is the interface presented to the model end user. Some software tools use the same interface for the simulationist and for the end user. This means that the end user has access to simulation primitives that may be confusing, no matter how "easy" the package is to use.

In all likelihood, the simulator end user only wants to change some data, or toggle among operational algorithms that have been pre-defined by the software vendor or by the simulationist (as described above). (Unless the end user is also the simulationist, it is not reasonable to expect the end user to create new algorithms. Algorithm description is programming, no matter how you do it or what you call it.)

If we restrict the end user to changing data or toggling among pre-defined algorithms, then most simulation tools can be used as simulators, given sufficient model development resources. We have seen various GPSS/H-based simulators that operate in any one of four modes:

(1) The end user edits a data file
(2) The model automatically queries a data file that has been updated in another context (this technique is used with models that are used for "day-to-day" operational decision making)
(3) The end user supplies data and parameters through a question-and-answer interface
(4) The end user supplies data and parameters through a forms-like user interface, which may or may not be mouse-driven

Case (4) is what many packages aspire to be "out of the box" for a particular application area. (Whether they succeed depends on the package and on the application.) Some of the slickest simulation applications we have seen use a forms-like interface to let a non-simulationist operator run a sophisticated language-based model of a particular system. Model development time is still an issue, which restricts the scope of potential applications somewhat. Software vendors must continue to strive to make building complex models—and the user interfaces that control them—a faster process.

1.5 All of the Above

The 1990s will see a blurring of the lines among languages, animators, packages, and simulators. Every vendor will push in the direction of providing quality implementations of all four capabilities. However, it is not easy to be all things to anyone. One has only to look at the limited success of the so-called "integrated" business applications (word processor, spreadsheet, database, etc., all rolled up into one program).

Multitasking, interapplication communication, and dynamic data exchange have begun to assert themselves in he IBM PC-compatible world. Soon it will be much easier to create a complete end user simulation

application in which multiple, interoperable tools are brought to bear on the model-building, end user interface, animation, and simulation engine parts of the problem.

2 GPSS/H IN THE 1980s

A history of GPSS/H in the 1980s is also a history of Wolverine. GPSS/H and Wolverine began the 1980s as a one-person operation. The first commercial version of GPSS/H was licensed (to General Motors) in 1977, and word of mouth brought a trickle and later a small stream of mainframe GPSS users who were hungry for the efficient execution offered by GPSS/H. By the end of the 1980s, GPSS/H was a major language in its own right, and Wolverine was a major vendor.

2.1 Platforms

GPSS/H milestones in the 1980s were largely platform-based. Wolverine hired its second employee in 1981, and also acquired a VAX computer. Thanks to the trend toward departmental computing, the VAX was expected to play a significant role in analytical computing, which it eventually did.

By 1983, VAX GPSS/H was shipping. Soon thereafter, Wolverine turned its attention to the new engineering workstations. Another simulation software vendor wanted to use GPSS/H as the engine for a workstation-based product. By 1986, GPSS/H was available for Silicon Graphics, Sun-3 and Apollo workstations, and for Integrated Solutions and NCR multiuser Unix-based microcomputers. All of these boxes used Motorola 68000 family processors.

But in 1986 and 1987, potential customers were only rarely asking for workstation simulation software. They wanted products to run on the IBM PC. So, in 1988, Wolverine introduced Personal GPSS/H. Limited by the 640K DOS memory ceiling, Personal GPSS/H gained momentum steadily but slowly. In 1989 Wolverine introduced Student GPSS/H, a full-speed, full-featured, limited-size modification of Personal GPSS/H, which is now available with either of two tutorial books.

Thanks to the emergence of "DOS Extender" technology, Wolverine in 1990 introduced a large-memory 32-bit version GPSS/H running under MS-DOS on 80386, 80386SX, and 80486 computers. GPSS/H 386 has quickly become very popular. We are very pleased to have permanently avoided 16-bit OS/2 1.x.

2.2 Features

When mainframe GPSS/H reached the 1.0 release designation in 1981, it contained the following basic enhancements over GPSS V:

- Built-in File and Screen I/O
- Use of any arithmetic expression as a Block operand
- Interactive debugger
- Improved External Routine Interface
- Dramatically faster execution
- Expanded Control Statement Language
- Ampervariables

GPSS/H Release 2.0, which appeared on various machines between 1986 and 1989, added several key enhancements:

- Floating Point Clock
- Built-in Math Functions
- Built-in Random Variate Generators
- Indexed Lehmer Random Number Generator

3 GPSS/H IN THE 1990s

As we move into the early '90s, GPSS/H is ready for more enhancements.

3.1 Language Features

Two key language features top users' lists: More Functional Assembly Sets, and Modifiable Function Definitions.

3.1.1 Assembly Set Extensions

GPSS has long offered Assembly Sets for managing grouped or batched transaction entities. If two or more transactions are part of the same Assembly Set, they can be automatically gathered (using GATHER) at a single point for simultaneous release (e.g. in a case palletizing system), or automatically consolidated into a single Transaction (using ASSEMBLE) (e.g. printed circuit boards being batched for transport). Two "siblings" can also wait at different points in the model for each other using a MATCH Block (e.g. in a communication system model).

Unfortunately, there is only one way for GPSS/H Transactions to become members of an Assembly Set, and that is through a SPLIT Block. A SPLIT Block creates one or more "children" that are in the same Assembly Set as their "parent." This means that two or more Transactions that come from any GENERATE Block can never be "siblings." Assembly Set membership cannot be changed after the SPLIT operation. This lack of flexibility has apparently caused many users to avoid the built-in Assembly Set capabilities of GPSS/H in favor of a "roll-your-own" approach.

Relief, in the form of more flexible GPSS/H Assembly Sets, is on the way. At this writing, three proposals are on the table. One is to have a special MATCH, GATHER, and ASSEMBLE based on Group membership. Another is to allow direct modification of the Assembly Set number of a Transaction. Finally, it might be possible to MATCH, GATHER, or ASSEMBLE based on a Transaction Parameter.

3.1.2 Function Re-definition

GPSS/H provides a well-known method for specifying discrete or continuous-valued (interpolated) user-defined Functions. Given an expression (often a random number sample) to use as the argument, and a list of X-Y pairs, GPSS/H will return the appropriate Y value (interpolated, in the case of a continuous Function) each time the Function is used in the model.

Functions are commonly used to represent inverse cumulative distribution functions for sampling from an empirical (non-fitted) probability distribution. (The old use of Functions to provide approximations for continuous fitted distributions is obsolete for distributions supported directly in GPSS/H.)

Unfortunately, Functions lose their luster if the empirical distribution is changing over time. Many users have requested the ability to change the list of X values "on the fly" (this can already be achieved for the Y values). Through "Function redefinition," GPSS/H will soon offer this capability. Using a new statement (probably (B)REBUILD), it will be possible to specify new X- and Y-lists (probably by giving the names of Ampervariable arrays).

3.2 Operating Platform

As PC system software makes the step-by-step transition from awkward 16-bit environments (MS-DOS, OS/2 1.x, Windows 3.0) to workstation-like 32-bit, large-memory systems (DOS Extenders, OS/2 2.0, "Win32," Unix, etc.), software developers must continue to adapt. Each new event (the release of MS-DOS 5.0 is an example) changes the ground rules for users of large memory software.

A main concern is to make Personal GPSS/H and GPSS/H 386 compatible with Windows—to the same extent as other text-mode DOS applications are compatible with Windows. This is not a problem for Personal GPSS/H, but has presented challenges for GPSS/H 386 because standards for managing conflicting demands for extended memory have only just begun to settle down. GPSS/H 386 needs to access this memory, as does Windows 3.0 in 386 Enhanced Mode and also as does MS-DOS 5.0.

By publication time, GPSS/H should be able to run under any Windows mode in conjunction with current versions of MS-DOS. This will be a major boon for GPSS/H users who are Windows devotees.

Soon we hope to allow GPSS/H 386 to share data with Windows applications via the Windows clipboard. We are particularly interested in allowing GPSS/H 386 to pass output data directly to Windows-based external analysis and graphics programs such SIMSTAT.

Further into the future is a Windows-based GPSS/H implementation with a redesigned debugger/monitor interface including graphical display of system internals.

3.3 Data Handling

GPSS/H should soon offer more tools for input modeling and output analysis.

3.3.1 Input Modeling

For those who like to fit raw data sets to closed-form continuous distributions (e.g. Exponential, Weibull, Gamma), third-party packages such as Unifit II can help with the fitting operation. Once a closed-form distribution has been identified, users seek the quickest method for incorporating the distribution into a model.

GPSS/H has lagged in supporting a wide array of closed-form distributions. However, we expect to offer either direct or indirect support for nearly every distribution in Unifit II in the very near future.

3.3.2 Output Analysis

GPSS/H automatically collects a variety of standard statistics, and can be made to gather just about any kind of statistical observations. The problem with most simulation languages is that with any model of meaningful size, there are so many individual data points that they must be aggregated while the simulation runs.

What does this aggregation mean? Consider GPSS/H Queues and Storages. GPSS/H will report (among other things) the total number of entries, the average contents, and the average time per transaction. However, these are summary statistics. GPSS/H Tables Tables can place raw observations in "bins," preserving enough information to create a histogram with a fixed format, and also reporting the mean and standard deviation. Still, these are summary statistics.

For time-varying statistics such as the contents of Queues, User Chains, or Storages, or for non-time-dependent output random variables such as Time In System, we need a way to preserve enough information to do more exhaustive statistical analysis and graphing. In short, we need to preserve each raw observation.

One can do this now by placing raw observations of a particular output variable into an array for later use, or directly into a file using (B)PUTPIC. However, this doesn't allow much analysis flexibility after the model has run. So, we are looking into ways to track voluminous output observations so that they can be easily exported, graphed, and analyzed during or after model execution. With a powerful analysis and graphing tool such as SIMSTAT running under Windows alongside GPSS/H, such analysis features as instant confidence interval calculation or quick autocorrelation analysis will be only a mouse click away.

3.4 Other Language Enhancements

Another enhancement is a new technique for declaring the number of Parameters in Transactions. Instead of requiring users to specify this at every GENERATE Block, it will be possible to change the system default by using a REALLOCATE statement such as REALLOCATE PH,4,PL,8. This will make GPSS/H not only easier to use but also easier to teach.

A new compiler directive, INCLUDE FILE=filename, will allow source to be pulled in from another file and included at compilation. This will be useful for Macros (Proof ATF generators, code libraries, etc.) and for externally generated probability distributions, especially lengthy empirical C-Type Functions, and probably for many other things that we have not thought of yet.

A new Standard Numerical Attribute will expand the already impressive built-in character handling capabilities of GPSS/H. The CHR() SNA will convert any integer from 0 through 255 to a Character data type.

4 SUMMARY

GPSS/H is a living language. Enhancements that improve the tool are a constant goal. All simulation software is moving inexorably in the direction of more sophistication. Users will expect speed, flexibility, animation, model building, and user interface building capabilities in every tool. GPSS/H is well positioned, through planned near- and medium-term improvements, to maintain its role as a powerful, state-of-the-art simulation tool in this environment.

REFERENCES

Banks, Jerry. 1991. Selecting Simulation Software. In *Proceedings of the 1991 Winter Simulation Conference*, eds. B.L. Nelson, W.D. Kelton, and G.C. Clark. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey.

Banks, Jerry, J.S. Carson II, and J.N. Sy. 1989. *Getting Started With GPSS/H.* Annandale, Virginia: Wolverine Software Corporation.

Blaisdell, W. 1991. *SIMSTAT for Windows 3.0 User's Manual.* Troy, New York: MC^2 Analysis Systems.

Brunner, D.T., N.J. Earle, and J.O. Henriksen. 1991. Proof Animation: The General Purpose Animator. In *Proceedings of the 1991 Winter Simulation Conference*, eds. B.L. Nelson, W.D. Kelton, and G.C. Clark. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey.

Crain, R.C. and D.T. Brunner. 1989. Extended Features of GPSS/H. In *Proceedings of the 1989 Winter Simulation Conference*, eds. E.A. MacNair, K.J. Musselman, and P. Heidelberger, 249-253. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey.

Crain, R.C. and D.T. Brunner. 1990. Modeling Efficiently With GPSS/H. In *Proceedings of the 1990 Winter Simulation Conference*, eds. O. Balci, R.P. Sadowski, and R. Nance, 89-93. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey.

Henriksen, J.O. and R.C. Crain. 1989. *GPSS/H Reference Manual*, Third Edition. Annandale, Virginia: Wolverine Software Corporation.

Schriber, T.J. 1991. *An Introduction to Simulation Using GPSS/H.* New York: John Wiley & Sons.

AUTHOR BIOGRAPHIES

DANIEL T. BRUNNER received a B.S. in Electrical Engineering from Purdue University in 1980, and an M.B.A. from The University of Michigan in 1986. He has been with Wolverine since 1986, where his responsibilities include product marketing, product development support, and simulation consulting. Mr. Brunner served as Publicity Chair for the 1988 Winter Simulation Conference and is Business Chair for the 1992 conference.

ROBERT C. CRAIN has been with Wolverine Software Corporation since 1981. He received a B.S. in Political Science from Arizona State University in 1971, and an M.A. in Political Science from The Ohio State University in 1975. Among his many Wolverine responsibilities is that of lead software developer for all PC and workstation implementations of GPSS/H. Mr. Crain is a member of IEEE/CS, SIGSIM, and ACM. He served as Business Chair of the 1988 Winter Simulation Conference and is General Chair of the 1992 Winter Simulation Conference.